
Epyk

Release 0.0.1

Epykure

May 01, 2023

CONTENTS

1	Presentation	3
2	Compatibility	5
3	Community & Support	7
4	Examples	9
5	Coming Soon	11
5.1	Getting started with Epyk	11
5.1.1	What is Epyk?	11
5.1.2	Installation	12
5.1.3	Dependencies	13
5.1.4	First Page	14
5.1.5	Quick start	14
5.1.6	Web Development	15
5.2	First Pages with Epyk	17
5.2.1	Step-by-step	17
5.2.1.1	Simple Page	17
5.2.1.2	Changing style	18
5.2.1.3	Adding events	18
5.2.1.4	Using external API	19
5.2.1.5	Offline mode	21
5.2.2	Other use cases	21
5.2.2.1	Dashboards	21
5.2.2.2	E-commerce	22
5.2.3	Using standard IDE	22
5.2.3.1	Epyk in IDE	22
5.2.3.1.1	Epyk in Visual Studio Code	22
5.2.3.1.2	Epyk in PyCharm	22
5.2.3.2	Epyk with Jupyter	22
5.2.3.2.1	Start a project	23
5.3	Ready to use examples	24
5.3.1	Important features	24
5.3.2	Epyk for beginners	25
5.3.3	Statics Page	25
5.3.3.1	Language comparison	25
5.3.3.2	COVID Dashboard	27
5.3.3.3	TESLA Shares	28
5.3.3.4	Pandas example	30

5.3.3.5	Advanced Dashboard	31
5.3.4	Web server integration	33
5.3.4.1	Flask	33
5.3.4.1.1	Show case report	33
5.3.4.1.2	Simple Workflow	34
5.3.4.2	Fast API	35
5.3.4.2.1	Log Viewer	35
5.3.4.2.2	Webscraping	37
5.3.4.2.3	Advanced Documentation viewer	38
5.4	Epyk and full stack development	41
5.4.1	Documentation	41
5.4.2	Profiling	42
5.4.3	Styles & Configuration	42
5.4.3.1	CSS Style	42
5.4.4	Properties	42
5.4.5	Virtual classes	43
5.4.5.1	CSS effects	43
5.4.6	Interactivity & events	43
5.4.6.1	Component events	43
5.4.7	Community	43
5.4.7.1	To Pyk	44
5.4.7.2	To Jupyter or JupyterLab	44
5.4.7.3	To CodePen	44
5.4.7.4	To W3Try	45
5.5	Design & Architecture	45
5.5.1	Architecture	45
5.5.1.1	Page	46
5.5.1.2	Components	46
5.5.1.3	HTML Component	47
5.5.2	Common API to easy the migration	49
5.5.3	Collaborative platform	49
5.5.4	Full stack development	50
5.5.5	Data Driven Design Architecture	50
5.5.6	Other related topics	50
5.5.6.1	A design for the web	50
5.5.6.1.1	Be an actor	51
5.5.6.2	Supported Libraries and Frameworks	52
5.5.6.3	127 Libraries	52
5.5.6.4	Framework	56
5.5.6.4.1	Import Manager	56
5.6	Web frameworks	62
5.6.1	Community	62
5.6.2	Jquery UI	62
5.6.3	ToastUI	63
5.6.4	Bootstrap	63
5.6.5	Material Design UI	63
5.6.6	Clarity	63
5.6.7	Evergreen	63
5.6.8	Common Interface	63
5.7	Security and Control	65
5.7.1	External libraries	65
5.7.2	Offline mode	65
5.7.2.1	Install packages	65
5.7.2.2	Update packages	66

5.7.2.3	Install all packages	66
5.7.2.4	Restrict the scope	66
5.7.2.5	Change versions	66
5.7.2.6	Add packages	66
5.7.2.7	Add packages	66
5.7.2.8	Checking packages and versions	66
5.7.3	Google extensions	67
5.8	Advanced features	67
5.8.1	CLI Features	67
5.8.1.1	Export (core) features	67
5.8.1.2	NPM Wrappers	68
5.8.1.3	Project features	70
5.8.2	Page	72
5.8.2.1	UI Interface	72
5.8.2.1.1	External Interfaces:	72
5.8.2.1.2	Interfaces per Categories:	72
5.8.2.1.3	Full list of 81 Interfaces:	479
5.8.2.2	Javascript Interface	492
5.8.2.2.1	Console	492
5.8.2.2.2	Window	495
5.8.2.2.3	Location	510
5.8.2.2.4	Shortcut and Features	516
5.8.2.2.5	Technical Documentation	516
5.8.2.3	Outputs Interface	537
5.8.2.4	Python Interface	541
5.8.2.4.1	Dates	541
5.8.2.4.2	Mails	546
5.8.2.4.3	Markdown	546
5.8.2.4.4	NPM	547
5.8.2.4.5	ReST	547
5.8.2.4.6	Database	550
5.8.2.4.7	Cryptography	558
5.8.2.4.8	Extension	559
5.8.2.4.9	Geo	562
5.8.2.4.10	Hash	562
5.8.2.5	Data Interface	563
5.8.2.5.1	Data Transformers	563
5.8.2.5.2	Interface Documentation	563
5.8.2.6	Web Framework Interfaces	569
5.8.2.7	Example	569
5.8.2.8	Technical Documentation	569
5.8.3	Themes	575
5.8.4	HTML Built-Ins	575
5.8.5	CSS Built-Ins	575
5.8.5.1	Core Modules	575
5.8.5.1.1	CSS Class	575
5.8.5.1.2	CSS Style	577
5.8.5.1.3	Colors	580
5.8.6	Javascript Built-Ins	581
5.8.6.1	Core Modules	581
5.8.6.2	JavaScript features	602
5.8.6.2.1	Console	602
5.8.6.2.2	Json	605
5.8.6.2.3	BreadCrumb	606

	5.8.6.2.4	Screen	607	
	5.8.6.2.5	Maths	608	
	5.8.6.2.6	Location	614	
	5.8.6.2.7	Navigator	619	
	5.8.6.2.8	Window	619	
	5.8.6.2.9	WebWorkers	620	
	5.8.6.2.10	Websocket	620	
	5.8.6.2.11	Performance	623	
5.8.7	Standards		627	
	5.8.7.1	Header Module	627	
	5.8.7.2	Aria Module	636	
	5.8.7.3	KeyCodes Module	644	
5.8.8	Entities		648	
	5.8.8.1	SymbArrows Module	648	
	5.8.8.2	SymbCurrencies Module	648	
5.8.9	Symboles		648	
	5.8.9.1	EntHtml4 Module	648	
	5.8.9.2	EntHtml5_A Module	648	
	5.8.9.3	EntHtml5_B Module	648	
	5.8.9.4	EntHtml5_C Module	648	
	5.8.9.5	EntHtml5_D Module	648	
5.9	External resources		648	
	5.9.1	Python Packages	648	
		5.9.1.1	Libraries	
			5.9.1.1.1	Pandas
			5.9.1.1.2	pandas_datareader
			5.9.1.1.3	Matplotlib
		5.9.1.2	Servers	650
	5.9.2	JavaScript Packages		650
		5.9.2.1	Libraries	651
			5.9.2.1.1	Slideshow
			5.9.2.1.2	Json Formatter
			5.9.2.1.3	Font awesome
			5.9.2.1.4	Mathjax
			5.9.2.1.5	Jquery UI
			5.9.2.1.6	PivotTable.js
		5.9.2.2	UI Frameworks	657
		5.9.2.3	Web Frameworks	657
5.10	Library extensions			657
	5.10.1	Styles & configurations		658
		5.10.1.1	Component style	659
			5.10.1.1.1	Using CSS inline
			5.10.1.1.2	Using CSS Inline object
			5.10.1.1.3	Using CSS classes
			5.10.1.1.4	Using external CSS
		5.10.1.2	Derived component	661
		5.10.1.3	Package extension	662
		5.10.1.4	Page skins	662
5.11	Bugs & ToDo			663
	5.11.1	Style		663
		5.11.1.1	Global CSS	663
	5.11.2	Data		663
	5.11.3	JavaScript		663
	5.11.4	Packages		663

5.11.4.1	PivotTable	663
5.11.4.2	Sparklines	663
Python Module Index		665
Index		667



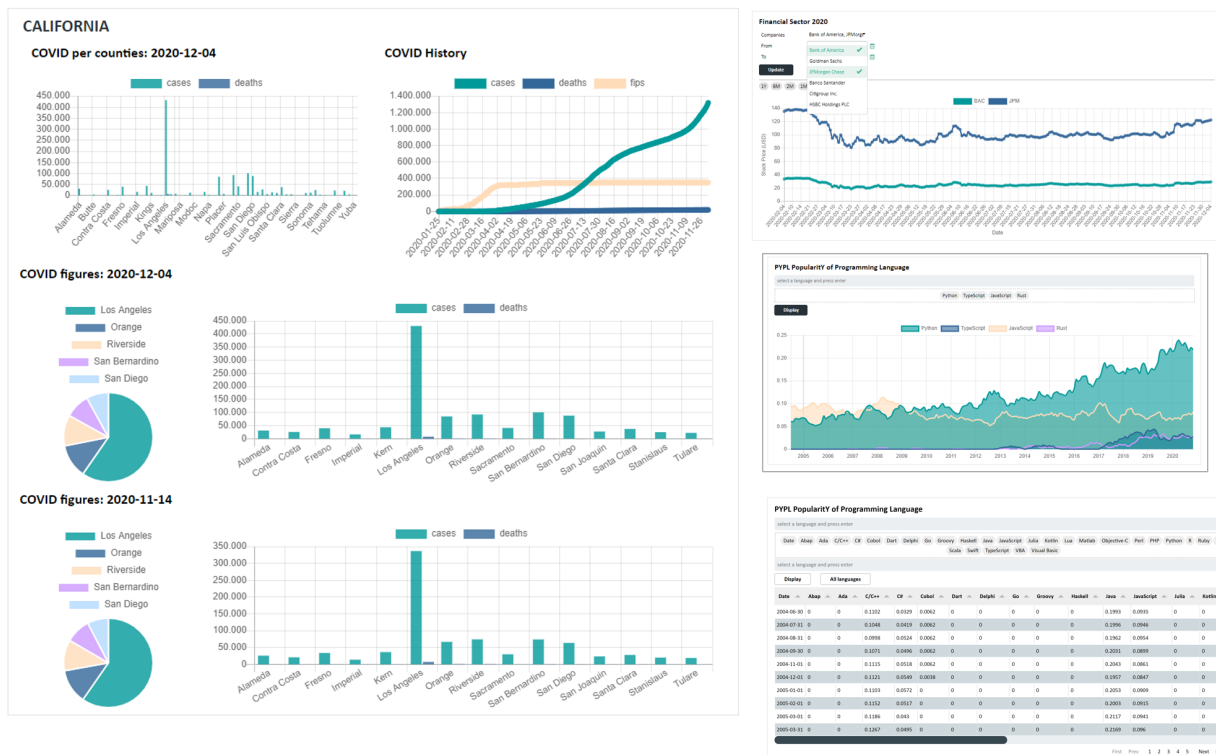
CHAPTER ONE

PRESENTATION

The target of Epyk is to ensure the implementation of a coherent system using a minimum of layers. With Epyk the user stays in the Python layer to drive and optimize the data transformation. This Framework also encourages the implementation of Micro services and cloud based architecture.

This package will allow you to easily create rich and interactive web interfaces to your projects.

No JavaScript, CSS or HTML5 knowledge needed, the platform will provide you the best experience thanks to the community.



COMPATIBILITY

Epyk is compatible with the most common Web Python Frameworks (Flask and Django). By default, the server package embeds a Flask app as it is easier to install and ready to use.

The Framework can be included within a jupyter/Jupyter or jupyter/JupyterLab project. But this will lead to some limitations - for example Ajax and Socket will not be available.

COMMUNITY & SUPPORT

Epyk is an **OpenSource** module dedicated to provide functions and components to improve the productivity. Do not hesitate to participate in improving the library.

Any help are welcome and this can be done by JavaScript developers interesting to learn Python or by Python developers.

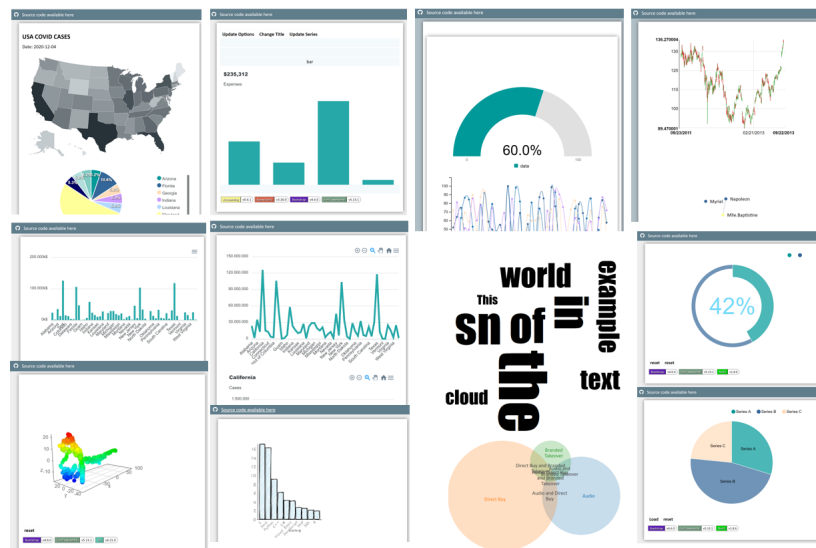
Also please do not forget to star our project on [Github](#) or to follow us on [Twitter](#) or on [LinkedIn](#) to get the latest news.

EXAMPLES

You can find examples of what Epyk can do right here:

- [Components](#)
- [Dashboards](#)
- [Websites](#)
- [Jupyter Playground](#)

Also get online demo from the [Epyk Gallery](#)



COMING SOON

Please see below the 2021 road map of notifications on Twitter:

Do not hesitate to follow us on [Twitter](#) or to help us improving our code by sending pull requests on [Github](#) !

5.1 Getting started with Epyk

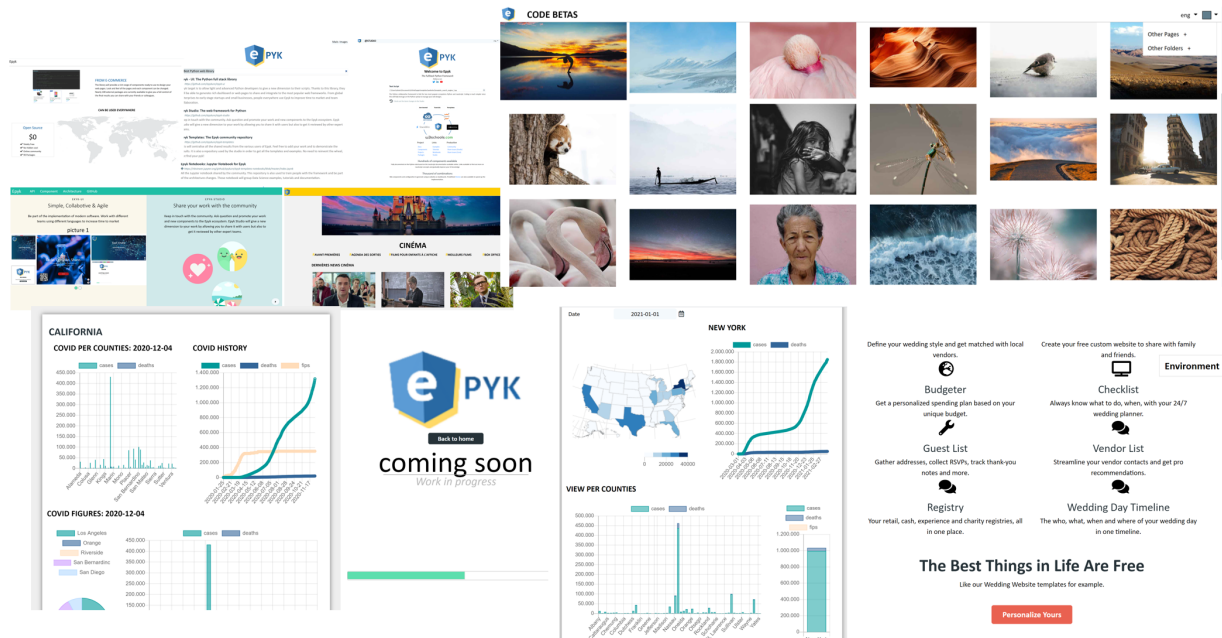
Epyk is designed for Python developers or users interested in implementing an entire product without having to change technologies.

5.1.1 What is Epyk?

Epyk is an OpenSource library designed to improve time to market for Data Scientists and any people eager to prototype a new / or an extension to a product in a quite flexible manner.

The main targets of Epyk are to:

- Allow Python developers to present and share their work to users.
- Simplify the prototyping by using **components** and **templates**.
- Easy to start for beginners.
- Help to **ramp up with web technologies** by learning those concepts.
- Be used as a **web toolbox** to improve time to market and the learn web technologies.



5.1.2 Installation

Assuming you have Python already, install Epyk:

```
pip install epyk
```

Create a directory inside your project to hold your ui and run the epyk-new command::

```
epyk.exe new
```

This command will create a first empty report in your folder. Then the below command will convert (transpile) it to a web page:

```
epyk.exe transpile
```

See also:

More details on the [CLI Features](#).

For a quick examples it is possible to use the below CLI:

```
epyk.exe demo
```

The will generate the below script epyk_demo.py in the current directly:

```
epyk.exe transpile -n=epyk_demo
```

This script will demo some common features available in the library:

```
import epyk as pk

# Module with mock data
from epyk.tests import mocks
```

(continues on next page)

(continued from previous page)

```

# Create a basic report object
page = pk.Page()
page.headers.dev()

page.body.template.style.configs.doc()

# Change the CSS style of the div template container
page.body.template.style.css.background = "white"

table = page.ui.table(mockes.popularity_2020)
table.options.paginationSize = 10

toggle = page.ui.toggle({"on": "Trend", "off": "Share"})
bar = page.ui.charts.bar(mockes.popularity_2020, y_columns=["Share"], x_axis="Language")

toggle.click([
    # Store the variable to myData on the JavaScript side
    pk.std.var("myData", sorted(mockes.popularity_2020, key=lambda k: k['Language'])),
    # Use the standard build and dom.content to respectively update and get the component_
    ↪ value
    pk.expr.if_(toggle.input.dom.content.toStr(), [
        # Use the variable to update the chart
        bar.build(pk.std.var("myData"), options={"y_columns": ["Trend"]})
    ]).else_([
        bar.build(pk.std.var("myData"), options={"y_columns": ["Share"]})
    ])
])

```

Another example is available in the [Github templates](#) repo to illustrate how to adapt a script to backend services. Very few changes are required to add a backend post to an underlying service:

```

toggle.click([
    page.js.post("/data", components=[toggle.input]).onSuccess([
        bar.build(pk.events.data["chart_data"], options={"y_columns": pk.events.data["columns
    ↪"]})
    ])
])

```

5.1.3 Dependencies

Epyk does not require any python dependency. The idea of this library is to code with any packages or frameworks used in both Python and JavaScript side. Thus it will not impose any library to run. Obviously component are done in a way to deal with records (list of dictionaries) which are common objects in Pandas.

The only thing required to work is an internet connection to be able to retrieve the external packages on which Epyk leverage to render the page.

Those packages are not part of the project and they are developed independently.

See also:

his is not a new Visualisation library with some core Javascript embedded modules, this is a unique library which link your python code to external and already popular JavaScript and CSS packages

5.1.4 First Page

In Epyk the object used to create the final web page is called a *Page*. The page object will be the one available to each component in charge of triggering the *.html()* on each components.

The unique page object will store all the components in a *page.components* dictionary. Each component will be in charge of defining its JavaScript bindings, its HTML5 structure and also its need in external resources guides/component-structure.

5.1.5 Quick start

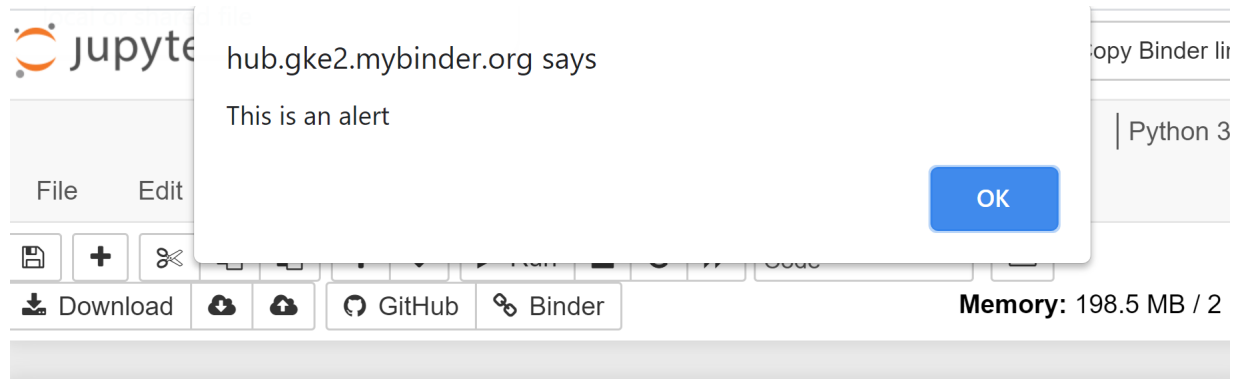
The below will illustrate how to start with Epyk and build your first report. This will write the web artifacts locally:

```
import epyk as pk

page = pk.Page()
text = page.ui.text("This is a test")
text.click([page.js.alert("This is an alert")])
page.outs.html()
```

the last line *page.outs.html()* is important since it will produce the final web files which can be used in a browser. It will be very often not mentioned in the examples because this can vary according to the framework used.

In the below example *page.outs.jupyter()* is used instead to render the example from an online session of Jupyter notebook.



```
In [1]: ► pip install epyk
```

```
In [7]: ► import epyk as pk

page = pk.Page()
text = page.ui.text("This is a test")
text.click([page.js.alert("This is an alert")])

page.outs.jupyter()
```

```
Out[7]: This is a test
```

Also this is not needed if you use the *CLI Features* to render the page.

The best to get more familiar with Epyk is to use PyCharm and the code autocompletion or to start with examples on the template [Github repository](#)

5.1.6 Web Development

From Epyk it is possible to interface any Python project using any Backend technology. It can be used like Jinja to render rich HTML content on the fly.

The below example illustrate how to generate an interactive dashboard with few lines of codes and without any server:

```
page = pk.Page()
#page.theme.color_palette("brewer.PuBu8")

# Set the page layout
template = page.body.add_template(defined_style="doc")
template.style.css.background = page.theme.greys[0]

page.ui.title("Pandas tutorial #5")
page.ui.texts.references.website(url="https://towardsdatascience.com/data-visualization-
↪with-bokeh-in-python-part-ii-interactions-a4cf994e2512")
source_data = "https://raw.githubusercontent.com/WillKoehrsen/Bokeh-Python-Visualization/
↪master/interactive/data/complete_flights.csv"
```

(continues on next page)

(continued from previous page)

```

df = pd.DataFrame(page.py.requests.csv(source_data, store_location=r"C:\tmps"))
df["dep_delay"] = pd.to_numeric(df["dep_delay"], downcast="float")
df["distance"] = pd.to_numeric(df["distance"], downcast="float")
agg_df = df.groupby(["name", "month", "distance"])["dep_delay"].sum()
agg_df = agg_df.reset_index()

max_value = float(df["distance"].max())
checks = page.ui.lists.checks(list(df["name"].unique()))
dist = page.ui.fields.static("[0, %s]" % max_value, label="Min, Max distance")
slider = page.ui.sliders.range([0, max_value], maximum=max_value)

chart = page.ui.charts.chartJs.bar(y_columns=["dep_delay"], x_axis="month")
table = page.ui.table(rows=["name", "month", "distance"], cols=["dep_delay"])
table.config.pagination = "local"
table.config.paginationSize = 10

pie = page.ui.charts.chartJs.pie(y_columns=["dep_delay"], x_axis="name")
pie_count = page.ui.charts.chartJs.pie(y_columns=["count"], x_axis="name")

grp = page.data.js.record(agg_df.to_dict(orient="records")).filterGroup("aggData")
grp2 = page.data.js.record(agg_df.to_dict(orient="records")).filterGroup("aggData2")

update_button = page.ui.buttons.colored("update")

row = page.ui.row([checks, update_button, chart], position="top")
row.set_size_cols(4)

total_delay = page.ui.titles.subtitle("Total Delay")
count_delay = page.ui.titles.subtitle("Count Delay")
page.ui.row([total_delay, pie], [count_delay, pie_count], position="top")

toggle = page.ui.buttons.toggle({'on': "More than 3 hours", 'off': 'All delays'}, label=
↪ "Delay Type",)

hamburger = page.ui.panels.hamburger([dist, slider, toggle], title="Details")
table.move()

page.ui.layouts.hr()
page.ui.titles.subtitle("Report powered by")
page.ui.rich.powered()

```

The below will add the interactivity and the JavaScript data transformation:

```

toggle.input.click([
    page.js.if_(
        toggle.input.dom.content.isTrue(), [
            chart.build(grp.includes("name", checks.dom.content, empty_all=False).sup("dep_
↪ delay", 180).sup("distance", slider.dom.min_select).inf("distance", slider.dom.max_
↪ select).group().sumBy(["dep_delay"], ["month"], cast_vals=True)),
            table.build(grp.includes("name", checks.dom.content, empty_all=False).sup("dep_
↪ delay", 180).sup("distance", slider.dom.min_select).inf("distance", slider.dom.max_
↪ select)),

```

(continues on next page)

(continued from previous page)

```

    pie.build(grp.includes("name", checks.dom.content, empty_all=False).sup("distance",
    ↪ slider.dom.min_select).inf("distance", slider.dom.max_select).group().sumBy(["dep_
    ↪ delay"], ["name"], cast_vals=True)),
    pie_count.build(grp.includes("name", checks.dom.content, empty_all=False).sup(
    ↪ "distance", slider.dom.min_select).inf("distance", slider.dom.max_select).group().
    ↪ countBy(["name"])))
    ]).else_([
        chart.build(grp2.includes("name", checks.dom.content, empty_all=False).sup(
    ↪ "distance", slider.dom.min_select).inf("distance", slider.dom.max_select).group().
    ↪ sumBy(["dep_delay"], ["month"], cast_vals=True)),
        table.build(grp2.includes("name", checks.dom.content, empty_all=False).sup(
    ↪ "distance", slider.dom.min_select).inf("distance", slider.dom.max_select)),
        pie.build(grp2.includes("name", checks.dom.content, empty_all=False).sup("distance
    ↪ ", slider.dom.min_select).inf("distance", slider.dom.max_select).group().sumBy(["dep_
    ↪ delay"], ["name"], cast_vals=True)),
        pie_count.build(grp2.includes("name", checks.dom.content, empty_all=False).sup(
    ↪ "distance", slider.dom.min_select).inf("distance", slider.dom.max_select).group().
    ↪ countBy(["name"])))
    ]),
    dist.input.build(slider.dom.content)
])

# Reuse the code generated by the toggle click for the below components.
update_button.click(toggle.input.event_fnc("click"))
slider.change(toggle.input.event_fnc("click"))

```

By adding `page.outs.html()` it will generate a file which can be opened in a browser

5.2 First Pages with Epyk

It is possible to implement simple static page to rich and interactive ones. Epyk will provide interfaces to components and events which can be used as an simple interface to a model or data source to a more complex ones interacting with any backend servers.

Warning: The line `page.outs.jupyter()` need to be added to run in online Jupyter notebooks.

5.2.1 Step-by-step

5.2.1.1 Simple Page

Let's start with simple chart with the hard coded values, to do so we will add components available to `page.ui`:

```

import epyk as pk

data = [
    {"x": "label 1", "y": 23},

```

(continues on next page)

(continued from previous page)

```
    {"x": "label 2", "y": 10},
]
page = pk.Page()
page.ui.charts.chartJs.line(data, y_columns=["y"], x_axis="x")
```

Note: By convention common components are attached directly to the `page.ui` property.

Plural names are used by convention to provide different flavour of them `page.ui.fields`

5.2.1.2 Changing style

The below will change the color of the lines:

```
import epyk as pk

data = [
    {"x": "label 1", "y": 23, "y2": 53},
    {"x": "label 2", "y": 10, "y2": 26},
]
page = pk.Page()
chart = page.ui.charts.chartJs.line(data, y_columns=["y", "y2"], x_axis="x")
chart.colors(["red", "green"])
```

5.2.1.3 Adding events

This will create:

```
import epyk as pk

data = [
    {"x": "label 1", "y": 23, "y2": 53},
    {"x": "label 2", "y": 10, "y2": 26},
]
page = pk.Page()
chart = page.ui.charts.chartJs.line(data, y_columns=["y", "y2"], x_axis="x")
chart.colors(["red", "green"])
console = page.ui.rich.console()
chart.click([
    console.dom.write(chart.activePoints().value.toString().prepend("Clicked point: "))
])
```

DESIGNER eng

PREVIEWS

Editor Web

▲ External report

◀ Undo ▶ Redo

```

1 # Epyk Designer
2 |
3 import epyk as pk
4
5 data = [
6     {"x": "label 1", "y": 23, "y2": 53},
7     {"x": "label 2", "y": 10, "y2": 26},
8 ]
9 page = pk.Page()
10 chart = page.ui.charts.chartJs.line(data,
11     y_columns=["y", "y2"], x_axis="x")
12 chart.colors(["red", "green"])
13 console = page.ui.rich.console()
14 chart.click([
15     console.dom.write(chart.activePoints().value
16         .toString().prepend("Clicked point: "))
17 ])

```

DESIGNER eng

PREVIEWS

Editor Web

new_page_2.py

label 1 label 2

```

> 2021-04-30 19:02:10, Clicked point: 53
> 2021-04-30 19:02:12, Clicked point: 10
> 2021-04-30 19:02:12, Clicked point: 10

```

5.2.1.4 Using external API

This will create:

```

import epyk as pk

# Socket server url
SERVER_SOCKET_HOST = "127.0.0.1"
SERVER_SOCKET_PORT = 5000

from flask import Flask
app = Flask(__name__)

def create_page():
    page = pk.Page()
    page.headers.dev()

    title = page.ui.title("Flask - First example")
    input = page.ui.inputs.left(placeholder="Enter your name", html_code="msg")
    button = page.ui.buttons.colored("Click")
    text = page.ui.text()
    simple_modal = page.ui.modals.popup([text])
    input.enter([button.dom.events.trigger("click")])
    button.click([page.js.post("/test_event", components=[input]).onSuccess([
        text.build(pk.events.data["message"]),
        simple_modal.dom.show()
    ])])

```

(continues on next page)

(continued from previous page)

```

box = page.ui.div()
box.extend([title, input, button])
box.style.configs.doc()
return page

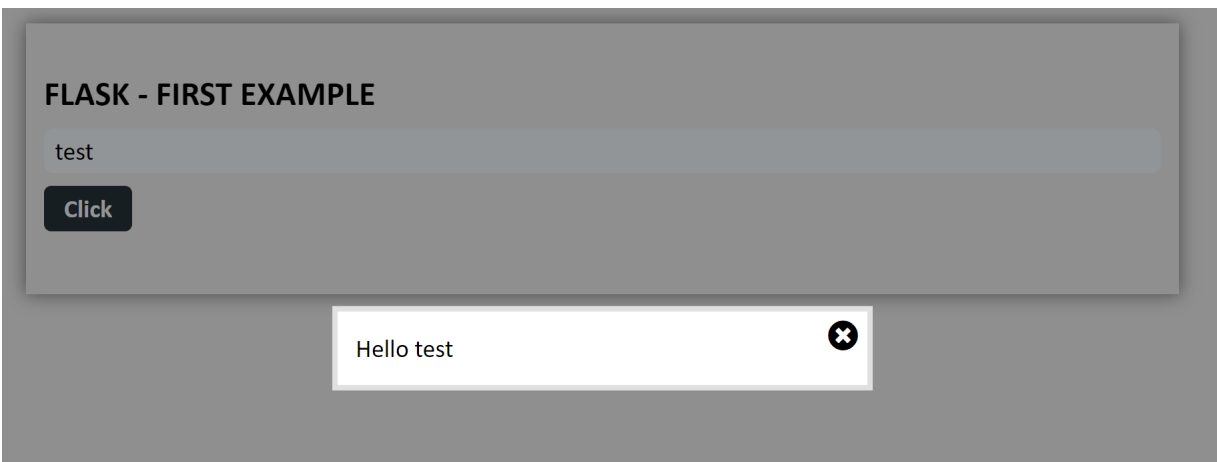
if __name__ == "__main__":
    from flask import Flask, jsonify, request

    @app.route('/')
    def ui():
        return create_page().outs.html()

    @app.route('/test_event', methods=['POST'])
    def test_event():
        data = request.get_json()
        return jsonify({"message": "Hello %s" % data['msg']})

    Flask.run(app, host=SERVER_SOCKET_HOST, port=SERVER_SOCKET_PORT, debug=True)

```

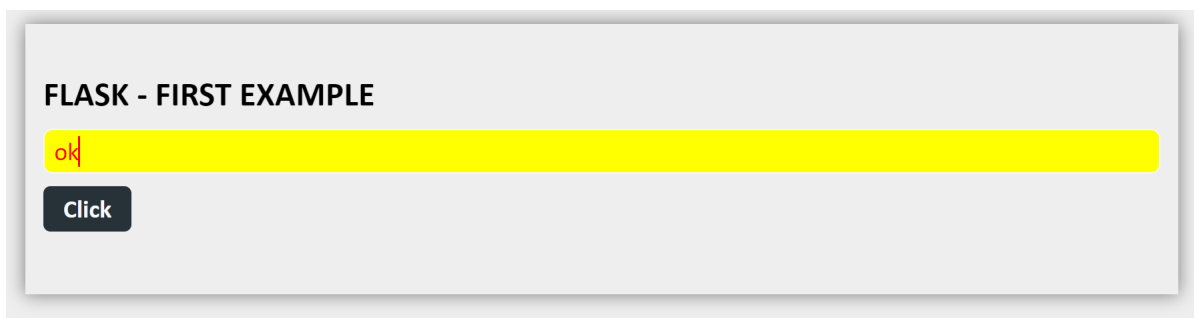


In the above example it is possible to change all the CSS properties of the components. For example, the below will change the color and the background color:

```

input = page.ui.inputs.left(placeholder="Enter your name", html_code="msg")
input.style.css.color = "red"
input.style.css.background = "yellow"

```



5.2.1.5 Offline mode

It is possible to run offline but this will require to get the list of external packages downloaded first.

To do so from a script some functions and CLI are embedded in the library.

1. Get the list of requirements.

At the end of any script it is possible to call `page.imports.requirements` to get the list of all the external packages. This will return the different alias (the npm aliases) used to defined an external package (some JavaScript and CSS pages):

```
page = pk.Page()
page.headers.dev()

page.body.template.style.configs.doc(background="white")
...

print(", ".join([r for r in page.imports.requirements]))

epyk_npm.exe install -pkg=promise-polyfill,@popperjs/core,bootstrap,showdown,jquery,
↪accounting,tabulator-tables,moment,chart.js
```

2. Install the packages locally

A CLI is available to install all the packages to a dedicated path:

```
epyk_npm.exe install -pkg=promise-polyfill,@popperjs/core,bootstrap,showdown,jquery,
↪accounting,tabulator-tables,moment,chart.js
```

By default this will download the package and create a statics folder at the root

3. Set a offline mode

Then change the outs definition to set the path of the external packages:

```
page.imports.static_url = "C:\epyks\statics"
page.outs.html_file(name="test.html", options={"split": True, "minify": False, "static_
↪path": page.imports.static_url})
```

In the function `page.outs.html_file` some options are available to split the outputs and change the formatting.

5.2.2 Other use cases

Epyk can be used to do more complicated dashboards with multiple events and interaction with the backend server.

5.2.2.1 Dashboards

This one will get data from datareader and return the result to be then display in a pivotTable. To insert those dependencies to the final web page it is only required to call those objects from the ui interfaces.

Thus we can find the below pieces in the code:

```
ticker = page.ui.fields.select(
cpns.select.from_dict(tickers_info), label="Tickers", multiple=True, html_code="ticker_
↪value")
```

(continues on next page)

(continued from previous page)

....

```
pivot = page.ui.tables.pivots.plotly()
```

FLASK - ADVANCED EXAMPLE 2

Bootstrap v4.5.3
Bootstrap-select v1.13.18
Font-awesome v5.13.1
Jquery v3.6.0
Jqueryui v1.12.1
Pivot-plotly v2.23.0

Pivottable v2.23.0
Showdown v1.9.1

Tickers Goldman Sachs ▼
From 2021-02-26
To 2021-04-30
Click

FINANCIAL SECTOR

Table ▼
Sum ▼
Date ▼

value ▼
value ▼
type ▼

value ▼
type ▼

	Close	High	Low	Open	Price	V
2021-02-26	319.48	328.68	319.04	325.77	322.63	3,7
2021-03-01	329.92	331.04	323.31	325.00	327.46	2,9

5.2.2.2 E-commerce

5.2.3 Using standard IDE

Epyk can be used in any IDE and it will provide autocompletion to simplify the implementation.

5.2.3.1 Epyk in IDE

5.2.3.1.1 Epyk in Visual Studio Code

5.2.3.1.2 Epyk in PyCharm

5.2.3.2 Epyk with Jupyter

Epyk can be used either with a local instance of Jupyter or directly online from Jupyter Notebooks.

In this section we will provide online examples.

5.2.3.2.1 Start a project

Go to [Jupyter](#) and start a Python project.

Add Epyk to your online notebook

```
In [1]: ► pip install epyk

Collecting epyk
  Downloading epyk-1.5.15-py2.py3-none-any.whl (3.0 MB)
    |████████████████████████████████████████| 3.0 MB 4.3 MB/s eta 0:00:
01
Installing collected packages: epyk
Successfully installed epyk-1.5.15
Note: you may need to restart the kernel to use updated packages.
```

Let's start with a simple cell and a slider component:

```
import epyk as pk

page = pk.Page()
slider = page.ui.slider()
text = page.ui.text(0)
slider.change([
    text.build(slider.dom.content)
])
page.outs.jupyter()
```

```
In [4]: ► import epyk as pk

page = pk.Page()
slider = page.ui.slider()
text = page.ui.text(0)
slider.change([
    text.build(slider.dom.content)
])
page.outs.jupyter()
```

Out[4]:



21

We can add tables to this notebook:

```
import epyk as pk
from pandas_datareader import data as pddr

data = [
    {"x": "label 1", "y": 23, "y2": 53},
    {"x": "label 2", "y": 10, "y2": 26},
```

(continues on next page)

(continued from previous page)

```

]
page = pk.Page()
title = page.ui.title("Data Reader")
s = pddr.DataReader("BAC", 'yahoo', "2021-04-01", "2021-04-20")
page.ui.tables.datatable(s.to_dict(orient="records"), cols=['Close'])
page.outs.jupyter()

```

```

In [28]: import epyk as pk
         from pandas_datareader import data as pddr

         data = [
             {"x": "label 1", "y": 23, "y2": 53},
             {"x": "label 2", "y": 10, "y2": 26},
         ]
         page = pk.Page()
         title = page.ui.title("Data Reader")
         s = pddr.DataReader("BAC", 'yahoo', "2021-04-01", "2021-04-20")
         page.ui.tables.datatable(s.to_dict(orient="records"), cols=['Close'])
         page.outs.jupyter()

```

Out[28]: **DATA READER**

Show entries Search:

Close
38.08000183105469
38.7400016784668
39.150001525878906
39.16999816894531
39.31999969482422

More example should be provided soon in this section.

You can also refer to the previous examples created to show case the concept of [Epyk](#)

Caution: Include paths are relative to the file in the document project, not the file in shared content.

5.3 Ready to use examples

This section will details some examples available on [Github](#)

It will illustrate some example and provide more explanation concerning the modules and functions used.

More ready to use dashboards are available in the [Epyk Gallery](#)

5.3.1 Important features

Only **12** functions are needed to be able to write interactive dashboards.

Table 1: Epyk common features

Scope	Function	Description
Page	theme	Change theme of the page
Page	theme.colors	Get the theme colors
Page	ui	Get all the components in the framework
Page	body	Get the body (main component) of the page on which components will be attached to.
Page	js	Get access to the JavaScript standard features
Page	js.post / js.get	Call a backend REST API
Page	outs	Page output property (html, codepen, Jupyter....)
Component	style.css	Get all the CSS properties
Component	js	Get the component specific js features
Component	dom.content	Get the component value on the UI side
Component	click	Add UI click event on the component
Component	build	Update a component in a UI event function

Note: This is a collaborative framework so not hesitate to share your work to add more examples to this list.

5.3.2 Epyk for beginners

This should be a easy start for people not really familiar with either Python or JavaScript but eager to learn and produce results quickly (without to deal with thousand of packages, technologies and concepts).

The below links will propose two approaches according to your appetite:

5.3.3 Statics Page

Those examples are interactive pages working as standalone HTML pages. No server is required to run them.

This is usually the starting point of any web pages.

5.3.3.1 Language comparison

The below line to create a new web page:

```
page = pk.Page()
page.headers.dev()
page.theme = ThemeBlue.BlueGrey()
```

This will add an external JavaScript page online:

```
page.js.customFile("FR.js", r"http://pypl.github.io/PYPL")
```

Add two components:

```
title = page.ui.title("PYPL Popularity of Programming Language")
items = page.ui.inputs.autocomplete(placeholder="select a language and press enter",
    options={"select": True})
cols_keys = page.ui.lists.drop(html_code="cols_agg_keys")
```

Change the slyde and add event to the list object:

```
cols_keys.style.css.min_height = 20
cols_keys.items_style(style="bullets")
cols_keys.drop()

button = page.ui.buttons.colored("Display")
button.style.css.margin_top = 5

items.options.on_select([
    cols_keys.dom.add(events.value),
    button.dom.events.trigger("click")
])
```

Add the chart component and change the style:

```
line = page.ui.charts.chartJs.line(x_axis="Date", profile=True)
line.options.scales.x_axes().type = "time"
line.options.elements.point.radius = 0
line.options.scales.x_axes().distribution = 'linear'
```

Add a powered component to illustrate the external modules used from Epykt:

```
tag = page.ui.rich.powered()
tag.style.css.margin_bottom = 5
tag.style.css.margin_top = 5

box = page.studio.containers.box()
box.extend([title, tag, items, cols_keys, button, line])
box.style.standard()
```

Add extra events to the components:

```
items.enter([cols_keys.dom.add(items.dom.content), items.dom.empty()])

button.click([
    std.var("graphData").fromArrayToRecord().setVar("records"),
    line.build(std.var("records"), options={"y_columns": cols_keys.dom.content, "x_axis":
    "Date"})
])
```

Run some functions when the page is loaded in the browser:

```
page.body.onReady([items.js.source(std.var("graphData"))[0]])
```

This example is available [here](#)

More example of the templates on [Github](#)

5.3.3.2 COVID Dashboard

This will create the page object and add a predefined configuration to the page.body:

```
page = Report()
page.theme = ThemeBlue.BlueGrey()
```

Get the external data from a URL directly:

```
data = page.py.requests.csv(data_urls.COUNTRY_WISE_COVID)
```

Add components to the page:

```
title = page.ui.title("Country wise COVID")
button_all = page.ui.button("All")
button_all.style.css.margin_top = 5
button_clear = page.ui.button("Clear")
```

Change the CSS style of the button:

```
button_clear.style.css.margin_top = 5
button_clear.style.css.margin_left = 5

cols_keys = page.ui.panels.filters(html_code="data_filters", options={"max_height": 90})
cols_keys.style.css.min_height = 20
```

Add an autocomplete input component and set an event when enter is pressed:

```
items = page.ui.inputs.autocomplete(placeholder="select a country", options={"select":_
↳ True})
items.enter([cols_keys.dom.add(items.dom.content, category='Country/Region'), items.dom.
↳ empty()])

button = page.ui.button("Show")
button.style.css.margin_top = 5
```

Add component specific options, in this case it is an event when selected:

```
items.options.on_select([
    cols_keys.dom.add(events.value, category='Country/Region'),
    button.dom.events.trigger("click")
])
```

Add other components to the page:

```
bar = page.ui.charts.chartJs.bar([], ['Confirmed', 'Deaths', 'Recovered'], 'Country/_
↳ Region')
bar.options.scales.y_axis().ticks.toNumber()

ref = page.ui.texts.references.website(author="rsharankumar", name="Learn Data Science_
↳ in 100Days", site="github",
                                url="https://github.com/rsharankumar/Learn_Data_
↳ Science_in_100Days")

box = page.studio.containers.box()
```

(continues on next page)

(continued from previous page)

```
box.extend([title, items, page.ui.div([button_all, button_clear]), cols_keys, button,
↪bar, ref])
box.style.standard()

grp = page.data.js.record(std.var("covidData", global_scope=True)).filterGroup("aggData")
```

Add events on the button and button_all object

```
button.click([
    page.js.console.log(cols_keys.dom.content),
    bar.build(grp.match(cols_keys.dom.content).group().sumBy(['Confirmed', 'Deaths',
↪'Recovered'], ['Country/Region']))
])

countries = set()
for rec in data:
    countries.add(rec['Country/Region'])

button_all.click([
    cols_keys.dom.clear(), cols_keys.dom.add(list(countries), category='Country/Region',
↪no_duplicate=False)])
button_clear.click([cols_keys.dom.clear()])
```

This example is available [here](#)

More example of the templates on [Github](#)

5.3.3.3 TESLA Shares

This is another example of interactive report running without server.

All data are extracted on the Python side and the page will be built based on the data retrieved

This will create the page object and add a predefined configuration to the `page.body`:

```
page = Report()
page.theme = ThemeBlue.BlueGrey()
```

Add components to the page:

```
from_dt = page.ui.fields.date(value=None, html_code="from_date", label="From")
to_dt = page.ui.fields.date(value=None, html_code="to_date", label="To")
button = page.ui.buttons.colored("Update")
text = page.ui.calendars.pill("1Y", group="chart_time")
text_6m = page.ui.calendars.pill("6M", group="chart_time")
text_2m = page.ui.calendars.pill("2M", group="chart_time")
text_1m = page.ui.calendars.pill("1M", group="chart_time")
text_all = page.ui.calendars.pill("All", group="chart_time")
title = page.ui.title("Tesla Share Price")
buttons = page.ui.div([text, text_6m, text_2m, text_1m, text_all])

chart = page.ui.charts.chartJs.line([], y_columns=columns, x_axis='Date')
```

Set some specific ChartJs options:

```

chart.options.scales.y_axis().ticks.scale(1000)
chart.options.scales.y_axis().add_label("Stock Price (USD)")
chart.options.scales.x_axes().add_label("Date")
chart.options.tooltips.callbacks.labelCurrency("$")

```

Load the records on the JavaScript side and create a filter group to be able to apply transformations:

```
grp = page.data.js.record(records).filterGroup("aggData")
```

Add events:

```

text.click([from_dt.input.build(text.dom.content), button.dom.events.trigger("click")])
text_6m.click([from_dt.input.build(text_6m.dom.content), button.dom.events.trigger("click
↪")])
text_2m.click([from_dt.input.build(text_2m.dom.content), button.dom.events.trigger("click
↪")])
text_1m.click([from_dt.input.build(text_1m.dom.content), button.dom.events.trigger("click
↪")])
text_all.click([from_dt.input.build(records[0]["Date"]), chart.build(grp)])

tag = page.ui.rich.powered()
tag.style.css.margin_bottom = 5
tag.style.css.margin_top = 5

box = page.studio.containers.box()
box.extend([
  title, from_dt, to_dt, button, page.ui.layouts.hr(margins=5), buttons, chart,
  page.ui.layouts.hr().css({"margin-top": "20px"}), tag])
box.style.configs.doc(background="white")

button.click([
  text.dom.classList.select(False),
  text_all.dom.classList.select(False),
  text_2m.dom.classList.select(False),
  text_1m.dom.classList.select(False),
  text_6m.dom.classList.select(False),

```

Use the group to be able to filter on values defined in the components on the UI:

```

chart.build(grp.sup("Date", from_dt.dom.content).inf("Date", to_dt.dom.content).group().
↪sumBy(columns, ['Date']))
])

```

This example is available [here](#)

More example of the templates on [Github](#)

5.3.3.4 Pandas example

Simple dashboard using interactive components and Pandas.

This will create the page object:

```
page = pk.Page()
```

Add component and change the style of the body:

```
page.ui.title("Pandas tutorial #1")
page.ui.texts.references.github("https://github.com/PatrikHlobil/Pandas-Bokeh")
template = page.body.add_template(defined_style="doc")
template.style.css.background = page.theme.greys[0]
page.ui.titles.subtitle("Google versus Apple")
page.ui.charts.apex.line(df.to_dict(orient="records"), y_columns=["Google", "Apple"], x_
↪axis="Date")
```

Create a line Chart from a DataFrame:

```
df = pd.DataFrame({"Animal": ["Mouse", "Rabbit", "Dog", "Tiger", "Elefant", "Wale"],
                    "Weight [g]": [19, 3000, 40000, 200000, 6000000, 50000000]})

line = page.ui.charts.apex.line(df.to_dict(orient="records"), y_columns=["Weight [g]"],
↪x_axis="Animal")
line.options.xaxis.title.text = "Animals"
line.options.yaxis.title.text = "Weight"
```

Create a scatter Chart from a DataFrame:

```
scatter = page.ui.charts.apex.scatter(df.to_dict(orient="records"), y_columns=["Weight_
↪[g]"], x_axis="Animal")
scatter.options.xaxis.title.text = "Animals"
scatter.options.yaxis.title.text = "Weight"
scatter.options.yaxis.labels.formatters.scale(1000000)
page.ui.row([line, scatter])

data = {
    'fruits': ['Apples', 'Pears', 'Nectarines', 'Plums', 'Grapes', 'Strawberries'],
    '2015': [2, 1, 4, 3, 2, 4],
    '2016': [5, 3, 3, 2, 4, 6],
    '2017': [3, 2, 4, 4, 5, 3]
}
df = pd.DataFrame(data)
```

Create a bar Chart from a DataFrame:

```
page.ui.titles.subtitle("Fruit price per year")
bar = page.ui.charts.apex.bar(df.to_dict(orient="records"), y_columns=["2015", "2016",
↪"2017"], x_axis="fruits")
bar.colors(["blue", "green", "orange"])

page.ui.layouts.hr()
page.ui.titles.subtitle("Report powered by")
page.ui.rich.powered()
```

This example is available [here](#)

More example of the templates on [Github](#)

5.3.3.5 Advanced Dashboard

The below will illustrate an advanced web page without server.

This will create the page object:

```
page = pk.Page()
#page.theme.color_palette("brewer.PuBu8")
template = page.body.add_template(defined_style="doc")
template.style.css.background = page.theme.greys[0]
```

Add UI components to the page:

```
page.ui.title("Pandas tutorial #5")
page.ui.texts.references.website(url="https://towardsdatascience.com/data-visualization-
↳with-bokeh-in-python-part-ii-interactions-a4cf994e2512")
source_data = "https://raw.githubusercontent.com/WillKoehrsen/Bokeh-Python-Visualization/
↳master/interactive/data/complete_flights.csv"

df = pd.DataFrame(page.py.requests.csv(source_data, store_location=r"C:\tmps"))
df["dep_delay"] = pd.to_numeric(df["dep_delay"], downcast="float")
df["distance"] = pd.to_numeric(df["distance"], downcast="float")
agg_df = df.groupby(["name", "month", "distance"])["dep_delay"].sum()
agg_df = agg_df.reset_index()

max_value = float(df["distance"].max())
checks = page.ui.lists.checks(list(df["name"].unique()))
dist = page.ui.fields.static("[0, %s]" % max_value, label="Min, Max distance")
slider = page.ui.sliders.range([0, max_value], maximum=max_value)

chart = page.ui.charts.chartJs.bar(y_columns=["dep_delay"], x_axis="month")
table = page.ui.table(rows=["name", "month", "distance"], cols=["dep_delay"])
table.config.pagination = "local"
table.config.paginationSize = 10

pie = page.ui.charts.chartJs.pie(y_columns=["dep_delay"], x_axis="name")
pie_count = page.ui.charts.chartJs.pie(y_columns=["count"], x_axis="name")
```

Create data transformation groups for the data loaded from dataframes:

```
grp = page.data.js.record(agg_df.to_dict(orient="records")).filterGroup("aggData")
grp2 = page.data.js.record(agg_df.to_dict(orient="records")).filterGroup("aggData2")

update_button = page.ui.buttons.colored("update")

row = page.ui.row([checks, update_button, chart], position="top")
row.set_size_cols(4)

total_delay = page.ui.titles.subtitle("Total Delay")
count_delay = page.ui.titles.subtitle("Count Delay")
```

(continues on next page)

(continued from previous page)

```

page.ui.row([[total_delay, pie], [count_delay, pie_count]], position="top")

toggle = page.ui.buttons.toggle({'on': "More than 3 hours", 'off': 'All delays'}, label=
↪ "Delay Type",)

hamburger = page.ui.panels.hamburger([dist, slider, toggle], title="Details")
table.move()

```

Add an event to select the data to be used in the components:

```

toggle.input.click([
    page.js.if_(
        toggle.input.dom.content.isTrue(), [
            chart.build(grp.includes("name", checks.dom.content, empty_all=False).sup("dep_
↪ delay", 180).sup("distance", slider.dom.min_select).inf("distance", slider.dom.max_
↪ select).group().sumBy(["dep_delay"], ["month"], cast_vals=True)),
            table.build(grp.includes("name", checks.dom.content, empty_all=False).sup("dep_
↪ delay", 180).sup("distance", slider.dom.min_select).inf("distance", slider.dom.max_
↪ select)),
            pie.build(grp.includes("name", checks.dom.content, empty_all=False).sup("distance",
↪ slider.dom.min_select).inf("distance", slider.dom.max_select).group().sumBy(["dep_
↪ delay"], ["name"], cast_vals=True)),
            pie_count.build(grp.includes("name", checks.dom.content, empty_all=False).sup(
↪ "distance", slider.dom.min_select).inf("distance", slider.dom.max_select).group().
↪ countBy(["name"])))
        ]).else_([
            chart.build(grp2.includes("name", checks.dom.content, empty_all=False).sup(
↪ "distance", slider.dom.min_select).inf("distance", slider.dom.max_select).group().
↪ sumBy(["dep_delay"], ["month"], cast_vals=True)),
            table.build(grp2.includes("name", checks.dom.content, empty_all=False).sup(
↪ "distance", slider.dom.min_select).inf("distance", slider.dom.max_select)),
            pie.build(grp2.includes("name", checks.dom.content, empty_all=False).sup("distance
↪ ", slider.dom.min_select).inf("distance", slider.dom.max_select).group().sumBy(["dep_
↪ delay"], ["name"], cast_vals=True)),
            pie_count.build(grp2.includes("name", checks.dom.content, empty_all=False).sup(
↪ "distance", slider.dom.min_select).inf("distance", slider.dom.max_select).group().
↪ countBy(["name"])))
        ]),
    dist.input.build(slider.dom.content)
])

# Reuse the code generated by the toggle click for the below components.
update_button.click(toggle.input.event_fnc("click"))
slider.change(toggle.input.event_fnc("click"))

```

Add extra components to the page:

```

page.ui.layouts.hr()
page.ui.titles.subtitle("Report powered by")
page.ui.rich.powered()

```

This example is available [here](#)

More example of the templates on [Github](#)

5.3.4 Web server integration

5.3.4.1 Flask

5.3.4.1.1 Show case report

This is the demo page available from the below command line:

```
epyk.exe demo
epyk.exe transpile
```

Python script will have the below sections.

This will create the page object and add a predefined configuration to the page.body:

```
page = pk.Page()
page.headers.dev()
page.body.template.style.configs.doc(background="white")
```

Add the 3 components to the page (they will be automatically added to the body):

```
table = page.ui.table(mockes.popularity_2020)
table.options.paginationSize = 10

toggle = page.ui.toggle({"on": "Trend", "off": "Share"})
bar = page.ui.charts.bar(mockes.popularity_2020, y_columns=["Share"], x_axis="Language")
```

Create a JavaScript click event on the toggle component:

```
toggle.click([
    # Store the variable to myData on the JavaScript side
```

Then use the standard library in order to use core JavaScript features like: - Creating / updating a variable:

```
pk.std.var("myData", sorted(mockes.popularity_2020, key=lambda k: k['Language'])),
# Use the standard build and dom.content to respectively update and get the component.
↪ value
```

- Defining a if / else statement:

```
pk.expr.if_(toggle.input.dom.content.toStr(), [
    # Use the variable to update the chart
    bar.build(pk.std.var("myData"), options={"y_columns": ["Trend"]})
]).else_(
    bar.build(pk.std.var("myData"), options={"y_columns": ["Share"]})
)
])
```

The examples using a Flask app is available [here](#)

More example of the templates on [Github](#)

5.3.4.1.2 Simple Workflow

This is a more complex example illustrating how to use workflow components and tabs in a page.

This will create the page object and add a predefined configuration to the body:

```
page = pk.Page()
page.body.template.style.configs.doc(background="white")
```

Add a stepper component and put it in a container for the display:

```
stepper = page.ui.steppers.arrow([
    {"value": 'test 1', "status": 'pending', 'label': 'test'},
    {"value": 'test 2'},
    {"value": 'test 3', "status": 'waiting'}],
    options={"media": False, "line": False})
page.ui.div(stepper, align="center")
```

Attach to this component a JavaScript event. This will only display a JavaScript standard alert:

```
stepper[1].click([
    page.js.alert("This ")
])
```

Note: Components and event can be added anywhere in the page. The page.components variable will keep the order when it will transpile the script.

Add a tabs and button components:

```
tabs = page.ui.panels.tabs()
tabs.add_panel("Status 1", "Description for status 1", selected=True)
tabs.add_panel("Status 2", "Description for status 2")
tabs.add_panel("Status 3", "Description for status 3")

btn1 = page.ui.button("Previous", icon="fas fa-caret-left")
btn1.style.css.padding_h = 5
```

Add a click event to the button:

```
btn1.click([
    stepper.dom[pk.std.var("state")].arrow(),
    stepper.dom[pk.std.var("state")].waiting(),
    stepper.dom[pk.std.var("state")].text("Waiting", color="black"),
    stepper.dom[pk.std.var("state")].css({"border-bottom": "1px solid white", "padding-
↪bottom": "5px"}),
    pk.std.var("state", pk.std.maths.max(pk.std.parseInt(pk.std.var("state")) - 1, 0), ↪
↪global_scope=True),
    stepper.dom[pk.std.var("state")].css({"border-bottom": "1px solid green", "padding-
↪bottom": "5px"}),
    stepper.dom[pk.std.var("state")].circle(),
    stepper.dom[pk.std.var("state")].pending(),
    stepper.dom[pk.std.var("state")].text("Pending", color="black"),
    tabs.dom[pk.std.var("state")].select(),
```

(continues on next page)

(continued from previous page)

```

])

btn2 = page.ui.button("Next", icon="fas fa-caret-right")
btn2.style.css.padding_h = 5
btn2.click([

```

Use the DOM and JS properties in the click events since this will be run by the browser once the code will be transpiled to JavaScript. Python will only perform some sanity checks on the pre defined features:

```

    stepper.dom[pk.std.var("state")].success(),
    stepper.dom[pk.std.var("state")].arrow(),
    stepper.dom[pk.std.var("state")].text("Completed", color="black"),
    stepper.dom[pk.std.var("state")].css({"border-bottom": "1px solid white", "padding-
↪bottom": "5px"}),
    pk.std.var("state", pk.std.maths.min(pk.std.parseInt(pk.std.var("state")) + 1, 2), ↵
↪global_scope=True),
    stepper.dom[pk.std.var("state")].css({"border-bottom": "1px solid green", "padding-
↪bottom": "5px"}),
    stepper.dom[pk.std.var("state")].circle(),
    stepper.dom[pk.std.var("state")].pending(),
    stepper.dom[pk.std.var("state")].text("Pending", color="black"),
    tabs.dom[pk.std.var("state")].select(),
    page.js.console.log(tabs.dom[pk.std.var("state")].innerText())
])

```

Create a variable when the browser will load the page:

```

page.body.onReady([
    pk.std.var("state", 0, global_scope=True)
])

```

This example is available [here](#)

More example of the templates on [Github](#)

5.3.4.2 Fast API

5.3.4.2.1 Log Viewer

This will create the page object and add a predefined configuration to the `page.body`:

```

page = pk.Page()
template = page.body.add_template(defined_style="margins")
template.style.css.background = "white"

```

Add a date object:

```

dt = page.ui.texts.date("Y-0", html_code='date_from', width=(100, "%"))

```

Create a navbar with components:

```
page.ui.navigation.shortcut([
    page.ui.layouts.hr(),
    page.ui.titles.title("Dates"),
    page.ui.titles.bold("From:", align="left"),
    dt,
    page.ui.titles.bold("To:", align="left"),
    page.ui.texts.date(html_code='date_to', width=(100, "%"), options={"date_from_js": "COB
→"}),
    page.ui.layouts.hr(),
    page.ui.titles.title("Actions"),
```

Use `html_Code` on the components in order to get this alias when they are passed to services:

```
page.ui.input("GS", html_code="input"),
page.ui.buttons.refresh("Load", html_code="button"),
page.ui.icons.date(),
page.ui.div([
    page.ui.icons.awesome("far fa-file-pdf", width=15),
    page.ui.icons.awesome("fas fa-at", width=15),
]).css({"bottom": '10px', 'position': 'absolute', 'display': 'block'})
], size=(100, 'px'), options={"position": 'left'})

page.body.style.css.margin_left = 10
content = ""
title = page.ui.title(content, options={"markdown": True})
table = page.ui.tables.tabulators.figures(
    rows=["Date", "Name"], cols=["Low", "High", 'Open', 'Close', 'Volume', 'Adj Close'])

footer = page.ui.navigation.footer('@Data from pandas_datareader using yahoo as source.')
```

Create a click event on the page by finding the component from the `page.components` dictionary based on its `html_code`:

```
page.components['button'].click([
    page.components["button"].icon.dom.spin(True),
```

Define an AJAX post to an underlying service:

```
page.js.post("/viewer", {"button": 'Data 1'}, components=[page.components["input"],
```

Update the components:

```
page.components['date_from'], page.components['date_to'])).onSuccess([
    title.build(pk.events.data["title"]),
    table.build(pk.events.data["table"]),
    page.components["button"].icon.dom.spin(False)
]),
])
page.components["input"].enter([page.components['button'].dom.events.trigger("click")])
footer.style.css.padding_left = 110
```

This example is available [here](#)

More example of the templates on [Github](#)

5.3.4.2.2 Webscraping

This will focus on the UI section of the template used to get data from an external website.

This will create the page object and add a predefined configuration to the `page.body`:

```
page = pk.Page()
page.body.template.style.configs.margins()
```

Add a QR Code component with the link of the example:

```
qrcode = page.ui.qrcode("https://github.com/epykure/epyk-templates/blob/master/tutos/
↳onepy/fastapi_webscraping.py")
qrcode.style.css.fixed(bottom=60, right=70)
qrcode.style.css.cursor = "pointer"
qrcode.style.css.z_index = 300
```

Add date components and change the CSS properties:

```
dt_from = page.ui.date(html_code='from', width=(120, "px"))
dt_from.input.style.css.margin_bottom = 10
dt_to = page.ui.date("2021-05-15", html_code='to', width=(120, "px"))
dt_to.input.style.css.margin_bottom = 10
```

Add a button component and change the CSS properties:

```
prices = page.ui.button("Get Prices", icon="fab fa-python")
prices.style.css.padding_left = 10
prices.style.css.padding_right = 10
prices.style.css.color = page.theme.colors[-1]
```

Create a navbar and attach the components to it:

```
bar = page.ui.navbar()
bar.style.css.background = page.theme.colors[-1]
bar.style.css.color = "white"
bar.add(dt_from)
bar.add(dt_to)
bar.add(prices)

page.ui.title("Eurostar average prices")
line = page.ui.charts.chartJs.line(y_columns=['standard', 'premier', 'business'], x_axis=
↳'full_date')
bar = page.ui.charts.chartJs.bar(y_columns=['average'], x_axis='category')
row = page.ui.row([line, bar])
row.set_size_cols(8)
```

Create a event when button is clicked:

```
prices.click([
    prices.icon.dom.spin(True),
```

Link this example to an entry point on the server:

```
page.js.post("/web_scrapping", components=[dt_from, dt_to]).onSuccess([
    line.build(pk.events.data["prices"]),
    bar.build(pk.events.data["average"]),
    prices.icon.dom.spin(False)
])
])
```

This example is available [here](#)

More example of the templates on [Github](#)

5.3.4.2.3 Advanced Documentation viewer

This example is a bit more sophisticated as it will use a FAST Api server and a database in order to store version of the documentation.

Database structure will be defined when the server will start.

First create the page object:

```
page = pk.Page()
```

Create a bespoke CSS inline object:

```
css = pk.CssInline()
css.margin_bottom = 5
css.margin_left = 5
css.important(["margin_bottom", "margin_left"])
```

Create an autocomplete input components and change some options:

```
autocomp = page.ui.inputs.autocomplete(placeholder="script name", html_code="name",
↪options={"borders": "bottom"})
autocomp.options.select = True
version = page.ui.select(width=(100, 'px'), html_code="selected_version")
```

Add the CSS object to this components with a specific name:

```
version.attr["class"].add(css.to_class("cssTestClass"))
version.options.noneSelectedText = "None"
```

Add components to the page:

```
button = page.ui.buttons.colored("Load")
button.style.css.margin_left = 10
page.ui.navbar(components=[autocomp, version, button])
script = page.ui.text("script name", html_code="script")
script.options.editable = True
script.style.css.bold()
pkg_version = page.ui.text("Version")
pkg_number = page.ui.text("0.0.0", html_code="version")
```

Change css styles and options:

```
pkg_number.options.editable = True
pkg_number.style.css.margin_left = 10
v = page.ui.div([pkg_version, pkg_number], width="auto")
v.style.css.float = "right"
v.style.css.display = "inline-block"
```

Add icon components:

```
i1 = page.ui.icon("fas fa-edit")
i2 = page.ui.icon("fas fa-lock")
i3 = page.ui.icon("fas fa-save")
actions = page.ui.div([i1, i2, i3], width=(20, 'px'))
```

Add CSS style properties using `actions.style.css`:

```
actions.style.css.position = "absolute"
actions.style.css.top = 60
actions.style.css.right = 0
actions.style.css.padding_left = "3px"
actions.style.css.border_radius = "5px 0 0 5px"
actions.style.css.background = page.theme.greys[3]

header = page.ui.div([script, v])
header.style.css.background = page.theme.greys[2]
header.style.css.display = "block"
header.style.css.padding_h = 15

title = page.ui.title("Documentation Viewer", html_code="title")
title.options.editable = True
content = page.ui.rich.markdown(__doc__, html_code="content")
content.options.editable = True

banner = page.ui.text("Editable", width=(75, 'px'))
banner.style.css.background = page.theme.success[1]
banner.style.css.border_radius = "0 0 20px 0"
banner.style.css.padding_h = 10
banner.style.css.font_factor(-2)
banner.style.css.color = page.theme.greys[-1]
banner.style.css.position = "absolute"
banner.style.css.bold()
banner.style.css.top = 0
banner.style.css.left = 0

container = page.ui.div([banner, header, title, content, actions])
container.style.configs.doc(background="white")
container.style.css.position = "relative"
container.style.css.padding_top = 30

updt = page.ui.rich.update(align="right", html_code="last_update")
updt.style.css.italic()
updt.style.css.font_factor(-2)
container.add(updt)
```

Add events to the components:

```
i3.click([
```

Call an underlying service in the FAST API server:

```
page.js.post("/save", components=[banner, script, pkg_number, title, content, updt]).
  .onSuccess([
    page.js.msg.status(),
    updt.refresh()
  ])]
```

Add click event on to the first icon:

```
i1.click([
```

Change the dom properties using the common JavaScript features `dom.setAttribute`:

```
script.dom.setAttribute("contenteditable", True).r,
pkg_number.dom.setAttribute("contenteditable", True).r,
title.dom.setAttribute("contenteditable", True).r,
content.dom.setAttribute("contenteditable", True).r,
```

Display a temporary message in the page:

```
page.js.msg.text("Components editable"),
```

Update the banner component:

```
banner.build("Editable"),
banner.dom.css({"background": page.theme.success[1], "color": page.theme.greys[-1]})
])
```

In the same way click events are added on the other components:

```
i2.click([
  script.dom.setAttribute("contenteditable", False).r,
  pkg_number.dom.setAttribute("contenteditable", False).r,
  title.dom.setAttribute("contenteditable", False).r,
  content.dom.setAttribute("contenteditable", False).r,
  page.js.msg.text("Components locked"),
  banner.build("Locked"),
  banner.dom.css({"background": page.theme.colors[-1], "color": page.theme.greys[0]})
])

autocomp.enter([
  page.js.post("/versions", components=[autocomp]).onSuccess([
    version.build(pk.events.data["versions"]),
    version.js.val(pk.events.data["selected"]),
    version.js.refresh(),
    page.js.msg.status()
  ])
])

button.click([
  page.js.post("/details", components=[autocomp, version]).onSuccess([
```

(continues on next page)

(continued from previous page)

```

    title.build(pk.events.data["title"]),
    content.build(pk.events.data["content"]),
    script.build(pk.events.data["script"]),
    pkg_number.build(pk.events.data["number"]),
    updt.build(pk.events.data["last_date"]),
  ])
])

```

Add an `body.onReady` to load the autocompletion when the page is ready:

```

page.body.onReady([
  page.js.post("/templates").onSuccess([
    autocomp.js.source(pk.events.data["values"])
  ])
])

```

This example is available [here](#)

More example of the templates on [Github](#)

5.4 Epyk and full stack development

5.4.1 Documentation

Epyk documentation will point you to the appropriate web concept.

The docstrings will link to either the online precised documentation or to the wrapped package. Epyk's target is to make the link between Python and the web in a transparent manner.

```

def click(self, js_funcs, profile=False, source_event=None, on_ready=False):
    """
    Description:
    -----
    When a user clicks on a sparkline, a sparklineClick event is generated.
    The event object contains a property called "sparklines" that holds an array of the sparkline objects under
    the mouse at the time of the click.
    For non-composite sparklines, this array will have just one entry.

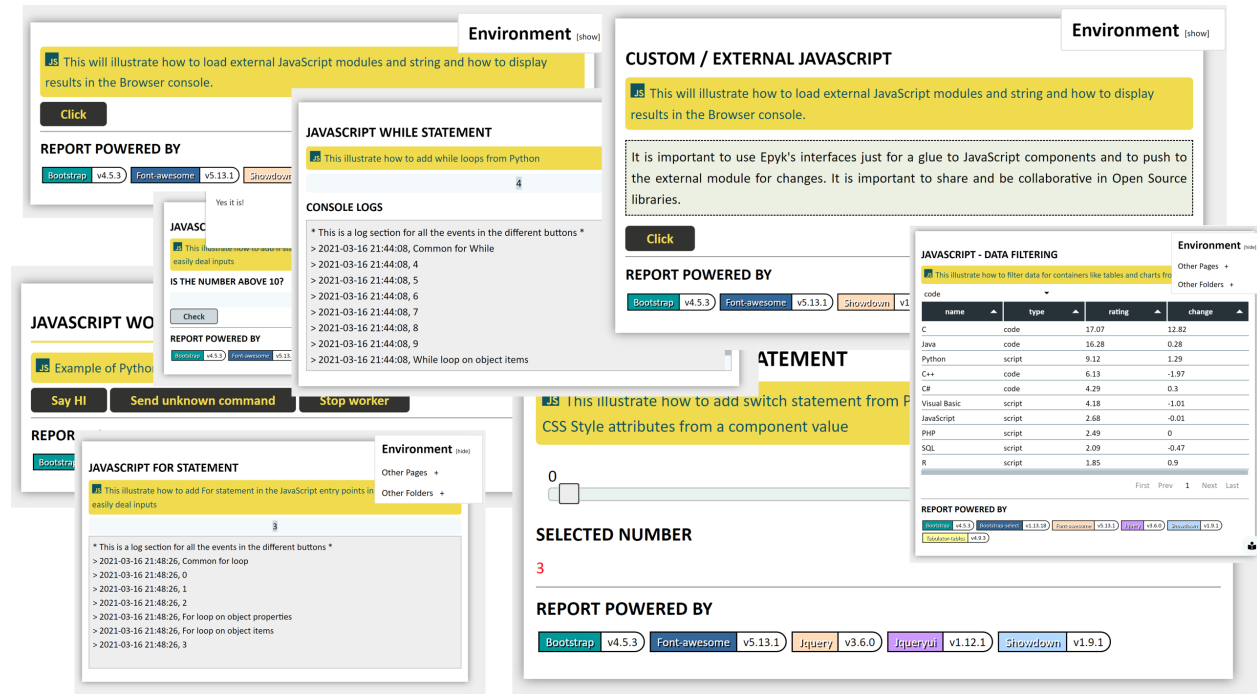
    Usage::

    Related Pages:
    |
    | https://omnipotent.net/jquery.sparkline/#interactive
    |
    Attributes:
    -----
    :param js_funcs: List | String. Required. Javascript functions.
    :param profile: Boolean | Dictionary. Required. A flag to set the component performance storage.
    :param source_event: String. Required. The source target for the event.
    :param on_ready: Boolean. Required. Specify if the event needs to be trigger when the page is loaded.
    """

```

Also components and entry points are structure to make sure you learn the various concepts of a web page. Thus the keys words `dom`, `console.log`, `InlineCss`, `css` properties, `page.bodt`, `component.onReady()` very popular in web development will be familiar to you as well.

Those concept are illustrated in the local section of the [templates Github repo](#)



5.4.2 Profiling

Checking performances is important on the backend but also on the front end side. Epyk will rely on the native features to profile the python and on the JavaScript side it will provide the profile keyword to write log messages to the console (F12 in the browser):

```
def click(self, js_funcs, profile=None, source_event=None, on_ready=False):
```

5.4.3 Styles & Configuration

5.4.3.1 CSS Style

5.4.4 Properties

It is possible to use CSS inline properties or bespoke CSS classes in order to change the display of components. Quite a few examples are available but the easiest is to use `style.css` properties:

```
button = page.ui.buttons.colored("Test")
button.style.css.color = "yellow" # Change the text color
button.style.css.position = "fixed"
button.style.css.bottom = 10 # Default will use px
button.style.css.right = 10 # Default will use px
```

More examples are available in the section [Library extensions](#)

5.4.5 Virtual classes

There are some shortcuts available to perform common and standard style changes. For example to change a style by putting the mouse hover a component can be done:

```
p = page.ui.texts.paragraph("This is a paragraph", helper="Paragraph helper")
p.style.hover({"color": "red"})
```

Or a bit more complex by doing, it is possible to add a full CSS class structure to a specific component:

```
p = page.ui.texts.paragraph("This is a paragraph", helper="Paragraph helper")
p.style.add_custom_class({"_attrs": {"color": "green"}, "_hover": {"color": "blue"}}, to_
↪ component=True)
```

5.4.5.1 CSS effects

It is also possible to use predefined CSS effects and animates:

```
title2 = page.ui.titles.title("Epyk in few words", align="center")
title2.style.effects.shiny_text("green")

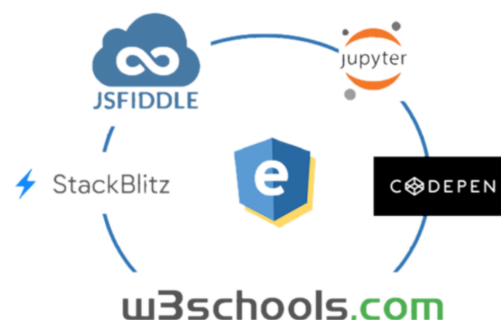
title1 = page.ui.titles.title("A vast selection of Charts", align="center")
title1.style.effects.down(start=-10)
```

5.4.6 Interactivity & events

5.4.6.1 Component events

5.4.7 Community

Epyk is dedicated to be a bridge to JavaScript. Thus it will provide ways to share pieces of code in both worlds. It is possible to structure your Python components and share them using Pyk files or to transpile a dashboard to then be shared in the popular online editors.



The purpose of this framework is to benefit from both worlds but also to contribute by providing feedbacks to the external libraries

5.4.7.1 To Pyk

5.4.7.2 To Jupyter or JupyterLab

Epyk can be imported to any Notebook. More details will come in this section.

5.4.7.3 To CodePen

The below is a code generated using the codepen outs to illustrate a problem:

```
import epyk as pk

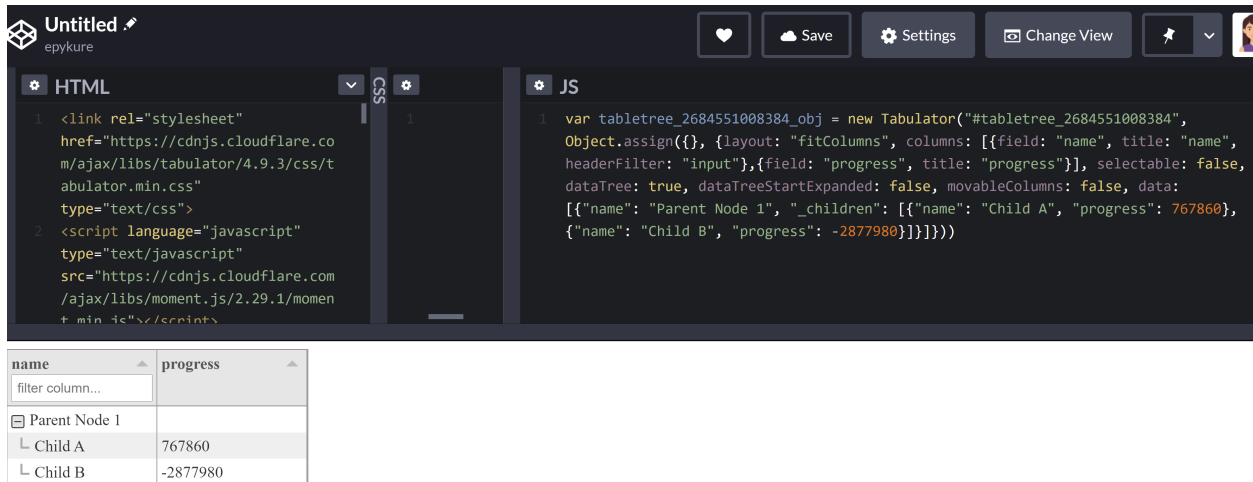
page = pk.Page()

records = {"data": [
    {"name": 'Test', "_children": [
        {"name": 'AAAA', "progress": 767860},
        {"name": 'BBB', "progress": -2877980},
    ]}
], "columns": [
    {"title": "Task Name", "field": "name"},
    {"title": "Progress", "field": "progress"},
]}

data = [
    {"name": 'Test', "_children": [
        {"name": 'AAAA', "progress": 767860},
        {"name": 'BBB', "progress": -2877980},
    ]}
]

table = page.ui.tables.tabulators.hierarchy(data, cols=["name"], rows=["progress"],
↪width=(300, 'px'))
table.get_column("name").headerFilter = "input"
page.outs.codepen()
```

[Codepend](#)

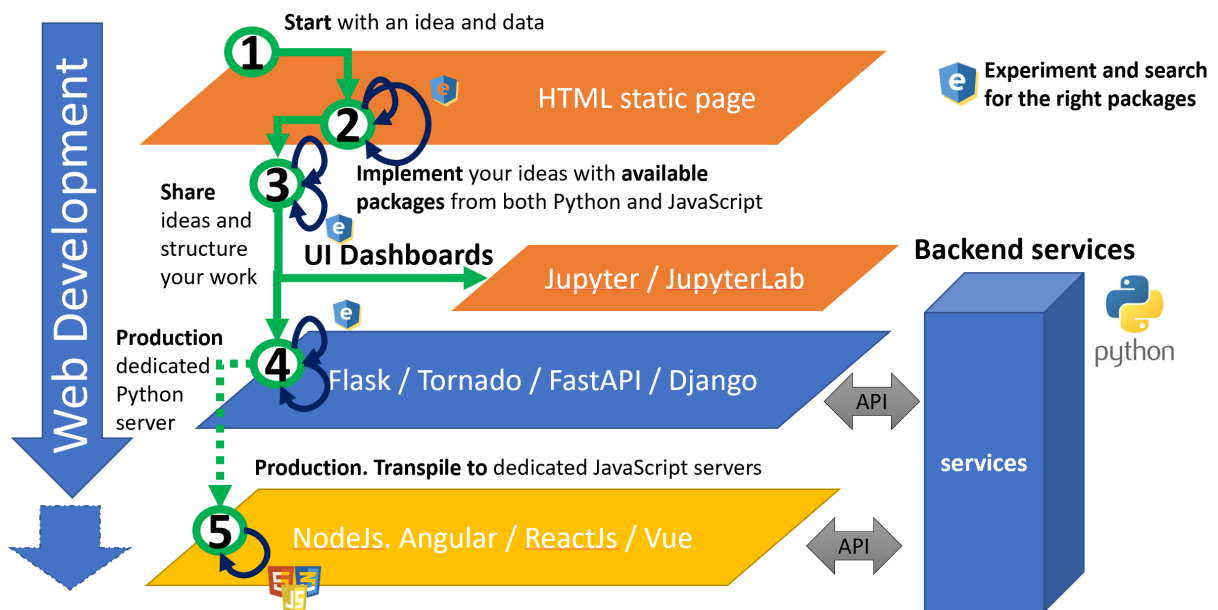


5.4.7.4 To W3Try

5.5 Design & Architecture

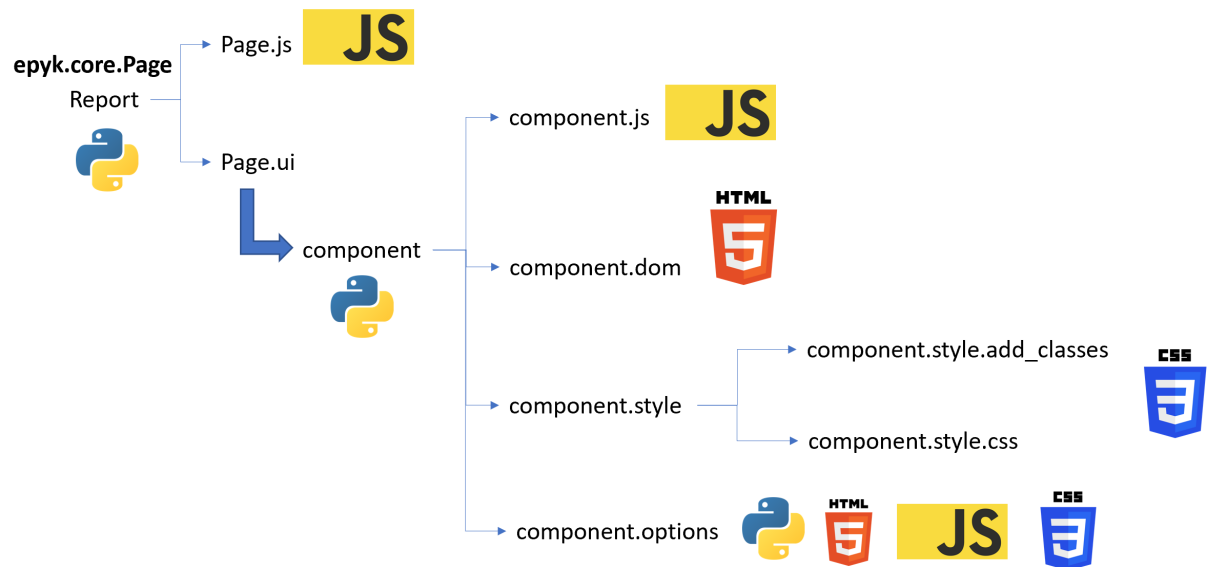
5.5.1 Architecture

Improve time to Market for your projects with Epyk



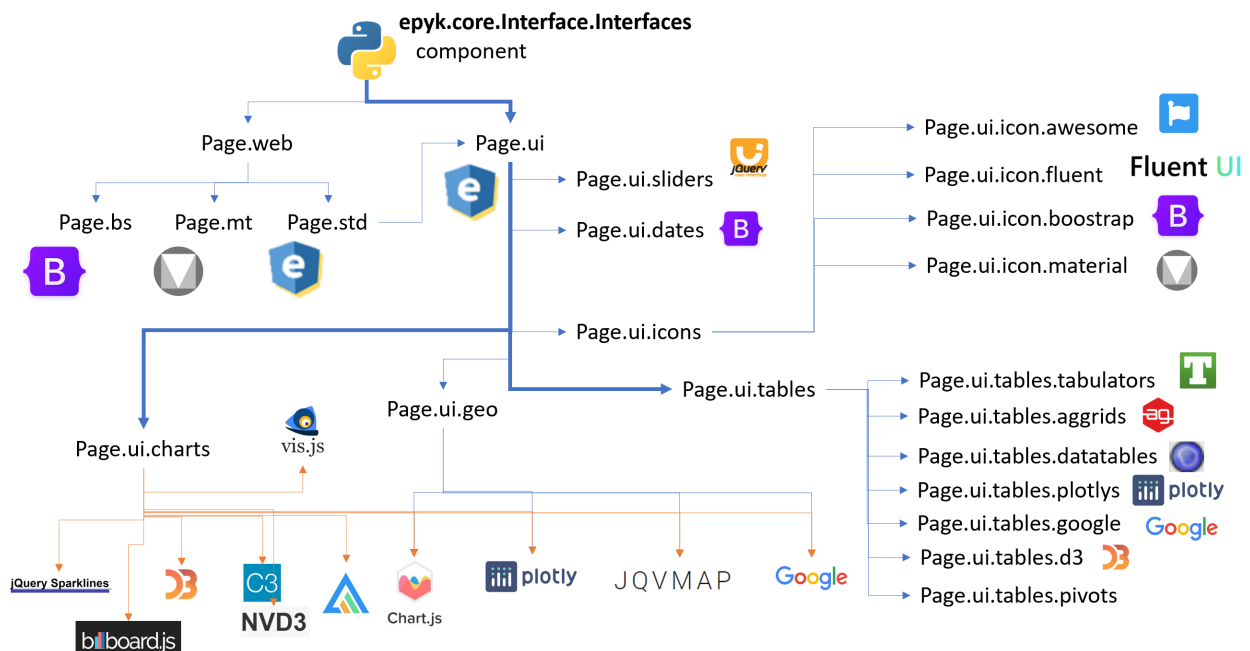
5.5.1.1 Page

From the page object it is possible to get all the entry points



The page object will also have dedicated properties to simplify the link with Python modules.

5.5.1.2 Components



5.5.1.3 HTML Component

The below sections details the work required to add a new HTML component to the framework. Frappe-Charts will be used as reference here:

First the creation of teh key modules for the page object:

- An interface in the interface component: CompChartsFrappe.py
- An HTML component page: GraphFrappe.py

An entry point in the interface

1

```
class CompChartFrappe:
    def __init__(self, ui):
        self.page = ui.page

    def plot(self, record=None, y=None, x=None, kind="line", profile=None,
            options=None, html_code=None):...

    def line(self, record=None, y_columns=None, x_axis=None, profile=None,
            options=None, html_code=None):...

    def bar(self, record=None, y_columns=None, x_axis=None, profile=None,
            options=None, html_code=None):...

    def percentage(self, record=None, y_columns=None, x_axis=None, profile=None,
            options=None, html_code=None):...

    def donut(self, record=None, y_columns=None, x_axis=None, profile=None,
            options=None, html_code=None):...

    def pie(self, record=None, y_columns=None, x_axis=None, profile=None,
            options=None, html_code=None):...
```

A function which link the interface to the HTML

2

```
def line(self, record=None, y_columns=None, x_axis=None, profile=None,
        options=None, html_code=None):
    line_chart = graph.GraphFrappe.Frappe(self.page, width, height)
    line_chart.options.height = height[0]
    line_chart.colors(self.page.theme.charts)
    data = self.page.data.c3.y(record or [], y_columns, x_axis)
    line_chart.labels(data["labels"])
    for i, dataset in enumerate(data["datasets"]):
        line_chart.add_dataset(dataset, data["series"][i])
    return line_chart
```

A class inheriting from HTML

3

```
class Frappe(Html.Html):
    requirements = ('frappe-charts', )
    name = 'Frappe Mixed'
    _chart_type = 'axis=mixed'
    option_cls = OptChartFrappe.FrappeLine

    def __init__(self, report, width, height, html_code, options=None):
        super().__init__(report, width, height, html_code, options)

    @property
    def shared(self):...

    @property
    def js(self):...

    @property
    def options(self):...

    def colors(self, hex_values):...

    def labels(self, values):...

    def add_dataset(self, data, label, colors=None, opacity=None):...

    def build(self, data=None, options=None, profile=None, components=None):...

    def __str__(self):...
```

An HTML footprint

4

```
def __str__(self):
    self.page.properties.js.add_builders(self.build())
    return '<div %s></div>' % self.get_attrs(pyClassNames=self.style.get_classes())
```

This will make the component available from the *page.ui* property

After to add interactivity and configuration for this components two extra modules are required:

- A dedicated modules for the component (Js, Py and Css) options: OptChartFrappe.py
- A module to wrap the external library API: JsFrappe.py

A class inheriting from HTML

```
class Frappe(Html.Html):
    requirements = ('frappe-charts',)
    name = 'Frappe Mixed'
    _chart_type = 'axis-mixed'
    _option_cls = OptChartFrappe.FrappeLine

    def __init__(self, report, width, height, html_code, options):
        pass

    @property
    def shared(self):
        pass

    @property
    def js(self):
        pass

    @property
    def options(self):
        pass

    def colors(self, hex_values):
        pass

    def labels(self, values):
        pass

    def add_dataset(self, data, label, colors=None, opacity=None):
        pass

    def build(self, data=None, options=None, profile=None, components=None):
        pass

    def __str__(self):
        pass
```

A JavaScript and Python entry for configuration

```
@property
def options(self):
    """
    Description:
    ~~~~~
    Chart specific options.

    :rtype: OptChartFrappe.FrappeLine
    """
    return super().options
```



A JavaScript entry point for the API package

```
@property
def js(self):
    """
    Description:
    ~~~~~
    JavaScript entry point for the API package.

    :rtype: str
    """
    if self._js is None:
        self._js = JsFrappe.FrappeCharts(selector="window['%s']" % self.chart_type)
    return self._js
```



+

A standard entry for all Charts in the library

```
@property
def shared(self):
    """
    Description:
    ~~~~~
    Standard entry for all Charts in the library.

    :rtype: OptChartFrappe.OptionsChartSharedFrappe
    """
    return OptChartFrappe.OptionsChartSharedFrappe(self)
```



Those modules will have link to the online documentation of the different functions in order to provide further details and examples.

Also most of the functions are exactly the same than the ones defined in the API to simplify the communication between developers.

Then if some external module is required to build the component those are defined from the npm alias in the requirements class variable.

This will point to two references in the *Imports.py* module one for the JavaScript and the other one for the CSS. *Imports.py* is the module used to add the external files to the page. It will manage the dependencies between configurations and the version of the packages.

```
# Frappe-Charts module
'frappe-charts': {
    'website': 'https://frappe.io/charts/docs',
    'version': '1.5.1',
    'register': {'alias': 'Frappe', 'module': 'frappe-charts.min.life'},
    'modules': [
        {'script': 'frappe-charts.min.life.js', 'path': 'frappe-charts@%(version)s/dist/',
         'cdnis': "https://cdn.jsdelivr.net/npm"}
    ],
}
```

With requirements which point to an entry in Imports.py

```
class Frappe(Html.Html):
    requirements = ('frappe-charts',)
    name = 'Frappe Mixed'
    _chart_type = 'axis-mixed'
    _option_cls = OptChartFrappe.FrappeLine
```



```
'frappe-charts': {
    'modules': [
        {'script': 'frappe-charts.min.css', 'path': 'frappe-charts@%(version)s/dist/',
         'cdnis': "https://cdn.jsdelivr.net/npm"}
    ],
}
```


5.5.2 Common API to easy the migration

The spirit of the framework is to benefit from the rich ecosystem but to not be attached to one library or framework. Thus there is a common API for components to easy this transition.

For example for charts:

```
import epyk as pk

url_data = "https://raw.githubusercontent.com/vega/datalib/master/test/data/stocks.csv"

page = pk.Page()

data = page.py.requests.csv(url_data)
formatted_data = []
agg_data = {}
for rec in data:
    agg_data.setdefault(rec["date"], {})[rec["symbol"]] = float(rec["price"])
    agg_data[rec["date"]]["date"] = rec["date"]
chart = page.ui.charts.chartJs.bar(list(agg_data.values()), y_columns=["MSFT", "AMZN",
↪ "IBM"], x_axis="date")
page.outs.jupyter()
```

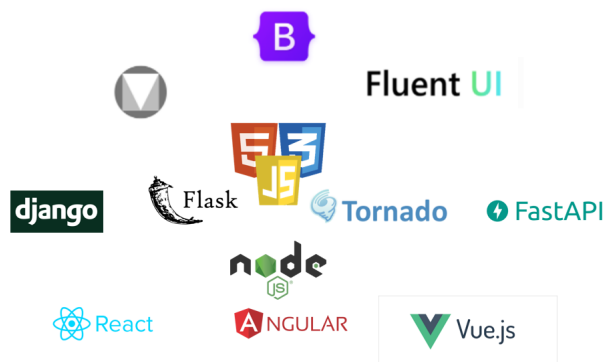
Note: Specific `.options` or `.js` functions will not be necessarily compatible between libraries. We are introducing the concept of `.shared` properties to solve this issue.

5.5.3 Collaborative platform

The main target of this library is to allow a smooth move from Python to JavaScript but also for the JavaScript developers to bring their knowledge in order to provide better integration and performances to the transpiled results.

By using Epyk and improving this layer, Python developers will be able to test JavaScript code and provide quick feedbacks. In the same way they will be able to produce web results which could be easily integrated to the IT stack.

To achieve this Epyk is implemented to be compatible with the current UI libraries and the most popular framework.

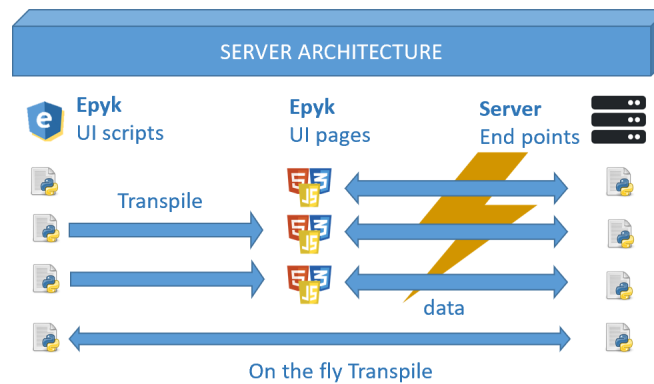


Integration to JavaScript web framework is still in progress on this.

5.5.4 Full stack development

Epyk ensure a full stack development and also it will guarantee that your front end will benefit from the community updates. Indeed this library is used for producing a configurable UI build from pre defined JavaScript and CSS definition, if the structure or the code as changed on the component side a simple update of the library and a new transpiling of the script will render better web pages.

It will also help you upgrading your packages in a safe manner. Indeed by using the *shared* property, this will guarantee that despite the changes performed in the packages this will remain unchanged.



5.5.5 Data Driven Design Architecture

This is the main concept which lead the implementation of Epyk. We realised that most of the users were familiar with Python hence were able to write algorithms but they were always blocked to promote and share their work easily.

Quite a few great proprietary platform (Tableau, Power BI, Dataiku...) offer this service but we wanted to have something free and flexible enough to go to the last stage which is the integration to existing web Ecosystems.

5.5.6 Other related topics

5.5.6.1 A design for the web

Epyk is fully integrated to the web ecosystem and its main target is to simplify the step from Python.

Python is becoming a very popular language nowadays and it is sometimes quite complicated to move to other technologies.

Epyk should be the right perfect library to deal with concept easily but also thanks to the documentation, structure and functions names to learn and get the good practices.

5.5.6.1.1 Be an actor

By using Epyk, it will be possible to create rich and interactive web content from the most popular libraries but it will also allow you to use those UI tools to communicate, provide feedback and ask for new features. Thus you will be able to point a bug or mention a nice feature missing a library.

Ask a question in Stack overflow with Codepen examples

Epyk structure will allow you to isolate a component or a feature, generate the web content to the desired formant and then check the result (and potentially adapt).

For example the below Stack overflow request will be there to ask for a change or details on a recently introduced Charting library:

1. Isolate the component and illustrate the problem in a report:

```
import epyk as pk

page = pk.Page()
languages = [
    {"name": 'C', 'type': 'code', 'rating': 17.07, 'change': 12.82},
    {"name": 'Java', 'type': 'code', 'rating': 16.28, 'change': 0.28},
    {"name": 'Python', 'type': 'script', 'rating': 9.12, 'change': 1.29},
    {"name": 'C++', 'type': 'code', 'rating': 6.13, 'change': -1.97},
    {"name": 'C#', 'type': 'code', 'rating': 4.29, 'change': 0.3},
    {"name": 'Visual Basic', 'type': 'script', 'rating': 4.18, 'change': -1.01},
    {"name": 'JavaScript', 'type': 'script', 'rating': 2.68, 'change': -0.01},
    {"name": 'PHP', 'type': 'script', 'rating': 2.49, 'change': 0},
    {"name": 'SQL', 'type': 'script', 'rating': 2.09, 'change': -0.47},
    {"name": 'R', 'type': 'script', 'rating': 1.85, 'change': 0.90},
]
b = page.ui.charts.roughviz.plot(languages, y=["rating", 'change'], x='name',
    width=300)
```

2. Use a specific outputs to generate Codepen files:

```
page.outs.codepen()
```

This will create to the root directory an output files with 3 files corresponding to the different Codepen boxes. Copy paste the content, save your work and create a question in Stack Overflow.

The screenshot shows a web browser with a Codepen example and a Stack Overflow question. The Codepen example displays a line chart with 'rating' and 'change' data for various programming languages. The Stack Overflow question asks about passing structured data to the line chart.

Codepen Example:

```
roughviz_2885952098800_obj = new roughViz.Line({width: 300, element: "#roughviz_2885952098800", height: 330, xlabel: "name", data: [{"name": "C", "rating": 17.07, "change": 12.82}, {"name": "Java", "rating": 16.28, "change": 0.28}, {"name": "Python", "rating": 9.12, "change": 1.29}, {"name": "C++", "rating": 6.13, "change": -1.97}, {"name": "C#", "rating": 4.29, "change": 0.3}, {"name": "Visual Basic", "rating": 4.18, "change": -1.01}, {"name": "JavaScript", "rating": 2.68, "change": -0.01}, {"name": "PHP", "rating": 2.49, "change": 0}, {"name": "SQL", "rating": 2.09, "change": -0.47}, {"name": "R", "rating": 1.85, "change": 0.90}]});
```

Stack Overflow Question:

Input data for Line Chart in roughViz

Asked yesterday Active yesterday Viewed 8 times

Is it possible to pass structured data for the line chart in the same way it is defined for a scatter chart (and by the way is it possible to use y1, y2 for a scatter)?

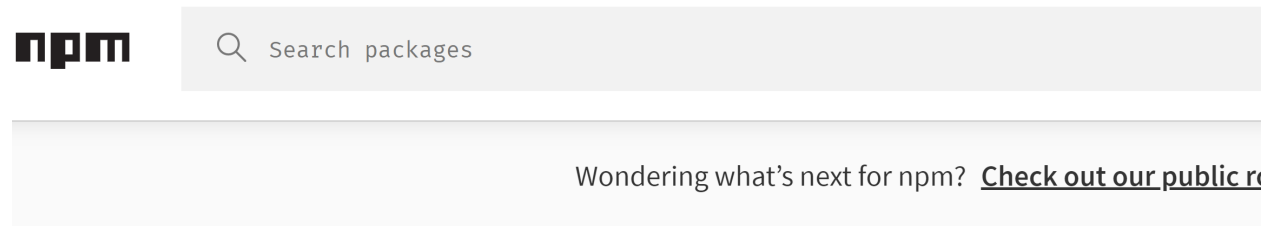
I find this library really interesting and I would like to add it to our python UI libraries but I would need to get this fixed first.

Here is an example of issue I have in codepen: <https://codepen.io/epykure/pen/WNpRWp>

roughviz_2885952098800_obj = new roughViz.Line({width: 300, element: "#roughviz_2885952098800", height: 330, xlabel: "name", data: [{"name": "C", "rating": 17.07, "change": 12.82}, {"name": "Java", "rating": 16.28, "change": 0.28}, {"name": "Python", "rating": 9.12, "change": 1.29}, {"name": "C++", "rating": 6.13, "change": -1.97}, {"name": "C#", "rating": 4.29, "change": 0.3}, {"name": "Visual Basic", "rating": 4.18, "change": -1.01}, {"name": "JavaScript", "rating": 2.68, "change": -0.01}, {"name": "PHP", "rating": 2.49, "change": 0}, {"name": "SQL", "rating": 2.09, "change": -0.47}, {"name": "R", "rating": 1.85, "change": 0.90}]});


2. Add extra feature to enrich an existing library

In the same way it is possible to create and propose change to existing libraries. For example for Tabulator we create a module dedicated to enrich the number of existing Formatter, Aggregator and Mutator to ease the configuration for anybody using this library.



tabulator-extensions

0.0.29 • Public • Published 3 months ago

 [Readme](#)

 [Explore](#) BETA

 [1 Dependency](#)

Tabulator extensions

A set of components for Tabulator in order to customise the display of columns and cells.

Those components are maintained by Epyk Team and they are designed to work with the Python Epyk-ui. The goal of this project being to extend the rich Python ecosystem with selected JavaScript features and render modern and interactive pages.

All Codepen source codes used to create the stack overflow requests are available [here](#)

5.5.6.2 Supported Libraries and Frameworks

A toolbox to multiple external libraries. Epyk will interface with the most popular JavaScript and Css libraries from the vast number of components.

This will be fully transparent all components will include to the page their external resources.

5.5.6.3 127 Libraries

- accounting, version 0.4.1
- qrcodejs, version 1.0.0
- underscore, version 1.12.0
- promise-polyfill, version 8.2.0
- url-search-params, version 1.1.0
- babel-polyfill, version 7.4.4

- bootstrap, version 4.6.0
- moment, version 2.29.1
- ag-grid-community, version 25.0.1
- tabulator-tables, version 4.9.3
- tabulator-inputs, version
- tabulator-drop, version
- tabulator-mutators-inputs, version
- editors-inputs, version
- editors-dates, version
- editors-selects, version
- tabulator-icons, version
- tabulator-numbers, version
- font-awesome, version 5.13.1
- **`datatables, version 1.10.19 <>`_**
- datatables-buttons, version 1.6.1
- datatables-select, version 1.3.1
- datatables-scroller, version 2.0.1
- datatables-searchPanels, version 1.0.1
- datatables-responsive, version 2.2.3
- datatables-keytable, version 2.5.1
- datatables-autoFill, version 2.1.0
- datatables-rows-group, version 1.0.0
- datatables-row-group, version 1.1.1
- datatables-fixed-columns, version 3.2.2
- datatables-fixed-header, version 3.1.3
- datatables-export, version 1.5.2
- datatables-col-order, version 1.5.1
- jszip, version 3.5.0
- json-formatter-js, version 2.3.4
- pivottable, version 2.23.0
- requirejs, version 2.3.6
- topojson, version 3.0.2
- subtotal, version
- pivot-c3, version 2.23.0
- pivot-plotly, version 2.23.0
- pivot-d3, version 2.23.0

- jquery, version 3.6.0
- jqvmap, version 1.5.1
- qunit, version 2.13.0
- jquery-sparkline, version 2.1.2
- jqueryui, version 1.12.1
- jquery-bracket, version 0.11.1
- timepicker, version 1.13.18
- jquery-context-menu, version
- jquery-scrollbar, version
- **`pdfmake, version 0.1.70 <>`_**
- html2canvas, version 0.4.1
- dompurify, version 2.2.6
- jsPDF, version 2.3.0
- clipboard, version 2.0.6
- d3, version 6.3.1
- **`d3-tip, version 0.9.1 <>`_**
- d3-axis, version 3.0.0
- d3-ease, version 3.0.1
- d3-dsv, version 3.0.1
- d3-dispatch, version 3.0.1
- d3-transition, version 3.0.1
- d3-selection, version 3.0.0
- d3-interpolate, version 3.0.1
- d3-time-format, version 4.0.0
- d3-time, version 3.0.0
- d3-array, version 3.0.1
- d3-format, version 3.0.1
- d3-timer, version 3.0.1
- d3-collection, version 1.0.7
- d3-scale, version 4.0.0
- d3-color, version 3.0.1
- d3-brush, version 3.0.0
- d3-drag, version 3.0.0
- d3-shape, version 3.0.1
- d3-zoom, version 3.0.0
- d3-path, version 3.0.1

- plotly.js, version 2.3.0
- nvd3, version 1.8.6
- c3, version 0.7.20
- crossfilter, version 1.3.12
- **`svgjs, version 2.6.2 <>`_**
- apexcharts, version 3.27.1
- dc, version 4.2.7
- vega, version 5.20.2
- vega-tooltip, version 0.25.1
- vega-util, version 1.16.1
- vega-lite, version 5.1.0
- vega-embed, version 6.18.2
- billboard.js, version 3.1.1
- rough-viz, version 1.0.6
- frappe-charts, version 1.5.1
- @chartshq/muze, version 2.0.0
- chart.js, version 3.5.0
- chartjs-plugin-dragdata, version latest
- chartjs-plugin-annotation, version 0.5.7
- chartjs-plugin-datalabels, version 0.7.0
- chartjs-plugin-labels, version 1.1.0
- chartjs-plugin-crosshair, version 1.1.6
- chartjs-plugin-zoom, version 0.7.7
- chartjs-chart-geo, version 3.1.0
- hammer, version 2.0.8
- @popperjs/core, version 2.10.1
- bootstrap-select, version 1.13.18
- ajax-bootstrap-select, version 1.4.5
- vis, version 4.21.0
- vis-timeline, version 7.3.7
- mathjax, version 3.1.2
- socket.io, version 3.0.4
- codemirror, version 5.59.2
- codemirror-search, version
- codemirror-placeholder, version
- codemirror-trailingspace, version

- codemirror-fullscreen, version
- codemirror-highlighter, version
- codemirror-hint, version
- codemirror-panel, version
- codemirror-fold, version
- highlight.js, version 10.4.1
- leaflet, version 1.7.1
- showdown, version 1.9.1
- sortablejs, version 1.10.2
- google-platform, version
- facebook-sdk, version 0.3.3
- tiny-slider, version 2.9.3

5.5.6.4 Framework

5.5.6.4.1 Import Manager

The import Manager is one of the entry point directly accessible from Epyk.

class `epyk.core.js.Imports.ImportManager`(*page=None*)

The main class in charge of defining the order of the imports in the header.

There is no check on the presence of the modules on the server. The only purpose of this module is to produce the string with the module names and the correct paths to your final HTML report.

add(*alias: str*)

Add package to the page external required modules.

Parameters

alias – The external module alias

addPackage(*alias: str, config: dict*)

Add a new package or update an existing one with new parameters.

Only few parameters are available here in order to limit the changes.

Usage:

```
i.addPackage('test',
{
    'req': [{ 'alias': 'd3' }],
    'modules': [
        { 'script': 'dc.min.css', 'version': '3.0.9', 'path': 'dc/%(version)s/',
        ↪ 'cdnjs': 'https://cdnjs.cloudflare.com/ajax/libs',
        { 'script': 'dc.min.js', 'version': '3.0.9', 'path': 'dc/%(version)s/',
        ↪ 'cdnjs': 'https://cdnjs.cloudflare.com/ajax/libs',
        ]},
    ]
})
```

Parameters

- **alias** – The package alias
- **config** – The Python dictionary with the package details

Returns

The import Manager.

cleanImports(*imports: List[str], import_hierarchy: Optional[dict] = None, use_require_js: bool = False*)

Remove the underlying imports to avoid duplicated entries.

Usage:

```
>>> ImportManager().cleanImports(['c3'], JS_IMPORTS)
```

```
['jquery', 'd3', 'c3']
```

Parameters

- **imports** – An array with the list of aliases for the external packages
- **import_hierarchy** – Optional. The package definition (Javascript | CSS) from the above import list
- **use_require_js** – Optional. Define if this is using requirejs to load imports. Default False

Returns

Return the list with the full list of aliases (including dependencies)

cssGetAll()

To retrieve the full list of available modules on the server.

This will return the dependencies as they should be included in the HTML page. The order and the path resolution is already performed.

If split is True the generated css file will be not included.

Usage:

```
print(page.imports.cssGetAll())
```

cssResolve(*css_aliases: List[str], local_css: Optional[dict] = None, excluded: Optional[List[str]] = None*)

Return the list of CSS modules to add to the header.

Usage:

```
>>> ImportManager().cssResolve(['c3'])
```

```
<link rel="stylesheet" href="/static/c3/0.6.12/c3.min.css" type="text/css">
```

Parameters

- **css_aliases** – An array with the list of aliases for the external packages
- **local_css** – Optional. The external file overrides with the full path
- **excluded** – Optional. Packages excluded from the result object (mandatory for some frameworks already onboarding modules).

Returns

The string to be added to the header.

cssURLs(*css_str: str*)

Retrieve the list of CSS dependencies URL from a header.

Parameters

css_str – The CSS String in the page

Returns

A Python list with all the CSS external URL to be imported.

extend(*aliases: List[str]*)

Add multiple aliases to the external requirements.

Parameters

aliases – The list of package aliases to be added

getFiles(*css_alias: List[str], js_alias: List[str]*)

Retrieve the package definition from the list of module aliases

Usage:

```
>>> ImportManager().getFiles(['c3'], ['c3'])
```

```
f['css'][0]['file']['script']
```

Parameters

- **css_alias** – An array with the list of aliases for the CSS external packages
- **js_alias** – An array with the list of aliases for the Js external packages

Returns

A dictionary with the CSS and JS files definition.

getFullPackage(*alias: str, version: Optional[str] = None, static_path: Optional[str] = None, reload: bool = False*)

Download a full package (CSS and JS) locally for a server or full offline mode.

Usage:

```
Imports.ImportManager(report=Report()).getFullPackage('font-awesome')
```

Parameters

- **alias** – The package reference in the above list
- **version** – Optional. The package version to retrieve
- **static_path** – Optional. The path in which the files should be copied to
- **reload** – Optional. Flag to force the package reloading if the folder already exists. Default False

Returns

The Python Import manager.

getModules(*modules: dict, alias: Union[str, dict], folder: Optional[str] = None, module_details: Optional[dict] = None*)

Return the list of modules for a given entry.

This will be used recursively to resolve all the dependencies.

Usage:

```
modules = collections.OrderedDict()
ImportManager().getModules(modules, 'c3')
```

Parameters

- **modules** – The ordered definition of modules
- **alias** – The module reference in the above JS and CSS dictionaries
- **folder** – Optional. The folder name
- **module_details** – The module definition. Default check in the Javascript modules

Returns

The list of modules

getReq(*mod: str, modules: List[dict], import_hierarchy: Optional[dict] = None, use_require_js: bool = False*)

Set the list pf required modules for a given alias to the modules list.

Usage:

```
deps = []
page.imports.getReq("c3", deps)
print(deps)
```

Parameters

- **mod** – The alias of the external package
- **modules** – The list of packages aliases in the inverse dependency order
- **import_hierarchy** – Optional. The package definition (Javascript | CSS) from the above import list
- **use_require_js** – Optional. Define if this is using requirejs to load imports. Default False

google_products(*products: List[str], api_key: Optional[str] = None, site_key: str = '6LeIxAcTAAAAAJcZVRqyHh71UMIEGNQ_MXjiZKhI'*)

Enable the google predefined products.

Those are by default disabled as they are sharing data with Google.

TODO: Add the use of the API Key.

Usage:

```
page.imports.google_products(['charts'])
page.imports.google_products(['maps'])
page.imports.google_products(['tables'])

https://developers.google.com/recaptcha/docs/faq#id-like-to-run-automated-tests-
↪with-recaptcha.-what-should-i-do
```

Parameters

- **products** – The various Google products to enable in the report

- **api_key** – Optional. The Google developer API key
- **site_key** – Optional. The Google site key: <https://developers.google.com/recaptcha/docs/v3>

jsGetAll()

To retrieve the full list of available modules on the server.

This will return the dependencies as they should be included in the HTML page. The order and the path resolution is already performed.

If split is True the generated JS file will be not included.

Usage:

```
print(page.imports.jsGetAll())
```

jsResolve(*js_aliases: List[str], local_js: Optional[dict] = None, excluded: Optional[List[str]] = None*)

Return the list of Javascript modules to add to the header.

Usage:

```
>>> ImportManager().jsResolve(['c3'])
```

```
'<script language="javascript" type="text/javascript" src="/static/jquery/3.4.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript" src="/static/d3/5.9.7/d3.min.js"></script> <script
language="javascript" type="text/javascript" src="/static/c3/0.6.12/c3.min.js"></script>'
```

param js_aliases

An array with the list of aliases for the external packages

param local_js

Optional. The external file overrides with the full path

param excluded

Optional. Packages excluded from the result object (mandatory for some frameworks already onboarding modules)

return

The string to be added to the header

jsURLs(*expr: str*)

Retrieve the list of Javascript dependencies URL from a header.

Parameters

expr – The Javascript String in the page

Returns

A Python list with all the Javascript external URL to be imported.

locals(*aliases: List[str], end_points: Optional[str] = None*)

Short circuit the import mechanism and retrieve the selected ones from a local static path.

This could help on the debugging and the improvement of the packages before submitting them for review.

Parameters

- **aliases** – The list of aliases
- **end_points** – Optional. The end point on the server (The module static_path as default)

property pkgs: ImportPackages

Shortcut properties to the various package definitions.

This can be used in the script in order to change the path of the version of any external modules used.

property requirements: set

Retrieve all the mandatory requirements required to display the final HTML page.

Usage:

```
print(page.imports().requirements)
```

setVersion(alias: str, version: str, js: Optional[dict] = None, css: Optional[dict] = None)

Allow the use of different version of a package.

This will change the Import important to the Python env.

Usage:

```
page.imports.setVersion(page.imports.pkgs.popper_js.alias, "1.00.0")
```

Parameters

- **alias** – The package reference in the above list
- **version** – The new version to be used globally
- **js** – Optional. The JavaScript packages to be added
- **css** – Optional. The CSS packages to be added

show(all: bool = False)

Show all the underlying packages used in a report or available in the framework.

Parameters

- **all** – Optional. A flag to specify if only the one requested in the report should be displayed

to_requireJs(data: dict, excluded_packages: Optional[list] = None)**Parameters**

- **data** – The Report modules to resolve
- **excluded_packages** – Optional. The packages to exclude

website(alias: str)

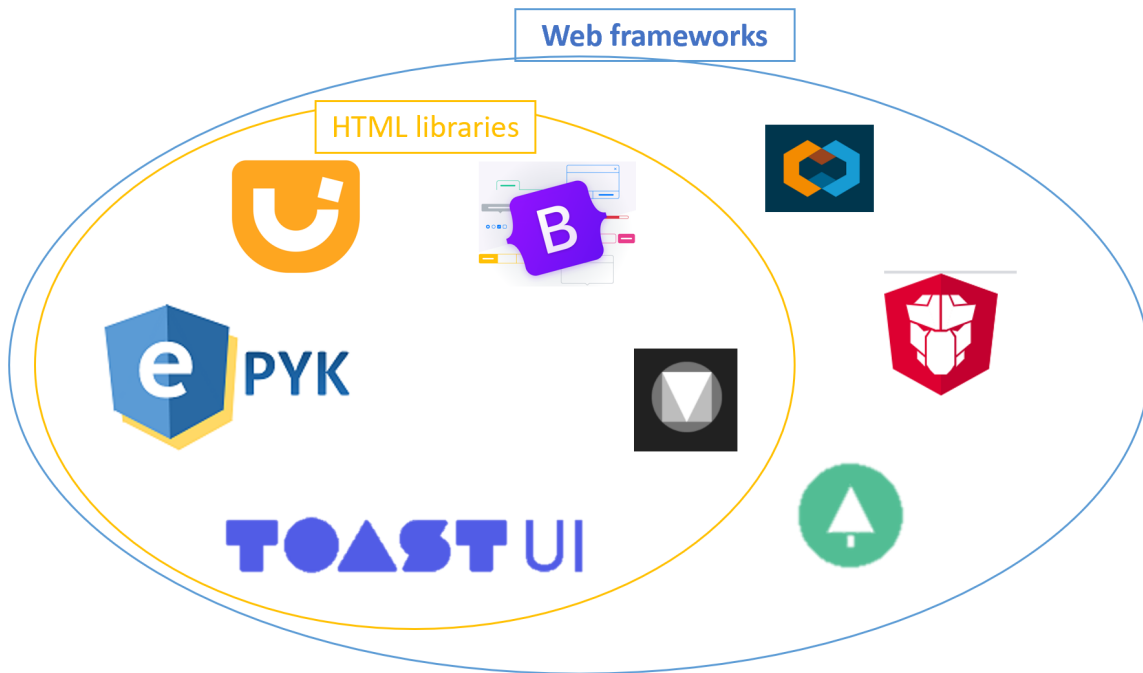
Get the official website for a JavaScript library.

Parameters

- **alias** – The JavaScript module alias (usually the one used by npm)

5.6 Web frameworks

Multiple external web frameworks with predefined components are being integrated to Epyk. It is possible to extend this list and to create your own wrapper to your preferred Web library. To do so please follow the architecture guidelines: [../guides/add_another_web_framework](#).



Warning: This part of the framework is still work in progress so feel free to help.

5.6.1 Community

This is a collaborative and open source framework so please do not hesitate to propose changes or extend the implementation using PR. Code is fully available in [Github](#) ! More details on the way to contribute using [PR](#)

5.6.2 JQuery UI

Jquery and JQuery UI are fully integrated to the core components. It means that those libraries will be added automatically when they are needed by a component.

[../report/web/jqueryui](#)

[Official website](#)

5.6.3 ToastUI

Use TOAST UI to Make Your Web Delicious!. JavaScript UI library and free open source project constantly managed by NHN.

`../report/web/toast`

[Official website](#)

5.6.4 Bootstrap

Build fast, responsive sites with Bootstrap Quickly design and customize responsive mobile-first sites with Bootstrap, the world's most popular front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins.

`../report/web/bootstrap`

[Official website](#)

5.6.5 Material Design UI

Design. Create intuitive and beautiful products with Material Design.

`../report/web/material`

[Official website](#)

5.6.6 Clarity

Work in progress (depend on the Angular integration)

5.6.7 Evergreen

Work in progress (depend on the Angular integration)

5.6.8 Common Interface

```
class epyk.interfaces.Interface.WebComponents(page: PageModel)
```

property bs: Components

Add the entire Bootstrap framework as a dependency to the framework.

This will enable more components to the framework.

More details on the Bootstrap `property` page

..note:

This will be using bootstrap 5.

Usage:

```
icon = page.web.bs.icons.danger()
```

property clr: Components

Clarity is a scalable, customizable, open source design system bolstered by the people that build with it, the people we build it for, and the community that makes us who we are.

Related Pages:

<https://clarity.design/>

Usage:

property evr: Components

Evergreen is a React UI Framework for building ambitious products on the web. Brought to you by Segment.

Related Pages:

<https://evergreen.segment.com/introduction/getting-started>

property ftw: Components

Simple components that focus on appearance and styling while showing the visual language of Office.

Usage:

```
page.web.ftw.check(label="Test Checkbox")
page.web.ftw.check(label="Test Checkbox 2")
page.web.ftw.buttons.small("Test Checkbox 2")
page.web.ftw.icon("add")
page.web.ftw.toggle(True)
page.web.ftw.loading("add", options={"large": True})
select = page.web.ftw.lists.select(selected="value 2")
data = ["value 1", "value 2", "value 3"]
select.data = select.parsers.from_list(data)
```

property jqui: Components

jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice.

More details on the JQuery property page

Related Pages:

<https://jqueryui.com/>

property mdc: Components

Set the material components entry point. This will be available in the same way than ui is available for anything else in the core framework.

More details on the Bootstrap property page

Usage:

Related Pages:

<https://material.io/develop/web/> <https://material.io/components?platform=web>

Returns

Python HTML object

property std: *Components*

The internal components.

property tui: Components

Add the entire TOAST UI framework as a dependency to the framework. This will enable more components to the framework.

Related Pages:

<https://ui.toast.com/>

Usage:

```
dt = page.web.tui.date()
cal = page.web.tui.calendar()
```

5.7 Security and Control

Epyk is an open source platform allowing to display data transformed from Python modules thanks to the existing ecosystem. For this reason the security and control of the packages used is really important.

Epyk will provide a full control on the stack used from the data retrieval to the final display in the page.

5.7.1 External libraries

It is possible to restrict the list of packages or even to update / extend it. Those packages have been tested to work with a dedicated version but it is quite easy to check if the latest version will not put any regression to your page.

5.7.2 Offline mode

To speed up performances but also to run without internet connection on a server some features are available to download the external packages. The folders structure created is the same one used by node.js to facilitate migration to other JavaScript or TypeScript web frameworks.

The property `page.imports.requirements` will allow to get the list of external packages used in the page.

It is important to put this at the end as packages are added during the run when component are attached to the page

5.7.2.1 Install packages

It is possible to install a specific package or a list of packages using the below CLI. If the package is already installed the process will not try to update it:

```
epyk_npm.exe install -pkg=promise-polyfill,@popperjs/core,bootstrap,showdown,jquery,
↪accounting,tabulator-tables,moment,chart.js
```

5.7.2.2 Update packages

It is possible to update a specific package or a list of packages using the below CLI:

```
epyk_npm.exe update -pkg=promise-polyfill,@popperjs/core,bootstrap,showdown,jquery,  
↪accounting,tabulator-tables,moment,chart.js
```

5.7.2.3 Install all packages

It is also possible to install all the packages (except the Google ones) used in the framework by using the following CLI:

```
epyk_npm.exe install_all
```

5.7.2.4 Restrict the scope

5.7.2.5 Change versions

5.7.2.6 Add packages

Adding a new package means creating a new component. It is very easy to add a new package by just using the `importManager`.

5.7.2.7 Add packages

5.7.2.8 Checking packages and versions

It is possible to check for a report or a list of reports the list of external packages. This is quite useful to get a global view of the various dependencies but also it get information about the version of the packages compared to the actual ones in [NPM](#):

```
::C:\Python\Python39\Scripts\epyk_npm.exe required -r=epyk_page_0.py  
C:\Python\Python39\Scripts\epyk_npm.exe required
```

```
C:\epyks>C:\Python\Python39\Scripts\epyk_npm.exe required  
showdown,jquery,tabulator-tables,accounting,chart.js,promise-polyfill,@popperjs/core,bootstrap,moment  
[{'name': 'showdown', 'version': '1.9.1', 'count': 2, 'latest': '1.9.1', 'usage': '40.0%'}, {'name': 'jquery', 'version':  
: '3.6.0', 'count': 3, 'latest': '3.6.0', 'usage': '60.0%'}, {'name': 'tabulator-tables', 'version': '4.9.3', 'count': 2  
, 'latest': '4.9.3', 'usage': '40.0%'}, {'name': 'accounting', 'version': '0.4.1', 'count': 2, 'latest': '0.4.1', 'usage'  
: '40.0%'}, {'name': 'chart.js', 'version': '2.9.4', 'count': 2, 'latest': '3.2.1', 'usage': '40.0%'}, {'name': 'promis  
e-polyfill', 'version': '8.2.0', 'count': 2, 'latest': '8.2.0', 'usage': '40.0%'}, {'name': '@popperjs/core', 'version':  
'2.9.2', 'count': 3, 'latest': '2.9.2', 'usage': '60.0%'}, {'name': 'bootstrap', 'version': '4.6.0', 'count': 3, 'lates  
t': '5.0.1', 'usage': '60.0%'}, {'name': 'moment', 'version': '2.29.1', 'count': 2, 'latest': '2.29.1', 'usage': '40.0%'  
}, {'name': 'font-awesome', 'version': '5.13.1', 'count': 1, 'latest': '4.7.0', 'usage': '20.0%'}, {'name': 'tiny-slider'  
, 'version': '2.9.3', 'count': 1, 'latest': '2.9.3', 'usage': '20.0%'}]
```

5.7.3 Google extensions

As google extensions are used to collect the information by default components using those modules are blocked. It will require a specific line of code in order to enable them otherwise the transpilation will raise an error:

```
# Enable Google Maps

# By default all Google products are disabled
page.imports.google_products(['maps'])

map = page.ui.geo.google.terrain(-33.92, 151.25)

# Click event to add interactivity on the page
page.ui.button("Click").click([
  map.js.setMapTypeId('satellite'),
  map.js.setHeading(45),
])
```

5.8 Advanced features

5.8.1 CLI Features

Epyk library will provide CLI features to assist in doing standard events:

1. epyk

Those CLI will provide features to transpile a script to different framework. Multiple entry points are available to allow the transpiling to different format. Functions to create new reports are also available here.

2. epyk_npm

Those CLI features will provide helper to deal with packages. It will allow to:

- check packages in a or multiple reports.
- Install external packages locally

3. epyk_project

Those CLI features will help in creating the right folder structure for projects. This will ensure an easy integration with the Studio.

5.8.1.1 Export (core) features

epyk_export.exe

`epyk.core.cli.cli_export.angular(args)`

Generate an Angular Application from the Epyk page

`epyk.core.cli.cli_export.demo(args)`

Create a page to demonstrate a example of report.

Parameters

args –

`epyk.core.cli.cli_export.html(args)`

Transpile a specific report.

Parameters

- **path** – -p, The path where the new environment will be created: -p /foo/bar
- **name** – -n, The name of the page to be transpiled: -n home
- **split** – -split, Y / N Flag to specify if the result should be splitting in several files

`epyk.core.cli.cli_export.main()`

The main function for all the export CLI entry points.

`epyk.core.cli.cli_export.page(args)`

Create a new page.

Parameters

args –

`epyk.core.cli.cli_export.transpile(args)`

Transpile a specific report

Parameters

- **name** – -p, The path where the new environment will be created: -p /foo/bar
- **path** – -n, The name of the page to be transpiled: -n home.
- **split** – -s, Y / N Flag, to specify if the files should be split input 3 modules.
- **output** – -o. String. Optional. The output destination path.
- **output** – -o. String. Optional. The output destination path.
- **colors** – String. Optional. The list of colors as string commas delimited.

5.8.1.2 NPM Wrappers

Dedicated CLI for the External packages management.

`epyk_npm.exe`

`epyk.core.cli.cli_npm.angular(args)`

Create an Angular application derived from the main NodeJs server. Then Angular CLI must be available on the NodeJs server.

Parameters

- **parser** – -s, The nodeJs server path
- **parser** – -n, The Angular application name

`epyk.core.cli.cli_npm.angular_parser(parser)`

Paser for the angular CLI

Parameters

subparser – subparser

`epyk.core.cli.cli_npm.install(args)`

Install only the defined packages locally. Those packages can be only the ones that the React or Vue scripters will be using.

The install will rely on the version and configuration in the Imports module

Usage:

```
print(", ".join(list(page.imports.requirements)))
```

```
epyk_npm.exe install -pkg=promise-polyfill,@popperjs/core,bootstrap,showdown,jquery,  
↪accounting,tabulator-tables,moment,chart.js
```

Parameters

- **packages** – -pkg. The packages list comma separated.
- **path** – -p. Optional. The project path. Default current path.
- **force** – -f. Optional. Force the update of the already installed packages. Default N.

`epyk.core.cli.cli_npm.install_all(args)`

Install all the internally defined packages locally. This will mimic the structure of NPM in order to facilitate the links.

Parameters

parser – -p, The project path

`epyk.core.cli.cli_npm.main()`

`epyk.core.cli.cli_npm.npm(args)`

Install the external packages relying on the NPM Javascript command line available on the NodeJs server. This will not install the packages using the definition in Imports but on the ones in the NPM configuration.

Parameters

- **parser** – -pkg, String, The packages list comma separated
- **parser** – -s, Path of the NodeJs server

`epyk.core.cli.cli_npm.react(args)`

Create the React application

Parameters

- **parser** – -s, The nodeJs server path
- **parser** – -n, The Angular application name

`epyk.core.cli.cli_npm.react_parser(parser)`

Parser for the vue CLI

Parameters

subparser – subparser

`epyk.core.cli.cli_npm.requirements(args)`

Get the list of external modules required for a script.

Parameters

- **path** – -p, the workspace path (Optional if run directly in the project root)

- **exception** – -e, Y/N flag
- **page** – -r, The page name (without the .py extension)

`epyk.core.cli.cli_npm.update(args)`

Install only the defined packages locally.

The install will rely on the version and configuration in the Imports module.

See also:

This is equivalent to `epyk_npm.exe install -f=Y`

Usage:

```
print(", ".join(list(page.imports.requirements)))

epyk_npm.exe update -pkg=promise-polyfill,@popperjs/core,bootstrap,showdown,jquery,
↪accounting,tabulator-tables,moment,chart.js
```

param packages

-pkg. The packages list comma separated.

param path

-p. Optional. The project path. Default current path.

`epyk.core.cli.cli_npm.vue(args)`

Create the VueJs application

Parameters

- **parser** – -s, The nodeJs server path
- **parser** – -n, The Angular application name

`epyk.core.cli.cli_npm.vue_parser(parser)`

Paser for the vue CLI

Parameters

subparser – subparser

5.8.1.3 Project features

Main command lines for Epyk

`epyk.exe`

`epyk.core.cli.cli_project.add(args)`

Add the UI structure to an existing project. This will not create a new workspace it will only add the mandatory structure for a valid UI project.

The project structure is as below: /ui

ui end points definition and data structures communication with the backend

/reports	Folder with all the Python scripts
/templates	Folder with the shared report structure
/views	Folder with the transpiled scripts

ui_settings.py, configuration module for the UI framework

Parameters

parser – -p, The path where the new environment will be created: -p /foo/bar

`epyk.core.cli.cli_project.app(args)`

Parameters

args –

`epyk.core.cli.cli_project.compile(args)`

Compile a markdown file to a valid HTML page.

Parameters

args –

`epyk.core.cli.cli_project.main()`

Main entry point for the various project command lines.

`epyk.core.cli.cli_project.new(args)`

Create a new Epyk Structure.

The project structure is as below: /ui

/reports	Folder with all the Python scripts
/templates	Folder with the shared report structure
/views	Folder with the transpiled scripts

ui_settings.py, configuration module for the UI framework

Parameters

- **parser** – -p, The path where the new environment will be created: -p /foo/bar
- **parser** – -n, The name of the new environment: -n MyEnv

`epyk.core.cli.cli_project.page(args)`

Create a new page in the current project.

Parameters

- **parser** – -p, The page / report name to be created (without the extension)
- **parser** – -n, The path where the new environment will be created: -p /foo/bar

`epyk.core.cli.cli_project.translate(args)`

Translate a markdown file to a valid Epyk python page.

Parameters

args –

`epyk.core.cli.cli_project.transpile_all(args)`

Transpile to HTML all the reports in the project Views are generated by default at the same level than the ui_setting file

Parameters

args –

5.8.2 Page

This class is your main entry point to the epyk framework, this is where you'll be able to access UI components through the report/ui interfaces or the report/js features. It takes care of writing the `body` section of your html page and also populating the `style` and `scripts` parts.

The below 2 lines of codes will show how to create a page object:

```
import epyk as pk

page = pk.Page()
```

There are 6 main interfaces available through this object which will enable you to build a complete HTML page:

- *UI Interface*
- components
- *Javascript Interface*
- *Outputs Interface*
- *Python Interface*
- *Data Interface*
- *Web Framework Interfaces*

5.8.2.1 UI Interface

The UI interface allows you to access the different parts of the framework without having to know the underlying details how these components are built.

5.8.2.1.1 External Interfaces:

Bootstrap 4

Material UI

These interfaces are grouped per category as follows:

5.8.2.1.2 Interfaces per Categories:

components Interface

Buttons Interface

```
class epyk.interfaces.components.CompButtons.Buttons(ui)
```

Buttons Interface.

```
absolute(text: str, size_notch=None, icon: str = "", top: Union[tuple, int, str] = (50, '%'), left: Union[tuple, int, str] = (50, '%'), bottom=None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None)
```


Display a button on the page regardless the current layout of components. By default, the button will be center on the page.

Tags

Categories

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Usage:

```
page.ui.buttons.absolute("Test")
```

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **size_notch** – Optional. A value to be added to the number font size
- **bottom** – Optional. The position of the component
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **top** – Optional. A tuple with the integer for the component's distance to the top of the page
- **left** – Optional. A tuple with the integer for the component's distance to the left of the page
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

button(*text: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*) → Button

Standard button

Tags

Categories

Usage:

```
page.ui.button("Test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/alerts.py>
https://github.com/epykure/epyk-templates/blob/master/locals/components/button_link.py
<https://github.com/epykure/epyk-templates/blob/master/locals/components/checkbox.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

cancel(*text: str = 'Cancel', width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, align: str = 'left', tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

Button with cross icon to cancellation actions.

Tags

Categories

Usage:

```
page.ui.buttons.cancel("Cancel")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **align** – Optional. The text-align property within this component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

check(*flag: bool = False, tooltip: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = (20, 'px'), label: Optional[str] = None, icon: Optional[str] = None, html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*)

Wrapper to the checkbox button object.

Tags

Categories

Usage:

```
page.ui.buttons.check(label="Label")
page.ui.buttons.check(True, label="Label")
page.ui.buttons.check(True, label="Label", icon="fas fa-align-center")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.CheckButton`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/checkbox.py>

Parameters

- **flag** – Optional. The value of the checkbox. Default False
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **label** – Optional. The component label content
- **icon** – Optional. The icon to be used in the check component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

checkboxes(*record=None, color: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: str = "", options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Python wrapper to the HTML checkbox elements.

Tips: record data should be in the format expected by the component. If needed a data helper can be used. from the data package in the component property, the various functions available for the checkboxes will help.

Tags**Categories**

Usage:

```
page.ui.buttons.checkboxes(data)

cb2 = page.ui.buttons.checkboxes(data, color="red", width=(100, "px"))
cb2.style.configs.shadow()
cb2.click([page.js.console.log(cb2.dom.current)])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Checkbox`

Related Pages:

https://www.w3schools.com/howto/howto_css_custom_checkbox.asp

Templates:

https://github.com/epykure/epyk-templates/blob/master/locals/components/button_checkboxes.py

Parameters

- **record** – Optional. The list of dictionaries with the data
- **color** – Optional. The color code
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. The text-align property within this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

clear(text: str = "", icon: str = 'fas fa-eraser', color: Optional[str] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None) → Button

Standard clear button with a font-awesome icon.

Usage:

```
page.ui.buttons.clear("Clear")
```

Tags**Categories**

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/alerts.py>
https://github.com/epykure/epyk-templates/blob/master/locals/components/button_link.py
<https://github.com/epykure/epyk-templates/blob/master/locals/components/checkbox.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

colored(*text: str = "", icon: Optional[str] = None, color: Optional[str] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*) → Button

Standard colored button.

Tags

Categories

Usage:

```
page.ui.buttons.colored("Test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/alerts.py>
https://github.com/epykure/epyk-templates/blob/master/locals/components/button_link.py
<https://github.com/epykure/epyk-templates/blob/master/locals/components/checkbox.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

data(filename, text: str = "", icon: Optional[Union[str, bool]] = None, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None)

Standard refresh button with a font-awesome icon.

Tags

Categories

Usage:

```
page.ui.buttons.refresh("Refresh")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/alerts.py>
https://github.com/epykure/epyk-templates/blob/master/locals/components/button_link.py
<https://github.com/epykure/epyk-templates/blob/master/locals/components/checkbox.py>

Parameters

- **filename** – Optional. The filename
- **text** – Optional. The value to be displayed to the button
- **width** – Optional. A tuple with the integer for the component width and its unit

- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

filter(text: str = "", is_number: bool = False, width: Union[tuple, int] = ('auto', ''), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None)

Tags

Categories

Usage:

```
:param text: Optional. The filter value
:param is_number: Optional. The filter property type
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
:param tooltip: Optional. A string with the value of the tooltip
:param profile: Optional. A flag to set the component performance storage
:param options: Optional. Specific Python options available for this component
```

important(text: str = "", icon: Optional[str] = None, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None)

Same as Standard button but used to attract user attention.

Tags

Categories

Underlying HTML Objects:

- epyk.core.html.HtmlButton.Button

Usage:

```
page.ui.buttons.important("Important")
```

Underlying HTML Objects:

- epyk.core.html.HtmlButton.Button

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

large(*text: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*)

Large button.

Usage:

```
page.ui.buttons.large("Test")
```

Tags

Categories

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/alerts.py>
https://github.com/epykure/epyk-templates/blob/master/locals/components/button_link.py
<https://github.com/epykure/epyk-templates/blob/master/locals/components/checkbox.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **width** – Optional. The integer for the component width and its unit
- **height** – Optional. The integer for the component height and its unit
- **align** – Optional. The horizontal position of the component

- **icon** – Optional. The value of the icon to display from font-awesome
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. The value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

live(*time: int, js_funcs: Union[List[Union[str, JsDataModel]], str], icon: Optional[Union[str, bool]] = 'fas fa-circle', width: Union[tuple, int, str] = ('auto', 'px'), height: Union[tuple, int, str] = ('auto', 'px'), align: str = 'left', html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*)

Live component which will trigger event every x second. This will then allow other components to be refreshed in the page.

Tags

Categories

Usage:

```
b7 = page.ui.buttons.live(3, page.js.console.log("Click"), options={"started":  
↪False})  
b8 = page.ui.buttons.live(2, page.js.console.log("refresh data"), profile=True)
```

Templates:

https://github.com/epykure/epyk-templates/blob/master/locals/components/button_icon.py

Parameters

- **time** – Interval time in second
- **js_funcs** – The Javascript functions
- **icon** – Optional. The font awesome icon reference
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

mail(*text: str = "", width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, align: str = 'left', tooltip=None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

Add a mail button with a predefined icon from font-awesome.

Tags

Categories

Usage:

```
page.ui.buttons.mail()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

menu(*record: Optional[list] = None, text: str = "", icon: Optional[Union[str, bool]] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*)

Button with underlying items menu.

Tags

Categories

Usage:

```
tree5 = page.ui.buttons.menu(["A", "B", "C"], 'Menu')
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.ButtonMenu`

Related Pages:

https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_js_dropdown_hover

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/tree.py>

Parameters

- **record** – Optional. The list of dictionaries with the data

- **text** – Optional. The value to be displayed to the button
- **icon** – Optional. The icon to be used in the check component
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

more(items, text: str = 'More', width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None) → ButtonMore

Tags

Categories

Usage:

```
b = page.ui.buttons.more([
    {"text": "Add", "target": "_blank", "icon": "fab fa-500px",
     "url": "https://stackoverflow.com/questions/5884066/hashing-a-dictionary"},
    {"text": "Delete", "target": "_blank", "icon": "fab fa-500px",
     "url": "https://stackoverflow.com/questions/5884066/hashing-a-dictionary"},
])
b.click([page.js.console.log("Click event")])
```

Templates:

Related Pages:

Parameters

- **items** – List of items to be added to the menu
- **text** – Optional. The text visible in the button
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

normal(text: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None)

Standard button with a standard layout.

Tags**Categories**

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Usage:

```
page.ui.buttons.normal("Standard button")
```

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

phone(*text: str = "", width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, align: str = 'left', tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

Add a phone button with a predefined icon from font-awesome.

Tags**Categories**

Usage:

```
page.ui.buttons.phone()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>

Parameters

- **text** – Optional. The value to be displayed to the button

- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

pill(*text: str, value=None, group: Optional[str] = None, width: Union[tuple, int] = ('auto', ''), height: Union[tuple, int] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

Add a pill button.

Tags

Categories

Usage:

Templates:

Parameters

- **text** – Optional. The text to be displayed to the button
- **value** – Optional. The value to be displayed in the pill
- **group** – Optional. The group value for the pill
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

radio(*record: Optional[List[dict]] = None, html_code: Optional[str] = None, group_name: Optional[str] = None, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (None, 'px'), align: str = 'left', options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Radio

Creates a radio HTML component.

Tips: record data should be in the format expected by the component. If needed a data helper can be used. from the data package in the component property, the various functions available for the radio will help.

Tags

Categories

Usage:

```
page.ui.buttons.radio(df, dfColumn="A", htmlCode="test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlRadio.Radio`

Related Pages:

https://www.w3schools.com/bootstrap/bootstrap_forms_inputs.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>

Parameters

- **record** – Optional. The Python list of dictionaries
- **group_name** – Optional. Group name for multi radio buttons
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

refresh(*text: str = 'Refresh', icon: Optional[Union[str, bool]] = 'fas fa-sync-alt', width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

Standard refresh button with a font-awesome icon.

Tags

Categories

Usage:

```
page.ui.buttons.refresh("Refresh")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/alerts.py>
https://github.com/epykure/epyk-templates/blob/master/locals/components/button_link.py
<https://github.com/epykure/epyk-templates/blob/master/locals/components/checkbox.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **align** – Optional. A string with the horizontal position of the component
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

remove(*text: str = "*, *width: Union[tuple, int] = (None, '%')*, *height: Union[tuple, int] = (None, 'px')*, *html_code: Optional[str] = None*, *align: str = 'left'*, *tooltip: Optional[str] = None*, *profile: Optional[Union[bool, dict]] = None*, *options: Optional[dict] = None*)

Button with cross icon.

Tags

Categories

Usage:

```
page.ui.buttons.remove("Remove")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

run(*text: str = "*, *width: Union[tuple, int] = (None, '%')*, *height: Union[tuple, int] = (None, 'px')*, *align: str = 'left'*, *html_code: Optional[str] = None*, *tooltip: Optional[str] = None*, *profile: Optional[Union[bool, dict]] = None*, *options: Optional[dict] = None*)

Add a run button with a predefined icon from font awesome.

Tags

Categories

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Usage:

```
page.ui.buttons.run("Run")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. The text-align property within this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

small(*text: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*)

Standard button with a small layout.

Tags

Categories

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Usage:

```
page.ui.buttons.small("Small button")
```

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>

Parameters

- **text** – Optional. The value to be displayed to the button

- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

store(*image, url, width: Union[tuple, int, str] = (7.375, 'rem'), height: Union[tuple, int, str] = (2.375, 'rem'), html_code: Optional[str] = None, align: str = 'left', options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Button for a badge which point to the various application stores (Google and Apple). The badge must be issued from the Google Play Store.

Tags

Categories

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlImage.Image`

Related Pages:

https://play.google.com/intl/en_us/badges/

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>

Parameters

- **image** – The url of the image
- **url** – The link to the app in the store
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **align** – Optional. The text-align property within this component
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

text(*text: str, icon: Optional[Union[str, bool]] = None, width: Union[tuple, int] = ('auto', ''), tooltip: Optional[str] = None, height: Union[tuple, int] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

Add a text button.

Tags
Categories

Usage:

Templates:

Parameters

- **text** – Optional. The value to be displayed to the button
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

thumbs_down(*width: Union[tuple, int] = ('auto', ''), height: Union[tuple, int] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

Button with the font awesome icon far fa-thumbs-down.

Tags
Categories

Usage:

Templates:

Parameters

- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

thumbs_up(*width: Union[tuple, int] = ('auto', ''), height: Union[tuple, int] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

Button with the font awesome icon far fa-thumbs-up.

Tags
Categories

Usage:

```
b6 = page.ui.buttons.thumbs_up(tooltip="Like")
```

Templates:

Parameters

- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

toggle(*record: Optional[dict] = None, label: Optional[str] = None, color: Optional[str] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Add a toggle component.

Component Structure:

```
component.input: :class:`html.HtmlRadio.Switch`
component.label: :class:`html.HtmlText.Label`
```

Tags

Categories

Usage:

```
page.ui.buttons.toggle({'on': "true", 'off': 'false'})
```

Underlying HTML Objects:

- `epyk.core.html.HtmlRadio.Switch`

Related Pages:

<http://thecodeplayer.com/walkthrough/pure-css-on-off-toggle-switch>
mburnette/pen/LxNxNg

<https://codepen.io/>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/switch.py>

Parameters

- **record** – Optional. component data
- **label** – Optional. The toggle static label displayed
- **color** – Optional. String. Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit

- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

validate(*text: str = "", width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, align: str = 'left', tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

Add a validate button with a predefined icon from font awesome.

Tags

Categories

Usage:

```
page.ui.buttons.validate("Validate")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Button`

Related Pages:

https://www.w3schools.com/tags/tag_button.asp <http://www.kodingmadesimple.com/2015/04/custom-twitter-bootstrap-buttons-icons-images.html>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>

Parameters

- **text** – Optional. The value to be displayed to the button
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **align** – Optional. The text-align property within this component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

Calendar Interface

class epyk.interfaces.components.CompCalendars.**Calendar**(*ui*)

agenda(*task, start, end, details=None, location=None, icon: str = 'calendar', text: str = 'Add to Calendar', options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
:tags:
:categories:
```

Templates:

Related Pages:

<https://stackoverflow.com/questions/5179760/add-events-to-google-calendar-yahoo-calendar-outlook-and-ical>
<https://codepen.io/vlemoine/pen/MLwygX>

TODO: improve the time management in this component

Parameters

- **task** –
- **start** –
- **end** –
- **details** – Optional.
- **location** – Optional.
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **text** – Optional. The value to be displayed to the button
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

days(*month: Optional[int] = None, content=None, year: Optional[int] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), align: Optional[str] = None, options: Optional[dict] = None, html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
content = {
    "2020-07-02": {'task1': 50, 'task2': 50},
    "2020-07-03": {'task1': 100},
    "2020-07-21": {'task4': 100},
    "2020-07-22": {'task4': 100}
}

july = page.ui.calendars.days(7, content, align="center", options={"colors": {
    ↪ "task4": 'red'}})
july.task('task1', start="2020-07-10", capacity=[50, 30, 10, 80])
july.task('task4', start="2020-07-20", capacity=[50, 40, 10])
```

(continues on next page)

(continued from previous page)

```
july.weekly("task6", start="2020-07-02", capacity=3, frequency=2, options={'unit'  
↪': 8})
```

Templates:

Related Pages:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/calendar.py>**Parameters**

- **month** – Optional. The month number
- **content** –
- **year** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. The text-align property within this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

forecast(*month_period: int, content: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), position: str = 'top', options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Display a forecast based on a dictionary containing the values for several months

Usage:

```
:tags:  
:categories:
```

Templates:

Related Pages:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/calendar.py>**Parameters**

- **month_period** – Number of months of forecast
- **content** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **position** – Optional. The position compared to the main component tag
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

google(*task*, *start*: str, *end*: str, *details*=None, *location*=None, *icon*: str = 'google_plus', *text*: str = 'Add to Google Calendar', *options*: Optional[Union[bool, dict]] = None, *profile*: Optional[Union[bool, dict]] = None)

Add link to the google calendar. Will add the event to the Calendar.

Usage:

```
page.ui.calendars.google("hrehr", "Test", "20200801T153000Z", "20200802T163000Z",
↪")
```

Templates:

Tags

Categories

Related Pages:

<https://stackoverflow.com/questions/5179760/add-events-to-google-calendar-yahoo-calendar-outlook-and-ical>
<https://codepen.io/vlemoine/pen/MLwygX>

TODO: improve the time management in this component

Parameters

- **task** –
- **start** – Date format YYYYMMDD
- **end** – Date format YYYYMMDD
- **details** – Optional.
- **location** – Optional.
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **text** – Optional. The value to be displayed to the button
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

legend(*record*: list, *width*: Union[tuple, int, str] = (None, '%'), *height*: Union[tuple, int, str] = (None, 'px'), *align*: Optional[str] = None, *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

Add a legend to a Calendar component.

Tags

Categories

Usage:

```
monthly = page.ui.calendars.months(content=records, align="center")
page.ui.calendars.legend(monthly.tasks)
```

Templates:

Related Pages:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/calendar.py>

Parameters

- **record** – Optional. The list of dictionaries with the input data

- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. The text-align property within this component
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

months(*content: Optional[dict] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), align: Optional[str] = None, options: Optional[dict] = None, html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
records = {
    1: {"Project 1": 12, "Project 2": 30},
    2: {"Project 1": 12, "Project 2": 30},
    3: {"Project 1": 42, "Project 2": 30},
    4: {"Project 1": 15, "Project 2": 30},
    5: {"Project 1": 12, "Project 2": 30},
    6: {"Project 1": 12, "Project 2": 30},
    7: {"Project 1": 12, "Project 2": 30},
}

monthly = page.ui.calendars.months(content=records, align="center")
monthly.style.css.margin_top = 10
```

Templates:

Related Pages:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/calendar.py>

Parameters

- **content** – Optional. The Pie charts values
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

pill(*text: str, value=None, group=None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

Usage:


```
pill = page.ui.calendars.pill("4D")
page.ui.button("Click").click([page.js.alert(pill.dom.content)])
```

Templates:

Tags

Categories

Parameters

- **text** – Optional. The value to be displayed to the button
- **value** – Optional.
- **group** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

timer(minutes: int, text: str = "", width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), align: Optional[str] = None, options: Optional[dict] = None, html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None)

Tags

Categories

Usage:

```
page = pk.Page()
dt = page.ui.calendars.timer(5)
```

Parameters

- **minutes** –
- **text** – Optional. The value to be displayed to the timer
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. The text-align property within this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage.
- **options** – Optional. Specific Python options available for this component.

Code Interface

class epyk.interfaces.components.CompCodes.**Code**(ui)

code(language: str, text: str = "", color: Optional[Union[str, bool]] = None, width: Union[tuple, int, str] = (90, '%'), height: Union[tuple, int, str] = (200, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None)

Generic code editor.

Underlying HTML Objects:

- epyk.core.html.HtmlTextEditor.Code

Related Pages:

<https://codemirror.net/index.html>

Usage:

Templates:

Parameters

- **language** – The language
- **text** – Optional. The text
- **color** – Optional. The color code
- **width** – Optional. The with details in the format(value, unit)
- **height** – Optional. The height details in the format(value, unit)
- **html_code** – Optional. The unique component ID
- **options** – Optional. The object properties
- **helper** – Optional. The helper
- **profile** – Optional. A flag to set the component performance storage

css(text: str = "", color: Optional[Union[str, bool]] = None, width: Union[tuple, int, str] = (90, '%'), height: Union[tuple, int, str] = (200, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None)

CSS Text editor.

Tags

Categories

Underlying HTML Objects:

- epyk.core.html.HtmlTextEditor.Code

Related Pages:

<https://codemirror.net/index.html>

Usage:

```
page.ui.codes.css('''
.cssdivnoborder {margin: 0 ;clear: both ;padding: 0 ;border: 0 ;}
.cssdivnoborder:focus {outline: 1px solid #B4BABF ;}
''')
```

Templates:

Parameters

- **text** – Optional. The text
- **color** – Optional. The color code
- **width** – Optional. The with details in the format(value, unit)
- **height** – Optional. The height details in the format(value, unit)
- **html_code** – Optional. The unique component ID
- **options** – Optional. The object properties
- **helper** – Optional. The helper
- **profile** – Optional. A flag to set the component performance storage

javascript(*text: str = "", color: Optional[Union[str, bool]] = None, width: Union[tuple, int, str] = (90, '%'), height: Union[tuple, int, str] = (200, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*)

Javascript Text editor.

Tags

Categories

Underlying HTML Objects:

- `epyk.core.html.HtmlTextEditor.Code`

Related Pages:

<https://codemirror.net/index.html>

Usage:

Templates:

Parameters

- **text** – Optional. The text
- **color** – Optional. The color code
- **width** – Optional. The with details in the format(value, unit)
- **height** – Optional. The height details in the format(value, unit)
- **html_code** – Optional. The unique component ID
- **options** – Optional. The object properties
- **helper** – Optional. The helper
- **profile** – Optional. A flag to set the component performance storage

markdown(*text: str = "", color: Optional[Union[str, bool]] = None, width: Union[tuple, int, str] = (90, '%'), height: Union[tuple, int, str] = (200, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*)

Markdown Text editor.

Underlying HTML Objects:

- `epyk.core.html.HtmlTextEditor.Code`

Related Pages:

<https://codemirror.net/index.html>

Usage:

Templates:

Parameters

- **text** – Optional. The text
- **color** – Optional. The color code
- **width** – Optional. The with details in the format(value, unit)
- **height** – Optional. The height details in the format(value, unit)
- **html_code** – Optional. The unique component ID
- **options** – Optional. The object properties
- **helper** – Optional. The helper
- **profile** – Optional. A flag to set the component performance storage

python(*text: str = "", color: Optional[Union[str, bool]] = None, width: Union[tuple, int, str] = (90, '%'), height: Union[tuple, int, str] = (200, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*)

Python Text editor.

Tags

Categories

Underlying HTML Objects:

- `epyk.core.html.HtmlTextEditor.Code`

Related Pages:

<https://codemirror.net/index.html>

Usage:

Templates:

Parameters

- **text** – Optional. The text
- **color** – Optional. The color code
- **width** – Optional. The with details in the format(value, unit)
- **height** – Optional. The height details in the format(value, unit)
- **html_code** – Optional. The unique component ID
- **options** – Optional. The object properties
- **helper** – Optional. The helper
- **profile** – Optional. A flag to set the component performance storage

r(text: str = "", color: Optional[Union[str, bool]] = None, width: Union[tuple, int, str] = (90, '%'), height: Union[tuple, int, str] = (200, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None)

R Text editor.

Tags

Categories

Underlying HTML Objects:

- `epyk.core.html.HtmlTextEditor.Code`

Related Pages:

<https://codemirror.net/index.html>

Usage:

Templates:

Parameters

- **text** – Optional. The text
- **color** – Optional. The color code
- **width** – Optional. The with details in the format(value, unit)
- **height** – Optional. The height details in the format(value, unit)
- **html_code** – Optional. The unique component ID
- **options** – Optional. The object properties
- **helper** – Optional. The helper
- **profile** – Optional. A flag to set the component performance storage

rst(text: str = "", color: Optional[Union[str, bool]] = None, width: Union[tuple, int, str] = (90, '%'), height: Union[tuple, int, str] = (200, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None)

RestructuredText editor.

Underlying HTML Objects:

- `epyk.core.html.HtmlTextEditor.Code`

Related Pages:

<https://codemirror.net/index.html>

Usage:

Templates:

Parameters

- **text** – Optional. The text
- **color** – Optional. The color code
- **width** – Optional. The with details in the format(value, unit)
- **height** – Optional. The height details in the format(value, unit)
- **html_code** – Optional. The unique component ID
- **options** – Optional. The object properties

- **helper** – Optional. The helper
- **profile** – Optional. A flag to set the component performance storage

sql(*text: str = "", color: Optional[Union[str, bool]] = None, width: Union[tuple, int, str] = (90, '%'), height: Union[tuple, int, str] = (200, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*)

SQL Text editor.

Tags

Categories

Underlying HTML Objects:

- `epyk.core.html.HtmlTextEditor.Code`

Related Pages:

<https://codemirror.net/index.html>

Usage:

Templates:

Parameters

- **text** – Optional. The text
- **color** – Optional. The color code
- **width** – Optional. The with details in the format(value, unit)
- **height** – Optional. The height details in the format(value, unit)
- **html_code** – Optional. The unique component ID
- **options** – Optional. The object properties
- **helper** – Optional. The helper
- **profile** – Optional. A flag to set the component performance storage

xml(*text: str = "", color: Optional[Union[str, bool]] = None, width: Union[tuple, int, str] = (90, '%'), height: Union[tuple, int, str] = (200, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*)

XML Text editor.

Tags

Categories

Underlying HTML Objects:

- `epyk.core.html.HtmlTextEditor.Code`

Related Pages:

<https://codemirror.net/index.html>

Usage:

Templates:

Parameters

- **text** – Optional. The text
- **color** – Optional. The color code

- **width** – Optional. The with details in the format(value, unit)
- **height** – Optional. The height details in the format(value, unit)
- **html_code** – Optional. The unique component ID
- **options** – Optional. The object properties
- **helper** – Optional. The helper
- **profile** – Optional. A flag to set the component performance storage

Drawers Interface

class epyk.interfaces.components.CompDrawers.**Drawers**(ui)

drawer(width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (100, '%'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None) → Drawer

Bespoke drawer with handle on the right.

tags
categories

Usage:

```
d1 = page.ui.drawer()
d1.add_panel(page.ui.button("Test"), "ok")
d1.drawers[0].click([d1.panels[0].dom.css({"display": 'block'})])
```

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/drawers.py>

- **param width**
Optional. A tuple with the integer for the component width and its unit
- param height**
Optional. A tuple with the integer for the component height and its unit
- param options**
Optional. A dictionary with the components properties
- param profile**
Optional. A flag to set the component performance storage
- param helper**
Optional. A tooltip helper

left(width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (200, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None) → Drawer

Bespoke drawer with handle on the left.

tags
categories

Usage:

- **param width**
Optional. A tuple with the integer for the component width and its unit
- param height**
Optional. A tuple with the integer for the component height and its unit
- param options**
Optional. A dictionary with the components properties
- param profile**
Optional. A flag to set the component performance storage
- param helper**
Optional. A tooltip helper

multi(*component: Html, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (200, 'px'), options: Optional[dict] = None, profile: Optional[dict] = None, helper: Optional[str] = None*) → DrawerMulti

tags
categories

Usage:

Templates:

https://github.com/epykure/epyk-templates/blob/master/locals/components/multi_drawers.py

- **param component**
Object in charge of managing the panel display
- param width**
Optional. A tuple with the integer for the component width and its unit
- param height**
Optional. A tuple with the integer for the component height and its unit
- param options**
Optional. A dictionary with the components properties
- param profile**
Optional. A flag to set the component performance storage
- param helper**
Optional. A tooltip helper

no_handle(*component: Html, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (200, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*) → Drawer

Bespoke drawer without handle. The event to display the panel will be attached to the component.

tags
categories

Usage:

```
page.ui.drawers.no_handle(page.ui.button("No Handle"))
```


- **param component**
Object in charge of managing the panel display
- param width**
Optional. A tuple with the integer for the component width and its unit
- param height**
Optional. A tuple with the integer for the component height and its unit
- param options**
Optional. A dictionary with the components properties
- param profile**
Optional. A flag to set the component performance storage
- param helper**
Optional. A tooltip helper

right(width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (200, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None)
→ Drawer

Bespoke drawer with handle on the left.

tags
categories

Usage:

```
button = page.ui.button("Test")

d = page.ui.drawers.right()
d.add_panel(page.ui.button("Test1"), "ok1")
d.set_handle(button)

d.drawers[0].click([
    d.dom.hide(),
    d.panels[0].dom.css({"display": 'block'}).r,
    page.js.console.log(d.dom.content)
])
```

- **param width**
Optional. A tuple with the integer for the component width and its unit
- param height**
Optional. A tuple with the integer for the component height and its unit
- param options**
Optional. A dictionary with the components properties
- param profile**
Optional. A flag to set the component performance storage
- param helper**
Optional. A tooltip helper

Fields Interface

class epyk.interfaces.components.CompFields.**Fields**(ui)

autocomplete(value: str = "", label: Optional[str] = None, placeholder: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None) → FieldAutocomplete

Tags

Categories

Usage:

```
page.ui.fields.autocomplete("", label="Range Example", icon="fas fa-unlock-alt")
```

Underlying HTML Objects:

- epyk.core.html.HtmlInput.FieldAutocomplete

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component. Default empty string
- **label** – Optional. The text of label to be added to the component
- **placeholder** – Optional. The text to be displayed when the input is empty
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

checkbox(value: bool = False, label: Optional[str] = None, icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None) → FieldCheckBox

Tags

Categories

Usage:

```
page.ui.fields.checkbox(True, label="Check")
```

Underlying HTML Objects:

- epyk.core.html.HtmlInput.FieldCheckBox

Related Pages:

https://www.w3schools.com/tags/att_input_type_checkbox.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component. Default empty string
- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

cob(*label: Optional[str] = None, icon: str = 'calendar', color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*) → DatePicker

This component is based on the JQuery Date Picker object.

Tags

Categories

Usage:

```
page.ui.fields.cob(label="COB Date")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlDates.DatePicker`

Related Pages:

<https://jqueryui.com/datepicker/>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. The component icon content from font-awesome references
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component
- **helper** – Optional. A tooltip helper

column_date(label, value: str = 'T', align: str = 'left', width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None) → Div

Tags

Categories

Usage:

```
:param label: Optional. The text of label to be added to the component
:param value: Optional. The value to be displayed to this component. Default T
:param align: Optional. The text-align property within this component
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

column_text(label, text: str = "", align: str = 'left', width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None) → Div

Tags

Categories

Usage:

```
:param label: Optional. The text of label to be added to the component
:param text: Optional. The string value to be displayed in the component
:param align: Optional. The text-align property within this component
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

date(value: Optional[str] = None, label: Optional[str] = None, icon: str = 'calendar', color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None) → DatePicker

This component is based on the JQuery Date Picker object.

Tags

Categories

Usage:

```
dt = page.ui.fields.date('2020-04-08', label="Date").included_dates(["2020-04-08", "2019-09-06"])
dt.select([page.js.alert(dt.dom.content)])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlDates.DatePicker`

Related Pages:

<https://jqueryui.com/datepicker/>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/dates.py>

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to the time component. Default now
- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. The component icon content from font-awesome references
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component
- **helper** – Optional. A tooltip helper

days(*value=None, label: Optional[str] = None, icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → FieldSelect

Tags

Categories

Usage:

```
page.ui.fields.select(["a", "b"], label="Check")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldSelect`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component
- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

file(*value: str = "", label: Optional[str] = None, placeholder: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → FieldFile

Tags

Categories

Usage:

```
page.ui.fields.integer(label="test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldFile`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component. Default empty string
- **label** – Optional. The text of label to be added to the component
- **placeholder** – Optional. The text to be displayed when the input is empty
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

filters(*items=None, button=None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (60, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, autocomplete: bool = False, kind: str = 'select', profile: Optional[Union[bool, dict]] = None*)

Parameters

- **items** –
- **button** –
- **width** –
- **height** –
- **html_code** –
- **helper** –
- **options** –
- **autocomplete** –
- **kind** –
- **profile** –

hidden(*value: str = "", label: Optional[str] = None, placeholder: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → `FieldInput`

Create a hidden HTML component. This is used to store values which are not visible on the page.

Tags

Categories

Usage:

```
page.ui.fields.hidden(label="readonly field")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldInput`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component. Default empty string
- **label** – Optional. The text of label to be added to the component
- **placeholder** – Optional. The text to be displayed when the input is empty
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper

- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

input(*value: str = "", label: Optional[str] = None, placeholder: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → FieldInput

Tags

Categories

Usage:

```
page.ui.fields.input("", label="Range Example", icon="fas fa-unlock-alt")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldInput`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component. Default empty string
- **label** – Optional. The text of label to be added to the component
- **placeholder** – Optional. The text to be displayed when the input is empty
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

integer(*value: str = "", label: Optional[str] = None, placeholder: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → FieldInteger

Tags

Categories

Usage:

```
page.ui.fields.integer(label="test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldInteger`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component. Default empty string
- **label** – Optional. The text of label to be added to the component
- **placeholder** – Optional. The text to be displayed when the input is empty
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

months(*value=None, label: Optional[str] = None, icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → FieldSelect

Tags

Categories

Usage:

```
page.ui.fields.select(["a", "b"], label="Check")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldSelect`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component
- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

now(*deltatime*: int = 0, *label*: Optional[str] = None, *icon*: str = 'clock', *color*: Optional[str] = None, *html_code*: Optional[str] = None, *profile*: Optional[Union[bool, dict]] = None, *options*: Optional[Union[bool, dict]] = None, *helper*: Optional[str] = None) → TimePicker

This component is based on the JQuery Time Picker object.

Tags

Categories

Usage:

```
page.ui.fields.now(label="timestamp", color="red", helper="This is the report_↵
↵timestamp")
page.ui.fields.now(label="Time field")
```

Underlying HTML Objects:

- epyk.core.html.HtmlDates.TimePicker

Related Pages:

<https://github.com/jonthornton/jquery-timepicker>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **deltatime** – Optional
- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. The component icon content from font-awesome references
- **color** – Optional. The font color in the component. Default inherit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component
- **helper** – Optional. A tooltip helper

number(*value*: str = "", *label*: Optional[str] = None, *placeholder*: str = "", *icon*: Optional[str] = None, *width*: Union[tuple, int, str] = (100, '%'), *height*: Union[tuple, int, str] = (None, 'px'), *html_code*: Optional[str] = None, *helper*: Optional[str] = None, *options*: Optional[Union[bool, dict]] = None, *profile*: Optional[Union[bool, dict]] = None) → FieldInput

Tags

Categories

Usage:

```
page.ui.fields.input("", label="Range Example", icon="fas fa-unlock-alt")
```

Underlying HTML Objects:

- epyk.core.html.HtmlInput.FieldInput

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component. Default empty string
- **label** – Optional. The text of label to be added to the component
- **placeholder** – Optional. The text to be displayed when the input is empty
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

password(*value: str = "", label: Optional[str] = None, placeholder: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → FieldPassword

Tags**Categories**

Usage:

```
page.ui.fields.password(label="password")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldPassword`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component. Default empty string
- **label** – Optional. The text of label to be added to the component
- **placeholder** – Optional. The text to be displayed when the input is empty
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

radio(*value: bool = False, label: str = "", group_name: Optional[str] = None, icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Radio

The `<input type="radio">` defines a radio button. Radio buttons are normally presented in radio groups (a collection of radio buttons describing a set of related options). Only one radio button in a group can be selected at the same time.

Tags

Categories

Usage:

```
page.ui.inputs.radio(False, label="radio")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.Radio`

Related Pages:

https://www.w3schools.com/tags/att_input_type_radio.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component. Default False
- **label** – Optional. The text of label to be added to the component
- **group_name** – Optional. Group different radio together to only have 1 value selected
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

range(*value: str = "", min: int = 0, max: int = 100, step: int = 1, label: Optional[str] = None, placeholder: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → FieldRange

The `<input type="range">` defines a control for entering a number whose exact value is not important. Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the attributes below. - **max** - specifies the maximum value allowed - **min** - specifies the minimum value allowed - **step** - specifies the legal number intervals - **value** - Specifies the default value

Tags

Categories

Usage:

```
page.ui.fields.range(54, min=20, label="Range Example", icon="fas fa-unlock-alt
↔")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldRange`

Related Pages:

https://www.w3schools.com/tags/att_input_type_range.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component. Default empty string
- **min** –
- **max** –
- **step** –
- **label** – Optional. The text of label to be added to the component
- **placeholder** –
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

select(*value: Optional[list] = None, label: Optional[str] = None, icon: Optional[str] = None, selected: Optional[bool] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, multiple: bool = False, profile: Optional[Union[bool, dict]] = None*) → FieldSelect

Tags

Categories

Usage:

```
page.ui.fields.select(["a", "b"], label="Check")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldSelect`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to the component. Default False
- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. The component icon content from font-awesome references
- **selected** – Optional. The selected value
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **multiple** – Optional. Flag to specify the number of items to be selectable
- **profile** – Optional. A flag to set the component performance storage

slider(value: float = 0, min: float = 0, max: float = 10, step: float = 1, orientation: str = 'horizontal', label: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, range: bool = False, profile: Optional[Union[bool, dict]] = None) → Div

Parameters

- **value** – Optional. The value to be displayed to this component. Default 0
- **min** – Optional.
- **max** – Optional.
- **step** – Optional.
- **orientation** – Optional.
- **label** – Optional. The toggle static label displayed
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **range** – Optional.
- **profile** – Optional. A flag to set the component performance storage

static(value: str = "", label: Optional[str] = None, placeholder: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None, input_tag: bool = False) → FieldInput

Tags**Categories**

Usage:

```

page.ui.fields.static(label="readonly field")

page.ui.fields.static(''
    Value Formatter €
    A Value Formatter is...
'', label="toto", options={"html_encode": True, "multiline": True})

```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldInput`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component. Default empty string
- **label** – Optional. The text of label to be added to the component
- **placeholder** – Optional. The text to be displayed when the input is empty
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component
- **input_tag** – Optional. Use an input field

text(*text: str = "", label: Optional[str] = None, color: Optional[str] = None, align: str = 'left', width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, tooltip: Optional[str] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → Text

Add the HTML text component to the page.

Tags

Categories

Usage:

```

page.ui.text("this is a test")

```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Text`

Related Pages:

https://www.w3schools.com/tags/tag_font.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/banners.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/contextmenu.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/image.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/markdown.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/postit.py>

Parameters

- **text** – Optional. The string value to be displayed in the component
- **label** – Optional. The text of label to be added to the component
- **color** – Optional. The color of the text
- **align** – Optional. The position of the icon in the line (left, right, center)
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** (*tuple*) – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **options** – Optional. Specific Python options available for this component
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage

textarea(*value: str = "", label: Optional[str] = None, placeholder: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → FieldTextArea

Tags

Categories

Usage:

```
page.ui.fields.textarea(label="Date")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldTextArea`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component. Default empty string
- **label** – Optional. The text of label to be added to the component
- **placeholder** – Optional. The text to be displayed when the input is empty
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

time(*value: Optional[str] = None, label: Optional[str] = None, icon: str = 'clock', color: Optional[str] = None, html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*) → TimePicker

This component is based on the JQuery Time Picker object.

Tags

Categories

Usage:

```
page.ui.fields.time(label="timestamp", color="red", helper="This is the report_
↪timestamp")
page.ui.fields.time(label="Time field")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlDates.TimePicker`

Related Pages:

<https://github.com/jonthornton/jquery-timepicker>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to the time component. Default now
- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. The component icon content from font-awesome references
- **color** – Optional. The font color in the component. Default inherit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component
- **helper** – Optional. A tooltip helper

today(*label: Optional[str] = None, icon: str = 'calendar', color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*) → DatePicker

This component is based on the JQuery Date Picker object.

Tags

Categories

Usage:

```
page.ui.fields.today(label="Date").included_dates(["2019-09-01", "2019-09-06"])
page.ui.fields.today()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlDates.DatePicker`

Related Pages:

<https://jqueryui.com/datepicker/>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/dates.py>

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Categories

Inputs, Texts

Tags

Dates

Parameters

- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **color** – Optional. The font color in the component. Default inherit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component
- **helper** – Optional. A tooltip helper

toggle(*record=None*, *label: Optional[str] = None*, *is_on: bool = False*, *color: Optional[str] = None*, *width: Union[tuple, int, str] = (100, '%')*, *height: Union[tuple, int, str] = (20, 'px')*, *html_code: Optional[str] = None*, *options: Optional[Union[bool, dict]] = None*, *profile: Optional[Union[bool, dict]] = None*)
→ Div

Add a toggle component.

Tags

Categories

Usage:

```
page.ui.buttons.toggle({'on': "true", 'off': 'false'})
```

Underlying HTML Objects:

- `epyk.core.html.HtmlRadio.Switch`

Related Pages:

<http://thecodeplayer.com/walkthrough/pure-css-on-off-toggle-switch>
mburnette/pen/LxNxNg

<https://codepen.io/>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/button.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/switch.py>

Parameters

- **record** – Optional. The list of dictionaries with the data
- **label** – Optional. The toggle static label displayed
- **is_on** – Optional.
- **color** – Optional. String. Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

weeks(*value=None, label: Optional[str] = None, icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → FieldSelect

Tags

Categories

Usage:

```
page.ui.fields.select(["a", "b"], label="Check")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldSelect`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to this component
- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. The component icon content from font-awesome references
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

years(*value=None, label: Optional[str] = None, icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → FieldSelect

tags

categories

Usage:

```
page.ui.fields.select(["a", "b"], label="Check")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.FieldSelect`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

- **param value**
Optional. The value to be displayed to this component
- param label**
Optional. The text of label to be added to the component
- param icon**
Optional. The component icon content from font-awesome references
- param width**
Optional. A tuple with the integer for the component width and its unit
- param height**
Optional. A tuple with the integer for the component height and its unit
- param html_code**
Optional. An identifier for this component (on both Python and Javascript side)
- param helper**
Optional. A tooltip helper
- param options**
Optional. Specific Python options available for this component
- param profile**
Optional. A flag to set the component performance storage

Timelines Interface

class `epyk.interfaces.components.CompFields.Timelines(ui)`

categories(*value=None, label: Optional[str] = None, icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → FieldSelect

Tags

Categories

Usage:

```

:param value:
:param label: Optional.
:param icon: Optional.
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side).
:param helper: Optional.
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage

```

issues(records=None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = ('auto', ''), options: Optional[dict] = None, html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None) → Items

Tags

Categories

Usage:

```

:param records:
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param options: Optional. Specific Python options available for this component
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
:param profile: Optional. A flag to set the component performance storage
:param helper:

```

meeting(time, icon: Optional[str] = None, width: Union[tuple, int, str] = (25, 'px'), height: Union[tuple, int, str] = (25, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None) → IconEdit

Tags

Categories

Usage:

```

:param time:
:param icon: Optional.
:param width: Optional. A tuple with the integer for the component width and
↳ its unit.
:param height: Optional. A tuple with the integer for the component height and
↳ its unit.
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side).
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.

```

milestone(completion_date: Union[datetime, str], icon: Optional[str] = None, width: Union[tuple, int, str] = (25, 'px'), height: Union[tuple, int, str] = (25, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None) → IconEdit

Tags
Categories

Usage:

```
:param completion_date:
:param icon: Optional.
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param html_code: Optional.
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

period(*start_date: Union[datetime, str]*, *days: int*, *width: Union[tuple, int, str] = (100, '%')*, *height: Union[tuple, int, str] = (None, 'px')*, *options: Optional[Union[bool, dict]] = None*, *profile: Optional[Union[bool, dict]] = None*) → Div

Tags
Categories

Usage:

```
:param start_date:
:param days:
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

view(*start_date: Union[datetime, str]*, *end_date: Union[datetime, str]*, *width: Union[tuple, int, str] = (100, '%')*, *height: Union[tuple, int, str] = (None, 'px')*, *options: Optional[Union[bool, dict]] = None*, *profile: Optional[Union[bool, dict]] = None*) → Div

Tags
Categories

Usage:

```
:param start_date:
:param end_date:
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

week(*start_date: Union[datetime, str]*, *width: Union[tuple, int, str] = (100, '%')*, *height: Union[tuple, int, str] = (None, 'px')*, *options: Optional[Union[bool, dict]] = None*, *profile: Optional[Union[bool, dict]] = None*) → Div

Tags Categories

Usage:

```
:param start_date:
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

workload(*value*, *width*: Union[tuple, int, str] = (25, 'px'), *html_code*: Optional[str] = None, *options*: Optional[Union[bool, dict]] = None, *profile*: Optional[Union[bool, dict]] = None) → Div

Tags Categories

Usage:

```
:param value: The workload percentage.
:param width: Optional. A tuple with the integer for the component width and
↳ its unit.
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side).
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.
```

Forms Interface

class epyk.interfaces.components.CompForms.**Forms**(*ui*)

date(*html_code*: str = 'Current', *profile*: Optional[Union[bool, dict]] = None, *options*: Optional[Union[bool, dict]] = None, *helper*: Optional[str] = None)

Create a DatePicker object.

tags
categories

Usage:

```
page.ui.forms.date("http://127.0.0.1:5000", "POST")
```

Underlying HTML Objects:

- epyk.core.html.HtmlContainer.Form
- epyk.core.html.HtmlContainer.Col
- epyk.core.html.HtmlDates.DatePicker
- **param html_code**
Optional. An identifier for this component (on both Python and Javascript side)
- param helper**
Optional. A tooltip helper

param profile

Optional. A flag to set the component performance storage

param options

Optional. Specific Python options available for this component

dates(*html_code: str, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*) → Form

Create two DatePicker objects for current and previous.

tags**categories**

Usage:

```
page.ui.forms.dates("http://127.0.0.1:5000", "POST")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Form`
- `epyk.core.html.HtmlContainer.Col`
- `epyk.core.html.HtmlDates.DatePicker`
- **param html_code**
An identifier for the prefix of the date components (on both Python and Javascript side)

param profile

Optional. A flag to set the component performance storage

param options

Optional. Specific Python options available for this component

param helper

Optional. A tooltip helper

input(*html_code: str, value: str = "", label: Optional[str] = None, placeholder: str = "", icon: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*) → Form

tags**categories**

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Form`
- `epyk.core.html.HtmlInput.FieldInput`
- **param html_code**
An identifier for this component (on both Python and Javascript side)
- **param value**
Optional. The value to be displayed to this component. Default empty
- **param label**
Optional. The text of label to be added to the component

param placeholder

Optional. The text to be displayed when the input is empty

param icon

Optional. The component icon content from font-awesome references

param profile

Optional. A flag to set the component performance storage

param options

Optional. Specific Python options available for this component

param helper

Optional. A tooltip helper

inputs(*record: List[dict], helper: Optional[str] = None, html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False*) → Form

tags**categories**

Usage:

```
page.ui.forms.inputs([
    {"label": "name", "htmlCode": "input"},
    {"label": "name 2", "htmlCode": "input2"},
])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Form`
- `epyk.core.html.HtmlContainer.Col`
- `epyk.core.html.HtmlInput.FieldInput`
- **param record**
The Python list of dictionaries
- param html_code**
Optional. An identifier for this component (on both Python and Javascript side)
- param helper**
Optional. A tooltip helper
- param options**
Optional. Specific Python options available for this component
- param profile**
Optional. A flag to set the component performance storage

new(*components: Optional[List[Html]] = None, helper: Optional[str] = None*) → Form

Creates a new empty form.

tags**categories**

Usage:

```
f = page.ui.form()
```

- **param helper**
Optional. A tooltip helper

param components

Optional. The different HTML objects to be added to the component

subscribe(*value: str = "", placeholder: str = 'Enter email address', button: Union[Html, str] = 'Subscribe', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False*) → Div

tags**categories**

Usage:

- **param value**
Optional. The value to be displayed to this component. Default empty
- param placeholder**
Optional. The text to be displayed when the input is empty
- param button**
Optional. The button component
- param width**
Optional. A tuple with the integer for the component width and its unit
- param height**
Optional. A tuple with the integer for the component height and its unit
- param options**
Optional. Specific Python options available for this component
- param profile**
Optional. A flag to set the component performance storage

Icons Interface

```
class epyk.interfaces.components.CompIcons.Icons(ui)
```

avatar(*img: str = "", name: Optional[str] = None, width: Union[tuple, int, str] = (30, 'px'), height: Union[tuple, int, str] = (None, ""), options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Display an avatar component.

Usage:

```
:param img: The image full path
:param name: Optional. The tooltip name
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

awesome(*icon: str, text: Optional[str] = None, tooltip: Optional[str] = None, position: Optional[str] = None, width: Union[tuple, int, str] = (25, 'px'), height: Union[tuple, int, str] = (25, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None, align: str = 'left', size: Union[tuple, int, str] = (None, 'px')*)

Generic function to create icon components. Default icon library used is font awesome but this might change depending on the web framework used.

Usage:

```
page.ui.icons.awesome(icon="fas fa-align-center")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/banners.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/icons.py>

Parameters

- **icon** – Optional. The font awesome icon reference
- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

badge(*text: str = "", icon: Optional[str] = None, width: Union[tuple, int, str] = (25, 'px'), height: Union[tuple, int, str] = (25, 'px'), background_color: Optional[str] = None, color: Optional[str] = None, url: Optional[str] = None, tooltip: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Display a badge component using Bootstrap

Usage:

```
page.ui.images.badge("Test badge", "Label", icon="fas fa-align-center")
page.ui.images.badge("This is a badge", background_color="red", color="white")
page.ui.images.badge(12, icon="far fa-bell", options={"badge_position": 'right'})
↪

b = rptObj.ui.images.badge(
    7688, icon="fab fa-python", options={'badge_css': {'color': 'white',
```

(continues on next page)

(continued from previous page)

```
↪ "background": 'red'}})
b.options.badge_css = {"background": 'green'}
```

Underlying HTML Objects:

- `epyk.core.html.HtmlImage.Badge`

Related Pages:

<https://getbootstrap.com/docs/4.0/components/badge/>

Parameters

- **text** – Optional. The content of the badge
- **icon** – Optional. A String with the icon to display from font-awesome
- **background_color** – Optional. The background color of the badge
- **color** – Optional. The text color of the badge
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **url** – Optional
- **tooltip** – Optional. The text to display in the tooltip
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

bar(*records=None, color: Optional[str] = None, width: Union[tuple, int, str] = (70, 'px'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Add a bespoke options / actions bar with icons

Usage:

Related Pages:

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/chips.py>

Parameters

- **records** –
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

capture(*text=None, position=None, tooltip='Save to clipboard', width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Usage:

```
page.ui.icons.capture()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

clear(*text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Same as `epyk.interfaces.components.CompIcons.Icons.awesome()` with a fas fa-times-circle icon

Usage:

```
page.ui.icons.clear()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **align** –
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component

- **profile** – Optional. A flag to set the component performance storage
- **size** – Optional. A tuple with the integer for the component size and its unit

clock(*text: Optional[str] = None, position: Optional[str] = None, tooltip: str = 'Last Updated Time', width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options=None, profile: Optional[Union[bool, dict]] = None, align: str = 'left', size: Union[tuple, int, str] = (None, 'px')*)

Usage:

```
page.ui.icons.clock()
page.ui.icons.clock().color("red")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/icons.py>

Parameters

- **text** – Optional. The text to be displayed to this component. Default `None`
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

danger(*text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Usage:

```
:param text: Optional. The text to be displayed to this component. Default None
:param position: Optional. The position of the icon in the line (left, right,
↳ center)
:param tooltip: Optional. A string with the value of the tooltip
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

(continues on next page)

(continued from previous page)

```
:param align: Optional. A string with the horizontal position of the component
:param size: Optional. A tuple with the integer for the component size and its
↪ unit
```

```
date(value: Optional[str] = None, label: Optional[str] = None, icon: str = 'calendar', color: Optional[str] =
None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = (None, 'px'),
html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options:
Optional[Union[bool, dict]] = None, helper: Optional[str] = None)
```

This component is based on the JQuery Date Picker object.

Usage:

```
page.ui.fields.date('2020-04-08', label="Date").included_dates(["2020-04-08",
↪ "2019-09-06"])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlDates.DatePicker`

Related Pages:

<https://jqueryui.com/datepicker/>

Parameters

- **value** – Optional. The value to be displayed to the time component. Default now
- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. The component icon content from font-awesome references
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component
- **helper** – Optional. A tooltip helper

```
delete(text=None, position=None, tooltip='Delete Component on the page', width=(None, 'px'),
height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None,
'px'))
```

Usage:

```
page.ui.icons.delete()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – Optional. The text to be displayed to this component. Default None

- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **align** – Optional. The text-align property within this component
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **size** – Optional. A tuple with the integer for the component size and its unit

download(*text=None, position=None, tooltip='Download', width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Usage:

```
page.ui.icons.download()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

edit(*text: Optional[str] = None, position: Optional[str] = None, tooltip: str = 'Edit', width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None, align: str = 'left', size: Union[tuple, int, str] = (None, 'px')*)

Usage:

```
page.ui.icons.edit()
page.ui.icons.edit().color("red")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/icons.py>

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

epyk(align: str = 'center', size: Union[tuple, int, str] = (32, 'px'))

Add the Epyk Icon.

Usage:

```
page.ui.icons.epyk()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlImage.Image`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/image.py>

Parameters

- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional.

error(text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

Usage:

```
:param text: Optional. The text to be displayed to this component. Default None
:param position: Optional. The position of the icon in the line (left, right, center)
:param tooltip: Optional. A string with the value of the tooltip
:param width: Optional. A tuple with the integer for the component width and its unit
:param height: Optional. A tuple with the integer for the component height and its unit
:param html_code: Optional. An identifier for this component (on both Python and Javascript side)
```

(continues on next page)

(continued from previous page)

```

↪and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
:param align: Optional. A string with the horizontal position of the component
:param size: Optional. A tuple with the integer for the component size and its
↪unit

```

```

excel(text=None, position=None, tooltip='Convert to Excel', width=(None, 'px'), height=(None, 'px'),
        html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

```

Usage:

```
page.ui.icons.excel()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center...)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

```

facebook(text=None, url='https://en-gb.facebook.com/', position=None, tooltip='Facebook', width=(25,
        'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

```

Usage:

```
page.ui.icons.facebook()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** –
- **url** –
- **position** –
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

fixed(*icon: Optional[str] = None, family: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, height: Union[tuple, int, str] = (None, 'px'), color: Optional[str] = None, tooltip: Optional[str] = None, align: str = 'left', options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Parameters

- **icon** – Optional. The icon value
- **family** – Optional. The icon family
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **color** – Optional.
- **tooltip** – Optional. A string with the value of the tooltip
- **align** – Optional. A string with the horizontal position of the component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

fluent(*icon: str, text: Optional[str] = None, tooltip: Optional[str] = None, position: Optional[str] = None, width: Union[tuple, int, str] = (25, 'px'), height: Union[tuple, int, str] = (25, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
page.ui.icons.awesome(icon="fas fa-align-center")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

ms-Icon ms-Icon-AdminDLogoInverse32

Parameters

- **icon** – Optional. The FluentUI icon reference
- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

gallery(*icons=None, columns=6, width=(None, '%'), height=('auto', ''), options=None, profile=None*)

Mosaic of pictures.

Tags

Categories

Usage:

Related Pages:

Underlying HTML Objects:

Templates:

Parameters

- **icons** – List. Optional. The list with the pictures
- **columns** – Integer. Optional. The number of column for the mosaic component
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit
- **options** – Dictionary. Optional. Specific Python options available for this component
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage

github(*text=None, url='https://github.com/epykure/epyk-ui', position=None, tooltip='Go the the Github project', width=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Link to a GitHub repository.

By default this icon button will redirect to the Epyk UI repository.

Usage:

```
page.ui.icons.github()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – optional. The text on the icon
- **url** – Optional. The url link
- **position** –
- **tooltip** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component

- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

hamburger(width=(15, 'px'), height=(2, 'px'), color=None, options=None, profile=None)

https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_menu_icon_js

Parameters

- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit
- **color** – String. Optional. The font color in the component. Default inherit
- **options** – Dictionary. Optional. Specific Python options available for this component
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage

info(text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

Usage:

```
:param text: Optional. The text to be displayed to this component. Default None
:param position: Optional. The position of the icon in the line (left, right, ↵
↵center)
:param tooltip: Optional. A string with the value of the tooltip
:param width: Optional. A tuple with the integer for the component width and ↵
↵its unit
:param height: Optional. A tuple with the integer for the component height and ↵
↵its unit
:param html_code: Optional. An identifier for this component (on both Python ↵
↵and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
:param align: Optional. A string with the horizontal position of the component
:param size: Optional. A tuple with the integer for the component size and its ↵
↵unit
```

instagram(text=None, url='https://www.instagram.com/?hl=en', position=None, tooltip='Twitter', width=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

Usage:

```
page.ui.icons.twitter()
```

Underlying HTML Objects:

- epyk.core.html.HtmlButton.IconEdit

Parameters

- **text** –
- **url** –
- **position** –

- **tooltip** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

large(*icon=None, family=None, width=(None, 'px'), height=(None, 'px'), html_code=None, color=None, tooltip=None, align='left', options=None, profile=None*)

Usage:

```
:param icon:
:param family:
:param width: Tuple. Optional. A tuple with the integer for the component width,
↳ and its unit
:param height: Tuple. Optional. A tuple with the integer for the component,
↳ height and its unit
:param html_code: String. Optional. An identifier for this component (on both,
↳ Python and Javascript side)
:param color:
:param tooltip:
:param align: String. Optional. The text-align property within this component
:param options: Dictionary. Optional. Specific Python options available for,
↳ this component
:param profile: Boolean | Dictionary. Optional. A flag to set the component,
↳ performance storage
```

linkedIn(*text=None, url='https://www.linkedin.com/in/epykure-python-58278a1b8/', position=None, tooltip="", width=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Create a LinkedIn icon button which will by default point to the Epykure account. Epykure account is the official account for the development of this framework.

Usage:

```
page.ui.icons.linkedIn()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – Optional. The text for the Icon
- **url** – Optional. The url when clicked
- **position** –
- **tooltip** – Optional. The tooltip when the mouse is hover
- **width** – Optional. A tuple with the integer for the component width and its unit

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

mail(*text=None, url="", position=None, tooltip='Share by mail', width=(25, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Same as [epyk.interfaces.components.CompIcons.Icons.awesome\(\)](#) with a fab fa-stack-overflow icon

Usage:

```
page.ui.icons.mail()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** –
- **url** –
- **position** –
- **tooltip** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

menu(*data, width=(100, '%'), height=(None, 'px'), options=None, profile=False, align: str = 'left', size=(None, 'px')*)

Add a menu bar with multiple icons.

Usage:

```
menu = page.ui.icons.menu([
    {"icon": "fab fa-github-square", "tooltip": "Github path", 'url': 'test'},
    {"icon": "far fa-eye"},
    {"icon": "fas fa-file-code"}
])
```

Parameters

- **data** – List. The icons definition.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit
- **size** – Tuple. Optional. A tuple with the integer for the icon size and its unit
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit
- **align** – String. Optional. A string with the horizontal position of the component
- **options** – Dictionary. Optional. Specific Python options available for this component
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage

messenger(*text=None, url='https://en-gb.facebook.com/', position=None, tooltip='Facebook', width=(25, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))*)

Usage:

```
page.ui.icons.facebook()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – Optional. The text to be displayed to this component. Default `None`
- **url** –
- **position** –
- **tooltip** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

next(*text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))*)

Usage:

```
:param text: Optional. The text to be displayed to this component. Default None
:param position: Optional. The position of the icon in the line (left, right, ↵
↵center)
:param tooltip: Optional. A string with the value of the tooltip
:param width: Optional. A tuple with the integer for the component width and ↵
↵its unit
:param height: Optional. A tuple with the integer for the component height and ↵
↵its unit
:param html_code: Optional. An identifier for this component (on both Python ↵
```

(continues on next page)

(continued from previous page)

```

↪and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
:param align: Optional. A string with the horizontal position of the component
:param size: Optional. A tuple with the integer for the component size and its_
↪unit

```

pdf(text=None, position=None, tooltip='Convert to PDF', width=(None, 'px'), height=(None, 'px'),
html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

Usage:

```
page.ui.icons.pdf(tooltip="helper")
```

Underlying HTML Objects:

- epyk.core.html.HtmlButton.IconEdit

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

play(text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None,
options=None, profile=None, align: str = 'left', size=(None, 'px'))

Shortcut to the play icon.

Usage:

```
page.ui.icons.play()
```

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

plus(text=None, position=None, tooltip='Add line', width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

Usage:

```
page.ui.icons.plus()
```

Underlying HTML Objects:

```
- :class:`epyk.core.html.HtmlButton.IconEdit`
```

```
:param text: Optional. The text to be displayed to this component. Default None
:param position: Optional. The position of the icon in the line (left, right, ↵
↵center)
:param tooltip: Optional. A string with the value of the tooltip
:param width: Optional. A tuple with the integer for the component width and ↵
↵its unit
:param height: Optional. A tuple with the integer for the component height and ↵
↵its unit
:param html_code: Optional. An identifier for this component (on both Python ↵
↵and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
:param align: Optional. A string with the horizontal position of the component
:param size: Optional. A tuple with the integer for the component size and its ↵
↵unit
```

previous(text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

Usage:

```
:param text: Optional. The text to be displayed to this component. Default None
:param position: Optional. The position of the icon in the line (left, right, ↵
↵center)
:param tooltip: Optional. A string with the value of the tooltip
:param width: Optional. A tuple with the integer for the component width and ↵
↵its unit
:param height: Optional. A tuple with the integer for the component height and ↵
↵its unit
:param html_code: Optional. An identifier for this component (on both Python ↵
↵and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
:param align: Optional. A string with the horizontal position of the component
```

(continues on next page)

(continued from previous page)

```
:param size: Optional. A tuple with the integer for the component size and its unit
```

```
python(text=None, url='https://pypi.org/', position=None, tooltip="", width=(25, 'px'), html_code=None,
        options=None, profile=None, align: str = 'left', size=(None, 'px'))
```

Usage:

```
page.ui.icons.python()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** –
- **url** –
- **position** –
- **tooltip** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

```
refresh(text=None, position=None, tooltip='Refresh Component', width=(None, 'px'), height=(None, 'px'),
        html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))
```

Usage:

```
page.ui.icons.refresh()
page.ui.icons.refresh().color("red")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

remove(*text=None, position=None, tooltip='Remove Item', width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Usage:

```
page.ui.icons.remove()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

rss(*text='RSS', position=None, tooltip="", width=('auto', ''), height=(25, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Usage:

```
page.ui.icons.rss()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **align** – Optional. A string with the horizontal position of the component
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **size** – Optional. A tuple with the integer for the component size and its unit

save(text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

Usage:

```
:param text: Optional. The text to be displayed to this component. Default None
:param position: Optional. The position of the icon in the line (left, right, ↵
↵center)
:param tooltip: Optional. A string with the value of the tooltip
:param width: Optional. A tuple with the integer for the component width and ↵
↵its unit
:param height: Optional. A tuple with the integer for the component height and ↵
↵its unit
:param html_code: Optional. An identifier for this component (on both Python ↵
↵and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
:param align: Optional. A string with the horizontal position of the component
:param size: Optional. A tuple with the integer for the component size and its ↵
↵unit
```

signin(text: str, width: Union[tuple, int, str] = (40, 'px'), icon: Optional[str] = None, colored: bool = True)

Usage:

```
page.ui.icons.signin("test")
```

Underlying HTML Objects:

- :class:`epyk.core.html.HtmlEvent.SignIn

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/icons.py>

Parameters

- **text** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **icon** – Optional. The component icon content from font-awesome references
- **colored** – Optional.

stackoverflow(text=None, url='https://stackoverflow.com/', position=None, tooltip='Share your comments', width=(25, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

Usage:

```
page.ui.icons.stackoverflow()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** –
- **url** –
- **position** –
- **tooltip** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

stop(*text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Shortcut to the stop icon.

Usage:

`page.ui.icons.stop()`

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

success(*text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Usage:

```
:param text: Optional. The text to be displayed to this component. Default None
:param position: Optional. The position of the icon in the line (left, right, center)
:param tooltip: Optional. A string with the value of the tooltip
:param width: Optional. A tuple with the integer for the component width and
    its unit
:param height: Optional. A tuple with the integer for the component height and
    its unit
:param html_code: Optional. An identifier for this component (on both Python and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
:param align: Optional. A string with the horizontal position of the component
:param size: Optional. A tuple with the integer for the component size and its
    unit
```

```
table(text=None, position=None, tooltip='Convert to Table', width=(None, 'px'), height=(None, 'px'),
      html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))
```

Usage:

```
page.ui.icons.table(tooltip="helper")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – Optional. The text to be displayed to this component. Default `None`
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

```
tick(flag=True, text=None, icons=('fa fa-check', 'fa fa-times'), position=None, tooltip="", width=(None, 'px'),
      html_code=None, options=None, profile=None)
```

Display a tick box component

Usage:

```
page.ui.icons.tick()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlRadio.Tick`

Parameters

- **flag** – Optional. The state for the tick component
- **text** – optional. The text for this component. Default none
- **icons** – Optional. The two icons to use for the component state
- **position** – Optional. A string with the vertical position of the component
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

timer(time, js_funcs, icon='clock', width=(15, 'px'), height=(15, 'px'), options=None, profile=None, align: str = 'left', size=(None, 'px'))

Usage:

```
:param time: Integer. Interval time in second
:param js_funcs: String | List. The Javascript functions
:param icon: String. The font awesome icon reference
:param width: Tuple. Optional. A tuple with the integer for the component width
↳and its unit
:param size: Tuple. Optional. A tuple with the integer for the icon size and
↳its unit
:param height: Tuple. Optional. A tuple with the integer for the component
↳height and its unit
:param options: Dictionary. Optional. Specific Python options available for
↳this component
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳performance storage
:param align: A string with the horizontal position of the component
```

property toggles

More custom toggles icons.

twitch(text=None, url='https://www.twitch.tv/epykure1', position=None, tooltip="", width=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

Usage:

```
page.ui.icons.twitter()
```

Underlying HTML Objects:

- epyk.core.html.HtmlButton.IconEdit

Parameters

- **text** –
- **url** –
- **position** –

- **tooltip** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

twitter(*text=None, url='https://twitter.com/Epykure1', position=None, tooltip="", width=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Usage:

```
page.ui.icons.twitter()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** –
- **url** –
- **position** –
- **tooltip** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

warning(*text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px')*)

Usage:

```
:param text: Optional. The text to be displayed to this component. Default None
:param position: Optional. The position of the icon in the line (left, right, ↵
↵center)
:param tooltip: Optional. A string with the value of the tooltip
:param width: Optional. A tuple with the integer for the component width and ↵
↵its unit
:param height: Optional. A tuple with the integer for the component height and ↵
↵its unit
:param html_code: Optional. An identifier for this component (on both Python ↵
```

(continues on next page)

(continued from previous page)

```

↪and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
:param align: Optional. A string with the horizontal position of the component
:param size: Optional. A tuple with the integer for the component size and its
↪unit

```

```

wrench(text=None, position=None, tooltip='Processing Time', width=(None, 'px'), height=(None, 'px'),
       html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

```

Usage:

```
page.ui.icons.wrench()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

```

youtube(text=None, url='https://www.youtube.com/', position=None, tooltip='Follow us on Youtube',
        width=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None,
        'px'))

```

Usage:

```
page.ui.icons.youtube()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.IconEdit`

Parameters

- **text** –
- **url** –
- **position** –
- **tooltip** –

- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Dictionary. Optional. Specific Python options available for this component
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

zoom(text=None, position=None, tooltip='Zoom on Component', width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

Usage:

```
page.ui.icons.zoom()
```

Underlying HTML Objects:

- epyk.core.html.HtmlButton.IconEdit

Parameters

- **text** – Optional. The text to be displayed to this component. Default None
- **position** – Optional. The position of the icon in the line (left, right, center)
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **align** – Optional. A string with the horizontal position of the component
- **size** – Optional. A tuple with the integer for the component size and its unit

zoom_in(text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None, options=None, profile=None, align: str = 'left', size=(None, 'px'))

Usage:

```
:param text: Optional. The text to be displayed to this component. Default None
:param position: Optional. The position of the icon in the line (left, right,
↳center)
:param tooltip: Optional. A string with the value of the tooltip
:param width: Optional. A tuple with the integer for the component width and
↳its unit
:param height: Optional. A tuple with the integer for the component height and
↳its unit
:param html_code: Optional. An identifier for this component (on both Python
↳and Javascript side)
```

(continues on next page)

(continued from previous page)

```
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
:param align: Optional. A string with the horizontal position of the component
:param size: Optional. A tuple with the integer for the component size and its_
↪unit
```

```
zoom_out(text=None, position=None, tooltip="", width=(None, 'px'), height=(None, 'px'), html_code=None,
          options=None, profile=None, align: str = 'left', size=(None, 'px'))
```

Usage:

```
:param text: Optional. The text to be displayed to this component. Default None
:param position: Optional. The position of the icon in the line (left, right,↪
↪center)
:param tooltip: Optional. A string with the value of the tooltip
:param width: Optional. A tuple with the integer for the component width and↪
↪its unit
:param height: Optional. A tuple with the integer for the component height and↪
↪its unit
:param html_code: Optional. An identifier for this component (on both Python_↪
↪and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
:param align: Optional. A string with the horizontal position of the component
:param size: Optional. A tuple with the integer for the component size and its_↪
↪unit
```

Toggles Interface

```
class epyk.interfaces.components.CompIcons.Toggles(ui)
```

```
collapse(icon_on='compress', icon_off='fas fa-expand', family=None, width=(None, 'px'), html_code=None,
          height=(None, 'px'), color=None, tooltip=None, align='left', options=None, profile=None)
```

Usage:

```
page.ui.images.icon("fab fa-angellist")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlImage.Icon`

Related Pages:

<https://fontawesome.com/icons?m=free>

Parameters

- **icon_on** – Optional. The component icon content from font-awesome references
- **icon_off** – Optional. The component icon content from font-awesome references
- **family** – Optional. The Icon library reference
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **color** – Optional. The font color in the component. Default inherit
- **tooltip** – Optional. A string with the value of the tooltip
- **align** – Optional. The text-align property within this component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

lock(*icon_on='lock_open', icon_off='fas fa-lock', family=None, width=(None, 'px'), html_code=None, height=(None, 'px'), color=None, tooltip=None, align='left', options=None, profile=None*)

Add a lock toggle button.

Usage:

```
page.ui.icons.toggles.lock()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlImage.Icon`

Related Pages:

<https://fontawesome.com/icons?m=free>

Parameters

- **icon_on** – Optional. The component icon content from font-awesome references
- **icon_off** – Optional. The component icon content from font-awesome references
- **family** – Optional. The Icon library reference
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **color** – Optional. The font color in the component. Default inherit
- **tooltip** – Optional. A string with the value of the tooltip
- **align** – Optional. The text-align property within this component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

Images Interface

class epyk.interfaces.components.CompImages.**Images**(ui)

animated(image: str = "", text: str = "", title: str = "", url: Optional[str] = None, path: Optional[str] = None, width: Union[tuple, int, str] = (200, 'px'), height: Union[tuple, int, str] = (200, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None) → AnimatedImage

Advance image with mask and gallery link. This will display some details when the mouse is on the container.

Tags

Categories

Usage:

```
c = page.ui.images.animated("epykIcon.PNG", text="This is a comment", title=
↪ "Title", url="#")
c.style.css.borders()
```

Underlying HTML Objects:

- epyk.core.html.HtmlImage.AnimatedImage

Related Pages:

<https://tympanus.net/Tutorials/OriginalHoverEffects/>

Parameters

- **image** – Optional. The image file name
- **text** – Optional. String. The image file path
- **title** – Optional. The image title displayed in to the mask on mouse hover the container
- **url** – Optional. The link to the gallery
- **path** – Optional. String. The image file path
- **width** – Optional. Tuple. The component width in pixel or percentage
- **height** – Optional. Tuple. The component height in pixel or percentage
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

avatar(text: str = "", image: Optional[str] = None, path: Optional[str] = None, status: Optional[str] = None, width: Union[tuple, int, str] = (30, 'px'), height: Union[tuple, int, str] = (30, 'px'), align: str = 'center', html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, menu: Optional[Html] = None, options: Optional[Union[bool, dict]] = None) → Div

Generate or load an avatar.

Tags

Categories

Usage:

```
page.ui.images.avatar("Epyk", status='out')
page.ui.images.avatar(image="epykIcon.PNG", path=config.IMG_PATH, status=False)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Div`
- `epyk.core.html.HtmlImage.Image`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/image.py>

Parameters

- **text** – Optional. The value to be displayed to the component
- **image** – Optional. The url of the image
- **path** – Optional. String. The image file path
- **status** – Optional. The avatar status code. Default no status
- **width** – Optional. Tuple. The component width in pixel or percentage
- **height** – Optional. Tuple. The component height in pixel or percentage
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

background(*url: str, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (300, 'px'), size: str = 'cover', margin: int = 0, align: str = 'center', html_code: Optional[str] = None, position: str = 'middle', profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*) → Div

Add a background image.

Tags

Categories

Usage:

```
:param url: Optional. The link to the gallery
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param size: Optional. The type of background in
:param margin: Optional. The CSS margin properties are used to create space
↳ around elements, outside of any defined
```

borders

Parameters

- **align** – Optional. A string with the horizontal position of the component

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **position** – Optional. A string with the vertical position of the component
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

badge(text: str = "", label: Optional[str] = None, icon: Optional[str] = None, width: Union[tuple, int, str] = (25, 'px'), height: Union[tuple, int, str] = (25, 'px'), background_color: Optional[str] = None, color: Optional[str] = None, url: Optional[str] = None, tooltip: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None) → Badge

Display a badge component using Bootstrap.

Tags

Categories

Usage:

```
page.ui.images.badge("Test badge", "Label", icon="fas fa-align-center")
page.ui.images.badge("This is a badge", background_color="red", color="white")
page.ui.images.badge(12, icon="far fa-bell", options={"badge_position": 'right'})
↪)

b = page.ui.images.badge(
    7688, icon="fab fa-python", options={'badge_css': {'color': 'white',
↪ "background": 'red'}})
b.options.badge_css = {"background": 'green'}
```

Underlying HTML Objects:

- `epyk.core.html.HtmlImage.Badge`

Related Pages:

<https://getbootstrap.com/docs/4.0/components/badge/>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/image.py>

<https://github.com/epykure/epyk-templates/blob/master/locals/components/links.py>

Parameters

- **text** – Optional. The content of the badge
- **label** – Optional. The label to display close to the badge
- **icon** – Optional. A String with the icon to display from font-awesome
- **background_color** – Optional. The background color of the badge
- **color** – Optional. The text color of the badge
- **url** – Optional. The underlying url link for the badge
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **tooltip** – String. Optional, The text to display in the tooltip
- **options** – Optional. Specific Python options available for this component

- **profile** – Optional. A boolean to store the performances for each components

carousel(*images: Optional[list] = None, path: Optional[str] = None, selected: int = 0, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (300, 'px'), options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → **ImgCarousel**

Carousel component for pictures.

Tags

Categories

Usage:

```
car = page.ui.images.carousel(["epykIcon.PNG", "epyklogo.ico", "epyklogo_whole_
↪big.png"],
                             path=r"../../static/images", height=(200, 'px'))
car.click([page.js.console.log('data', skip_data_convert=True)])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlImage.ImgCarousel`

Related Pages:

<https://www.cssscript.com/basic-pure-css-slideshow-carousel/>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/image.py>

Parameters

- **images** – Optional. With the different picture file names
- **path** – Optional. The common path for the pictures
- **width** – Optional. Tuple. The component width in pixel or percentage
- **height** – Optional. Tuple. The component height in pixel
- **selected** – Optional. The selected item index
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

circular(*image: Optional[str] = None, path: Optional[str] = None, width: Union[tuple, int, str] = (200, 'px'), height: Union[tuple, int, str] = (200, 'px'), align: str = 'center', html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*) → **Image**

Add an HTML image to the page. The path can be defined either in a absolute or relative format.

Tip: The absolute format does not work on servers. It is recommended to use relative starting to the root of the server.

Tags

Categories

Usage:

```
page.ui.circular("epykIcon.PNG", path=r"../../static/images", height=(50, "px
↪"))
```

Underlying HTML Objects:

- `epyk.core.html.HtmlImage.Image`

Related Pages:

https://www.w3schools.com/bootstrap/bootstrap_ref_css_images.asp https://www.w3schools.com/cssref/css3_pr_border-radius.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/image.py>

Parameters

- **image** – Optional. The image file name
- **path** – Optional. String. The image file path
- **width** – Optional. Tuple. The component width in pixel or percentage
- **height** – Optional. Tuple. The component height in pixel or percentage
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

color(*code: str, color: Optional[str] = None, width: Union[tuple, int, str] = (110, 'px'), height: Union[tuple, int, str] = (25, 'px'), options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Simple vignette to display a color with it is code.

TODO: Return the hex code of the color when dom.content used

Tags

Categories

Usage:

```
page.ui.images.color("FFFFFF")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Div`

Parameters

- **code** – The color code
- **color** – Optional. The font color
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage

container(*components: List[Html], max_width: Union[tuple, int, str] = (900, 'px'), align: str = 'center', profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*) → Div

Empty container for images.

Tags

Categories

Usage:

```
:param components: List of HTML Component. internal components
:param max_width: Optional. The maximum width for this container
:param align: Optional. A string with the horizontal position of the component
:param profile: Optional. A flag to set the component performance storage
:param options: Optional. Specific Python options available for this component
```

emoji(*symbol: Optional[str] = None, top: Union[tuple, int, str] = (20, 'px'), options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Emoji

Tags

Categories

Usage:

```
page.ui.images.emoji(page.symbols.smileys.DISAPPOINTED_FACE)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlImage.Emoji`

Related Pages:

<https://github.com/wedgies/jquery-emoji-picker>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/image.py>

Parameters

- **symbol** – Optional. The emoji code
- **top** – Optional. The number of pixel from the top of the page
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

epyk(*align: str = 'center', width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = ('auto', ''), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, tooltip: Optional[str] = None, options: Optional[Union[bool, dict]] = None*)

Add the Epyk Icon.

Usage:

```
page.ui.icons.epyk()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlImage.Image`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/image.py>

Parameters

- **align** – Optional. A string with the horizontal position of the component.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **tooltip** – Optional. A string with the value of the tooltip
- **options** – Optional. Specific Python options available for this component

figure(*image: Optional[str] = None, caption: Optional[str] = None, path: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'center', html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, tooltip: Optional[str] = None, options: Optional[Union[bool, dict]] = None*) → Figure

Display a picture as a figure component with an attached caption object.

Tags

Categories

Related Pages:

https://www.w3schools.com/tags/tag_figcaption.asp

Usage:

```
page.ui.images.figure("33c33735-8a1e-4bef-8201-155b4775304a.jpg", "test caption  
→",  
path=picture_path, width=(100, 'px'))
```

Parameters

- **image** – Optional. The url path of the image
- **caption** – Optional.
- **path** – Optional. The image file path
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **tooltip** – Optional. A string with the value of the tooltip
- **options** – Optional. Specific Python options available for this component

gallery(*images: Optional[List[Union[dict, Html]]] = None, columns: int = 6, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = ('auto', ''), options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Grid

Mosaic of pictures.

Tags

Categories

Usage:

Related Pages:

Underlying HTML Objects:

Templates:

Parameters

- **images** – Optional. The list with the pictures
- **columns** – Optional. The number of column for the mosaic component
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

icon(*icon: Optional[str] = None, family: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, height: Union[tuple, int, str] = (None, 'px'), color: Optional[str] = None, tooltip: Optional[str] = None, align: str = 'left', options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Icon

Add an icon to the page.

Tags

Categories

Usage:

```
page.ui.images.icon("fab fa-angellist")
icon = page.ui.icon("fab fa-python")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlImage.Icon`

Related Pages:

<https://fontawesome.com/icons?m=free>

Parameters

- **icon** – Optional. The component icon content from font-awesome references
- **family** – Optional. The Icon framework reference
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **color** – Optional. The font color in the component. Default inherit

- **tooltip** – Optional. A string with the value of the tooltip
- **align** – Optional. The text-align property within this component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

img(*image: Optional[str] = None, path: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'center', html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, tooltip: Optional[str] = None, options: Optional[Union[bool, dict]] = None*) → Image

Add an HTML image to the page. The path can be defined either in a absolute or relative format.

Tip: The absolute format does not work on servers. It is recommended to use relative starting to the root of the server.

Tags

Categories

Usage:

```
page.ui.img("epykIcon.PNG", path=r"../../static/images", height=(50, "px"))
```

Underlying HTML Objects:

- `epyk.core.html.HtmlImage.Image`

Related Pages:

https://www.w3schools.com/bootstrap/bootstrap_ref_css_images.asp https://www.w3schools.com/cssref/css3_pr_border-radius.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/image.py>

Parameters

- **image** – Optional. The image file name
- **path** – Optional. Optional. The image file path
- **width** – Optional. Optional. The component width in pixel or percentage
- **height** – Optional. Optional. The component height in pixel or percentage
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **tooltip** – Optional. A string with the value of the tooltip
- **options** – Optional. Specific Python options available for this component

logo(*url: str, width: Union[tuple, int, str] = (160, 'px'), height: Union[tuple, int, str] = (60, 'px'), top: Union[tuple, int, str] = (16, 'px'), left: Union[tuple, int, str] = (16, 'px'), profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*) → Div

Tags

Categories

Usage:

```
:param url: Optional. The link to the gallery
:param width: Optional. The component width in pixel or percentage
:param height: Optional. The component height in pixel or percentage
:param top: Optional. A tuple with the integer for the component's distance to,
↳the top of the page
:param left: Optional. A tuple with the integer for the component's distance to,
↳the left of the page
:param profile: Optional. A flag to set the component performance storage
:param options: Optional. Specific Python options available for this component
```

section(*image: str, name: str, title: str, text: str, url: Optional[str] = None, path: Optional[str] = None, width: Union[tuple, int, str] = (200, 'px'), height: Union[tuple, int, str] = (200, 'px'), profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*) → Div

Tags

Categories

Usage:

```
page.ui.images.section("epykIcon.PNG", "# Test", "Epyk Test", 'This is a test',
↳path=r"../../static/images")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Div`
- `epyk.core.html.HtmlText.Span`
- `epyk.core.html.HtmlText.Paragraph`
- `epyk.core.html.HtmlText.Title`
- `epyk.core.html.HtmlImage.Image`

Parameters

- **image** – The url of the image
- **name** – The name of the image
- **title** – Optional. A panel title. This will be attached to the title property
- **text** – Optional. The value to be displayed to the component
- **url** – Optional. The link to the gallery
- **path** – Optional. The image file path
- **width** – Optional. The component width in pixel or percentage
- **height** – Optional. The component height in pixel or percentage
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

wallpaper(*url: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (100, '%'), size: str = 'cover', margin: int = 0, align: str = 'center', html_code: Optional[str] = None, position: str = 'middle', profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*) → Background

Tags

Background

Categories

Usage:

```
:param url: Optional. The link to the gallery
:param width: Optional. Tuple. The component width in pixel or percentage
:param height: Optional. Tuple. The component height in pixel or percentage
:param size: Optional. The type of background in
:param margin: Optional. The CSS margin properties are used to create space.
↪around elements,
```

outside of any defined borders

Parameters

- **align** – Optional. The text-align property within this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **position** – Optional. The position compared to the main component tag
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

youtube(*video_id: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'center', html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*) → Image

Get teh picture used by youtube.

Tags**Categories**

Usage:

```
:param video_id: Optional. The youtube video ID
:param width: Optional. The component width in pixel or percentage
:param height: Optional. The component height in pixel or percentage
:param align: Optional. A string with the horizontal position of the component
:param html_code: Optional. An identifier for this component (on both Python.
↪and Javascript side)
:param profile: Optional. A flag to set the component performance storage
:param options: Optional. Specific Python options available for this component
```


Inputs Interface

class epyk.interfaces.components.CompInputs.**Inputs**(*ui*)

autocomplete(*text: str = "", placeholder: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → AutoComplete

Enables users to quickly find and select from a pre-populated list of values as they type, leveraging searching and filtering.

Usage:

```
page.ui.inputs.autocomplete("Test")
```

Underlying HTML Objects:

- epyk.core.html.HtmlInput.AutoComplete

Related Pages:

<https://jqueryui.com/autocomplete/>

Parameters

- **text** – Optional. The value to be displayed to the component
- **placeholder** – Optional. Text visible when the input component is empty
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** – Optional. Specific HTML tags to be added to the component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

cell(*text: str = "", language: str = 'python', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (60, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Cell

Usage:

```
page.ui.inputs.cell()
```

Underlying HTML Objects:

- epyk.core.html.HtmlTextEditor.Cell

Parameters

- **text** – Optional. The value to be displayed to the componen
- **language** –
- **width** – Optional. A tuple with the integer for the component width and its unit

- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

checkbox(*flag: bool, label: str = "", group_name: Optional[str] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, tooltip: str = "", profile: Optional[Union[bool, dict]] = None*) → InputCheckbox

Usage:

```
page.ui.inputs.checkbox(False)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.Checkbox`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/checkbox.py>

Parameters

- **flag** –
- **label** – Optional. The text of label to be added to the component
- **group_name** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** –
- **tooltip** – Optional. A string with the value of the tooltip.
- **profile** – Optional. A flag to set the component performance storage

d_date(*text: str, placeholder: str = "", width: Union[tuple, int, str] = (140, 'px'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → InputDate

Usage:

```
date = page.ui.inputs.d_date()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.InputDate`

Parameters

- **text** – Optional. The value to be displayed to the component

- **placeholder** – Optional. Text visible when the input component is empty
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** – Optional. Specific HTML tags to be added to the component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

d_int(value: str = "", placeholder: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None) → InputInteger

Usage:

```
date = page.ui.inputs.d_int()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.InputInteger`

Parameters

- **value** – Optional. The value of this input number field
- **placeholder** – Optional. Text visible when the input component is empty
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** – Optional. Specific HTML tags to be added to the component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

d_radio(flag: bool = False, group_name: Optional[str] = None, placeholder: str = "", tooltip: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None) → InputRadio

Add a radio component.

Usage:

```
page.ui.inputs.d_radio()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.InputRadio`

Parameters

- **flag** – Optional. The component init value
- **group_name** – Optional. The radio group name
- **placeholder** – Optional. Text visible when the input component is empty
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** – Optional. Specific HTML tags to be added to the component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

d_range(value, min_val: float = 0, max_val: float = 100, step: float = 1, placeholder: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), tooltip: Optional[str] = None, html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None) → InputRange

Usage:

```
:param value: Optional. The value of the component
:param min_val: Optional. The minimum value
:param max_val: Optional. The maximum value
:param step: Optional. The step when the handle is moved
:param placeholder: Optional. Text visible when the input component is empty
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
:param options: Optional. Specific Python options available for this component
:param attrs: Optional. Specific HTML tags to be added to the component
:param tooltip: Optional. A string with the value of the tooltip
:param profile: Optional. A flag to set the component performance storage
```

d_search(text: str = "", placeholder: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None) → Input

One of the new types of inputs in HTML5 is search

Usage:

```
page.ui.inputs.d_search("")
```

Underlying HTML Objects:

- epyk.core.html.HtmlInput.Input

Related Pages:

<https://developer.mozilla.org/fr/docs/Web/HTML/Element/Input/search> <https://css-tricks.com/webkit-html5-search-inputs/>

Parameters

- **text** – Optional. The value to be displayed to the component
- **placeholder** – Optional. Text visible when the input component is empty
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** – Optional. Specific HTML tags to be added to the component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

d_text(*text: str = "", placeholder: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), tooltip: Optional[str] = None, html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Input

Usage:

```
page.ui.inputs.d_text()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.Input`

Parameters

- **text** – Optional. The value to be displayed to the component
- **placeholder** – Optional. Text visible when the input component is empty
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** – Optional. Specific HTML tags to be added to the component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

d_time(*text: str = "", placeholder: str = "", width: Union[tuple, int, str] = (139, 'px'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → InputTime

A lightweight, customizable javascript timepicker plugin for jQuery inspired by Google Calendar.

Usage:

```
date = page.ui.inputs.d_time()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.InputTime`

Parameters

- **text** – Optional. The value to be displayed to the component
- **placeholder** – Optional. Text visible when the input component is empty
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** – Optional. Specific HTML tags to be added to the component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

editor(*text: str = "", language: str = 'python', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (300, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Editor

Usage:

```
page.ui.inputs.editor()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTextEditor.Editor`

Parameters

- **text** – Optional. The value to be displayed to the componen
- **language** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

file(*text: str = "", placeholder: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → InputFile

Input field that will hide characters typed in

Usage:

```
page.ui.inputs.file()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.InputFile`

Parameters

- **text** – Optional. The value to be displayed to the component
- **placeholder** – Optional. Text visible when the input component is empty
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** – Optional. Specific HTML tags to be added to the component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

filters (*items: Optional[List[Html]] = None, button: Optional[Html] = None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (60, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, autocomplete: bool = False, profile: Optional[Union[bool, dict]] = None*) → Div

Parameters

- **items** –
- **button** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –
- **helper** –
- **autocomplete** –
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

hidden (*text: str = "", placeholder: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → Input

Add a hidden input component to the page. This could be used to store data to then be passed to underlying services,

Usage:

```
rptObj.ui.inputs.hidden("Test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.Input`

Parameters

- **text** – Optional. The value to be displayed to the component
- **placeholder** – Optional. Text visible when the input component is empty
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** – Optional. Specific HTML tags to be added to the component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

input(*text: str = "", placeholder: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → Input

Add a standard input component.

Usage:

```
page.ui.inputs.input("Test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.Input`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py>
[//github.com/epykure/epyk-templates/blob/master/locals/components/modal.py](https://github.com/epykure/epyk-templates/blob/master/locals/components/modal.py)
[//github.com/epykure/epyk-templates/blob/master/locals/components/popup_info.py](https://github.com/epykure/epyk-templates/blob/master/locals/components/popup_info.py)

[https:](https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py)
[https:](https://github.com/epykure/epyk-templates/blob/master/locals/components/modal.py)

Parameters

- **text** – Optional. The value to be displayed to the component
- **placeholder** – Optional. Text visible when the input component is empty
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** – Optional. Specific HTML tags to be added to the component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

label(*label: str, text: str = "", placeholder: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, tooltip: Optional[str] = None, options: Optional[dict] = None, attrs: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Add an input label component.

Usage:

```
page.ui.inputs.label()
page.ui.inputs.label("test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.Input`
- `epyk.core.html.HtmlText.Label`
- `epyk.core.html.HtmlContainer.Div`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/links.py>

Parameters

- **label** –
- **text** –
- **placeholder** –
- **width** –
- **height** –
- **html_code** –
- **tooltip** – Optional. A string with the value of the tooltip.
- **options** –
- **profile** –
- **attrs** –

left(*text: str = "", placeholder: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → Input

Add a standard input component.

Usage:

```
page.ui.inputs.left("Test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.Input`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/modal.py>
https://github.com/epykure/epyk-templates/blob/master/locals/components/popup_info.py

[https:](#)
[https:](#)

Parameters

- **text** – Optional. The value to be displayed to the component
- **placeholder** – Optional. Text visible when the input component is empty
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** – Optional. Specific HTML tags to be added to the component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

password(text: str = "", placeholder: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, attrs: Optional[dict] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None) → Input

Input field that will hide characters typed in.

Usage:

```
page.ui.inputs.password(placeholder="Password")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.Input`

Parameters

- **text** – Optional. The value to be displayed to the component
- **placeholder** – Optional. Text visible when the input component is empty
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **attrs** – Optional. Specific HTML tags to be added to the component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

radio(flag: bool, label: Optional[str] = None, group_name: Optional[str] = None, icon: Optional[str] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None) → Radio

Usage:

```
page.ui.radio(['Single', 'Multiple'], html_code="type")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.Radio`

Related Pages:

https://www.w3schools.com/tags/att_input_type_radio.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/radio.py>

Parameters

- **flag** –
- **label** –
- **group_name** –
- **icon** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** –
- **tooltip** – Optional. A string with the value of the tooltip
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

search(*text: str = "", placeholder: str = 'Search..', align: str = 'left', color: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, tooltip: Optional[str] = None, extensible: bool = False, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Search

Add an input search component.

Usage:

```
page.ui.inputs.search()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTextEditor.Cell`

Related Pages:

https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_anim_search

Templates:

https://github.com/epykure/epyk-templates/blob/master/locals/components/list_filter.py

Parameters

- **text** – Optional. The value to be displayed to the componen
- **placeholder** –
- **align** – Optional. The text-align property within this component

- **color** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **extensible** –
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

textarea(*text: str = "", width: Union[tuple, int, str] = (100, '%'), rows: int = 5, placeholder: Optional[str] = None, background_color: Optional[str] = None, html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, tooltip: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → TextArea

Add textarea component.

Usage:

```
page.ui.inputs.textarea("Test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.TextArea`

Related Pages:

https://www.w3schools.com/tags/tag_textarea.asp

Parameters

- **text** – Optional. The value to be displayed to the component
- **width** – Optional. A tuple with the integer for the component width and its unit
- **rows** –
- **placeholder** – Optional. Text visible when the input component is empty
- **background_color** –
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage

Delimiter Interface

class epyk.interfaces.components.CompLayouts.**Delimiter**(*ui*)

dashed(*count: int = 1, width: Union[tuple, int, str] = (100, '%'), align: str = 'center', options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Wrapper around the HT html tag.

Tags

Categories

Usage:

Templates:

Parameters

- **count** – Optional. The number of HR tag to be added.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **align** – Optional. The content position. Values (left, right, center). Default center.
- **options** – Optional. Dictionary. Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

dotted(*count: int = 1, width: Union[tuple, int, str] = (100, '%'), align: str = 'center', options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Wrapper around the HT html tag.

Tags

Categories

Usage:

Templates:

Parameters

- **count** – Optional. The number of HR tag to be added.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **align** – Optional. The content position. Values (left, right, center). Default center.
- **options** – Optional. Dictionary. Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

double(*count: int = 1, width: Union[tuple, int, str] = (100, '%'), align: str = 'center', options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Wrapper around the HT html tag.

Tags

Categories

Usage:

Templates:

Parameters

- **count** – Optional. The number of HR tag to be added.

- **width** – Optional. A tuple with the integer for the component width and its unit.
- **align** – Optional. The content position. Values (left, right, center). Default center.
- **options** – Optional. Dictionary. Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

line(*count: int = 1, width: Union[tuple, int, str] = (100, '%'), align: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Wrapper around the HT html tag.

Tags

Categories

Usage:

Templates:

Parameters

- **count** – Optional. The number of HR tag to be added.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **align** – Optional. The content position. Values (left, right, center). Default center.
- **options** – Optional. Dictionary. Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

Layouts Interface

class epyk.interfaces.components.CompLayouts.**Layouts**(*ui*)

accentuate(*width: Union[tuple, int, str] = (10, '%'), height: Union[tuple, int, str] = (1, 'px'), align: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Add a styles hr component to lightly underline another component.

Tags

Categories

Usage:

`page.ui.layouts.accentuate()`

Templates:

Parameters

- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **align** – Optional. The content position. Values (left, right, center). Default center.
- **options** – Optional. Dictionary. Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

br(*count: int = 1, profile: Optional[Union[bool, dict]] = None*) → Newline

Wrapper around the Br html tag.

The
 tag inserts a single line break.

Tags

Categories

Usage:

```
page.ui.layouts.new_line(10)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlOthers.Newline`

Related Pages:

https://www.w3schools.com/tags/tag_br.asp

Parameters

- **count** – Integer. Optional. The number of empty line to put. Default 1.
- **profile** – Boolean | Dictionary. Optional. Activate the profiler.

centered(*components: Optional[List[Html]] = None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Tags

Categories

Usage:

Templates:

Parameters

- **components** – The different HTML objects to be added to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

col(*components: Optional[List[Html]] = None, position: str = 'middle', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), align: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Col

Python wrapper for a column of HTML elements from Bootstrap.

This component is a container and it is used to display multiple Ares components in column. You can first add a component in the data list then add the + operator to add more.

Tags

Categories

Usage:

```
page.ui.layouts.col([
    page.ui.text("test C"),
    page.ui.text("test D"),
])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Col`

Related Pages:

<https://getbootstrap.com/docs/4.0/layout/grid/1493-css3-flexbox-layout-module.html>

<https://www.alsacreations.com/tuto/lire/>

Templates:

Parameters

- **components** – The different HTML objects to be added to the component.
- **position** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **align** – Optional. A string with the horizontal position of the component.
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

columns(*components: List[Html], cols, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), align: Optional[str] = None, position: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Grid

Tags

Categories

Usage:

Templates:

Parameters

- **components** – List. The different HTML objects to be added to the component.
- **cols** –
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **align** – String. Optional. A string with the horizontal position of the component.
- **position** – String. Optional. A string with the vertical position of the component.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean or Dictionary. Optional. A flag to set the component performance storage.

dialogs(*text: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (20, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Dialog

Simple JQuery UI modal with a text.

Tags

Categories

Underlying HTML Objects:

- `epyk.core.html.HtmlEvent.Dialog`

Related Pages:

<https://jqueryui.com/dialog/>

Usage:

```
:param text: Optional. The value to be displayed to the component.
:param width: Optional. A tuple with the integer for the component width and
↳ its unit.
:param height: Optional. A tuple with the integer for the component height and
↳ its unit.
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side).
:param helper: Optional. A tooltip helper.
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.
```

div(*components: Optional[Union[Html, List[Html]]] = None, label: Optional[str] = None, color: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), icon: Optional[str] = None, height: Union[tuple, int, str] = (None, 'px'), editable: bool = False, align: str = 'left', padding: Optional[int] = None, html_code: Optional[str] = None, tag: str = 'div', helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None, position: Optional[Union[bool, dict]] = None*) → Div

Tags

Categories

Usage:

```
div = page.ui.div([html])
div += html_2
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Div`

Related Pages:

https://www.w3schools.com/tags/tag_div.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/bars.py>

Parameters

- **components** – The different HTML objects to be added to the component.
- **label** –

- **color** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **icon** –
- **position** – Optional.
- **editable** –
- **align** – Optional. A string with the horizontal position of the component
- **padding** –
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tag** –
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **helper** – Optional.

form(*components: Optional[List[Html]] = None, helper: Optional[str] = None*) → Form

Tags

Categories

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Form`

Templates:

Parameters

- **components** – The different HTML objects to be added to the component.
- **helper** – Optional. A tooltip helper.

grid(*rows=None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), align: Optional[str] = None, position: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Grid

Python wrapper to the HTML Bootstrap Grid.

Tags

Categories

Usage:

```
gr = page.ui.layouts.grid()
gr += [page.ui.text("test %s" % i) for i in range(5)]
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Grid`

Related Pages:

<https://getbootstrap.com/docs/4.0/layout/grid/>

Templates:

Parameters

- **rows** –
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **align** – Optional. A string with the horizontal position of the component.
- **position** – Optional. A string with the vertical position of the component.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

header(*components: Optional[List[Html]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Header

The HTML <header> element represents introductory content, typically a group of introductory or navigational aids. It may contain some heading elements but also a logo, a search form, an author name, and other elements.

Tags

Categories

Usage:

```
div = page.ui.header([html])
div += html_2
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Header`

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/header>

Templates:

Parameters

- **components** – The different HTML objects to be added to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **helper** –
- **options** – Optional. Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

hr(*count: int = 1, background_color: Optional[str] = None, margins: int = 0, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), align: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Wrapper around the HT html tag.

The <hr> tag defines a thematic break in an HTML page (e.g. a shift of topic).

Tips: If `background_color` is `True`, the theme color will be used.

Tags

Categories

Usage:

```
page.ui.layouts.hr(10)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlOthers.Hr`

Related Pages:

https://www.w3schools.com/tags/tag_hr.asp

Templates:

Parameters

- **count** – Optional. The number of HR tag to be added.
- **background_color** – Optional. The component background color.
- **margins** – Optional. The margin top and bottom in pixels.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **align** – Optional. The content position. Values (left, right, center). Default center.
- **options** – Optional. Dictionary. Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

icons(*icon_names=None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → IconsMenu

Tags

Categories

Usage:

```
menu = page.ui.layouts.icons(["fas fa-bell", "fas fa-calendar-check"])
menu.icon.click([menu.icon.dom.css({"color": 'red'})])
menu[0].click([menu[0].dom.css({"color": 'red'})])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.IconsMenu`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/icons.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/image.py>

Parameters

- **icon_names** –
- **width** – Optional. A tuple with the integer for the component width and its unit.

- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **helper** – Optional. A tooltip helper.
- **profile** – Optional. A flag to set the component performance storage.

iframe(*url: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (100, '%'), helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → IFrame

Add a iframe component to the page.

Tags

Categories

Usage:

```
page.ui.layouts.iframe("http://www.google.com")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.IFrame`

Templates:

Parameters

- **url** – Optional. The link to the underlying page.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **helper** – Optional. A tooltip helper.
- **profile** – Optional. A flag to set the component performance storage.

inline(*components: Optional[List[Html]] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Tags

Categories

Usage:

Templates:

Parameters

- **components** – The different HTML objects to be added to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

new_line(*count*: int = 1, *profile*: Optional[Union[bool, dict]] = None) → Newline

Wrapper around the Br html tag.

The
 tag inserts a single line break.

Tags

Categories

Usage:

```
page.ui.layouts.new_line(10)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlOthers.Newline`

Related Pages:

https://www.w3schools.com/tags/tag_br.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/contextmenu.py>

<https://github.com/epykure/epyk-templates/blob/master/locals/components/links.py>

Parameters

- **count** – Integer. Optional. The number of empty line to put. Default 1.
- **profile** – Boolean | Dictionary. Optional. Activate the profiler.

panel(*components*: Optional[List[Html]] = None, *title*: Optional[str] = None, *color*: Optional[str] = None, *width*: Union[tuple, int, str] = (100, '%'), *height*: Union[tuple, int, str] = (None, 'px'), *html_code*: Optional[str] = None, *helper*: Optional[str] = None, *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = False) → Panel

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Panel`

Tags

Categories

Usage:

Templates:

Parameters

- **components** – The different HTML objects to be added to the component.
- **title** –
- **color** – Optional. The font color in the component. Default inherit.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.

- **profile** – Optional. A flag to set the component performance storage.

popup(*components: Optional[List[Html]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Popup

Add a generic popup component to the page.

Tags

Categories

Usage:

```
popup = page.ui.layouts.popup(page.ui.title('Test'), color="red")
popup.add(page.ui.texts.paragraph('Test'))
```

Underlying HTML Objects:

- `epyk.core.html.HtmlPopup.Popup`

Related Pages:

https://www.w3schools.com/tags/tag_div.asp

Templates:

Parameters

- **components** – The different HTML objects to be added to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

row(*components: Optional[List[Html]] = None, position: str = 'middle', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), align: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Row

Python wrapper for a column of HTML elements from Bootstrap.

This component is a container and it is used to display multiple Ares components in column. You can first add a component in the data list then add the + operator to add more.

Tags

Categories

Usage:

```
row = page.ui.layouts.row()
row += page.ui.layouts.col([
    page.ui.text("test A"),
    page.ui.text("test B"),
])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Row`

Templates:

Related Pages:

<https://getbootstrap.com/docs/4.0/layout/grid/1493-css3-flexbox-layout-module.html>

<https://www.alsacreations.com/tuto/lire/>

Parameters

- **components** – The different HTML objects to be added to the component.
- **position** – Optional. The CSS justify-content attribute
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **align** – Optional. A string with the horizontal position of the component.
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

section(*components: Optional[List[Html]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Section

The <section> tag defines sections in a document, such as chapters, headers, footers, or any other sections of the document.

Tags

Categories

Usage:

```
div = page.ui.header([html])
div += html_2
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Header`

Related Pages:

https://www.w3schools.com/tags/tag_section.asp

Templates:

Parameters

- **components** – The different HTML objects to be added to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

table(*components: Optional[List[Html]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Table

table layout for HTML components.

Tags

Categories

Usage:

```
row = page.ui.layouts.table()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Table`

Templates:

Parameters

- **components** – The different HTML objects to be added to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

underline(*width: Union[tuple, int, str] = (10, '%'), height: Union[tuple, int, str] = (3, 'px'), align: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Add a styles hr component to underline another component.

Tags

Categories

Usage:

```
page.ui.layouts.underline()
```

Templates:

Parameters

- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **align** – Optional. The content position. Values (left, right, center). Default center.
- **options** – Optional. Dictionary. Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

Links Interface

class epyk.interfaces.components.CompLinks.**Links**(*ui*)

button(*text: str = "", url: str = "", icon: Optional[str] = None, helper: Optional[str] = None, height: Union[tuple, int, str] = (None, 'px'), decoration: bool = False, html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Underlying HTML Objects:

- `epyk.core.html.HtmlLinks.ExternalLink`

Usage:

```
page.ui.links.button()
```

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/links.py>

Parameters

- **text** – Optional. The string value to be displayed in the component
- **url** – The destination page when clicked
- **icon** – Optional. The component icon content from font-awesome references
- **helper** – Optional. A tooltip helper
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit
- **decoration** –
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

colored(*text: str = "", url: str = "", icon: Optional[str] = None, helper: Optional[str] = None, color: Optional[str] = None, height: Union[tuple, int, str] = (None, 'px'), decoration: bool = False, html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Display a link with the same layout than a buttons.colored HTML component.

Usage:

```
page.ui.links
```

Parameters

- **text** – Optional. The string value to be displayed in the component.
- **url** – Optional. The string url of the link.
- **icon** – Optional. A string with the value of the icon to display from font-awesome.
- **helper** – Optional. A tooltip helper.
- **color** – Optional. The font color in the component. Default inherit.
- **height** – Optional. A tuple with the integer for the component height and its unit.

- **decoration** –
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. Optional. A flag to set the component performance storage.

data(*text: str, value, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), fmt: str = 'txt', options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Python interface to the Hyperlink to retrieve data.

Usage:

```
data_link = page.ui.links.data("link", "test#data")
data_link.build({"text": 'new link Name', 'data': "new content"})
```

Underlying HTML Objects:

- `epyk.core.html.HtmlLinks.DataLink`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/links.py>

Parameters

- **text** – The string value to be displayed in the component
- **value** – The value to be displayed to this component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **fmt** – Optional. The downloaded data format.
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage.

download(*url: str = '#', text: str = '', icon: str = 'download', helper: Optional[str] = None, height: Union[tuple, int, str] = (None, 'px'), decoration: bool = False, align: str = 'left', html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

HTML component to upload files.

Usage:

```
page.ui.links
```

Parameters

- **text** – Optional. The string value to be displayed in the component
- **url** – Optional. The string url of the link
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **helper** – Optional. A tooltip helper
- **height** – Optional. A tuple with the integer for the component height and its unit

- **decoration** –
- **align** – Optional. The text-align property within this component.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

external(*text: str, url: str, icon: Optional[str] = None, align: str = 'left', helper: Optional[str] = None, height: Union[tuple, int, str] = (None, 'px'), decoration=False, html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
page.ui.links.external('data', 'www.google.fr', icon="fas fa-align-center",  
↪options={"target": "_blank"})
```

Underlying HTML Objects:

- `epyk.core.html.HtmlLinks.ExternalLink`

Related Pages:

https://www.w3schools.com/TagS/att_a_href.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/links.py>

Parameters

- **text** – Optional. The string value to be displayed in the component
- **url** – Optional. The string url of the link
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **align** – Optional. The text-align property within this component.
- **helper** – Optional. A tooltip helper
- **height** – Optional. A tuple with the integer for the component height and its unit
- **decoration** –
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

link(*text: str = "", url: str = "", icon: Optional[str] = None, align: str = 'left', tooltip: Optional[str] = None, helper: Optional[str] = None, height: Union[tuple, int, str] = (None, 'px'), decoration: bool = False, html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Python interface to the common Hyperlink.

Usage:

```

page.ui.link({"text": "Profiling results", "url": '#'})
l = page.ui.links.link('data', 'www.google.fr', icon="fas fa-align-center",
options={"target": "_blank"})
b = page.ui.images.badge("new")
l.append_child(b)

```

Underlying HTML Objects:

- `epyk.core.html.HtmlLinks.ExternalLink`

Parameters

- **text** – Optional. The string value to be displayed in the component.
- **url** – Optional. The string url of the link.
- **align** – Optional. The text-align property within this component.
- **icon** – Optional. A string with the value of the icon to display from font-awesome.
- **tooltip** – Optional. The tooltip displayed when the mouse is on the component.
- **helper** – Optional. A tooltip helper.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **decoration** –
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

upload(url: str = '#', text: str = "", icon: str = 'upload', helper: Optional[str] = None, height: Union[tuple, int, str] = (None, 'px'), decoration: bool = False, align: str = 'left', html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

HTML component to upload files.

Usage:

```
page.ui.links
```

Parameters

- **text** – Optional. The string value to be displayed in the component
- **url** – Optional. The string url of the link
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **helper** – Optional. A tooltip helper
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit
- **decoration** –
- **align** – Optional. The text-align property within this component.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component

- **profile** – Optional. A flag to set the component performance storage

Lists Interface

class epyk.interfaces.components.CompLists.**Lists**(ui)

alpha(data=None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'),
html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool,
dict]] = None, helper: Optional[str] = None)

Usage:

```
page.ui.lists
```

Underlying HTML Objects:

- epyk.core.html.HtmlList.List

Parameters

- **data** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

badges(data=None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'),
html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None,
profile: Optional[Union[bool, dict]] = None)

Usage:

```
page.ui.lists.badges([{'label': 'Python', 'value': 12}, {'label': 'R', 'value': 3}])

i = page.ui.lists.badges([
    {"text": 'text', 'icon': 'fas fa-times', 'checked': True, 'value': 8},
    {"text": 'text', 'icon': 'fas fa-times', 'checked': True, 'value': 5},
], options={"badge": {"background": 'green'}})
i.click([page.js.console.log(i.dom.content)])
```

Underlying HTML Objects:

- epyk.core.html.HtmlList.Badges

Related Pages:

https://www.w3schools.com/bootstrap/bootstrap_list_groups.asp <https://v4-alpha.getbootstrap.com/components/list-group/>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py>

Parameters

- **data** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

box(*records: Optional[list] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, '%'), options: Optional[dict] = None, html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*)

Special list configuration for a list of box with a title with a text and a list of icons

Usage:

```
page.ui.lists
```

Templates:

https://github.com/epykure/epyk-templates/blob/master/locals/components/list_box.py

Parameters

- **records** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **helper** – Optional. A tooltip helper

brackets(*records=None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (550, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
minimalData = {
    "teams": [
        ["Team 1", "Team 2"],
        ["Team 3", "Team 4"]
    ],
    "results": [
        [[1, 2], [3, 4]],
        [[4, 6], [2, 1]]
    ]
}
```

(continues on next page)

(continued from previous page)

```
}
bt = page.ui.lists.brackets(minimalData)
```

Parameters

- **records** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

checks(*data=None*, *width: Union[tuple, int, str] = ('auto', '')*, *height: Union[tuple, int, str] = (None, 'px')*, *html_code: Optional[str] = None*, *helper: Optional[str] = None*, *options: Optional[dict] = None*, *profile: Optional[Union[bool, dict]] = None*)

Add a list component with checkbox items.

Usage:

```
data = [{"label": "python", "value": False}, {"label": "Java", "value": 5}]
checks = page.ui.lists.checklist(data)

ts = page.ui.lists.items(["menu %s" % i for i in range(10)])
its.options.checked_key = "selected"
#its.options.max_selected = 2
its.options.items_type = "check"
its.options.checked_key = "selected"
its.options.text_click = True
```

Underlying HTML Objects:

- `epyk.core.html.HtmlList.Checks`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/checkbox.py>

Parameters

- **data** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

chips(items=None, category: str = 'group', placeholder: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (60, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Add a chip (filter) html component

Usage:

```
chips = page.ui.chips([])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlEvent.Filters`

Related Pages:

https://www.w3schools.com/howto/howto_css_contact_chips.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/chips.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py>

Parameters

- **items** – Selected items
- **category** – Optional. The group of the items.
- **placeholder** – Optional. The input field placeholder
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

disc(data=None, width: Union[tuple, int, str] = ('auto', ""), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Usage:

```
page.ui.lists
```

Underlying HTML Objects:

- `epyk.core.html.HtmlList.List`

Related Pages:

https://www.w3schools.com/cssref/pr_list-style-type.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/paragraph.py>

Parameters

- **data** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

drop(*data=None, color=None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
cols_keys = page.ui.lists.drop(html_code="cols_agg_keys")
cols_keys.style.css.min_height = 20
cols_keys.items_style(style_type="bullets")
cols_keys.drop()
```

Parameters

- **data** –
- **color** –
- **width** –
- **height** –
- **html_code** –
- **helper** –
- **options** –
- **profile** –

dropdown(*records=None, text: str = "", width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Create a dropdown item.

Underlying HTML Objects:

- `epyk.core.html.HtmlTrees.DropDown`

Related Pages:

<http://getbootstrap.com/docs/4.0/components/dropdowns/> https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_js_dropdown_multilevel_css&stacked=h
<https://codepen.io/svnt/pen/beEgre>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/dropdown.py> <https://github.com/epykure/epyk-templates/blob/master/locals/components/tree.py>

Parameters

- **records** – Optional. The list of dictionaries with the input data.
- **text** – Optional. The value to be displayed to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

groups(*data=None, categories=None, color: Optional[str] = None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
l = page.ui.lists.groups(["AWW", "B"])
l.add_list(["D", "E"], category="Test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlList.Groups`

Related Pages:

<http://designbump.com/create-a-vertical-accordion-menu-using-css3-tutorial/>
[//thecodeplayer.com/walkthrough/vertical-accordion-menu-using-jquery-css3](http://thecodeplayer.com/walkthrough/vertical-accordion-menu-using-jquery-css3)

[http:](http://)

Parameters

- **data** –
- **categories** –
- **color** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

icons(*data=None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
page.ui.lists.badges([{'label': 'Python', 'value': 12}, {'label': 'R', 'value': 3}])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlList.Badges`

Related Pages:

https://www.w3schools.com/bootstrap/bootstrap_list_groups.asp <https://v4-alpha.getbootstrap.com/components/list-group/>

Parameters

- **data** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

item(*text: Optional[str] = None, tag: Optional[str] = None, options: Optional[dict] = None*) → Li

Add a dynamic and configurable list component.

Usage:

```
l = page.ui.lists.list(["A", "B"])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlList.List`

Related Pages:

https://www.w3schools.com/bootstrap/bootstrap_list_groups.asp <http://astronautweb.co/snippet/font-awesome/>

Parameters

- **text** –
- **tag** –
- **options** –

items(*records: Optional[list] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = ('auto', ''), options: Optional[dict] = None, html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*) → Items

Add a JavaScript based list component.

Usage:

```
c = page.ui.lists.items([
    {"text": 'value 1', 'icon': 'fas fa-times', 'checked': True, 'value': 8000},
    {"text": 'value 1', 'icon': 'fas fa-times', 'checked': True, 'value': 50000},
], options={"style": {"background": 'green', 'color': 'white'}})
```

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py> https://github.com/epykure/epyk-templates/blob/master/locals/components/list_custom.py
https://github.com/epykure/epyk-templates/blob/master/locals/components/list_filter.py

Parameters

- **records** – Optional. The list of dictionaries with the input data.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **helper** – Optional. A tooltip helper

list(*data=None, color: Optional[str] = None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Add a static list component.

Usage:

```
l = page.ui.lists.list(["A", "B"])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlList.List`

Related Pages:

https://www.w3schools.com/bootstrap/bootstrap_list_groups.asp <http://astronautweb.co/snippet/font-awesome/>

Templates:

https://github.com/epykure/epyk-templates/blob/master/locals/components/list_dragdrop.py

Parameters

- **data** – List. Optional. The list items.
- **color** – String. Optional. The font color in the component. Default inherit.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

- **helper** – String. Optional. A tooltip helper.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

lookup(*lookup=None, html_code: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, '%'), profile: Optional[Union[bool, dict]] = None, multiple: bool = False, options: Optional[dict] = None*)

HTML Select component.

Usage:

```
select1 = page.ui.select([
    {"value": "value 1", "text": "value 1"},
    {"value": "value 2", "text": "value 2"},
])
lookupData = {"value 1": [
    {"value": "A", 'text': "Example 1"},
    {"value": "B", 'text': "Example 2"}
]}
select2 = page.ui.lookup(lookupData)
select1.change([
    select2.build(select1.dom.content)
])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlSelect.Lookup`

Related Pages:

<https://silviomoreto.github.io/bootstrap-select/examples/>
[npmjs.com/package/bootstrap-select-v4](https://www.npmjs.com/package/bootstrap-select-v4)
[Bootstrap-4-Dropdown-Select-Plugin-jQuery.html](https://www.jqueryscript.net/form/Bootstrap-4-Dropdown-Select-Plugin-jQuery.html)

<https://www.jqueryscript.net/form/>

Parameters

- **lookup** – Dictionary. Optional. The mapping to the list of recs to be loaded.
- **html_code** – Optional. The component identifier code (for bot
- **width** – Tuple. Optional. Integer for the component width
- **height** – Tuple. Optional. Integer for the component height
- **profile** – Optional. A flag to set the component performance storage
- **multiple** – Boolean. To set if the component can handle multiple selections
- **options** – The select options as defined <https://developer.snapappointments.com/bootstrap-select/options/>

numbers(*data=None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*)

Usage:

```
page.ui.lists.numbers(["A", "B"])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlList.List`

Related Pages:

https://www.w3schools.com/html/html_lists.asp

<https://www.w3.org/wiki/CSS/Properties/list-style-type>

Parameters

- **data** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **helper** – Optional. A tooltip helper

pills(*records=None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, '%'), options: Optional[dict] = None, html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*)

Usage:

```
page.ui.lists
```

Parameters

- **records** –
- **width** –
- **height** –
- **options** –
- **html_code** –
- **profile** –
- **helper** –

points(*data=None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), align: Optional[str] = None, html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*)

Usage:

```
page.ui.lists
```

Underlying HTML Objects:

- `epyk.core.html.HtmlList.List`

Related Pages:

https://www.w3schools.com/html/html_lists.asp

Parameters

- **data** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** –
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

radios(*data=None, group_name: str = 'group', width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

`page.ui.lists`

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Div`
- `epyk.core.html.HtmlInput.Radio`

Parameters

- **data** –
- **group_name** –
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – String. Optional. A tooltip helper
- **options** – Dictionary. Optional. Specific Python options available for this component
- **profile** – Boolean or Dictionary. Optional. A flag to set the component performance storage

roman(*data=None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*)

Underlying HTML Objects:

- `epyk.core.html.HtmlList.List`

Usage:


```
page.ui.lists
```

Parameters

- **data** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

select(*records=None, html_code: Optional[str] = None, selected: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, '%'), profile: Optional[Union[bool, dict]] = None, multiple: bool = False, options: Optional[dict] = None*) → Select

HTML Select component.

Usage:

```
records = [
    {"text": 'Text 1', "value": "text 1"},
    {"text": 'Text 2', "value": "text 2"},
    {"text": 'Text 3', "value": "text 3"},
]

select = page.ui.select(records)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlSelect.Select`

Related Pages:

<https://silviomoreto.github.io/bootstrap-select/examples/>
[npmjs.com/package/bootstrap-select-v4](https://www.npmjs.com/package/bootstrap-select-v4)
[Bootstrap-4-Dropdown-Select-Plugin-jQuery.html](https://www.jqueryscript.net/form/Bootstrap-4-Dropdown-Select-Plugin-jQuery.html)

<https://www.jqueryscript.net/form/>

Parameters

- **records** – Optional. The list of dictionaries with the input data.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **selected** – Optional. The selected value or values.
- **width** – Optional. Integer for the component width.
- **height** – Optional. Integer for the component height.
- **profile** – Optional. A flag to set the component performance storage.
- **multiple** – Optional. To set if the component can handle multiple selections.

- **options** – The select options as defined <https://developer.snapappointments.com/bootstrap-select/options/>

squares(data=None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Usage:

```
page.ui.lists.squares(["A", "B"])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlList.List`

Related Pages:

https://www.w3schools.com/cssref/pr_list-style-type.asp

Parameters

- **data** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

tree(data=None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None) → Tree

Usage:

```
data = [{"label": 'test', 'items': [{"label": 'child 1', 'color': 'red'}]}]
page.ui.lists.tree(data)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTrees.Tree`

Parameters

- **data** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

Media Interface

class epyk.interfaces.components.CompMedia.**Media**(ui)

audio(value: str = "", path: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None)

Add a audio track from the server to the page. The format for the video must be mpeg.

Usage:

```
page.ui.media.video("CWWB3673.mpeg")
```

Underlying HTML Objects:

- epyk.core.html.HtmlMedia.Audio

Related Pages

https://www.w3schools.com/html/html5_video.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/>

Parameters

- **value** – Optional. The name of the audio object.
- **path** – Optional. The path to the audio object.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **profile** – Optional. A flag to set the component performance storage.
- **options** – Optional. Specific Python options available for this component.

camera(align: str = 'center', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None)

Add a video from the server to the page. The format for the video must be MP4.

Usage:

```
page.ui.media.camera()
```

Underlying HTML Objects:

- epyk.core.html.HtmlMedia.Camera

Related Pages

https://www.w3schools.com/html/html5_video.asp https://www.kirupa.com/html5/accessing_your_webcam_in_html5.htm

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/>

Parameters

- **align** – Optional. A string with the horizontal position of the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. The component identifier code (for both Python and Javascript).
- **profile** – Optional. A flag to set the component performance storage.
- **options** – Optional. Specific Python options available for this component.

video(*value: str = "", align: str = 'center', path: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

Add a video from the server to the page. The format for the video must be MP4.

Usage:

```
page.ui.media.video("CWWB3673.MP4")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlMedia.Media`

Related Pages:

https://www.w3schools.com/html/html5_video.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/>

Parameters

- **value** – Optional. The name of the video.
- **path** – Optional. The path to the video.
- **align** – Optional. A string with the horizontal position of the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. The component identifier code (for both Python and Javascript).
- **profile** – Optional. A flag to set the component performance storage.
- **options** – Optional. Specific Python options available for this component.

youtube(*link: str, align: str = 'center', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

This will add a youtube video using the shared line to embedded to a website.

Usage:

```
page.ui.media.youtube("https://www.youtube.com/embed/dfiHMtjh5Ac")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlMedia.Youtube`

Related Pages

https://www.w3schools.com/html/html5_video.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/>

Parameters

- **link** – The youtube link.
- **align** – Optional. A string with the horizontal position of the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. The component identifier code (for both Python and Javascript).
- **profile** – Optional. A flag to set the component performance storage.
- **options** – Optional. A dictionary with the components properties.

Menus Interface

class epyk.interfaces.components.CompMenus.**Menus**(*ui*)

bar(*data=None, align: str = 'left', position: str = 'top', color: Optional[str] = None, width: Union[tuple, int] = (350, 'px'), height: Union[tuple, int] = (None, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
page.ui.menus.bar([
    {"value": "File", "children": [
        {"url": "Test", "text": "Test"}
    ]}
])
```

Underlying HTML Objects:

- epyk.core.html.HtmlContainer.Div
- epyk.core.html.HtmlContainer.Col
- epyk.core.html.HtmlContainer.Grid
- epyk.core.html.HtmlText.Title
- epyk.core.html.HtmlList.List

Related Pages:

https://www.w3schools.com/bootstrap/bootstrap_list_groups.asp

<http://astronautweb.co/snippet/font-awesome/>

Parameters

- **data** –
- **align** – Optional. A string with the horizontal position of the component
- **position** – Optional. A string with the vertical position of the component

- `epyk.core.html.HtmlContainer.Div`
- `epyk.core.html.HtmlButton.Button`

Parameters

- **value** –
- **components** –
- **symbol** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

buttons(*data: Optional[list] = None, color: Optional[str] = None, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
bs = page.ui.buttons.buttons(["Button", "Button 2", "Button 3"])
bs[2].click([
    page.js.alert(bs[2].dom.content)
])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlButton.Buttons`

Parameters

- **data** –
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

contextual(*record: Optional[list] = None, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, visible: bool = False, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Set a bespoke Context Menu on an Item. This will create a popup on the page with action. This component is generic is need to be added to a component to work.

Usage:

```
menu = page.ui.contextual([{"text": 'text', 'event': 'alert("ok")'}])
page.ui.title("Test").attach_menu(menu)
```

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/contextmenu.py>

Parameters

- **record** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **visible** – Optional.
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

divisor(*data*, *divider*: *Optional[bool] = None*, *width*: *Union[tuple, int] = (100, '%')*, *height*: *Union[tuple, int] = (None, 'px')*, *options*: *Optional[dict] = None*, *profile*: *Union[bool, dict] = False*)

Add list of items separated by a symbol (default BLACK_RIGHT_POINTING_TRIANGLE). The components will be based on Links.

Usage:

```
record = []
page.ui.menus.divisor(record)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Div`
- `epyk.core.html.HtmlText.link`

Parameters

- **data** –
- **divider** – `symbols.shape | String`. The symbol between the links.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit
- **options** – Dictionary. Optional. Specific Python options available for this component
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage

icons(*data*: *Optional[list] = None*, *width*=(100, '%'), *height*: *Union[tuple, int] = (None, 'px')*, *align*: *str = 'center'*, *options*: *Optional[dict] = None*, *profile*: *Union[bool, dict] = False*)

Add a menu bar with font awesome icons.

Usage:


```
icons = page.ui.menus.icons([
    "bi-1-circle-fill",
    "bi-search-heart-fill",
    "bi-x-circle-fill",
], options={"icon_family": "bootstrap-icons"})
```

Parameters

- **data** – Optional. Parameter bar icons
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

images(*data: Optional[list] = None, path: Optional[str] = None, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (None, 'px'), align: str = 'center', options: Optional[dict] = None, profile: Union[bool, dict] = False*)

Usage:

```
page.ui.menus.images(["https://jupyter.org/favicon.ico", "https://codepen.io//
↪favicon.ico"])
```

Parameters

- **path** –
- **data** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. The text-align property within this component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

menu(*data: Optional[list] = None, color: Optional[str] = None, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
page.ui.menus.menu([
    {"value": "File", "children": [
        {"url": "Test", "text": "Test"}
    ]}
])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Div`
- `epyk.core.html.HtmlContainer.Col`

- `epyk.core.html.HtmlContainer.Grid`
- `epyk.core.html.HtmlText.Title`
- `epyk.core.html.HtmlList.List`

Related Pages:

https://www.w3schools.com/bootstrap/bootstrap_list_groups.asp
[snippet/font-awesome/](https://www.w3schools.com/bootstrap/bootstrap_list_groups.asp)

<http://astronautweb.co/>

Parameters

- **data** –
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

pills(*data: Optional[List] = None, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (50, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Union[bool, dict] = False*)

Usage:

```
page.ui.pills.
```

Parameters

- **data** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

right(*data=None, color: Optional[str] = None, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (30, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
page.ui.lists.
```

Parameters

- **data** –
- **color** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –
- **helper** –
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

selections(*data*, *width*=(150, 'px'), *height*=('auto', ''), *html_code*: *Optional[str]* = None, *helper*: *Optional[str]* = None, *options*: *Optional[dict]* = None, *profile*: *Optional[Union[bool, dict]]* = None)

Menu using JQuery UI external module.

Usage:

```
page.ui.menus.selections(["Item 1", "Item 2"])

page.ui.menus.selections([
    {'value': "fas fa-exclamation-triangle", 'items': [
        {"value": 'value 1'},
        {"value": 'value 2'},
        {"value": 'value 3'},
    ]},
    "fas fa-exclamation-triangle"])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlEvent.Menu`

Related Pages:

<https://jqueryui.com/menu/>

Parameters

- **data** –
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

toolbar(*data*: *Optional[list]* = None, *width*: *Union[tuple, int]* = ('auto', ''), *height*: *Union[tuple, int]* = (None, 'px'), *options*: *Optional[dict]* = None, *profile*: *Union[bool, dict]* = False)

Usage:

```

tb = page.ui.menus.toolbar(["fas fa-paint-brush", "fas fa-code"])
tb[1].link.val = 4589
tb[1].tooltip("This is a tooltip")
tb[0].style.css.color = 'red'

# with other icon families
page.ui.menus.toolbar(["face"], options={"icon_family": 'material-design-icons'})
page.ui.menus.toolbar(["Mail", "AdminALogo32"], options={"icon_family": 'office-
ui-fabric-core'})

```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Div`
- `epyk.core.html.HtmlImage.Badge`

Parameters

- **data** – Optional. The list of icons
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

top(*data: Optional[List[dict]] = None, color: Optional[str] = None, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (30, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Add a menu item at the top of the page. The menu will be fixed on the page, always visible

Usage:

```

page.ui.menus.top([{"value": "Menu 1", 'children': ["Item 1", "Item 2"]}, "Menu_
1 2"])

```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Div`
- `epyk.core.html.HtmlContainer.Col`
- `epyk.core.html.HtmlContainer.Grid`
- `epyk.core.html.HtmlText.Title`
- `epyk.core.html.HtmlList.List`

Related Pages:

https://www.w3schools.com/bootstrap/bootstrap_list_groups.asp
[snippet/font-awesome/](https://www.w3schools.com/bootstrap/bootstrap_list_groups.asp)

<http://astronautweb.co/>

Parameters

- **data** – Optional. The top menu values
- **color** – Optional. The font color in the component. Default inherit

- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

Modals Interface

class epyk.interfaces.components.CompModals.**Modals**(ui)

acknowledge(components: Optional[List[Html]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Display a popup with a ok button to validate the message has been displayed.

Usage:

```
popup = page.popup(page.ui.title('Test'), color="red")
popup + page.paragraph('Test')
```

Underlying HTML Objects:

- epyk.core.html.HtmlPopup.Popup

Related Pages:

https://www.w3schools.com/tags/tag_div.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/modals.py>

Parameters

- **components** – The different HTML objects to be added to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

dialog(text, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (20, 'px'), html_code: Optional[str] = None, helper: Optional[Union[HtmlModel, str]] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Simple JQuery UI modal with a text.

Underlying HTML Objects:

- epyk.core.html.HtmlEvent.Dialog

Related Pages:

<https://jqueryui.com/dialog/>

Usage:

```
:param text: Optional. The value to be displayed to the component.
:param width: Optional. A tuple with the integer for the component width and
↳ its unit.
:param height: Optional. A tuple with the integer for the component height and
↳ its unit.
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side).
:param helper: Optional. A tooltip helper.
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.
```

```
disclaimer(disc_list, header=None, footer=None, submit: bool = True, validation_text: str = 'AGREE',
            action: Optional[str] = None, add_buttons=None, helper: Optional[Union[HtmlModel, str]] =
            None)
```

Disclaimer that will appear as a modal.

Usage:

```
privacy_title = page.ui.texts.title('A privacy reminder', 2)
p1 = page.ui.texts.paragraph(''
    Scroll down and click "%s" when you're ready to continue, or explore other
↳ options on this page.
    '' % page.ui.tags.strong(''I agree'', options={'managed': False}))
disc = page.ui.modals.disclaimer([privacy_title, p1])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Modal`
- `epyk.core.html.HtmlContainer.Row`
- `epyk.core.html.HtmlButton.Button`

Parameters

- **disc_list** –
- **header** –
- **footer** –
- **submit** –
- **validation_text** –
- **action** –
- **add_buttons** –
- **helper** –

```
error(components: Optional[List[Html]] = None, width: Union[tuple, int, str] = (100, '%'), height:
      Union[tuple, int, str] = (None, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool,
      dict]] = None)
```

Display an error popup.

Usage:

```
popup = page.ui.popup(page.ui.title('Test'), color="red")
popup + page.ui.paragraph('Test')
```

Underlying HTML Objects:

- `epyk.core.html.HtmlPopup.Popup`

Related Pages:

https://www.w3schools.com/tags/tag_div.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/modals.py>

Parameters

- **components** – The different HTML objects to be added to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

forms(*components: Html, action: str, method: str, header=None, footer=None, helper: Optional[Union[HtmlModel, str]] = None*)

Simple interface to create an html form within a modal

Usage:

```
d = page.ui.fields.today('test')
i = page.ui.fields.input(placeholder='test2', label='test1')
i2 = page.ui.fields.input('test3', label='test2')
form_modal = page.ui.modals.forms([d, i, i2], "http://127.0.0.1:5000", "POST")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Modal`
- `epyk.core.html.HtmlContainer.Form`

Related Pages:

https://www.w3schools.com/w3css/w3css_modal.asp

Parameters

- **components** –
- **action** –
- **method** –
- **header** –
- **footer** –
- **helper** –

icon(*components: Optional[List[Html]] = None, icon: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Display a generic popup with an icon.

Usage:

```
popup = page.popup(page.ui.title('Test'), color="red")
popup + page.paragraph('Test')
```

Underlying HTML Objects:

- `epyk.core.html.HtmlPopup.Popup`

Related Pages:

https://www.w3schools.com/tags/tag_div.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/modals.py>

Parameters

- **components** – The different HTML objects to be added to the component.
- **icon** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

info(*components: Optional[List[Html]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Display an info popup.

Usage:

```
popup = page.ui.popup(page.ui.title('Test'), color="red")
popup + page.ui.paragraph('Test')
```

Underlying HTML Objects:

- `epyk.core.html.HtmlPopup.Popup`

Related Pages:

https://www.w3schools.com/tags/tag_div.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/modals.py>

Parameters

- **components** – The different HTML objects to be added to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.

- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

loading(*text: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Display a success popup.

Usage:

```
popup = page.ui.popup(page.ui.title('Test'), color="red")
popup + page.ui.paragraph('Test')
```

Underlying HTML Objects:

- `epyk.core.html.HtmlPopup.Popup`

Related Pages:

https://www.w3schools.com/tags/tag_div.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/modals.py>

Parameters

- **text** – Optional. The loading text.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

popup(*components: Optional[List[Html]] = None, title: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Display a generic popup.

Usage:

```
popup = page.ui.modals.popup(page.ui.title('Test'), color="red")
popup.add(page.ui.texts.paragraph('Test'))
```

Underlying HTML Objects:

- `epyk.core.html.HtmlPopup.Popup`

Related Pages:

https://www.w3schools.com/tags/tag_div.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/modals.py>

Parameters

- **components** – The different HTML objects to be added to the component.

- **title** –
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

stepper(*records=None, components: Optional[List[Html]] = None, shape: str = 'arrow', title: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Parameters

- **records** –
- **components** –
- **shape** –
- **title** –
- **width** –
- **height** –
- **options** –
- **profile** –

success(*components: Optional[List[Html]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Display a success popup.

Usage:

```
popup = page.ui.popup(page.ui.title('Test'), color="red")
popup + page.ui.paragraph('Test')
```

Underlying HTML Objects:

- `epyk.core.html.HtmlPopup.Popup`

Related Pages:

https://www.w3schools.com/tags/tag_div.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/modals.py>

Parameters

- **components** – The different HTML objects to be added to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

validation(*components: Optional[List[Html]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
popup = page.popup(page.ui.title('Test'), color="red")
popup + page.paragraph('Test')
```

Underlying HTML Objects:

- `epyk.core.html.HtmlPopup.Popup`

Related Pages:

https://www.w3schools.com/tags/tag_div.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/modals.py>

Parameters

- **components** – The different HTML objects to be added to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

Banners Interface

class `epyk.interfaces.components.CompNavigation.Banners(ui)`

bottom(*data: Union[str, list] = '', background=None, align='center', width=(100, '%'), height=(None, 'px'), options=None, profile=False*)

Tags

Categories

Usage:

```
# Add a banner with HTML content
icon = page.ui.icon("fab fa-python")
text = page.ui.text("This is a text")

# Chang the option to have the content in one line
bottom = page.ui.banners.bottom([icon, text], options={"inline": True})
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Div`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/banners.py>

Parameters

- **data** –
- **background** – String. Optional. Background color code.
- **align** – String. The text-align property within this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

```
contact_us(title='Contact Us', background=None, width=(100, '%'), align='left', height=(None, 'px'),  
            html_code='contactus', options=None, profile=False)
```

Tags

Categories

Usage:

```
:param title: String. Optional. A panel title. This will be attached to the  
→ title property.  
:param background: String. Optional. Background color code.  
:param align: String. Optional. A string with the horizontal position of the  
→ component.  
:param width: Tuple. Optional. A tuple with the integer for the component width  
→ and its unit.  
:param height: Tuple. Optional. A tuple with the integer for the component  
→ height and its unit.  
:param html_code: String. Optional. An identifier for this component (on both  
→ Python and Javascript side).  
:param options: Dictionary. Optional. Specific Python options available for  
→ this component.  
:param profile: Boolean | Dictionary. Optional. A flag to set the component  
→ performance storage.
```

```
cookies(text, url, align='center', width=(100, '%'), height=(None, 'px'), options=None, profile=False)
```

Tags

Categories

Usage:

```
page.ui.banners.cookies("Test", "#")
```

Parameters

- **text** – String. The value to be displayed to the component.
- **url** – String. The url link.
- **align** – String. The text-align property within this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

corner(*data=""*, *background=None*, *position='bottom'*, *width=(180, 'px')*, *height=(None, 'px')*, *options=None*, *profile=False*)

Tags

Categories

Usage:

```
# Add a banner on the bottom right corner
b = page.ui.banners.corner("bottom", 'red')
# Add click event on the banner
b.click([
    # hide the bonner on click
    b.dom.hide()
])

# Add a banner on the top right conner
corner = page.ui.banners.corner("top", 'red', position='top')
# Add interactivity on the banner style
corner.style.hover({"background": "white", 'color': 'red'})
# display the banner
corner.hover([b.dom.show()])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Div`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/banners.py>

Parameters

- **data** –
- **background** – String. Optional. Background color code.
- **position** – String. Optional. A string with the vertical position of the component
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit
- **options** – Dictionary. Optional. Specific Python options available for this component
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage

disclaimer(*copyright=None*, *links=None*, *width=(100, '%')*, *height=('auto', '')*, *align='center'*, *options=None*, *profile=False*)

Tags

Categories

Usage:

```
:param copyright:
:param links:
:param align: String. Optional. A string with the horizontal position of the
↳ component.
:param width: Tuple. Optional. A tuple with the integer for the component width
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳ this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳ performance storage.
```

follow(text, width=(100, '%'), height=('auto', ''), align='left', options=None, profile=False, youtube=True, twitter=True, facebook=True, twitch=True, instagram=True, linkedIn=True)

Tags

Categories

Usage:

```
:param text: String. Optional. The value to be displayed to the component.
:param align: String. Optional. A string with the horizontal position of the
↳ component.
:param width: Tuple. Optional. A tuple with the integer for the component width
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳ this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳ performance storage.
:param youtube: Boolean. Optional. Add the icon to the follow bar. Default True.
:param twitter: Boolean. Optional. Add the icon to the follow bar. Default True.
:param facebook: Boolean. Optional. Add the icon to the follow bar. Default
↳ True.
:param twitch: Boolean. Optional. Add the icon to the follow bar. Default True.
:param instagram: Boolean. Optional. Add the icon to the follow bar. Default
↳ True.
:param linkedIn: Boolean. Optional. Add the icon to the follow bar. Default
↳ True.
```

info(data, icon='fas fa-info-circle', background=None, width=(100, '%'), height=(None, 'px'), options=None, profile=False)

Tags

Categories

Usage:

```
:param data:
:param icon: String. Optional. The component icon content from font-awesome
↳ references. Default fas fa-info-circle
:param background: String. Optional. Background color code.
```

(continues on next page)

(continued from previous page)

```
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit
:param options: Dictionary. Optional. Specific Python options available for.
↳this component
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage
```

quote(content, author, avatar=None, background=None, size_notch=0, width=(100, '%'), align='center', height=(None, 'px'), options=None, profile=False)

Tags**Categories**

Usage:

```
:param content: String. Optional. The value to be displayed to the component.
:param author:
:param avatar:
:param background: String. Optional. Background color code.
:param size_notch:
:param align: String. Optional. A string with the horizontal position of the.
↳component
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit
:param options: Dictionary. Optional. Specific Python options available for.
↳this component
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage
```

row(headers, links, size_notch=0, background=None, width=(100, '%'), align='left', height=(None, 'px'), options=None, profile=False)

Tags**Categories**

Usage:

```
:param headers:
:param links:
:param size_notch:
:param background: String. Optional. Background color code.
:param align: String. Optional. A string with the horizontal position of the.
↳component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit.
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
```

(continues on next page)

(continued from previous page)

```
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
```

sponsor(logos, title='Sponsors', content="", background=None, width=(100, '%'), height=('auto', ''), align='center', options=None, profile=False)

Tags

Categories

Usage:

```
:param logos:
:param title: String. Optional. A panel title. This will be attached to the
↳title property.
:param content:
:param background: String. Optional. Background color code.
:param align: String. Optional. A string with the horizontal position of the
↳component.
:param width: Tuple. Optional. A tuple with the integer for the component width
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
```

text(data="", size_notch=0, background=None, width=(100, '%'), align='center', height=(None, 'px'), options=None, html_code=None, profile=False)

Tags

Categories

Usage:

```
:param data:
:param size_notch:
:param background: String. Optional. Background color code.
:param align: String. Optional. A string with the horizontal position of the
↳component.
:param width: Tuple. Optional. A tuple with the integer for the component width
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳this component.
:param html_code: String. Optional. An identifier for this component (on both
↳Python and Javascript side).
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
```

title(title, content, size_notch=0, background=None, width=(100, '%'), align='center', height=(None, 'px'), options=None, profile=False)

Tags

Categories

Usage:

```
:param title: String. Optional. A panel title. This will be attached to the
↳ title property.
:param content: String. Optional. The value to be displayed to the component.
:param size_notch:
:param background: String. Optional. Background color code.
:param align: String. Optional. A string with the horizontal position of the
↳ component.
:param width: Tuple. Optional. A tuple with the integer for the component width
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳ this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳ performance storage.
```

```
top(data="", background=None, width=(100, '%'), height=(None, 'px'), options=None, profile=False)
```

Tags

Categories

Usage:

```
# to Change the CSS style
top = page.ui.banners.top("text")
top.style.css.font_size = '40px'
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Div`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/banners.py>

Parameters

- **data** –
- **background** – String. Optional. Background color code.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

NavBars Interface

class epyk.interfaces.components.CompNavigation.NavBars(ui)

dark(logo=None, title=None, width=(100, '%'), height=(40, 'px'), options=None, profile=False)

Tags

Categories

Usage:

```
:param logo:
:param title: String. Optional. A panel title. This will be attached to the
    ↳ title property.
:param width: Tuple. Optional. A tuple with the integer for the component width
    ↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
    ↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
    ↳ this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
    ↳ performance storage.
```

fixed(logo=None, title=None, width=(100, '%'), height=(40, 'px'), options=None, profile=False)

Tags

Categories

Usage:

```
:param logo:
:param title: String. Optional. A panel title. This will be attached to the
    ↳ title property.
:param width: Tuple. Optional. A tuple with the integer for the component width
    ↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
    ↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
    ↳ this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
    ↳ performance storage.
```

top(logo=None, title=None, width=(100, '%'), height=(40, 'px'), options=None, profile=False)

Tags

Categories

Usage:

```
:param logo:
:param title: String. Optional. A panel title. This will be attached to the
    ↳ title property.
:param width: Tuple. Optional. A tuple with the integer for the component width
    ↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
    ↳ height and its unit.
```

(continues on next page)

(continued from previous page)

```
:param options: Dictionary. Optional. Specific Python options available for
↳ this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳ performance storage.
```

transparent(*logo=None, title=None, width=(100, '%'), height=(40, 'px'), options=None, profile=False*)

Tags

Categories

Usage:

```
:param logo:
:param title: String. Optional. A panel title. This will be attached to the
↳ title property.
:param width: Tuple. Optional. A tuple with the integer for the component width
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳ this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳ performance storage.
```

Navigation Interface

class epyk.interfaces.components.CompNavigation.**Navigation**(*ui*)

banner(*image: str = "", text: str = "", link: str = "", width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (None, 'px'), options: Optional[dict] = None, profile: Union[dict, bool] = False*)

Tags

Categories

Underlying HTML Objects:

- epyk.core.html.HtmlContainer.Div
- epyk.core.html.HtmlImage.Image
- epyk.core.html.HtmlContainer.Col
- epyk.core.html.HtmlContainer.Row
- epyk.core.html.HtmlText.Text
- epyk.core.html.HtmlLinks.ExternalLink

Parameters

- **image** – Optional. The image full path
- **text** – Optional. The value to be displayed to the component
- **link** – Optional. The url link
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

bar(*logo=None, title=None, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (40, 'px'), options=None, html_code=None, avatar: Union[bool, str] = False, profile: Union[dict, bool] = False*)
→ HtmlNavBar

Tags

Categories

Usage:

```
nav = page.ui.navigation.bar(title="test")
nav.add_text("Test text")
nav + page.ui.button("Click")
```

Underlying HTML Objects:

- epyk.core.html.HtmlMenu.HtmlNavBar

Parameters

- **logo** –
- **title** – String. Optional. A panel title. This will be attached to the title property
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **avatar** – Optional.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage

dots(*count: int, selected: int = 1, position: str = 'right', width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Union[dict, bool] = False*)

Tags

Categories

Usage:

```
d = page.ui.navigation.dots(10)
```

Parameters

- **count** – Optional. The number of pages
- **selected** – Optional. The selected index
- **position** – Optional. A string with the vertical position of the component
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

down(*icon: str = 'fas fa-arrow-down', top: int = 20, right: int = 20, bottom: Optional[int] = None, tooltip: Optional[str] = None, width: Union[tuple, int] = (25, 'px'), height: Union[tuple, int] = (25, 'px'), options: Optional[dict] = None, profile: Union[bool, dict] = False*)

Navigation button to go to the bottom of the page directly.

Tags

Categories

Usage:

```
page.ui.navigation.down()
```

Parameters

- **icon** – Optional. The component icon content from font-awesome references. Default fas fa-arrow-up
- **top** – Optional. The top property affects the vertical position of a positioned element
- **right** – Optional. The right property affects the horizontal position of a positioned element
- **bottom** – Optional. The top property affects the vertical position of a positioned element
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

footer(*components=None, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (80, 'px'), fixed=False, options=None, profile=False*) → HtmlFooter

Will create a footer object in the body of the report.

Tags

Categories

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlMenu.HtmlFooter`

Parameters

- **components** – list of html components.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **fixed** – Boolean. Optional. Fix the component at the page bottom.
- **options** – Dictionary. Optional. Specific Python options available for this component.

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

indices(*count: int, selected: int = 1, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Union[bool, dict] = False*)

Tags

Categories

Usage:

```
page.ui.navigation.indices(10)
```

Parameters

- **count** – Optional. The number of pages
- **selected** – Optional. The selected index
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage

more(*text: str = 'See more', icon: Optional[Union[str, bool]] = None, width: Union[tuple, int] = ('auto', ''), tooltip: Optional[str] = None, height: Union[tuple, int] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*)

Add a see more button to get the number of calls for a pagination on the server side.

Usage:

```
t = page.ui.text("Rewind")
btn = page.ui.navigation.more()
btn.click([page.js.console.log(btn.dom.next())])
t.click([btn.dom.rewind()])
```

Parameters

- **text** – Optional. The value to be displayed to the button
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. A string with the horizontal position of the component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage

- **options** – Optional. Specific Python options available for this component

nav(*logo=None, title: Optional[str] = None, components=None, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (40, 'px'), options: Optional[dict] = None, avatar: bool = False, profile: Union[dict, bool] = False*) → HtmlNavBar

Tags

Categories

Usage:

```
page.ui.components_skin = {"nav": {"css": {"background-color": 'pink'}}}
nav = page.ui.navigation.nav(height=60, options={"center": True, "logo_height": 50})
```

Parameters

- **logo** – Optional. The picture for the logo
- **title** – Optional. A panel title. This will be attached to the title property
- **components** – Optional. The Components to be added to the navbar
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **avatar** – Optional. Add a avatar picture to the right in the navbar
- **profile** – Optional. A flag to set the component performance storage

panel(*width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (100, '%'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*) → PanelsBar

Tags

Categories

Usage:

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/bars.py>

Parameters

- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **helper** – Optional. A tooltip helper

path(*record: List[dict], divider: Optional[str] = None, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (None, 'px'), options: Optional[dict] = None, profile: Union[dict, bool] = False*)

Tags

Categories

Usage:

```
record = [{"text": "Lin 1", 'url': 'report_list.html'}, {"text": "Link 2"}]
page.ui.navigation.path(record)
```

Parameters

- **record** – Component input data
- **divider** – Optional. A path delimiter
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

pilcrow(text: str = "", html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Add an anchor on the page and move to this when it is clicked.

Tags

Categories

Usage:

```
:param text: Optional. The value to be displayed to the component
:param html_code: Optional. An identifier for this component (on both Python_
↪and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

pin(text: str, url: str = '#', icon: str = 'fas fa-map-pin', top: int = 20, right: int = 20, bottom: Optional[int] = None, tooltip: Optional[str] = None, width: Union[tuple, int] = (25, 'px'), height: Union[tuple, int] = (25, 'px'), options: Optional[dict] = None, profile: Union[bool, dict] = False)

Shortcut to a specific position in the page.

Tags

Categories

Usage:

```
page.ui.navigation.pin("anchor", tooltip="test", bottom=20)
```

Parameters

- **text** – The shortcut name
- **url** – Optional. The anchor name
- **icon** – Optional. The component icon content from font-awesome references. Default fas fa-arrow-up
- **top** – Optional. The top property affects the vertical position of a positioned element
- **right** – Optional. The right property affects the horizontal position of a positioned element
- **bottom** – Optional. The top property affects the vertical position of a positioned element

- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

points(*count: int, selected: int = 0, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Union[dict, bool] = False*) → Points

Tags

Categories

Usage:

```
p = page.ui.navigation.points(10)
for i, _ in enumerate(p):
    p.click_item(i, [])
```

Parameters

- **count** – The number of pages
- **selected** – Optional. The selected index
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

scroll(*progress: int = 0, height: Union[tuple, int] = (3, 'px'), options: Optional[dict] = None, profile: Union[bool, dict] = False*)

Add a horizontal progressbar to display the status of the page scrollbar.

Tags

Categories

Usage:

```
page.ui.navigation.scroll()
```

Parameters

- **progress** – Optional. The progression on the page
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

shortcut(*components=None, logo=None, size=(40, 'px'), options=None, profile=None, html_code=None*)
→ Shortcut

Tags

Categories

Usage:

```
:param components: List. The different HTML objects to be added to the
↳ component.
:param logo:
:param size: Integer. Optional. Panel's height in pixel.
:param options: Dictionary. Optional. Specific Python options available for
↳ this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳ performance storage.
:param html_code: String. Optional. An identifier for this component (on both
↳ Python and Javascript side).
```

side(*components=None, anchor=None, size=262, position='right', options=None, profile=False, z_index: int = 20, overlay: bool = False*)

Tags

Categories

Usage:

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/contextmenu.py>
https://github.com/epykure/epyk-templates/blob/master/locals/components/st_news.py

Parameters

- **components** – The different HTML objects to be added to the component.
- **anchor** – Optional. The panel button to show / hide.
- **size** – Optional. Panel's width in pixel.
- **position** – Optional. A string with the vertical position of the component.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.
- **z_index** – Optional.
- **overlay** – Optional.

to(*y, x: Optional[int] = None, icon: str = 'fas fa-map-pin', top: int = 20, right: int = 20, bottom: Optional[int] = None, tooltip: Optional[str] = None, width: Union[tuple, int] = (25, 'px'), height: Union[tuple, int] = (25, 'px'), options: Optional[dict] = None, profile: Union[bool, dict] = False*)

Navigation button to go to a specific point in the page directly.

Tags

Categories

Usage:

```
page.ui.navigation.to(100, tooltip="test")
```

Parameters

- **y** – The y position on the page
- **x** – Optional. The x position on the page
- **icon** – Optional. The component icon content from font-awesome references. Default fas fa-arrow-up
- **top** – Optional. The top property affects the vertical position of a positioned element
- **right** – Optional. The right property affects the horizontal position of a positioned element
- **bottom** – Optional. The top property affects the vertical position of a positioned element
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

up(*icon: str = 'fas fa-arrow-up', top: int = 20, right: int = 20, bottom: Optional[int] = None, tooltip: Optional[str] = None, width: Union[tuple, int] = (25, 'px'), height: Union[tuple, int] = (25, 'px'), options: Optional[dict] = None, profile: Union[bool, dict] = False*)

Navigation button to go to the top of the page directly.

Tags

Categories

Usage:

```
page.ui.navigation.up()
```

Parameters

- **icon** – Optional. The component icon content from font-awesome references. Default fas fa-arrow-up
- **top** – Optional. The top property affects the vertical position of a positioned element
- **right** – Optional. The right property affects the horizontal position of a positioned element
- **bottom** – Optional. The top property affects the vertical position of a positioned element
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

Network Interface

class epyk.interfaces.components.CompNetwork.**Network**(ui)

alert(category: str, value: str = "", width: Union[tuple, int, str] = (320, 'px'), height: Union[tuple, int, str] = (None, None), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False) → Alert

Function to add when the python run some tags to put on the top of your report messages.

The type of the messages can be different according to its criticality. This is fully defined and #driven in the Python and visible in the browser when the page is ready

All the notification can be hidden directly from the report by setting the flag alerts = False e.g: rptObj.alerts = False

Usage:

```
page.ui.messaging.alert('WARNING', 'Server URL not recognized', 'Please check')
```

Underlying HTML Objects:

- epyk.core.html.HtmlMessaging.Alert

Related Pages:

<https://getbootstrap.com/docs/4.0/components/alerts/>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/alerts.py>

Parameters

- **category** – The warning level.
- **value** – Optional. The content of the notification.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

assistant(image, name: str = "", path: Optional[str] = None, html_code: Optional[str] = None, size: Union[tuple, int, str] = (50, 'px'), profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None) → Assistant

Usage:

```
:param image:
:param name:
:param path:
:param html_code: Optional. An identifier for this component (on both Python,
↳ and Javascript side).
:param size: Optional. A tuple with the integer for the component width and its,
↳ unit.
```

(continues on next page)

(continued from previous page)

```
:param profile: Optional. A flag to set the component performance storage.
:param options: Optional. Specific Python options available for this component.
```

bot(*html_code: str, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (200, 'px'), profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*) → Bot

Usage:

```
container = page.ui.network.bot(html_code='bot_service')
```

Underlying HTML Objects:

- `epyk.core.html.HtmlMessaging.Bot`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/bot.py>

Parameters

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **profile** – Optional. A flag to set the component performance storage.
- **options** – Optional. Specific Python options available for this component.

chat(*html_code: str, record: Optional[List[dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (200, 'px'), profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*) → Chat

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlMessaging.Chat`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/chat.py>

Parameters

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **record** – Optional. The Python list of dictionaries.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **profile** – Optional. A flag to set the component performance storage.
- **options** – Optional. Specific Python options available for this component.

comments(*html_code: str, record: Optional[List[dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (200, 'px'), profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None*) → Comments

Python wrapper to a div item composed to several sub html items to display message

Usage:

```
db = page.db(database="test.db")
page.comments('Test', dbService={'db': db, 'com_table': 'comments', 'reply_table': 'replyComments',
    'reply_service': 'post_reply/url', 'user_coms': 'user_comments', 'privacy': 'public', 'service': your/url})
```

Underlying HTML Objects:

- `epyk.core.html.HtmlMessaging.Comments`

Related Pages:

<https://leaverou.github.io/bubbly/> <http://manos.malihu.gr/jquery-custom-content-scroller/>

Parameters

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **record** – Optional. The Python list of dictionaries.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **profile** – Optional. A flag to set the component performance storage.
- **options** – Optional. Specific Python options available for this component.

danger(*value: str = "", html_code: Optional[str] = None, width: Union[tuple, int, str] = (320, 'px'), height: Union[tuple, int, str] = (None, None), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*) → Alert

Function to add when the python run some tags to put on the top of your report messages.

The type of the messages can be different according to its criticality. This is fully defined and #driven in the Python and visible in the browser when the page is ready

All the notification can be hidden directly from the report by setting the flag `alerts = False` e.g: `rptObj.alerts = False`

Usage:

```
page.ui.messaging.alert('danger', 'Server URL not recognized', 'Please check')
```

Underlying HTML Objects:

- `epyk.core.html.HtmlMessaging.Alert`

Related Pages:

<https://getbootstrap.com/docs/4.0/components/alerts/>

Parameters

- **value** – Optional. The content of the notification.

- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

download(name: str, icon: Optional[str] = None, path: Optional[str] = None, width: Union[tuple, int, str] = (25, 'px'), height: Union[tuple, int, str] = (25, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None) → IconEdit

Usage:

```
:param name: Optional.
:param icon: Optional. The component icon content from font-awesome references.
:param path: Optional. String. The image file path.
:param width: Optional. A tuple with the integer for the component width and
↳ its unit.
:param height: Optional. A tuple with the integer for the component height and
↳ its unit.
:param html_code: Optional. The id for this component.
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.
```

dropfile(placeholder: str = "", delimiter: str = 'TAB', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = ('auto', ''), tooltip: Optional[str] = None, html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None) → DropFile

Add an HTML component to drop files. The files will be dropped by default to the OUTPUT folder of the defined environment.

Files will also be recorded in the database in order to ensure that those data will not be shared. The data sharing is and should be defined only by the user from the UI.

Underlying HTML Objects:

- `epyk.core.html.HtmlFiles.DropFile`

Usage:

```
page.ui.network.dropfile()
```

Related Pages:

Parameters

- **placeholder** – Optional. The placeholder text when input empty.
- **delimiter** – Optional. The column delimiter.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **tooltip** – Optional. A string with the value of the tooltip.
- **html_code** – Optional. The id for this component.
- **options** – Optional. Specific Python options available for this component.

- **profile** – Optional. A flag to set the component performance storage.

impression(*number: int = 0, icon: str = 'fas fa-chart-bar', options: Optional[dict] = None, html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Add an impression component. This is designed to use the viewport function to increment the value,

Usage:

```
page.ui.network.impression()
```

Parameters

- **number** – Optional. The initial value
- **icon** – Optional. The icon text
- **options** – Optional. The number component options
- **html_code** – Optional. The code used on the JavaScript side
- **profile** – Optional. The profiling options

info(*value: str = "", html_code: Optional[str] = None, width: Union[tuple, int, str] = (320, 'px'), height: Union[tuple, int, str] = (None, None), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*) → Alert

Function to add when the python run some tags to put on the top of your report messages. The type of the messages can be different according to its criticality. This is fully defined and #driven in the Python and visible in the browser when the page is ready

Usage:

```
page.ui.messaging.alert('info', 'Server URL not recognized', 'Please check')
```

Underlying HTML Objects:

- `epyk.core.html.HtmlMessaging.Alert`

Related Pages:

<https://getbootstrap.com/docs/4.0/components/alerts/>

Parameters

- **value** – Optional. The content of the notification.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

news(*value: str = "", html_code: Optional[str] = None, width: Union[tuple, int, str] = (320, 'px'), height: Union[tuple, int, str] = (None, None), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*) → News

Usage:


```
b = page.ui.button("Display")
n = page.ui.messaging.news("This is a title", "This is the content", link_
↪script="TestSlider")
b.click(n.jsGenerate("Updated content", isPyData=True))
```

Underlying HTML Objects:

- `epyk.core.html.HtmlMessaging.News`

Templates:

https://github.com/epykure/epyk-templates/blob/master/locals/components/st_news.py

Parameters

- **value** – Optional.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

room(*img: str, html_code: Optional[str] = None, width: Union[tuple, int, str] = (60, 'px'), height: Union[tuple, int, str] = (60, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*) → Room

Usage:

```
:param img: Optional. The image path on the server or locally to be used.
:param html_code: Optional. The id for this component.
:param width: Optional. A tuple with the integer for the component width and
↪its unit.
:param height: Optional. A tuple with the integer for the component height and
↪its unit.
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.
```

success(*value: str = "", html_code: Optional[str] = None, width: Union[tuple, int, str] = (320, 'px'), height: Union[tuple, int, str] = (None, None), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*) → Alert

unction to add when the python run some tags to put on the top of your report messages. The type of the messages can be different according to its criticality. This is fully defined and #driven in the Python and visible in the browser when the page is ready

Usage:

```
page.ui.messaging.alert('success', 'Server URL not recognized', 'Please check')
```

Underlying HTML Objects:

- `epyk.core.html.HtmlMessaging.Alert`

Related Pages:

<https://getbootstrap.com/docs/4.0/components/alerts/>

Parameters

- **value** – Optional. The content of the notification.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

upload(*icon: Optional[str] = None, width: Union[tuple, int, str] = (25, 'px'), height: Union[tuple, int, str] = (25, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → IconEdit

Usage:

```
:param icon: Optional. The component icon content from font-awesome references
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param html_code: Optional. The id for this component
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

votes(*number: int = 0, options: Optional[dict] = None, html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Add a vote component with two arrows and a number.

Usage:

```
vote = page.ui.network.votes()
vote.up.style.css.cursor = "pointer"
vote.up.click(vote.number.dom.add(1))
vote.down.click(vote.number.dom.add(-1))
```

Parameters

- **number** – Optional. The initial value
- **options** – Optional. The number component options
- **html_code** – Optional. The code used on the JavaScript side
- **profile** – Optional. The profiling options

warning(*value: str = "", html_code: Optional[str] = None, width: Union[tuple, int, str] = (320, 'px'), height: Union[tuple, int, str] = (None, None), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*) → Alert

Function to add when the python run some tags to put on the top of your report messages. The type of the messages can be different according to its criticality. This is fully defined and #driven in the Python and visible in the browser when the page is ready

Usage:

```
page.ui.messaging.alert('warning', 'Server URL not recognized', 'Please check')

danger = page.ui.network.warning()
danger.options.time = None
danger.options.close = True
```

Underlying HTML Objects:

- `epyk.core.html.HtmlMessaging.Alert`

Related Pages:

<https://getbootstrap.com/docs/4.0/components/alerts/>

Parameters

- **value** – Optional. The content of the notification.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

Numbers Interface

`class epyk.interfaces.components.CompNumbers.Numbers(ui)`

digits(*text: Optional[str] = None, color: Optional[str] = None, align: str = 'center', width: Optional[Union[tuple, int, str]] = None, height: Optional[Union[tuple, int, str]] = None, html_code: Optional[str] = None, tooltip: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

The `` tag is used to group inline-elements in a document.

The `` tag provides no visual change by itself.

The `` tag provides a way to add a hook to a part of a text or a part of a document.

Tags

Categories

Usage:

```
page.ui.texts.span("Test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Position`

Related Pages:

https://www.w3schools.com/tags/tag_span.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/numbers.py>

Parameters

- **text** – Optional. The string value to be displayed in the component.
- **color** – Optional. The color of the text.
- **align** – Optional. The position of the icon in the line (left, right, center).
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **tooltip** – Optional. A string with the value of the tooltip.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

dollar(*number: float = 0, title: Optional[str] = None, label: Optional[str] = None, icon: Optional[str] = None, color: Optional[str] = None, tooltip: Optional[str] = None, html_code: Optional[str] = None, options: Optional[dict] = None, helper: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), align: str = 'center', profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
page.ui.texts.dollar(289839898, label="test", helper="Ok", icon="fas fa-align-  
↔center")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Numeric`

Parameters

- **number** – Optional. The value to be displayed to the component. Default now.
- **title** – Optional. A panel title. This will be attached to the title property.
- **label** – Optional. The text of label to be added to the component.
- **icon** – Optional. A string with the value of the icon to display from font-awesome.
- **color** – Optional. The font color in the component. Default inherit.
- **tooltip** – Optional. A string with the value of the tooltip.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **helper** – Optional. A tooltip helper.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **align** – Optional. The text-align property within this component.
- **profile** – Optional. A flag to set the component performance storage.

euro(*number: float = 0, title: Optional[str] = None, label: Optional[str] = None, icon: Optional[str] = None, color: Optional[str] = None, tooltip: Optional[str] = None, html_code: Optional[str] = None, options: Optional[dict] = None, helper: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), align: str = 'center', profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
page.ui.texts.euro(289839898, label="test", helper="Ok", icon="fas fa-align-
↔center")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Numeric`

Parameters

- **number** – Optional. The value to be displayed to the component. Default now.
- **title** – Optional. A panel title. This will be attached to the title property.
- **label** – Optional. The text of label to be added to the component.
- **icon** – Optional. A string with the value of the icon to display from font-awesome.
- **color** – Optional. The font color in the component. Default inherit.
- **tooltip** – Optional. A string with the value of the tooltip.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **helper** – Optional. A tooltip helper.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **align** – Optional. The text-align property within this component.
- **profile** – Optional. A flag to set the component performance storage.

money(*symbol: str, number: float = 0, title: Optional[str] = None, label: Optional[str] = None, icon: Optional[str] = None, color: Optional[str] = None, tooltip: Optional[str] = None, html_code: Optional[str] = None, options: Optional[dict] = None, helper: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), align: str = 'center', profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
page.ui.texts.money(289839898, label="test", helper="Ok", icon="fas fa-align-
↔center")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Numeric`

Parameters

- **symbol** – The currency symbol.
- **number** – Optional. The value to be displayed to the component. Default now.
- **title** – Optional. A panel title. This will be attached to the title property.
- **label** – Optional. The text of label to be added to the component.
- **icon** – Optional. A string with the value of the icon to display from font-awesome.
- **color** – Optional. The font color in the component. Default inherit.
- **tooltip** – Optional. A string with the value of the tooltip.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **helper** – Optional. A tooltip helper.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **align** – Optional. The text-align property within this component.
- **profile** – Optional. A flag to set the component performance storage.

move(*current*, *previous*=None, *components*=None, *title*: Optional[str] = None, *align*: str = 'center', *width*: Union[tuple, int, str] = (100, '%'), *height*: Union[tuple, int, str] = (None, 'px'), *color*: Optional[str] = None, *label*: Optional[str] = None, *options*: Optional[dict] = None, *helper*: Optional[str] = None, *profile*: Optional[Union[bool, dict]] = None)

Tags

Categories

Usage:

```
page.ui.numbers.move(100, 60, helper="test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTextComp.Delta`

Parameters

- **current** – The current value.
- **previous** – Optional. Default the current value and not move.
- **components** – Optional. List of HTML component to be added.
- **title** – Optional. The title definition.
- **align** – Optional. The text-align property within this component.
- **color** – Optional. The text color.
- **label** – Optional. The label for the up and down component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **helper** – Optional. The value to be displayed to the helper icon.

- **profile** – Optional. A flag to set the component performance storage.

number(*number: float = 0, title: Optional[str] = None, label: Optional[str] = None, icon: Optional[int] = None, color: Optional[str] = None, tooltip: Optional[str] = None, html_code: Optional[str] = None, options: Optional[dict] = None, helper: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), align: str = 'center', profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
page.ui.texts.number(289839898, label="test", helper="Ok", icon="fas fa-align-
↩center")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Numeric`

Parameters

- **number** – Optional. The value to be displayed to the component. Default now.
- **title** – Optional. A panel title. This will be attached to the title property.
- **label** – Optional. The text of label to be added to the component.
- **icon** – Optional. A string with the value of the icon to display from font-awesome.
- **color** – Optional. The color of the value.
- **tooltip** – Optional. A string with the value of the tooltip.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **helper** – Optional. A tooltip helper.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **align** – The text-align property within this component.
- **profile** – Optional. A flag to set the component performance storage.

percent(*number: float = 0, title: Optional[str] = None, label: Optional[str] = None, icon: Optional[str] = None, color: Optional[str] = None, tooltip: Optional[str] = None, html_code: Optional[str] = None, options: Optional[str] = None, helper: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), align: str = 'center', profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
page.ui.texts.percent(289839898, label="test", helper="Ok", icon="fas fa-align-
↩center")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Numeric`

Parameters

- **number** – Optional. The value to be displayed to the component. Default now.
- **title** – Optional. A panel title. This will be attached to the title property.
- **label** – Optional. The text of label to be added to the component.
- **icon** – Optional. A string with the value of the icon to display from font-awesome.
- **color** – Optional. The color of the value.
- **tooltip** – Optional. A string with the value of the tooltip.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **helper** – Optional. A tooltip helper.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **align** – Optional. The text-align property within this component.
- **profile** – Optional. A flag to set the component performance storage.

plotly(*value: float = 0, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*)

Tags**Categories**

Usage:

Underlying HTML Objects:

- `epyk.core.graph.GraphPlotly.Indicator`

Parameters

- **value** – Optional. Number. a value.
- **profile** – Optional. A flag to set the component performance storage.
- **options** – Optional. Specific Python options available for this component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).

plotly_with_delta(*value, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*)

Tags**Categories**

Usage:

Underlying HTML Objects:

- `epyk.core.graph.GraphPlotly.Indicator`

Parameters

- **value** – Number. a value.
- **profile** – Optional. A flag to set the component performance storage.
- **options** – Optional. Specific Python options available for this component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).

pound(*number: float = 0, title: Optional[str] = None, label: Optional[str] = None, icon: Optional[str] = None, color: Optional[str] = None, tooltip: Optional[str] = None, html_code: Optional[str] = None, options: Optional[dict] = None, helper: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), align: str = 'center', profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
page.ui.texts.pound(289839898, label="test", helper="Ok", icon="fas fa-align-
center")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Numeric`

Parameters

- **number** – Optional. The value to be displayed to the component. Default now.
- **title** – Optional. A panel title. This will be attached to the title property.
- **label** – Optional. The text of label to be added to the component.
- **icon** – Optional. A string with the value of the icon to display from font-awesome.
- **color** – Optional. The font color in the component. Default inherit.
- **tooltip** – Optional. A string with the value of the tooltip.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **helper** – Optional. A tooltip helper.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **align** – Optional. The text-align property within this component.
- **profile** – Optional. A flag to set the component performance storage.

Panels Interface

class epyk.interfaces.components.CompPanels.Panels(ui)

arrows_down(color: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False)

Python wrapper for a multi Tabs component.

Tags

Categories

Underlying HTML Objects:

- epyk.core.html.HtmlContainer.TabsArrowsDown

Related Pages:

<https://getbootstrap.com/docs/4.0/components/navs/>

Usage:

```
:param color: Optional. The font color in the component. Default inherit.
:param width: Optional. A tuple with the integer for the component width and
↳ its unit.
:param height: Optional. A tuple with the integer for the component height and
↳ its unit.
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side).
:param helper: Optional. A tooltip helper.
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.
```

arrows_up(color: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False)

Python wrapper for a multi Tabs component.

Tags

Categories

Underlying HTML Objects:

- epyk.core.html.HtmlContainer.TabsArrowsUp

Related Pages:

<https://getbootstrap.com/docs/4.0/components/navs/>

Usage:

```
:param color: Optional. The font color in the component. Default inherit.
:param width: Optional. A tuple with the integer for the component width and
↳ its unit.
:param height: Optional. A tuple with the integer for the component height and
↳ its unit.
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side).
:param helper: Optional. A tooltip helper.
```

(continues on next page)

(continued from previous page)

```
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.
```

boxes(color: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False)

Python wrapper to the Bootstrap rectangle boxes interface.

Tags

Categories

Usage:

```
tab = page.ui.panels.boxes()
for i in range(5):
    tab.add_panel("Panel %s" % i, rptObj.ui.text("test %s" % i))
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Tabs`

Related Pages:

<https://getbootstrap.com/docs/4.0/components/navs/>

Parameters

- **color** – Optional. The font color in the component. Default inherit.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **align** – Optional. The text-align property within this component.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

filters(items=None, category: str = 'group', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (60, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Chip component with only the filtering section.

Tags

Categories

Usage:

```
filters = page.ui.panels.filters()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlEvent.Filters`

Related Pages:

https://www.w3schools.com/howto/howto_css_contact_chips.asp

Parameters

- **items** – Optional.
- **category** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

hamburger(*components: Optional[List[Html]] = None, title: Union[str, dict] = "", color: Optional[str] = None, align: str = 'center', width=(100, '%'), height=(None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Union[dict, bool] = False*)

Add hamburger panel.

Tags

Categories

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.PanelSlide`

Parameters

- **components** – Optional. The different HTML objects to be added to the component.
- **title** – Optional. A panel title. This will be attached to the title property.
- **color** – Optional. The font color in the component. Default inherit.
- **align** – Optional. The text-align property within this component (Default center).
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

menu(*color: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False*)

Python wrapper to the Bootstrap Pills interface.

Tags

Categories

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Tabs`

Related Pages:

<https://getbootstrap.com/docs/4.0/components/navs/>

Usage:

```
:param color: Optional. The font color in the component. Default inherit.
:param width: Optional. A tuple with the integer for the component width and
↳ its unit.
:param height: Optional. A tuple with the integer for the component height and
↳ its unit.
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side).
:param helper: Optional. A tooltip helper.
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.
```

```
nav(width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (100, '%'), options:
Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str]
= None)
```

Tags

Categories

Usage:

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/bars.py>

Parameters

- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. A dictionary with the components properties.
- **profile** – Optional. A flag to set the component performance storage.
- **helper** – Optional. A tooltip helper.

```
panel(components: Optional[List[Html]] = None, title: Optional[str] = None, color: Optional[str] = None,
width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code:
Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile:
Optional[Union[bool, dict]] = False)
```

Add a simple div panel to the page.

Tags

Categories

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Panel`

Usage:

```

:param components: Optional. The different HTML objects to be added to the
    component.
:param title: Optional. A panel title. This will be attached to the title
    property.
:param color: Optional. The font color in the component. Default inherit.
:param width: Optional. A tuple with the integer for the component width and
    its unit.
:param height: Optional. A tuple with the integer for the component height and
    its unit.
:param html_code: Optional. An identifier for this component (on both Python
    and Javascript side).
:param helper: Optional. A tooltip helper.
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.

```

pills(*color: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), align: str = 'left', html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*)

Python wrapper to the Bootstrap Pills interface.

Tags

Categories

Usage:

```

tab = page.ui.panels.pills()
for i in range(5):
    tab.add_panel("Panel %s" % i, rptObj.ui.text("test %s" % i))

```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Tabs`

Related Pages:

<https://getbootstrap.com/docs/4.0/components/navs/>

Parameters

- **color** – Optional. The font color in the component. Default inherit.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **align** – Optional. The text-align property within this component.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

sliding(*components, title, color: Optional[str] = None, align: str = 'center', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False*) → PanelSlide

Add a sliding panel.

TODO: Animate the CSS to make a transition.

Tags

Categories

Usage:

```
text = page.ui.text("Test")
page.ui.panels.sliding([text], title="Panel title")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.PanelSlide`

Parameters

- **components** – Optional. The different HTML objects to be added to the component.
- **title** – Optional. A panel title. This will be attached to the title property.
- **color** – Optional. The font color in the component. Default inherit.
- **align** – Optional. The text-align property within this component (Default center).
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

property slidings

More custom sliding panels.

split(*left: Optional[Html] = None, right: Optional[Html] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (200, 'px'), left_width: Union[tuple, int, str] = (160, 'px'), resizable: bool = True, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → PanelSplit

Tags

Categories

Usage:

```
number = page.ui.rich.number(500, "Test", height=(150, 'px'))
number_2 = page.ui.rich.number(500, "Test 2 ", options={"url": "http://www.
↪google.fr"})
div = page.ui.layouts.panelsplit(left=number, right=number_2)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.PanelSlide`

Related Pages:

<https://codepen.io/rstrahl/pen/eJZQej>

Parameters

- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **left_width** – Optional.
- **left** – Optional.
- **right** – Optional.
- **resizable** – Optional.
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

tabs(*color: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False*)

Python wrapper for a multi Tabs component.

Tags

Categories

Usage:

```
tab = page.ui.panels.tabs()
for i in range(5):
    tab.add_panel("Panel %s" % i, rptObj.ui.text("test %s" % i))
```

Underlying HTML Objects:

- `epyk.core.html.HtmlContainer.Tabs`

Related Pages:

<https://getbootstrap.com/docs/4.0/components/navs/>

Parameters

- **color** – Optional. The font color in the component. Default inherit.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

Slidings Interface

class epyk.interfaces.components.CompPanels.Slidings(ui)

left(components: List[Html], title: str = "", color: Optional[str] = None, align: str = 'center', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None)

Sliding panels with the arrow on the left.

Tags

Categories

Usage:

```
:param components: The different HTML objects to be added to the component.
:param title: Optional. A panel title. This will be attached to the title_
↳property.
:param color: Optional. The font color in the component. Default inherit.
:param align: The text-align property within this component.
:param width: Optional. A tuple with the integer for the component width and_
↳its unit.
:param height: Optional. A tuple with the integer for the component height and_
↳its unit.
:param html_code: Optional. An identifier for this component (on both Python_
↳and Javascript side).
:param helper: Optional. A tooltip helper.
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.
```

plus(components, title: str = "", color: Optional[str] = None, align: str = 'center', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None)

Same component than sliding with a different style.

Tags

Categories

Usage:

```
:param components: The different HTML objects to be added to the component.
:param title: Optional. A panel title. This will be attached to the title_
↳property.
:param color: Optional. The font color in the component. Default inherit.
:param align: Optional. The text-align property within this component.
:param width: Optional. A tuple with the integer for the component width and_
↳its unit.
:param height: Optional. A tuple with the integer for the component height and_
↳its unit.
:param html_code: Optional. An identifier for this component (on both Python_
↳and Javascript side).
:param helper: Optional. A tooltip helper.
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.
```

right(*components: List[Html]*, *title: str = ''*, *color=None*, *align='center'*, *width=(100, '%')*, *height=(None, 'px')*, *html_code=None*, *helper=None*, *options=None*, *profile=False*)

Sliding panels with the arrow on the right.

Tags

Categories

Usage:

```
:param components: The different HTML objects to be added to the component.
:param title: Optional. A panel title. This will be attached to the title_
↳property.
:param color: Optional. The font color in the component. Default inherit.
:param align: Optional. The text-align property within this component.
:param width: Optional. A tuple with the integer for the component width and
↳its unit.
:param height: Optional. A tuple with the integer for the component height and
↳its unit.
:param html_code: Optional. An identifier for this component (on both Python_
↳and Javascript side).
:param helper: Optional. A tooltip helper.
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.
```

Pictogram Interface

class epyk.interfaces.components.CompPictos.**Pictogram**(*ui*)

arrow(*width=(21, 'px')*, *height=(12, 'px')*)

Usage:

```
:param width: Tuple. Optional. A tuple with the integer for the component width_
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component_
↳height and its unit.
```

compass(*fill=None*, *border=None*, *width=(33, 'px')*, *height=(25, 'px')*)

Usage:

Related Pages:

<https://uxwing.com/compass-icon/>

Parameters

- **fill** –
- **border** –
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.

faq(*fill=None*, *border=None*, *width=(33, 'px')*, *height=(25, 'px')*)

Usage:

Related Pages:

<https://uxwing.com/faq-icon/>

Parameters

- **fill** –
- **border** –
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.

flam(width=(619, 'px'), height=(423, 'px'))

Usage:

```
:param width: Tuple. Optional. A tuple with the integer for the component width_
↪and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component_
↪height and its unit.
```

path(path, fill=None, stroke=None, width=(33, 'px'), height=(25, 'px'), viewBox=(150, 100), options=None, profile=None)

Usage:

```
:param path:
:param fill:
:param stroke:
:param width: Tuple. Optional. A tuple with the integer for the component width_
↪and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component_
↪height and its unit.
:param viewBox:
:param options:
:param profile:
```

paths(paths, fill=None, stroke=None, width=(33, 'px'), height=(25, 'px'), viewBox=(150, 100))

Usage:

```
:param paths:
:param fill:
:param stroke:
:param width: Tuple. Optional. A tuple with the integer for the component width_
↪and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component_
↪height and its unit.
:param viewBox:
```

people(fill=None, border=None, width=(20, 'px'), height=(48, 'px'))

Usage:

```
:param fill:
:param border:
:param width: Tuple. Optional. A tuple with the integer for the component width_
↪and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component_
↪height and its unit.
```

quote(*fill=None, border=None, width=(33, 'px'), height=(25, 'px')*)

Usage:

```
:param fill:
:param border:
:param width: Tuple. Optional. A tuple with the integer for the component width
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳height and its unit.
```

stack(*fill=None, border=None, width=(33, 'px'), height=(25, 'px')*)

Usage:

Related Pages:

<https://uxwing.com/stack-icon/>

Parameters

- **fill** –
- **border** –
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.

team(*fill=None, border=None, width=(50, 'px'), height=(30, 'px')*)

Usage:

Related Pages:

<https://uxwing.com/business-team-icon/>

Parameters

- **fill** –
- **border** –
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.

tick(*fill=None, border=None, width=(30, 'px'), height=(30, 'px')*)

Usage:

Related Pages:

<https://uxwing.com/check-mark-icon/>

Parameters

- **fill** –
- **border** –
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.

Poller Interface

class epyk.interfaces.components.CompPollers.**Poller**(ui)

live(time, js_funcs=None, components=None, icon: str = 'circle', width: Union[tuple, int] = (15, 'px'), height: Union[tuple, int] = (15, 'px'), align: str = 'left', html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None)

Tags

Categories

Usage:

```
:param time: Integer. Interval time in second.
:param js_funcs: String | List. The Javascript functions.
:param components: List. HTML components to be triggered when activated.
:param icon: String. Optional. The font awesome icon reference.
:param width: Tuple. Optional. A tuple with the integer for the component width,
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component,
↳ height and its unit.
:param align: String. Optional. A string with the horizontal position of the,
↳ component.
:param html_code: String. Optional. An identifier for this component (on both,
↳ Python and Javascript side).
:param options: Dictionary. Optional. Specific Python options available for,
↳ this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component,
↳ performance storage.
```

toggle(time, js_funcs=None, components=None, label: Optional[str] = None, color: Optional[str] = None, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (20, 'px'), align: str = 'left', html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Tags

Categories

Usage:

```
:param time: Integer. Interval time in second.
:param js_funcs: String | List. The Javascript functions.
:param components: List. HTML components to be triggered when activated.
:param label: String. Optional. The text of label to be added to the component
:param color: String. Optional. The font color in the component. Default,
↳ inherit.
:param width: Tuple. Optional. A tuple with the integer for the component width,
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component,
↳ height and its unit.
:param align: String. Optional. A string with the horizontal position of the,
↳ component.
:param html_code: String. Optional. An identifier for this component (on both,
↳ Python and Javascript side).
:param options: Dictionary. Optional. Specific Python options available for,
```

(continues on next page)

(continued from previous page)

```

↪this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↪performance storage.

```

Rich Interface

```
class epyk.interfaces.components.CompRich.Rich(ui)
```

```
adv_text(section, title, content, background: str = "", options: Optional[Union[bool, dict]] = None, profile:
Optional[Union[bool, dict]] = None)
```

Tags

Categories

Usage:

```

:param section:
:param title: String | Component. Optional. A panel title. This will be
↪attached to the title property.
:param content:
:param background:
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.

```

```
color(code, content: str = 'data copied to clipboard', width: Union[tuple, int, str] = (20, 'px'), height:
Union[tuple, int, str] = (20, 'px'), options: Optional[Union[bool, dict]] = None, profile:
Optional[Union[bool, dict]] = None)
```

Color component.

Tags

Categories

Usage:

```

:param code: Tuple or String. The color code.
:param content: Optional.
:param width: Optional. A tuple with the integer for the component width and
↪its unit.
:param height: Optional. A tuple with the integer for the component height and
↪its unit.
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.

```

```
composite(schema, width: Union[tuple, int, str] = (None, '%'), height: Optional[Union[bool, dict]] = (None,
'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] =
None, profile: Optional[Union[bool, dict]] = None)
```

Composite bespoke object.

This object will be built based on its schema. No specific CSS Style and class will be added to this object. The full definition will be done in the nested dictionary schema.

Tags

Categories

Usage:

```

schema = {'type': 'div', 'css': {}, 'class': , 'attrs': {} 'arias': {},
↪ 'children': [
    {'type': : {...}}
    ...
  ]}

```

Parameters

- **schema** – The component nested structure.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **helper** – Optional. The value to be displayed to the helper icon.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

console(*content: str = "*, *width: Union[tuple, int, str] = (100, '%')*, *height: Union[tuple, int, str] = (200, 'px')*, *html_code: Optional[str] = None*, *options: Optional[Union[bool, dict]] = None*, *profile: Optional[Union[bool, dict]] = None*)

Display an component to show logs.

Tags

Categories

Usage:

```

c = page.ui.rich.console(
    "* This is a log section for all the events in the different buttons *", ↪
↪ options={"timestamp": True})

```

Underlying HTML Objects:

- `epyk.core.html.HtmlTextEditor.Console`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/checkbox.py>

Parameters

- **content** – Optional. The console content.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

countdown(*day, month, year, hour: int = 0, minute: int = 0, second: int = 0, label: Optional[str] = None, icon: str = 'fas fa-stopwatch', time_ms_factor: int = 1000, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Add a countdown to the page and remove the content if the page has expired.

Tags

Categories

Usage:

```
page.ui.rich.countdown(24, 9 2021)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlDates.CountDownDate`

Related Pages:

https://www.w3schools.com/js/js_date_methods.asp https://www.w3schools.com/howto/howto_js_countdown.asp <https://fontawesome.com/icons/stopwatch?style=solid>

Parameters

- **day** – Day's number.
- **month** – Month's number.
- **year** – Year's number.
- **hour** – Optional. Number of hours.
- **minute** – Optional. Number of minutes.
- **second** – Optional. Number of seconds.
- **label** – Optional. The component label content.
- **icon** – Optional. The component icon content from font-awesome references.
- **time_ms_factor** – Optional. The format from the format in milliseconds.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. The component identifier code (for both Python and Javascript).
- **helper** – Optional. A tooltip helper.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

delta(*record=None, components=None, title: Optional[str] = None, align: str = 'center', width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = ('auto', ''), options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
page.ui.rich.delta({'number': 100, 'prevNumber': 60, 'thresold1': 100,  
↪ 'thresold2': 50}, helper="test")
```


Underlying HTML Objects:

- `epyk.core.html.HtmlTextComp.Delta`

Tags

Numbers |

Categories

Container |

Parameters

- **record** – Optional. The input data for this component.
- **components** – Optional. The HTML components to be added to this component.
- **title** – Optional. A panel title. This will be attached to the title property.
- **align** – The text-align property within this component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **helper** – Optional. A tooltip helper.
- **profile** – Optional. A flag to set the component performance storage.

elapsed(*day: Optional[int] = None, month: Optional[int] = None, year: Optional[int] = None, label=None, icon=None, width=(None, 'px'), height=(None, 'px'), html_code=None, helper=None, options=None, profile=None*)

Tags

Categories

Usage:

```
dt = page.ui.rich.elapsed(day=1, month=1, year=2021)
page.ui.button("Click").click([dt.build({"year": 2022, "month": 1, "day": 1})])
```

Parameters

- **day** – The day number
- **month** – The month number [1, 12]
- **year** – The year number
- **label** –
- **icon** –
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).
- **helper** – String. Optional. The value to be displayed to the helper icon.
- **options** – Dictionary. Optional. Specific Python options available for this component.

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

info(*text: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Display an info icon with a tooltip.

Tags

Categories

Usage:

```
page.ui.info("Test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlOthers.Help`

Related Pages:

<https://fontawesome.com/icons/question-circle?style=solid> <https://api.jqueryui.com/tooltip/>

Parameters

- **text** – Optional. The content of the tooltip.
- **profile** – Optional. A boolean to store the performances for each components.
- **options** – Optional. Specific Python options available for this component.

light(*color: Optional[str] = None, height: Union[tuple, int, str] = (None, 'px'), label: Optional[str] = None, align: str = 'left', tooltip: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Add a traffic light component to give a visual status of a given process.

Tags

Categories

Usage:

```
page.ui.rich.light("red", label="label", tooltip="Tooltip", helper="Helper")
page.ui.rich.light(True)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTextComp.TrafficLight`

Parameters

- **color** – Optional. A hexadecimal color code.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **label** – Optional. The text of label to be added to the component.
- **align** – Optional. A string with the horizontal position of the component.
- **tooltip** – Optional. A string with the value of the tooltip.
- **helper** – Optional. The filtering properties for this component.
- **options** – Optional. Specific Python options available for this component.

- **profile** – Optional. A flag to set the component performance storage.

markdown(*text: str = "", width: Union[tuple, int, str] = ('calc(100% - 10px)', ''), height: Union[tuple, int, str] = ('auto', ''), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
page.ui.inputs.editor()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTextEditor.Editor`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/markdown.py>

Parameters

- **text** – Optional. The value to be displayed to the component.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

powered(*by=None, width=(100, '%'), height=(None, 'px'), options=None, profile=None*)

Display badges for the specifies JavaScript modules.

Tip: If by is None. This will display only the main JavaScript module with the current version. It will not display the underlying components.

This component needs to be called at the end to ensure all the imported will be registered.

Tags

Categories

Parameters

- **by** – List. Optional. Name of JavaScript library aliases.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

search_input(*text: str = "", placeholder: str = 'Search..', color: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, tooltip: Optional[str] = None, extensible: bool = False, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Search bar.

Tags

Categories

Usage:

```
page.ui.inputs.search()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlInput.Search`

Related Pages:

https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_anim_search

Parameters

- **text** – Optional. The value to be displayed to the component.
- **placeholder** – Optional. The text display when empty.
- **color** – Optional. The font color in the component. Default inherit.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **tooltip** – Optional. A string with the value of the tooltip.
- **extensible** – Optional. Flag to specify the input style.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

search_results(*records=None, results_per_page: int = 20, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Display the search results. This will return the matches and the details.

Tags

Categories

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlTextComp.SearchResult`

Parameters

- **records** – Optional. The list of dictionaries with the input data.
- **results_per_page** – Optional. The page index.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.

- **profile** – Optional. A flag to set the component performance storage.

stars(*val=None, label: Optional[str] = None, color: Optional[str] = None, align: str = 'left', best: int = 5, html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Entry point for the Stars component.

Tags

Categories

Usage:

```
page.ui.rich.stars(3, label="test", helper="This is a helper")
stars = page.ui.rich.stars(3, label="test", helper="This is a helper")
stars.click()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlOthers.Stars`

Related Pages:

https://www.w3schools.com/howto/howto_css_star_rating.asp

Parameters

- **val** – Optional. Number of stars.
- **label** – Optional. The text of label to be added to the component.
- **color** – Optional. The font color in the component. Default inherit.
- **align** – Optional. A string with the horizontal position of the component.
- **best** – Optional. The max number of stars. Default 5.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **helper** – Optional. The value to be displayed to the helper icon.
- **profile** – Optional. A flag to set the component performance storage.

status(*status: str, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
:param status:
:param width: Optional. A tuple with the integer for the component width and
↳ its unit.
:param height: Optional. A tuple with the integer for the component height and
↳ its unit.
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side).
```

(continues on next page)

(continued from previous page)

```
:param profile: Optional. A flag to set the component performance storage.
:param options: Optional. Specific Python options available for this component.
```

```
update(label: Optional[str] = None, color: Optional[str] = None, align: Optional[str] = None, width:
        Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code:
        Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool,
        dict]] = None)
```

Last Update time component.

Tags

Categories

Usage:

```
page.ui.rich.update("Last update: ")

update = page.ui.rich.update()
page.ui.button("Click").click([update.refresh()])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlDates.LastUpdated`

Related Pages:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/calendar.py>

Parameters

- **label** – Optional. The label to be displayed close to the date. Default Last Update.
- **color** – Optional. The color code for the font.
- **align** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. The component identifier code (for both Python and Javascript).
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

Sliders Interface

```
class epyk.interfaces.components.CompSliders.Sliders(ui)
```

This module is relying on some JQuery IU components

The slider and progress bar components can be fully described on the corresponding website

- <https://jqueryui.com/progressbar/>
- <https://jqueryui.com/slider/>

As this module will return those object, all the properties and changes defined in the documentation can be done.

date(*value=None, minimum: Optional[float] = None, maximum: Optional[float] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (20, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlEvent.SliderDate`

Parameters

- **value** – Optional. The initial value
- **minimum** – Optional. The min value
- **maximum** – Optional. The max value
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

date_range(*value1: Optional[str] = None, value2: Optional[str] = None, minimum: Optional[float] = None, maximum: Optional[float] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (20, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlEvent.SliderDate`

Parameters

- **value1** – Optional. The initial min value
- **value2** – Optional. The initial max value
- **minimum** – Optional. The min value
- **maximum** – Optional. The max value
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

lower(*value=None, minimum: float = 0, maximum: float = 100, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (20, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlEvent.Range`

Parameters

- **value** – Optional. The initial value
- **minimum** – Optional. The min value. Default 0
- **maximum** – Optional. The max value. Default 100
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

progress(*number: float = 0, total: float = 100, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (20, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

```
:param number: Optional. The initial value
:param total: Optional. The total value
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
:param helper: Optional. A tooltip helper
:param profile: Optional. A flag to set the component performance storage
:param options: Optional. Specific Python options available for this component
```

progressbar(*number: float = 0, total: float = 100, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (20, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Add a progress bar component to the page

Usage:

```
page.ui.sliders.progressbar(300)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlEvent.ProgressBar`

Related Pages:

<https://jqueryui.com/progressbar/>

Parameters

- **number** – Optional. The initial value
- **total** – Optional. The total value
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

range(*values=None, minimum: float = 0, maximum: float = 100, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (20, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlEvent.Range`

Parameters

- **values** – Optional. The initial values
- **minimum** – Optional. The min value. Default 0
- **maximum** – Optional. The max value. Default 100
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

slider(*number: float = 0, minimum: float = 0, maximum: float = 100, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Slider

Add a JQuery UI slider object to the page

Usage:

```
page.ui.slider(40)
page.ui.slider([1, 2, 3, 4, 5, 6, 7])

# With even and circle progress
text = page.ui.pyk.progress.circle()
s = page.ui.slider(54)
page.ui.row([text, s])
s.output = text
s.options.slide()
```

Underlying HTML Objects:

- `epyk.core.html.HtmlEvent.Slider`

Related Pages:

<https://jqueryui.com/slider/>

Parameters

- **number** – Optional. The initial value
- **minimum** – Optional. The min value. Default 0
- **maximum** – Optional. The max value. Default 100
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

upper(*value=None, minimum: float = 0, maximum: float = 100, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (20, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlEvent.Range`

Parameters

- **value** – Optional. The initial value
- **minimum** – Optional. The min value. Default 0
- **maximum** – Optional. The max value. Default 100
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

Steppers Interface

```
class epyk.interfaces.components.CompSteps.Steppers(ui)
```

Tags Interface

```
class epyk.interfaces.components.CompTags.Tags(ui)
```

a(text: str, url: str, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

The <a> tag defines a hyperlink, which is used to link from one page to another.

The most important attribute of the <a> element is the href attribute, which indicates the link's destination.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGenericLink`

Related Pages:

https://www.w3schools.com/tags/tag_a.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/paragraph.py>

Usage:

```
:param text: String with the content to be added to the component.
:param url: String. Specifies the URL of the page the link goes to.
:param width: Tuple with the width value and its unit.
:param height: Tuple with the height value and its unit.
:param html_code: String. The code reference of the component.
:param tooltip: String. The tooltip to be display on the component.
:param options: Dictionary. Optional. Specific Python options available for
↳ this component.
:param profile: Boolean flag to set the profiling mode for the component.
```

abbr(text: str, width: Union[tuple, int] = (100, 'px'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

The <abbr> tag defines an abbreviation or an acronym, like “HTML”, “Mr.”, “Dec.”, “ASAP”, “ATM”.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_abbr.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

aside(*text: str = "", width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The <aside> tag defines some content aside from the content it is placed in.

The aside content should be related to the surrounding content.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGenericLink`

Related Pages:

https://www.w3schools.com/tags/tag_aside.asp

Parameters

- **text** – String with the content to be added to the component.
- **width** – Tuple with the width value and its unit.
- **height** – Tuple with the height value and its unit.
- **html_code** – String. The code reference of the component.
- **tooltip** – String. The tooltip to be display on the component.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component.

b(*text: str, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The tag specifies bold text without any extra importance.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_b.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit

- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

bdi(*text*, *width*: Union[tuple, int] = (100, '%'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

BDI stands for Bi-Directional Isolation. The <bdi> tag is new in HTML5.

Usage:

```
bdi = rptObj.ui.tags.bdi("bdi tag")
bdi.click(rptObj.js.alert("test"))
bdi.css({'cursor': 'pointer'})
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_bdi.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

bdo(*text*: str, *width*: Union[tuple, int] = (100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

bdo stands for Bi-Directional Override. The <bdo> tag is used to override the current text direction.

Usage:

```
bdo = rptObj.ui.tags.bdo("bdo tag")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_bdo.asp

Parameters

- **text** – String with the content to be added to the component

- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

cite(*text: str, width: Union[tuple, int] = (100, 'px'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The <cite> tag defines the title of a work (e.g. a book, a song, a movie, a TV show, a painting, a sculpture, etc.).

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_cite.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

comment(*text: str*)

Add an HTML comment to the code

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlComment`

Parameters

- **text** – String with the content to be added to the component

delete(*text: str, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit

- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

dfn(*text: str, width: Union[tuple, int] = (100, 'px'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The <dfn> tag represents the defining instance of a term in HTML.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_dfn.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

em(*text: str, width: Union[tuple, int] = (100, 'px'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The tag is a phrase tag. It renders as emphasized text.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_em.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.

- **profile** – Boolean flag to set the profiling mode for the component

figcaption(*text: str = "", width=(None, '%'), height=(None, 'px'), html_code=None, tooltip="", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The <figcaption> tag defines a caption for a <figure> element.

The <figcaption> element can be placed as the first or last child of the <figure> element.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGenericLink`

Related Pages:

https://www.w3schools.com/tags/tag_figcaption.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

h1(*text: str, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The <h1> to <h6> tags are used to define HTML headings.

<h1> defines the most important heading. <h6> defines the least important heading.

Underlying HTML Objects:

- `:class: epyk.core.html.HtmlTags.HtmlGeneric`s`

Related Pages:

https://www.w3schools.com/tags/tag_hn.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

h2(*text: str, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The <h1> to <h6> tags are used to define HTML headings.

<h1> defines the most important heading. <h6> defines the least important heading.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_hn.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

h3(*text: str, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The <h1> to <h6> tags are used to define HTML headings.

<h1> defines the most important heading. <h6> defines the least important heading.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_hn.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

hn(*level: int, text: str, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = '', options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The <h1> to <h6> tags are used to define HTML headings.

<h1> defines the most important heading. <h6> defines the least important heading.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_hn.asp

Parameters

- **level** – Integer.
- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

i(*text, width: Union[tuple, int] = (None, 'px'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = '', options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The <i> tag defines a part of text in an alternate voice or mood. The content of the <i> tag is usually displayed in italic.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_i.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

ins(*text*, *width*: Union[tuple, int] = (100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

The <ins> tag defines a text that has been inserted into a document.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_ins.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

kbd(*text*, *width*: Union[tuple, int] = (100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

The <kbd> tag is a phrase tag. It defines keyboard input.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_kbd.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

label(*text*: str = "", *width*: Union[tuple, int] = (100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

The <label> tag defines a label for several elements.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_label.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

mark(*text: str, width: Union[tuple, int] = (100, 'px'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The <mark> tag defines marked text.

Use the <mark> tag if you want to highlight parts of your text.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_mark.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

meter(*text: str, width: Union[tuple, int] = (100, 'px'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The <meter> tag defines a scalar measurement within a known range, or a fractional value. This is also known as a gauge.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_meter.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

nav(*text: Optional[str] = None, width: Union[tuple, int] = (100, 'px'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The HTML <nav> element represents a section of a page whose purpose is to provide navigation links, either within the current document or to other documents. Common examples of navigation sections are menus, tables of contents, and indexes.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

<https://fr.w3docs.com/apprendre-html/html-tag-nav.html> <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/nav>

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

no_tag(*text: str = "", width: Union[tuple, int] = (100, 'px'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Dummy HTML without any tag to add this to the list of a container objects.

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.

- **profile** – Boolean flag to set the profiling mode for the component

ol(*text*="", *width*: Union[tuple, int] = (100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

The tag defines an ordered list. An ordered list can be numerical or alphabetical.

Use the tag to define list items.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_ol.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

p(*text*, *width*: Union[tuple, int] = (None, '%'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

A paragraph is marked up as follows with the <p> tag: <p>This is some text in a paragraph.</p>

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_p.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

q(*text*, *width*: Union[tuple, int] = (100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

The <q> tag defines a short quotation.

Browsers normally insert quotation marks around the quotation.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_q.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

s(*text*, *width*=(100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

The <s> tag specifies text that is no longer correct, accurate or relevant.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_s.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **tooltip** – String. The tooltip to be display on the component
- **profile** – Boolean flag to set the profiling mode for the component

samp(*text*, *width*: Union[tuple, int] = (100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

The <samp> tag is a phrase tag. It defines sample output from a computer program.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_samp.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

small(*text*: str = "", *width*: Union[tuple, int] = (100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

The <small> tag defines smaller text (and other side comments).

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_small.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

span(*text*: str = "", *width*: Union[tuple, int] = (100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

The tag is an inline container used to mark up a part of a text, or a part of a document.

The tag is easily styled by CSS or manipulated with JavaScript using the class or id attribute.

The tag is much like the <div> element, but <div> is a block-level element and is an inline element.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_span.asp

Usage:

```
s2 = page.ui.tags.span(''  
    Value Formatter €  
    A Value Formatter is  
    ''', options={"multiline": True})
```

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

strong(*text*, *width*: Union[tuple, int] = (100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

The tag is a phrase tag. It defines important text.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_strong.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

sub(*text*, *width*: Union[tuple, int] = (100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

The <sub> tag defines subscript text. Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H₂O.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_sub.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

sup(text, width: Union[tuple, int] = (100, 'px'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

The <sup> tag defines superscript text. Superscript text appears half a character above the normal line, and is sometimes rendered in a smaller font. Superscript text can be used for footnotes, like WWW

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_sup.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

time(text, width: Union[tuple, int] = (100, 'px'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

The <time> tag defines a human-readable date/time.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_time.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

u(text, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Underline a misspelled word with the <u> tag: <p>This is a <u>paragraph</u>.</p>

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_u.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

var(text, width: Union[tuple, int] = (100, 'px'), height: Union[tuple, int] = (None, 'px'), html_code: Optional[str] = None, tooltip: str = "", options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

The <var> tag also supports the Global Attributes in HTML.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_var.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component

- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

wbr(*text*, *width*: Union[tuple, int] = (100, 'px'), *height*: Union[tuple, int] = (None, 'px'), *html_code*: Optional[str] = None, *tooltip*: str = "", *options*: Optional[dict] = None, *profile*: Optional[Union[bool, dict]] = None)

The <wbr> (Word Break Opportunity) tag specifies where in a text it would be ok to add a line-break.

Underlying HTML Objects:

- `epyk.core.html.HtmlTags.HtmlGeneric`

Related Pages:

https://www.w3schools.com/tags/tag_wbr.asp

Parameters

- **text** – String with the content to be added to the component
- **width** – Tuple with the width value and its unit
- **height** – Tuple with the height value and its unit
- **html_code** – String. The code reference of the component
- **tooltip** – String. The tooltip to be display on the component
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean flag to set the profiling mode for the component

TextReferences Interface

class `epyk.interfaces.components.CompTexts.TextReferences(ui)`

book(*text*, *author*=None, *name*=None, *edition*=None, *year*=None, *page*=None, *html_code*=None, *profile*=None, *options*=None)

Shortcut to quote an extra from a book.

Tags

Categories

Usage:

Related Pages:

Parameters

- **text** – String. Optional. The text of the quote.
- **author** – String. Optional. The author.
- **name** – String. Optional.
- **edition** –
- **year** –
- **page** –

- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **options** – Dictionary. Optional. Specific Python options available for this component.

github(*url=None, html_code=None, profile=None, options=None*)

Shortcut to data reference from github.

Tags

Categories

Usage:

Related Pages:

https://en.wikipedia.org/wiki/Wikipedia:Citing_sources <https://apastyle.apa.org/style-grammar-guidelines/references/examples/webpage-website-references>

Parameters

- **url** – String. Optional. The url link to the data.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **options** – Dictionary. Optional. Specific Python options available for this component.

website(*author=None, name=None, site=None, url=None, html_code=None, profile=None, options=None*)

Shortcut to data reference from another website.

Tags

Categories

Usage:

Related Pages:

https://en.wikipedia.org/wiki/Wikipedia:Citing_sources <https://apastyle.apa.org/style-grammar-guidelines/references/examples/webpage-website-references>

Parameters

- **author** – String. Optional. The author.
- **name** – String. Optional. The name of the page.
- **site** – String. Optional. The website name.
- **url** – String. Optional. The url link to the data.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **options** – Dictionary. Optional. Specific Python options available for this component.

Texts Interface

class epyk.interfaces.components.CompTexts.**Texts**(ui)

absolute(text: str, size_notch: Optional[int] = None, top: Union[tuple, int, str] = (50, '%'), left: Union[tuple, int, str] = (50, '%'), bottom: Optional[Union[tuple, int, str]] = None, align: str = 'left', width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None) → Text

Tags

Categories

Usage:

```
:param text: Optional. The value to be displayed to the component
:param size_notch:
:param top: Optional. A tuple with the integer for the component's distance to
↳ the top of the page
:param left: Optional. A tuple with the integer for the component's distance to
↳ the left of the page
:param bottom: Optional. A tuple with the integer for the component's distance
↳ to the bottom of the page
:param align: Optional. The text-align property within this component
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

alert(text: Optional[str] = None, title: Optional[str] = None, icon: Optional[str] = None, category: Optional[str] = None, color: Optional[str] = None, width: Union[tuple, int, str] = ('400', 'px'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None)

Provide contextual feedback messages for typical user actions with the handful of available and flexible alert messages.

Tags

Categories

Usage:

```
page.ui.texts.highlights("Test content", title="Test", icon="fab fa-angellist")
page.ui.texts.highlights("Server configuration at: %s" % SERVER_PATH, icon=
↳ "fas fa-exclamation-triangle")
```

Underlying HTML Objects:

- epyk.core.html.HtmlText.Highlights

Related Pages:

<https://getbootstrap.com/docs/4.3/components/alerts/>

Parameters

- **text** – Optional. The string value to be displayed in the component
- **title** – Optional.
- **icon** – Optional. The component icon content from font-awesome references
- **category** – Optional. The type of the warning. Can be (primary, secondary, success, danger, warning, info, light, dark). Default danger
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

block(*text: str = "", color: Optional[str] = None, align: str = 'left', width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, tooltip: Optional[str] = None, options: Optional[dict] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → Text

Add the HTML text component to the page.

Tags**Categories**

Usage:

```
page.ui.text("this is a test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Text`

Related Pages:

https://www.w3schools.com/tags/tag_font.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/banners.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/contextmenu.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/image.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/markdown.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/postit.py>

Parameters

- **text** – Optional. The string value to be displayed in the component.
- **color** – Optional. The color of the text.
- **align** – Optional. The position of the icon in the line (left, right, center).
- **width** – Optional. A tuple with the integer for the component width and its unit.

- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **tooltip** – Optional. A string with the value of the tooltip.
- **options** – Optional. Specific Python options available for this component.
- **helper** – Optional. The value to be displayed to the helper icon.
- **profile** – Optional. A flag to set the component performance storage.

blockquote(*text: Optional[str] = None, author: Optional[str] = None, color: Optional[str] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → BlockQuote

The <blockquote> tag specifies a section that is quoted from another source. Browsers usually indent <blockquote> elements.

Tags

Categories

Usage:

```
page.ui.texts.blockquote("This is a code")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.BlockQuote`

Related Pages:

<https://v4-alpha.getbootstrap.com/content/typography/> https://www.w3schools.com/TAGS/tag_blockquote.asp

Parameters

- **text** – Optional. The string value to be displayed in the component
- **author** – Optional. The quote's author
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. The value to be displayed to the helper icon
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

button(*text: str, icon: Optional[str] = None, width: Union[tuple, int, str] = ('auto', ''), tooltip: Optional[str] = None, height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

Templates:

Parameters

- **text** – Optional. The value to be displayed to the button
- **icon** – Optional. The component icon content from font-awesome references
- **tooltip** – Optional. A string with the value of the tooltip
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

code(*text: str = "", language: str = 'python', color: Optional[str] = None, width: Union[tuple, int, str] = (90, '%'), height: Union[tuple, int, str] = (200, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*)

Python Wrapper to the Bootstrap CODE Tag. This entry point compare to the ui.codes will be by default readonly.

Tags

Categories

Usage:

```
page.ui.texts.code("This is a code")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTextEditor.Code`

Related Pages:

<https://v4-alpha.getbootstrap.com/content/code/>

Parameters

- **text** – Optional. The string value to be displayed in the component
- **language** – Optional. The language used in the code cell
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **color** – Optional. The font color in the component. Default inherit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. The value to be displayed to the helper icon
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

col(*text: str, label: str, align: str = 'left', width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
:param text: Optional. The value to be displayed to the component
:param label: Optional. The text of label to be added to the component
:param align: Optional. The text-align property within this component
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

date(*value=None, label: Optional[str] = None, icon: str = False, color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None*)

This component is based on the JQuery Date Picker object.

Tags

Categories

Usage:

```
page.ui.texts.date('2020-04-08', label="Date").included_dates(["2020-04-08",
↳ "2019-09-06"])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlDates.DatePicker`

Related Pages:

<https://jqueryui.com/datepicker/>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/dates.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/fields.py>

Parameters

- **value** – Optional. The value to be displayed to the time component. Default now
- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. The component icon content from font-awesome references
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component
- **helper** – Optional. A tooltip helper

fieldset(*legend: str = "", width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

The <fieldset> tag is used to group related elements in a form. The <fieldset> tag draws a box around the related elements.

Tags

Categories

Usage:

```
page.ui.texts.fieldset("legend")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Fieldset`

Related Pages:

https://www.w3schools.com/tags/tag_legend.asp https://www.w3schools.com/tags/tag_fieldset.asp

Parameters

- **legend** – Optional. The legend value
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **helper** – Optional. The value to be displayed to the helper icon
- **profile** – Optional. A flag to set the component performance storage

formula(*text: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, color: Optional[str] = None, helper: Optional[str] = None, align: str = 'left', options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Formula

Interface to the mathjax Formulas object.

Tags

Categories

Usage:

```
page.ui.texts.formula("$x = \{-b \pm \sqrt{b^2-4ac} \over 2a\}.$$", helper="This is a formula")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTextComp.Formula`

Related Pages:

<https://mathjax.org/docs/index.html>

Parameters

- **text** – Optional. The string value to be displayed in the component
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **color** – Optional. The font color in the component. Default inherit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. The value to be displayed to the helper icon
- **align** – Optional. The text-align property within this component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

highlights(*text: Optional[str] = None, title: Optional[str] = None, icon: Optional[str] = None, type: str = 'danger', color: Optional[str] = None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Highlights

Provide contextual feedback messages for typical user actions with the handful of available and flexible alert messages.

Tags

Categories

Usage:

```
page.ui.texts.highlights("Test content", title="Test", icon="fab fa-angellist")
page.ui.texts.highlights("Server configuration at: %s" % SERVER_PATH, icon="fas_
↪ fa-exclamation-triangle")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Highlights`

Related Pages:

<https://getbootstrap.com/docs/4.3/components/alerts/>

Parameters

- **text** – Optional. The string value to be displayed in the component
- **title** – Optional.
- **icon** – Optional. The component icon content from font-awesome references
- **type** – Optional. The type of the warning. Can be (primary, secondary, success, danger, warning, info, light, dark). Default danger
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit

- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

label(text: str = "", color: Optional[str] = None, align: str = 'center', width: Union[tuple, int, str] = (140, 'px'), height: Union[tuple, int, str] = ('auto', ''), html_code: Optional[str] = None, tooltip: str = "", profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None) → Label

The <label> tag defines a label for a <button>, <input>, <meter>, <output>, <progress>, <select>, or <textarea> element...

The for attribute of the <label> tag should be equal to the id attribute of the related element to bind them together.

Tags Categories

Usage:

```
page.ui.texts.label("Test")
page.ui.texts.label("this is a test", color="red")
```

Underlying HTML Objects:

- epyk.core.html.HtmlText.Label

Related Pages:

https://www.w3schools.com/tags/tag_label.asp

Parameters

- **text** – Optional. The string value to be displayed in the component
- **color** – Optional. The color of the text
- **align** – Optional. The position of the icon in the line (left, right, center)
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

note(text: Optional[str] = None, title: str = "", icon: Optional[str] = None, category: str = 'success', color: Optional[str] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None) → Highlights

Provide contextual feedback messages for typical user actions with the handful of available and flexible alert messages.

Tags
Categories

Usage:

```
page.ui.texts.highlights("Test content", title="Test", icon="fab fa-angellist")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Highlights`

Related Pages:

<https://getbootstrap.com/docs/4.3/components/alerts/>

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/modal.py>
https://github.com/epykure/epyk-templates/blob/master/locals/components/popup_info.py**Parameters**

- **text** – Optional. The string value to be displayed in the component
- **title** –
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **category** – Optional. The type of the warning. Can be (primary, secondary, success, danger, warning, info, light, dark). Default danger
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. The value to be displayed to the helper icon
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

number(*number: int = 0, title: Optional[str] = None, label: Optional[str] = None, icon: Optional[str] = None, color: Optional[str] = None, align: str = 'left', tooltip: str = "", html_code=None, options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, width: Union[tuple, int, str] = (150, 'px'), profile: Optional[Union[bool, dict]] = None*)

Tags
Categories

Usage:

```
page.ui.texts.number(289839898, label="test", helper="Ok", icon="fas fa-align-  
↔center")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Numeric`

Related Pages:

<http://openexchangerates.github.io/accounting.js/>

Parameters

- **number** – Optional. The value to be displayed to the component. Default 0
- **title** – Optional. The text title
- **label** – Optional. The text of label to be added to the component
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **color** – Optional. The font color in the component. Default inherit
- **align** – Optional. The text-align property within this component
- **width** – Optional. A tuple with the integer for the component width and its unit
- **tooltip** – Optional. A string with the value of the tooltip
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **helper** – Optional. The value to be displayed to the helper icon
- **profile** – Optional. A flag to set the component performance storage

paragraph(text: str = "", color: Optional[str] = None, background_color: Optional[str] = None, border: bool = False, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, encoding: str = 'UTF-8', helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None) → Paragraph

Python Wrapper to the HTML P Tag.

Tags

Categories

Usage:

```
page.ui.texts.paragraph("This is a paragraph", helper="Paragraph helper")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Paragraph`

Related Pages:

https://www.w3schools.com/html/html_styles.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/paragraph.py>

Parameters

- **text** – Optional. The string value to be displayed in the component
- **color** – Optional. The font color in the component. Default inherit
- **background_color** –
- **border** –
- **width** – Optional. A tuple with the integer for the component width and its unit

- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **encoding** – Optional.
- **helper** – Optional. The value to be displayed to the helper icon
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

preformat(*text: Optional[str] = None, color: Optional[str] = None, width: Union[tuple, int, str] = (90, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → Pre

Preformatted text: The <pre> tag defines preformatted text. Text in a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks.

Tags

Categories

Usage:

```
page.ui.texts.preformat("This is a pre formatted text")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Pre`

Related Pages:

https://www.w3schools.com/html/html_styles.asp https://www.w3schools.com/tags/tag_pre.asp

Parameters

- **text** – Optional. The string value to be displayed in the component
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. The value to be displayed to the helper icon
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

property references: [TextReferences](#)

More custom toggles icons.

span(*text: str = "", color: Optional[str] = None, align: str = 'center', width: Optional[Union[tuple, int, str]] = None, height: Optional[Union[tuple, int, str]] = None, html_code: Optional[str] = None, tooltip: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*) → Span

The `` tag is used to group inline-elements in a document.

The `` tag provides no visual change by itself.

The `` tag provides a way to add a hook to a part of a text or a part of a document.

Tags

Categories

Usage:

```
page.ui.texts.span("Test")

span = page.ui.texts.span("youpi")
span.mouse([
    span.dom.css("color", "red"),
    span.dom.css("cursor", "pointer").r],
    span.dom.css("color", "blue").r)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Span`

Related Pages:

https://www.w3schools.com/tags/tag_span.asp

Parameters

- **text** – Optional. The string value to be displayed in the component
- **color** – Optional. The color of the text
- **align** – Optional. The position of the icon in the line (left, right, center)
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **tooltip** – Optional. A string with the value of the tooltip
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component

text(*text: str = "", color: Optional[str] = None, align: str = 'left', width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, tooltip: Optional[str] = None, options: Optional[dict] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*) → Text

Add the HTML text component to the page.

Usage:

```
page.ui.text("this is a test")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Text`

Related Pages:

https://www.w3schools.com/tags/tag_font.asp

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/banners.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/contextmenu.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/image.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/markdown.py>
<https://github.com/epykure/epyk-templates/blob/master/locals/components/postit.py>

Parameters

- **text** – The string value to be displayed in the component.
- **color** – Optional. The color of the text.
- **align** – Optional. The position of the icon in the line (left, right, center).
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **tooltip** – Optional. A string with the value of the tooltip.
- **options** – Optional. The component options.
- **helper** – Optional. A tooltip helper.
- **profile** – A flag to set the component performance storage.

title(*text: Union[str, dict] = "", level=None, name: Optional[str] = None, contents=None, color=None, picture: Optional[str] = None, icon: Optional[str] = None, top: int = 5, html_code: Optional[str] = None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), align: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Add a title.

Tags

Categories

Usage:

```
page.ui.title("Test")
page.ui.title("Test", level=2)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Title`

Related Pages:

https://www.w3schools.com/tags/tag_hn.asp

Parameters

- **text** – Optional. The value to be displayed to the component.
- **level** –
- **name** –

- **contents** –
- **color** – Optional. The font color in the component. Default inherit.
- **picture** –
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **top** – Optional. The margin top in pixel
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **align** – Optional. The text-align property within this component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

up_down(*record=None, components=None, color=None, label: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, helper: Optional[str] = None, profile: Optional[Union[bool, dict]] = None*)

Up and down Text component.

Tags

Categories

Usage:

```
page.ui.texts.up_down({'previous': 240885, 'value': 240985})
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.UpDown`

Related Pages:

<https://fontawesome.com/>

Parameters

- **record** – Optional. The component inputs
- **components** – List of HTML component to be added
- **color** – Optional. The font color in the component. Default inherit
- **label** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **helper** – Optional. The value to be displayed to the helper icon
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

Titles Interface

class epyk.interfaces.components.CompTitles.**Titles**(ui)

bold(text: str = "", options: Optional[dict] = None, tooltip: str = "", align: str = 'left', color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = ('auto', ''), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = False)

tags

categories

Usage:

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py>

- **param text**

Optional. The value to be displayed to the component

param options

Optional. Specific Python options available for this component

param tooltip

Optional. A string with the value of the tooltip

param align

Optional. The text-align property within this component

param color

Optional. The font color in the component. Default inherit

param width

Optional. A tuple with the integer for the component width and its unit

param height

Optional. A tuple with the integer for the component height and its unit

param html_code

Optional. An identifier for this component (on both Python and Javascript side)

param profile

Optional. A flag to set the component performance storage

caption(text: Union[str, dict] = "", options: Optional[dict] = None, tooltip: str = "", align: str = 'left', color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = ('auto', ''), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = False)

tags

categories

Usage:

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py>

- **param text**

Optional. The value to be displayed to the component

param options

Optional. Specific Python options available for this component

param tooltip

Optional. A string with the value of the tooltip

param align

Optional. The text-align property within this component

param color

Optional. The font color in the component. Default inherit

param width

Optional. A tuple with the integer for the component width and its unit

param height

Optional. A tuple with the integer for the component height and its unit

param html_code

Optional. An identifier for this component (on both Python and Javascript side)

param profile

Optional. A flag to set the component performance storage

category(*text: Union[str, dict] = "", options: Optional[dict] = None, tooltip: str = "", align: str = 'left', color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = ('auto', ''), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = False*)

tags**categories**

Usage:

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py>

- **param text**

Optional. The value to be displayed to the component

param options

Optional. Specific Python options available for this component

param tooltip

Optional. A string with the value of the tooltip

param align

Optional. The text-align property within this component

param color

Optional. The font color in the component. Default inherit

param width

Optional. A tuple with the integer for the component width and its unit

param height

Optional. A tuple with the integer for the component height and its unit

param html_code

Optional. An identifier for this component (on both Python and Javascript side)

param profile

Optional. A flag to set the component performance storage

head(*text: Union[str, dict] = "", options: Optional[dict] = None, tooltip: str = "", align: str = 'left', color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = ('auto', ''), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = False*)

Tags

Categories

Usage:

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py>

Parameters

- **text** – Optional. The value to be displayed to the component
- **options** – Optional. Specific Python options available for this component
- **tooltip** – Optional. A string with the value of the tooltip
- **align** – Optional. The text-align property within this component
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage

headline(*text: Union[str, dict] = "", options: Optional[dict] = None, tooltip: str = "", align: str = 'left', color: bool = True, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = ('auto', ''), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = False*)

Tags

Categories

Usage:

```
page.ui.titles.headline("Daily")
```

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/calendar.py> <https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py>

Parameters

- **text** – Optional. The value to be displayed to the component
- **options** – Optional. Specific Python options available for this component
- **tooltip** – Optional. A string with the value of the tooltip
- **align** – Optional. The text-align property within this component
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage

rubric(*text: Union[str, dict] = "", options: Optional[dict] = None, tooltip: str = "", align: str = 'left', color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = ('auto', ""), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = False*)

Tags

Categories

Usage:

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py>

Parameters

- **text** – Optional. The value to be displayed to the component
- **options** – Optional. Specific Python options available for this component
- **tooltip** – Optional. A string with the value of the tooltip
- **align** – Optional. The text-align property within this component
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage

section(*text: Union[str, dict] = "", options: Optional[dict] = None, tooltip: str = "", align: str = 'left', color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = ('auto', ""), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = False*)

Tags

Categories

Usage:

```
t0 = page.ui.titles.section("Available Items")
```

Parameters

- **text** – Optional. The value to be displayed to the component
- **options** – Optional. Specific Python options available for this component
- **tooltip** – Optional. A string with the value of the tooltip
- **align** – Optional. The text-align property within this component
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage

subtitle(*text: str = "", name: Optional[str] = None, contents=None, color: Optional[str] = None, picture: Optional[str] = None, icon: Optional[str] = None, top: int = 5, html_code: Optional[str] = None, width: Union[tuple, int, str] = ('auto', ''), height: Union[tuple, int, str] = (None, 'px'), align: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

tags

categories

Usage:

- **param text**
Optional. The value to be displayed to the component
- param name**
- param contents**
- param color**
Optional. The font color in the component. Default inherit
- param picture**
- param icon**
- param top**
- param html_code**
Optional. An identifier for this component (on both Python and Javascript side)
- param width**
Optional. A tuple with the integer for the component width and its unit
- param height**
Optional. A tuple with the integer for the component height and its unit
- param align**
Optional. The text-align property within this component
- param options**
Optional. Specific Python options available for this component
- param profile**
Optional. A flag to set the component performance storage

title(*text: Optional[Union[str, dict]] = None, options: Optional[dict] = None, tooltip: str = "", align: str = 'left', color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = ('auto', ''), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = False*)

Tags

Categories

Usage:

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py>
[//github.com/epykure/epyk-templates/blob/master/locals/components/paragraph.py](https://github.com/epykure/epyk-templates/blob/master/locals/components/paragraph.py)

[https:](https://github.com/epykure/epyk-templates/blob/master/locals/components/paragraph.py)

Parameters

- **text** – Optional. The value to be displayed to the component
- **options** – Optional. Specific Python options available for this component
- **tooltip** – Optional. A string with the value of the tooltip
- **align** – Optional. The text-align property within this component
- **color** – Optional. The font color in the component. Default inherit
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage

underline(*text: Union[str, dict] = "", options: Optional[dict] = None, tooltip: str = "", align: str = 'left', color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = ('auto', ''), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = False*)

tags
categories

Usage:

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py>

- **param text**
Optional. The value to be displayed to the component
- param options**
Optional. Specific Python options available for this component
- param tooltip**
Optional. A string with the value of the tooltip
- param color**
Optional. The font color in the component. Default inherit
- param align**
Optional. The text-align property within this component
- param width**
Optional. A tuple with the integer for the component width and its unit
- param height**
Optional. A tuple with the integer for the component height and its unit
- param html_code**
Optional. An identifier for this component (on both Python and Javascript side)
- param profile**
Optional. A flag to set the component performance storage

upper(*text: Union[str, dict] = "", options: Optional[dict] = None, tooltip: str = "", align: str = 'left', color: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = ('auto', ''), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] = False*)

tags
categories

Usage:

```
page.ui.titles.upper("Test")
page.ui.titles.upper("Test", color=True)
```

Templates:

```
https://github.com/epykure/epyk-templates/blob/master/locals/components/list.py
https://github.com/epykure/epyk-templates/blob/master/locals/components/paragraph.py
```

- **param text**
Optional. The value to be displayed to the component
- param options**
Optional. Specific Python options available for this component
- param tooltip**
Optional. A string with the value of the tooltip
- param align**
Optional. The text-align property within this component
- param color**
Optional. The font color in the component. Default inherit
- param width**
Optional. A tuple with the integer for the component width and its unit
- param height**
Optional. A tuple with the integer for the component height and its unit
- param html_code**
Optional. An identifier for this component (on both Python and Javascript side)
- param profile**
Optional. A flag to set the component performance storage

Trees Interface

```
class epyk.interfaces.components.CompTrees.Trees(ui)
```

```
dropdown(record: Optional[List[dict]] = None, text: str = "", width: Union[tuple, int, str] = (100, '%'),
height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper:
Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile:
Optional[Union[bool, dict]] = None) → DropDown
```

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlTrees.DropDown`

Related Pages:

```
http://getbootstrap.com/docs/4.0/components/dropdowns/      https://www.w3schools.com/
bootstrap/tryit.asp?filename=trybs_ref_js_dropdown_multilevel_css&stacked=h      https://codepen.io/svnt/pen/beEgre
```

<https://codepen.io/raneio/pen/NbbZEM> https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_js_dropdown_hover <https://codepen.io/antoniputra/pen/BzyWmb>

Parameters

- **record** – Optional. The records
- **text** – Optional. Dropdown label
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

folder(*folder: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Tree

Add a tree component from a folder structure.

Parameters

- **folder** – The path to be displayed
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

inputs(*data: Optional[List[dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → TreeInput

Usage:

Underlying HTML Objects:

- `epyk.core.html.HtmlTrees.TreeInput`

Parameters

- **data** – Optional. The records
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper

- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

tree(data: Optional[List[dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None) → Tree

Add a tree / hierarchy component to the page.

Usage:

```
data = [{"value": 'test', 'items': [{"value": 'child 1', 'color': 'red'}]}]
page.ui.lists.tree(data)

data = [{"value": 'test', 'icon': "fas fa-check", "css": {'color': 'green'},
↪ 'items': [{
    "value": 'child 1', "css": {'color': 'red'}, 'icon': "fas fa-times"}]},
    {"value": 'test 2', 'icon': "fas fa-exclamation-triangle", "css": {
↪ 'color': 'orange'}, 'items': [{
    "value": 'child 1', "css": {'color': 'red'}, 'icon': "fas fa-times"}]},
    {}]}

hyr = page.ui.tree(data)
hyr.options.icon_close = "fas fa-caret-right"
hyr.options.icon_open = "fas fa-caret-down"
hyr.options.with_badge = True
hyr.options.with_icon = "icon"
hyr.click_node([page.js.alert(pk.events.value)])
hyr.click([page.js.alert(pk.events.value)])
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTrees.Tree`

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/tree.py>

Parameters

- **data** – Optional. The records
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. A tooltip helper
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

Vignets Interface

class epyk.interfaces.components.CompVignets.**Vignets**(ui)

background(url: str, width: Union[tuple, int] = (90, '%'), height: Union[tuple, int] = (450, 'px'), size: str = 'contain', margin: int = 0, align: str = 'center', position: str = 'middle', options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Tags

Categories

Usage:

```
:param url: String. The url string.
:param align: String. Optional. The text-align property within this component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳ height and its unit.
:param size: String. Optional.
:param margin: Integer. Optional.
:param position: String. Optional.
:param options: Dictionary. Optional. Specific Python options available for.
↳ this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳ performance storage.
```

block(records=None, color: Optional[str] = None, border: str = 'auto', width: Union[tuple, int] = (300, 'px'), height: Union[tuple, int] = (None, 'px'), helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Every HTML element has a default display value depending on what type of element it is. The two display values are: block and inline.

Tags

Categories

Usage:

```
page.ui.vignets.block({"text": 'This is a brand new python framework', "title":
↳ 'New Python Web Framework',
                        "button": {"text": 'Get Started', 'url': "/getStarted"},
↳ 'color': 'green'})
```

Underlying HTML Objects:

- epyk.core.html.HtmlTextComp.BlockText

Related Pages:

https://www.w3schools.com/html/html_blocks.asp

Parameters

- **records** – List. Optional. The list of dictionaries with the input data.
- **color** – String. Optional. The font color in the component. Default inherit.
- **border** –

- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **helper** – String. Optional. The value to be displayed to the helper icon.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

bubble(*records=None, width: Union[tuple, int] = (70, 'px'), height: Union[tuple, int] = ('auto', ''), color: Optional[str] = None, background_color: Optional[str] = None, helper: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

The bubbles event property returns a Boolean value that indicates whether or not an event is a bubbling event. Event bubbling directs an event to its intended target, it works like this: A button is clicked and the event is directed to the button. If an event handler is set for that object, the event is triggered. If no event handler is set for that object, the event bubbles up (like a bubble in water) to the object's parent. The event bubbles up from parent to parent until it is handled, or until it reaches the document object.

Tags

Categories

Usage:

```
page.ui.vignets.bubble({"value": 23, "title": "Title"}, helper="This is a helper  
→")
```

Underlying HTML Objects:

- `epyk.core.html.HtmlText.Text`
- `epyk.core.html.HtmlContainer.Div`
- `epyk.core.html.HtmlLinks.ExternalLink`

Related Pages:

https://www.w3schools.com/jsref/event_bubbles.asp

Parameters

- **records** – List. Optional. The list of dictionaries.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **color** – String. Optional. The font color in the component. Default inherit.
- **background_color** – String. Optional. The hexadecimal color code.
- **helper** – String. Optional. The value to be displayed to the helper icon.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

image(*title: Optional[str] = None, content: str = "", image: Optional[str] = None, render: str = 'row', align: str = 'center', width: Union[tuple, int] = (90, '%'), height: Union[tuple, int] = (None, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
:param title: String. Optional. A panel title. This will be attached to the
↳title property.
:param content:
:param image:
:param render:
:param align: String. Optional. The text-align property within this component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳performance storage.
```

number(*number: float, label: str = "", title: Optional[str] = None, align: str = 'center', components=None, width: Union[tuple, int] = ('auto', ''), height: Union[tuple, int] = (None, 'px'), profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, helper: Optional[str] = None*)

The <input type="number"> defines a field for entering a number. Use the following attributes to specify restrictions: max - specifies the maximum value allowed min - specifies the minimum value allowed step - specifies the legal number intervals value - Specifies the default value

Tags

Categories

Usage:

```
number = page.ui.vignets.number(500, "Test")
number_2 = page.ui.vignets.number(500, "Test 2 ", options={"url": "http://www.
↳google.fr"})
number.span.add_icon(page.ui.icons.get.ICON_ENVELOPE)
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTextComp.Number`

Related Pages:

https://www.w3schools.com/tags/att_input_type_number.asp

Parameters

- **number** – Integer. The value.
- **label** – String. Optional. The label text.
- **title** – String | Component. Optional. A panel title. This will be attached to the title property.
- **align** – String. Optional. The text-align property within this component.
- **components** – List. Optional. The HTML components.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **helper** – String. Optional. The value to be displayed to the helper icon.

price(value, title: str, items: Optional[List[Html]] = None, components: Optional[List[Html]] = None, url: Optional[str] = None, align: str = 'center', width: Union[tuple, int] = (250, 'px'), height: Union[tuple, int] = ('auto', ''), currency: str = '£', options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None, helper: Optional[str] = None)

Tags

Categories

Usage:

```
page.ui.vignets.price(10, "This is the price", [])
```

Parameters

- **value** –
- **title** – String. Optional. A panel title. This will be attached to the title property.
- **items** –
- **components** –
- **url** –
- **align** – String. Optional. The text-align property within this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **currency** – String. Optional. The currency value.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **helper** – String. Optional. The value to be displayed to the helper icon.

slides(start: int = 0, width: Union[tuple, int] = (100, '%'), height: Union[tuple, int] = (100, '%'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Tags

Categories

Usage:

```
:param start:
:param width: Tuple. Optional. A tuple with the integer for the component width
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳ this component.
```

(continues on next page)

(continued from previous page)

```
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳ performance storage.
```

TODO: Fix layout issue with F11 (full screen)

```
text(records=None, width: Union[tuple, int] = (None, '%'), height: Union[tuple, int] = (None, 'px'), align:
    str = 'center', helper: Optional[str] = None, options: Optional[dict] = None, profile:
    Optional[Union[bool, dict]] = None)
```

Tags

Categories

Usage:

```
page.ui.vignets.text({"title": "New Python Framework", 'value': "A new Python
↳ Web Framework", 'color': 'green',
                    'icon': 'fab fa-python', 'colorTitle': 'darkgreen'})
```

Underlying HTML Objects:

- `epyk.core.html.HtmlTextComp.TextWithBorder`

Parameters

- **records** – List. Optional. The list of dictionaries with the input data.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **align** – String. Optional. The text-align property within this component.
- **helper** – String. Optional. The value to be displayed to the helper icon.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

```
video(title: str, content: str = "", video: Optional[str] = None, render: str = 'row', align: str = 'center', width:
    Union[tuple, int] = (90, '%'), height: Union[tuple, int] = (None, 'px'), options: Optional[dict] = None,
    profile: Optional[Union[bool, dict]] = None)
```

Component to allow creation of a vignet embedding a video.

Tags

Categories

Usage:

Related Pages:

Issue:

<https://github.com/epykure/epyk-ui/issues/92>

param title

String. Optional. A panel title. This will be attached to the title property.

param content**param video****param render****param align**

String. The text-align property within this component.

param width

Tuple. Optional. A tuple with the integer for the component width and its unit.

param height

Tuple. Optional. A tuple with the integer for the component height and its unit.

param options

Dictionary. Optional. Specific Python options available for this component.

param profile

Boolean | Dictionary. Optional. A flag to set the component performance storage.

vignet(title: str, content: str, icon: Optional[str] = None, render: str = 'col', align: str = 'center', width: Union[tuple, int] = (200, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Tags**Categories**

Usage:

```
:param title: String. Optional. A panel title. This will be attached to the_
↳title property.
:param content: String. Optional. The value to be displayed to the component.
:param icon: String. Optional. A string with the value of the icon to display_
↳from font-awesome.
:param render:
:param align: String. Optional. The text-align property within this component.
:param width: Tuple. Optional. A tuple with the integer for the component width_
↳and its unit.
:param options: Dictionary. Optional. Specific Python options available for_
↳this component.
:param profile: Boolean | Dictionary. Optional. A flag to set the component_
↳performance storage.
```

geo Interface

Geo Interface

class `epyk.interfaces.geo.CompGeo.Geo(ui)`

property `chartJs`: *ChartJs*

Property to the ChartJs Geo API.

Usage:

Related Pages:

<https://github.com/sgratzl/chartjs-chart-geo>

property `d3`: *D3*

Interactive maps for data visualizations. Bundled into a single Javascript file.

Related Pages:

<https://github.com/markmarkoh/datamaps>

property `google`: *GeoGoogle*

Property to the Google charts API.

Usage:

Related Pages:

<https://developers.google.com/chart>

property `jqv`: *JqueryVectorMap*

Property to the Jquery vector Map API.

Usage:

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

property `leaflet`: *GeoLeaflet*

Property to the Jquery vector Map API.

Usage:

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

property `mapbox`: *MapboxMaps*

A JavaScript library that uses WebGL to render interactive maps from vector tiles and Mapbox styles.

Usage:

```
page.imports.pkgs.mapbox.set_access_token(
    "XXXXXX",
    "mapboxgl.accessToken"
)

l = page.ui.geo.mapbox.globe() # page.ui.geo.leaflet.europe()
l.load([
```

(continues on next page)

(continued from previous page)

```

1.js.addControl([l.js._.FullscreenControl()]),
1.js.addControl([l.js._.GeolocateControl()]),
1.js.addControl([l.js._.NavigationControl()]),
1.js.addControl([l.js._.ScaleControl()]),
1.js.addSource('portland', {'type': 'raster', 'url': 'mapbox://examples.
↪32xkp0wd'}),
1.js.addLayer({'id': 'portland', 'source': 'portland', 'type': 'raster'}),
1.js.addSource('route', {'type': 'geojson', 'data': {'type': 'Feature',
↪'properties': {}, 'geometry': {
  'type': 'LineString', 'coordinates': [
    [-122.483696, 37.833818], [-122.483482, 37.833174], [-122.483396, 37.
↪8327], [-122.483568, 37.832056]
    [-122.48404, 37.831141], [-122.48404, 37.830497], [-122.483482, 37.82992],
↪ [-122.483568, 37.829548],
    [-122.48507, 37.829446], [-122.4861, 37.828802], [-122.486958, 37.82931],
↪ [-122.487001, 37.830802],
    [-122.487516, 37.831683], [-122.488031, 37.832158], [-122.488889, 37.
↪832971], [-122.489876, 37.832632],
    [-122.490434, 37.832937], [-122.49125, 37.832429], [-122.491636, 37.
↪832564], [-122.492237, 37.833378],
    [-122.493782, 37.833683]]}}}),
1.js.addLayer({'id': 'route', 'type': 'line', 'source': 'route', 'layout': {
  'line-join': 'round', 'line-cap': 'round'}, 'paint': {'line-color': '#888',
↪'line-width': 8}})
])
l.options.style = 'mapbox://styles/mapbox/streets-v11'

```

Related Pages:

<https://docs.mapbox.com/>**property plotly:** *Plotly*

Usage:

Related Pages:

<https://plotly.com/javascript/choropleth-maps/>**property plotly_map:** *Plotly*

Interface for the Plotly library.

Usage:

Related Pages:

<https://plotly.com/javascript/choropleth-maps/>

BubbleMaps Interface

class epyk.interfaces.geo.CompGeoChartJs.**BubbleMaps**(ui)

us(record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None)

Underlying HTML Objects:

- epyk.core.geo.GeoChartJs.Choropleth

Parameters

- **record** –
- **y_columns** –
- **x_axis** –
- **profile** –
- **options** –
- **width** –
- **height** –
- **html_code** –

world(record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None)

Underlying HTML Objects:

- epyk.core.geo.GeoChartJs.Choropleth

Parameters

- **record** –
- **y_columns** –
- **x_axis** –
- **profile** –
- **options** –
- **width** –
- **height** –
- **html_code** –

ChartJs Interface

```
class epyk.interfaces.geo.CompGeoChartJs.ChartJs(ui)
```

Choropleth Interface

```
class epyk.interfaces.geo.CompGeoChartJs.Choropleth(ui)
```

country(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None*)

Usage:

```
records = [{"name": "Liverpool", "value": 23}, {"name": "Leeds", "value": 60}]
uk = page.ui.geo.chartJs.choropleths.uk(records, y_columns=["value"], x_axis=
→ "name")
page.ui.button("Click").click([
    uk.build({"Birmingham": 40}),
])
```

Underlying HTML Objects:

- `epyk.core.geo.GeoChartJs.Choropleth`

Parameters

- **record** –
- **y_columns** –
- **x_axis** –
- **profile** –
- **options** –
- **width** –
- **height** –
- **html_code** –

uk(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None*)

Usage:

```
records = [{"name": "Liverpool", "value": 23}, {"name": "Leeds", "value": 60}]
uk = page.ui.geo.chartJs.choropleths.uk(records, y_columns=["value"], x_axis=
→ "name")
page.ui.button("Click").click([
    uk.build({"Birmingham": 40}),
])
```

Parameters

- **record** –

- **y_columns** –
- **x_axis** –
- **profile** –
- **options** –
- **width** –
- **height** –
- **html_code** –

Returns

us(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None*)

Usage:

```
records = [{"name": "Nevada", "value": 23}, {"name": "Texas", "value": 60}]
us = page.ui.geo.chartJs.choropleths.us(records, y_columns=["value"], x_axis=
↪ "name")
page.ui.button("Click").click([us.build({"Louisiana": 40})])
```

Underlying HTML Objects:

- `epyk.core.geo.GeoChartJs.Choropleth`

Parameters

- **record** –
- **y_columns** –
- **x_axis** –
- **profile** –
- **options** –
- **width** –
- **height** –
- **html_code** –

world(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None*)

Usage:

```
records = [{"name": "Italy", "value": 23}]
wl = page.ui.geo.chartJs.choropleths.world(records, y_columns=["value"], x_axis=
↪ "name")
page.ui.button("Click").click([
    wl.build({"Germany": 23, "Spain": 40, "Italy": 23}),
    us.build({"Louisiana": 40}),
    uk.build({"Birmingham": 40}),
])
```

Underlying HTML Objects:

- `epyk.core.geo.GeoChartJs.Choropleth`

Parameters

- `record` –
- `y_columns` –
- `x_axis` –
- `profile` –
- `options` –
- `width` –
- `height` –
- `html_code` –

D3 Interface

```
class epyk.interfaces.geo.CompGeoD3.D3(ui)
```

Dc Interface

```
class epyk.interfaces.geo.CompGeoDc.Dc(ui)
```

```
usa(record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, title: Optional[str] =  
None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int,  
tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None)
```

Underlying HTML Objects:

- `epyk.core.geo.GeoDc.ChartGeoChoroplethk`

Related Pages:

https://jsfiddle.net/djmartin_umich/9VJHe/ <http://bl.ocks.org/KatiRG/cccd23dd7a830da0de5c>

Parameters

- `record` –
- `y_columns` –
- `x_axis` –
- `title` –
- `profile` –
- `options` –
- `width` –
- `height` –
- `html_code` –

GeoGoogle Interface

class epyk.interfaces.geo.CompGeoGoogle.**GeoGoogle**(ui)

current(profile: *Optional[Union[bool, dict]] = None*, options: *Optional[dict] = None*, width: *Union[int, tuple] = (100, '%')*, height: *Union[int, tuple] = (330, 'px')*, html_code: *Optional[str] = None*)

Parameters

- **profile** –
- **options** –
- **width** –
- **height** –
- **html_code** –

maps(latitude: *float*, longitude: *float*, profile: *Optional[Union[bool, dict]] = None*, options: *Optional[dict] = None*, width: *Union[int, tuple] = (100, '%')*, height: *Union[int, tuple] = (330, 'px')*, html_code: *Optional[str] = None*)

Parameters

- **latitude** –
- **longitude** –
- **profile** –
- **options** –
- **width** –
- **height** –
- **html_code** –

satellite(latitude: *float*, longitude: *float*, profile: *Optional[Union[bool, dict]] = None*, options: *Optional[dict] = None*, width: *Union[int, tuple] = (100, '%')*, height: *Union[int, tuple] = (330, 'px')*, html_code: *Optional[str] = None*)

Parameters

- **latitude** –
- **longitude** –
- **profile** –
- **options** –
- **width** –
- **height** –
- **html_code** –

terrain(latitude: *float*, longitude: *float*, profile: *Optional[Union[bool, dict]] = None*, options: *Optional[dict] = None*, width: *Union[int, tuple] = (100, '%')*, height: *Union[int, tuple] = (330, 'px')*, html_code: *Optional[str] = None*)

Parameters

- **latitude** –

- **longitude** –
- **profile** –
- **options** –
- **width** –
- **height** –
- **html_code** –

JqueryVertorMap Interface

class epyk.interfaces.geo.CompGeoJqV.**JqueryVertorMap**(*ui*)

add_map(*name: str, continent: bool = False*)

Add the Javascript external package defining the map to the page.

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

Parameters

- **name** – String. The map alias.
- **continent** – Boolean. Optional.

africa(*record=None, y_column: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None*)

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

Usage:

```
:param record: List. Optional. The records
:param y_column: String. Optional. The column in the record for the keys.
:param x_axis: String. Optional. The column in the record for the values.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param options: Dictionary. Optional. Specific Python options available for
↳this component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit.
:param html_code: String. Optional. An identifier for this component (on both
↳Python and Javascript side).
```

asia(*record=None, y_column: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None*)

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

Usage:

```
:param record: List. Optional. The records
:param y_column: String. Optional. The column in the record for the keys.
:param x_axis: String. Optional. The column in the record for the values.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit.
:param html_code: String. Optional. An identifier for this component (on both.
↳Python and Javascript side).
```

australia(record=None, y_column: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None)

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

Usage:

```
:param record: List. Optional. The records
:param y_column: String. Optional. The column in the record for the keys.
:param x_axis: String. Optional. The column in the record for the values.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit.
:param html_code: String. Optional. An identifier for this component (on both.
↳Python and Javascript side).
```

europe(record=None, y_column: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None)

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

Usage:

```
:param record: List. Optional. The records
:param y_column: String. Optional. The column in the record for the keys.
:param x_axis: String. Optional. The column in the record for the values.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
```

(continues on next page)

(continued from previous page)

```
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit.
:param html_code: String. Optional. An identifier for this component (on both.
↳Python and Javascript side).
```

france(record=None, y_column: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None)

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

Usage:

```
:param record: List. Optional. The records
:param y_column: String. Optional. The column in the record for the keys.
:param x_axis: String. Optional. The column in the record for the values.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit.
:param html_code: String. Optional. An identifier for this component (on both.
↳Python and Javascript side).
```

germany(record=None, y_column: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None)

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

Usage:

```
:param record: List. Optional. The records
:param y_column: String. Optional. The column in the record for the keys.
:param x_axis: String. Optional. The column in the record for the values.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit.
:param html_code: String. Optional. An identifier for this component (on both.
↳Python and Javascript side).
```

map(name: str, record=None, y_column: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None)

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

Usage:

```
:param name: String. The map alias.
:param record: List. Optional. The records
:param y_column: String. Optional. The column in the record for the keys.
:param x_axis: String. Optional. The column in the record for the values.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit.
:param html_code: String. Optional. An identifier for this component (on both.
↳Python and Javascript side).
```

north_america(record=None, y_column: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None)

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

Usage:

```
:param record: List. Optional. The records
:param y_column: String. Optional. The column in the record for the keys.
:param x_axis: String. Optional. The column in the record for the values.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit.
:param html_code: String. Optional. An identifier for this component (on both.
↳Python and Javascript side).
```

south_america(record=None, y_column: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None)

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

Usage:

```
:param record: List. Optional. The records
:param y_column: String. Optional. The column in the record for the keys.
:param x_axis: String. Optional. The column in the record for the values.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit.
:param html_code: String. Optional. An identifier for this component (on both.
↳Python and Javascript side).
```

usa(*record=None, y_column: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None*)

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

Usage:

```
:param record: List. Optional. The records
:param y_column: String. Optional. The column in the record for the keys.
:param x_axis: String. Optional. The column in the record for the values.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit.
:param html_code: String. Optional. An identifier for this component (on both.
↳Python and Javascript side).
```

world(*record=None, y_column: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int, tuple] = (330, 'px'), html_code: Optional[str] = None*)

Related Pages:

<https://www.10bestdesign.com/jqvmap/>

Usage:

```
:param record: List. Optional. The records
:param y_column: String. Optional. The column in the record for the keys.
:param x_axis: String. Optional. The column in the record for the values.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
:param width: Tuple. Optional. A tuple with the integer for the component width.
```

(continues on next page)

(continued from previous page)

```

↪and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↪height and its unit.
:param html_code: String. Optional. An identifier for this component (on both
↪Python and Javascript side).

```

GeoLeaflet Interface

```
class epyk.interfaces.geo.CompGeoLeaflet.GeoLeaflet(ui)
```

Plotly Interface

```
class epyk.interfaces.geo.CompGeoPlotly.Plotly(ui)
```

property bubbles

Plotly Bubble charts

Related Pages:

<https://plotly.com/javascript/bubble-maps/>

```

chorolet(record, y_columns: Optional[list] = None, x_axis: Optional[str] = None, title: Optional[str] =
None, options: Optional[dict] = None, width: Union[int, tuple] = (100, '%'), height: Union[int,
tuple] = (330, 'px'), html_code: Optional[str] = None, profile: Optional[Union[bool, dict]] =
None)

```

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.Chorolet`

Related Pages:

<https://plotly.com/javascript/mapbox-county-choropleth/>

Parameters

- **record** –
- **y_columns** –
- **x_axis** –
- **title** –
- **profile** –
- **options** –
- **width** –
- **height** –
- **html_code** –

property choropleths

Plotly Choropleth charts

Related Pages:

<https://plotly.com/javascript/mapbox-county-choropleth/>

density(*record*, *y_columns*: *Optional*[*list*] = *None*, *x_axis*: *Optional*[*str*] = *None*, *profile*:
Optional[*Union*[*bool*, *dict*]] = *None*, *options*: *Optional*[*dict*] = *None*, *width*=(100, '%'),
height=(330, 'px'), *html_code*=*None*)

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.Scatter`

Related Pages:

<https://plot.ly/javascript/mapbox-density-heatmaps/>

Parameters

- **record** –
- **y_columns** –
- **x_axis** –
- **profile** –
- **options** –
- **width** –
- **height** –
- **html_code** –

scattermapbox(*record*, *lon_columns*: *Optional*[*list*] = *None*, *lat_columns*: *Optional*[*list*] = *None*,
text_columns: *Optional*[*list*] = *None*, *profile*: *Optional*[*Union*[*bool*, *dict*]] = *None*, *options*:
Optional[*dict*] = *None*, *width*: *Union*[*int*, *tuple*] = (100, '%'), *height*: *Union*[*int*, *tuple*] =
(430, 'px'), *html_code*: *Optional*[*str*] = *None*)

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.Scatter`

Related Pages:

<https://plot.ly/javascript/mapbox-layers/>

Parameters

- **record** –
- **lon_columns** –
- **lat_columns** –
- **text_columns** –
- **profile** –
- **options** –
- **width** –
- **height** –
- **html_code** –

PlotlyBubble Interface

class epyk.interfaces.geo.CompGeoPlotly.**PlotlyBubble**(ui)

africa(*record=None, size_col=None, country_col=None, long_col=None, lat_col=None, profile=None, options=None, width=(100, '%'), height=(430, 'px'), html_code=None*)

A bubble chart for Africa

Underlying HTML Objects:

- epyk.core.geo.GeoPlotly.BubbleGeo

Related Pages:

<https://plotly.com/javascript/bubble-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **long_col** – String. The column in the recordset used to retrieve the longitude (optional if country codes)
- **lat_col** – String. The column in the recordset used to retrieve the latitude (optional if country codes)
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

asia(*record=None, size_col=None, country_col=None, long_col=None, lat_col=None, profile=None, options=None, width=(100, '%'), height=(430, 'px'), html_code=None*)

A bubble chart for Asia

Underlying HTML Objects:

- epyk.core.geo.GeoPlotly.BubbleGeo

Related Pages:

<https://plotly.com/javascript/bubble-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **long_col** – String. The column in the recordset used to retrieve the longitude (optional if country codes)
- **lat_col** – String. The column in the recordset used to retrieve the latitude (optional if country codes)

- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

bubble(*scope*, *record=None*, *size_col=None*, *country_col=None*, *long_col=None*, *lat_col=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(430, 'px')*, *html_code=None*)

How to make a D3.js-based bubble map in JavaScript. A bubble map overlays a bubble chart on a map.

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.BubbleGeo`

Related Pages:

<https://plotly.com/javascript/bubble-maps/>

Parameters

- **scope** – String. The scope of the chart
- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **long_col** – String. The column in the recordset used to retrieve the longitude (optional if country codes)
- **lat_col** – String. The column in the recordset used to retrieve the latitude (optional if country codes)
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

europe(*record=None*, *size_col=None*, *country_col=None*, *long_col=None*, *lat_col=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(430, 'px')*, *html_code=None*)

A bubble chart for Europe

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.BubbleGeo`

Related Pages:

<https://plotly.com/javascript/bubble-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values

- **country_col** – String. The column in the recordset used to retrieve country code
- **long_col** – String. The column in the recordset used to retrieve the longitude (optional if country codes)
- **lat_col** – String. The column in the recordset used to retrieve the latitude (optional if country codes)
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

north_america(*record=None, size_col=None, country_col=None, long_col=None, lat_col=None, profile=None, options=None, width=(100, '%'), height=(430, 'px'), html_code=None*)

A bubble chart for North America

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.BubbleGeo`

Related Pages:

<https://plotly.com/javascript/bubble-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **long_col** – String. The column in the recordset used to retrieve the longitude (optional if country codes)
- **lat_col** – String. The column in the recordset used to retrieve the latitude (optional if country codes)
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

south_america(*record=None, size_col=None, country_col=None, long_col=None, lat_col=None, profile=None, options=None, width=(100, '%'), height=(430, 'px'), html_code=None*)

A bubble chart for South America

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.BubbleGeo`

Related Pages:

<https://plotly.com/javascript/bubble-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **long_col** – String. The column in the recordset used to retrieve the longitude (optional if country codes)
- **lat_col** – String. The column in the recordset used to retrieve the latitude (optional if country codes)
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

usa(*record=None, size_col=None, country_col=None, long_col=None, lat_col=None, profile=None, options=None, width=(100, '%'), height=(430, 'px'), html_code=None*)

A bubble chart for the USA.

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.BubbleGeo`

Related Pages:

<https://plotly.com/javascript/bubble-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **long_col** – String. The column in the recordset used to retrieve the longitude (optional if country codes)
- **lat_col** – String. The column in the recordset used to retrieve the latitude (optional if country codes)
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

world(*record=None, size_col=None, country_col=None, long_col=None, lat_col=None, profile=None, options=None, width=(100, '%'), height=(430, 'px'), html_code=None*)

A world bubble chart

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.BubbleGeo`

Related Pages:

<https://plotly.com/javascript/bubble-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **long_col** – String. The column in the recordset used to retrieve the longitude (optional if country codes)
- **lat_col** – String. The column in the recordset used to retrieve the latitude (optional if country codes)
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

PlotlyChoropleth Interface

class `epyk.interfaces.geo.CompGeoPlotly.PlotlyChoropleth(ui)`

africa(*record*, *size_col*=None, *country_col*=None, *profile*=None, *options*=None, *width*=(100, '%'), *height*=(430, 'px'), *html_code*=None)

A Choropleth Chart for african countries

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.Choropleth`

Related Pages:

<https://plotly.com/javascript/choropleth-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

asia(*record*, *size_col=None*, *country_col=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(430, 'px')*, *html_code=None*)

A Choropleth Chart for asian countries

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.Choropleth`

Related Pages:

<https://plotly.com/javascript/choropleth-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

europe(*record=None*, *size_col=None*, *country_col=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(430, 'px')*, *html_code=None*)

A Choropleth Chart for European countries

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.Choropleth`

Related Pages:

<https://plotly.com/javascript/choropleth-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

north_america(*record*, *size_col=None*, *country_col=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(430, 'px')*, *html_code=None*)

A Choropleth Chart for north american countries

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.Choropleth`

Related Pages:

<https://plotly.com/javascript/choropleth-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

south_america(*record*, *size_col=None*, *country_col=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(430, 'px')*, *html_code=None*)

A Choropleth Chart for south american countries

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.Choropleth`

Related Pages:

<https://plotly.com/javascript/choropleth-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

usa(*record*, *y_column=None*, *x_axis=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(330, 'px')*, *html_code=None*)

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.Choropleth`

Related Pages:

<https://plotly.com/javascript/choropleth-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

world(*record*, *size_col*=None, *country_col*=None, *profile*=None, *options*=None, *width*=(100, '%'),
height=(430, 'px'), *html_code*=None)

A world Choropleth Chart

Underlying HTML Objects:

- `epyk.core.geo.GeoPlotly.Choropleth`

Related Pages:

<https://plotly.com/javascript/choropleth-maps/>

Parameters

- **record** – Data. The recordset
- **size_col** – String. The column in the recordset used for the values
- **country_col** – String. The column in the recordset used to retrieve country code
- **profile** – A flag to set the component performance storage
- **options** – Dictionary. The charts options
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** –

graphs Interface

Chart2d Interface

class `epyk.interfaces.graphs.CompCharts.Chart2d(ui)`

property apex: [*ApexChart*](#)

Interface for the ApexChart library.

Category

Web application

Usage:

Related Pages:

<https://apexcharts.com/>

Returns

A Python ChartJs object

property billboard: *Billboard*

Interface to the Javascript Billboard module.

This will propose various charts for data analysis and visualisation based on D3. This project has been forked from Billboard.js.

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://naver.github.io/billboard.js/>

Returns

A Python Billboard Object

property c3: *C3*

Interface to the JavaScript C3 module.

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://c3js.org/>

Returns

A Python C3 object

property canvas: *Canvas*

The HTML <canvas> element is used to draw graphics on a web page.

The graphic to the left is created with <canvas>. It shows four elements: a red rectangle, a gradient rectangle, a multicolor rectangle, and a multicolor text.

Usage:

Related Pages:

https://www.w3schools.com/html/html5_canvas.asp

property chartJs: *ChartJs*

Interface for the ChartJs library

Category

Web application

Usage:

Related Pages:

<https://www.chartjs.org/>

Returns

A Python ChartJs object

property d3: *D3*

D3.js is a JavaScript library for manipulating documents based on data.

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://d3js.org/>

property dc: *DC*

dc.js is a javascript charting library with native crossfilter support, allowing highly efficient exploration on large multi-dimensional datasets (inspired by crossfilter's demo).

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://dc-js.github.io/dc.js/>

property nvd3: *Nvd3*

This project is an attempt to build re-usable charts and chart components for d3.js without taking away the power that d3.js gives you.

Category

Analytics, Web application

Usage:

Related Pages:

<http://nvd3.org/>

Returns

A Python NVD3 object

property plotly: *Plotly2D*

Built on top of d3.js and stack.gl, Plotly.js is a high-level, declarative charting library. plotly.js ships with over 40 chart types, including 3D charts, statistical graphs, and SVG maps.

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://plotly.com/javascript/>

Returns

A Python Plotly object

property svg: *SVG*

SVG stands for Scalable Vector Graphics.

SVG defines vector-based graphics in XML format.

Usage:

Related Pages:

https://www.w3schools.com/graphics/svg_intro.asp

property vis: *Vis2D*

Interface for the Vis library.

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://visjs.org/>

Returns

A Python Vis object

Chart3d Interface

class epyk.interfaces.graphs.CompCharts.**Chart3d**(*ui*)

property plotly: *Plotly3D*

Built on top of d3.js and stack.gl, Plotly.js is a high-level, declarative charting library. plotly.js ships with over 40 chart types, including 3D charts, statistical graphs, and SVG maps.

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://plotly.com/javascript/>

Returns

A Python Plotly object

property vis: *Vis3D*

Interface for the Vis library.

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://visjs.org/>

Returns

A Python Vis object

Graphs Interface

class epyk.interfaces.graphs.CompCharts.**Graphs**(ui)

property apex: *ApexChart*

Interface for the ApexChart library.

Category

Web application

Usage:

Related Pages:

<https://apexcharts.com/>

property bb: *Billboard*

Interface to the Javascript Billboard module.

This will propose various charts for data analysis and visualisation based on D£. This project has been forked from C3.js.

Category

Analytics, Dataviz

Usage:

```
languages = [
    {"name": 'C', 'type': 'code', 'rating': 17.07, 'change': 12.82},
    {"name": 'Java', 'type': 'code', 'rating': 16.28, 'change': 0.28},
]

c = page.ui.charts.bb.line(languages, y_columns=["rating", 'change'], x_axis=
↪ 'name')
```

Related Pages:

<https://naver.github.io/billboard.js/>

property billboard: *Billboard*

Interface to the Javascript Billboard module.

This will propose various charts for data analysis and visualisation based on D£. This project has been forked from C3.js.

Category

Analytics, Dataviz

Usage:

```
data = page.py.requests.csv(data_urls.DEMO_COUNTRY)
c3 = page.ui.charts.billboard.bar(results, y_columns=['Value'], x_axis="Year")
```

Related Pages:

<https://naver.github.io/billboard.js/>

property c3: *C3*

Interface to the JavaScript C3 module.

Category

Analytics, Dataviz

Usage:

```
data = page.py.requests.csv(data_urls.DEMO_COUNTRY)
c3 = page.ui.charts.c3.line(results, y_columns=['Value'], x_axis="Year")
```

Related Pages:

<https://c3js.org/>**property canvas:** [Canvas](#)

The HTML <canvas> element is used to draw graphics on a web page.

The graphic to the left is created with <canvas>. It shows four elements: a red rectangle, a gradient rectangle, a multicolor rectangle, and a multicolor text.

Usage:

Related Pages:

https://www.w3schools.com/html/html5_canvas.asp**property chartCss:** [CompChartCss](#)

Charts.css is a modern CSS framework. It uses CSS utility classes to style HTML elements as charts.

Category

Web application

Related Pages:

<https://chartscss.org/>**property chartJs:** [ChartJs](#)

Simple yet flexible JavaScript charting for designers & developers.

Category

Web Application

Usage:

Related Pages:

<https://www.chartjs.org/>**Returns**

A Python ChartJs object

property d3: [D3](#)

D3.js is a JavaScript library for manipulating documents based on data.

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://d3js.org/>

property dc: [DC](#)

dc.js is a javascript charting library with native crossfilter support, allowing highly efficient exploration on large multi-dimensional datasets (inspired by crossfilter's demo).

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://dc-js.github.io/dc.js/>

property frappe: [CompChartFrappe](#)

GitHub-inspired simple and modern SVG charts for the web with zero dependencies.

Category

Web application

Related Pages:

<https://frappe.io/charts>

property google: [ChartGoogle](#)

Google chart tools are powerful, simple to use, and free. Try out our rich gallery of interactive charts and data tools.

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://developers.google.com/chart>

menu(*chart: Html, height: Union[tuple, int, str] = (18, 'px'), options: Optional[dict] = None, post: Optional[Union[List[Union[str, JsDataModel]], str]] = None, profile: Optional[Union[bool, dict]] = None*) → Col

Add a standard menu on the table to trigger standard operation (add, empty, copy, download).

Parameters

- **chart** – The chart component
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **post** – Optional.
- **profile** – Optional. A flag to set the component performance storage

property nvd3: [Nvd3](#)

Interface to the Javascript NVD3 library.

Category

Analytics, Web application

Usage:

Related Pages:

<http://nvd3.org/>

plot(pkg: str = 'apex', record=None, y: Optional[list] = None, x: Optional[str] = None, kind: str = 'line', profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None)

Generic shortcut to plot a chart in the framework. Family and kind of chart are passed in parameter.

Usage:

```
:param pkg: Optional. The external chart package reference. Default ApexCharts
:param record: Optional. The list of dictionaries with the input data
:param y: Optional. The columns corresponding to keys in the dictionaries in
↳ the record
:param x: Optional. The column corresponding to a key in the dictionaries in
↳ the record
:param kind: Optional. The chart type
:param profile: Optional. A flag to set the component performance storage
:param width: Optional. The width of the component in the page, default (100, '%'
↳ ')
:param height: Optional. The height of the component in the page, default (330,
↳ "px")
:param options: Optional. Specific Python options available for this component
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
```

property plotly: *Plotly*

Built on top of d3.js and stack.gl, Plotly.js is a high-level, declarative charting library. plotly.js ships with over 40 chart types, including 3D charts, statistical graphs, and SVG maps.

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://plotly.com/javascript/>

Returns

A Python Plotly object

property roughviz: *CompRoughViz*

Reusable JavaScript library for creating sketchy/hand-drawn styled charts in the browser.

Category

Web application

Related Pages:

<https://github.com/jwilber/roughViz>

skillbars(records=None, y_column: Optional[str] = None, x_axis: Optional[str] = None, title: Optional[str] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False) → SkillBar

Python interface for the HTML Skill bars, simple bars chart done in pure Javascript and CSS.

Category

Web Application, Analytics

Usage:

```
records = [
    {"label": 'python', 'value': 12},
    {"label": 'Java', 'value': 5},
    {"label": 'Javascript', 'value': 80}]
page.ui.charts.skillbars(records, y_column='value', x_axis='label').css({"width
↪": '100px'})
```

Related Pages:

https://www.w3schools.com/howto/howto_css_skill_bar.asp

Parameters

- **records** – Optional. The Python list of dictionaries
- **y_column** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **title** – Optional. The chart title
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

sparkline(*chart_type: str, data, title: Optional[str] = None, options: Optional[dict] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), profile: Optional[Union[bool, dict]] = False*) → Sparklines

Display a sparkline component.

Category

Web Application, Analytics

Usage:

```
page.ui.charts.sparkline("box", [1, 2, 3, 4, 5, 4, 3, 2, 1])
page.ui.charts.sparkline("bar", [1, 2, 3, 4, 5, 4, 3, 2, 10])
```

Related Pages:

<https://omnipotent.net/jquery.sparkline/#s-about>

Parameters

- **chart_type** – The type of chart (bullet, line, bar, tristate, discrete, pie, box)
- **data** – A String corresponding to a JavaScript object
- **options** – Optional. Specific Python options available for this component
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **title** – Optional. A panel title. This will be attached to the title property

- **profile** – Optional. A flag to set the component performance storage

Returns

A python Sparkline object

property sparklines: *Sparkline*

Display a sparkline component.

Category

Web Application, Analytics

Usage:

```
page.ui.charts.sparklinea.box([1, 2, 3, 4, 5, 4, 3, 2, 1])
page.ui.charts.sparklines.bar([1, 2, 3, 4, 5, 4, 3, 2, 10])
```

Related Pages:

<https://plotly.com/javascript/>

property svg: *SVG*

SVG defines vector-based graphics in XML format.

Usage:

Related Pages:

https://www.w3schools.com/graphics/svg_intro.asp

property vega: *VegaEmbedded*

Vega – A Visualization Grammar.

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://vega.github.io/vega/>

Returns

A Python Vega object

property vis: *Vis*

A dynamic, browser based visualization library.

Category

Analytics, Dataviz

Usage:

Related Pages:

<https://visjs.org/>

ApexChart Interface

class epyk.interfaces.graphs.CompChartsApex.**ApexChart**(ui)

area(record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None)

Display an area chart from Apexcharts.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – The Python list of dictionaries
- **y_columns** – The columns corresponding to keys in the dictionaries in the record
- **x_axis** – The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

bar(record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None)

Display a bar chart from Apexcharts.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – The Python list of dictionaries
- **y_columns** – The columns corresponding to keys in the dictionaries in the record
- **x_axis** – The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

bubble(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)

Display a bubble chart from ApexCharts.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – The Python list of dictionaries
- **y_columns** – The columns corresponding to keys in the dictionaries in the record
- **x_axis** – The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

donut(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)

Display a donut chart from ApexCharts.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – The Python list of dictionaries
- **y_columns** – The columns corresponding to keys in the dictionaries in the record
- **x_axis** – The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

gauge(*values: float = 0, labels: str = "", profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Display a gauge chart from ApexCharts.

Tags

Categories

Related Pages:

<https://apexcharts.com/javascript-chart-demos/radialbar-charts/multiple-radialbars/>

Parameters

- **values** – Optional. The gauge value
- **labels** – Optional. The gauge label
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

hbar(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)

Display a horizontal bars chart from ApexCharts.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – The Python list of dictionaries
- **y_columns** – The columns corresponding to keys in the dictionaries in the record
- **x_axis** – The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

heatmap(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)

Display a heatmap chart from ApexCharts.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – The Python list of dictionaries
- **y_columns** – The columns corresponding to keys in the dictionaries in the record
- **x_axis** – The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

line(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Display a line chart from ApexCharts.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

pie(records=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None)

Display a pie chart from ApexCharts.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **records** – The Python list of dictionaries
- **y_columns** – The columns corresponding to keys in the dictionaries in the record
- **x_axis** – The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – The width of the component in the page, default (100, '%')
- **height** – The height of the component in the page, default (330, "px")
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

plot(record=None, y: Optional[list] = None, x: Optional[str] = None, kind: str = 'line', profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None)

Tags

Categories

Usage:

```
:param record: Optional. The list of dictionaries with the input data
:param y: Optional. The columns corresponding to keys in the dictionaries in
↳ the record
:param x: Optional. The column corresponding to a key in the dictionaries in
↳ the record
:param kind: Optional. The chart type
:param profile: Optional. A flag to set the component performance storage
:param width: Optional. The width of the component in the page, default (100, '%'
↳ ')
:param height: Optional. The height of the component in the page, default (330,
↳ "px")
:param options: Optional. Specific Python options available for this component
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
```

polar(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)

Display a polar chart from ApexCharts.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – The Python list of dictionaries
- **y_columns** – The columns corresponding to keys in the dictionaries in the record
- **x_axis** – The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

radar(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)

Display a radar chart from Apexcharts.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – The Python list of dictionaries
- **y_columns** – The columns corresponding to keys in the dictionaries in the record
- **x_axis** – The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

radial(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)

Display a radial chart from ApexCharts.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – The Python list of dictionaries
- **y_columns** – The columns corresponding to keys in the dictionaries in the record
- **x_axis** – The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

scatter(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)

Display a scatter chart from Apexchart.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – The Python list of dictionaries
- **y_columns** – The columns corresponding to keys in the dictionaries in the record
- **x_axis** – The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

treemap(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)

Display a treemap chart from ApexCharts.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – The Python list of dictionaries
- **y_columns** – The columns corresponding to keys in the dictionaries in the record
- **x_axis** – The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

Billboard Interface

class epyk.interfaces.graphs.CompChartsBillboard.**Billboard**(*ui*)

area(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a area chart from Billboard.

Tags

Categories

Usage:

```
data = page.py.requests.csv(data_urls.DEMO_COUNTRY)
c = page.ui.charts.bb.area(data, y_columns=["Value"], x_axis="Year",
    height=(500, "px"))
c.options.axis.y.tick.formats.scale(10000000)
```

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.AreaChart>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.

- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

area_step(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a area step chart from Billboard.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.AreaChart>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

bar(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a bar chart from Billboard.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.BarChart>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

bubble(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a bubble chart from Billboard.

Tags

Categories

Usage:

```
c = page.ui.charts.bb.bubble(y_columns=["Value"], x_axis="Year", height=(500,
↪ "px"))
c.options.axis.y.tick.formats.scale(1000000)
```

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.BubbleChart>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

donut(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a donut chart from Billboard.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.DonutChart>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

gauge(*value=0, text="", profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Display a gauge chart from Billboard.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.GaugeChart>

Parameters

- **value** – Integer. Optional. The gauge chart value.
- **text** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

hbar(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a horizontal bar chart from Billboard.

Tags

Categories

Usage:

```
data = page.py.requests.csv(data_urls.DEMO_COUNTRY)
c = page.ui.charts.bb.hbar(data, y_columns=["Value"], x_axis="Year",
    height=(500, "px"))
c.options.axis.y.tick.formats.scale(1000000)
c.options.axis.x.tick.count = 5
```

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.BarChart>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

line(record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None)

Display a line chart from Billboard.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

line_range(*record=None, y_columns=None, x_axis=None, range=5, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a line range chart from Billboard.

Tags

Categories

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

pie(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a pie chart from Billboard.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.PieChart>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.

- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

plot(*record=None, y=None, x=None, kind='line', profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Tags

Categories

Usage:

Related Pages:

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x** – Optional. The column corresponding to a key in the dictionaries in the record
- **kind** – Optional. The chart type
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, '%')
- **height** – Optional. The height of the component in the page, default (330, "px")
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

radar(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a radar chart from Billboard.

Tags

Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.RadarChart>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

scatter(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a scatter chart from Billboard.

Tags

Categories

Usage:

```
data = page.py.requests.csv(data_urls.DEMO_COUNTRY)
c = page.ui.charts.bb.scatter(data, y_columns=["Value"], x_axis="Year",
    height=(500, "px"))
c.options.axis.y.tick.formats.scale(1000000)
c.options.axis.x.tick.count = 5
```

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.ScatterPlot>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

spline(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a spline chart from Billboard.

Tags

Categories

Usage:

```
data = page.py.requests.csv(data_urls.DEMO_COUNTRY)
c = page.ui.charts.bb.spline(data, y_columns=["Value"], x_axis="Year",
    height=(500, "px"))
c.options.axis.y.tick.formats.scale(1000000)
```

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.SplineChart>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

stacked(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a stacked bar chart from Billboard.

Tags

Categories

Usage:

```
data = page.py.requests.csv(data_urls.DEMO_COUNTRY)
c = page.ui.charts.bb.stacked(data, y_columns=["Value"], x_axis="Year",
    height=(500, "px"))
c.options.axis.y.tick.formats.scale(1000000)
c.options.axis.x.tick.count = 5
```

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.StackedBarChart>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

step(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Display a step chart from Billboard.

Tags
Categories

Usage:

```
data = page.py.requests.csv(data_urls.DEMO_COUNTRY)
c = page.ui.charts.bb.step(data, y_columns=["Value"], x_axis="Year",
    height=(500, "px"))
c.options.axis.y.tick.formats.scale(1000000)
```

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.StepChart>**Parameters**

- **record** – Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

timeseries(*record=None, y_columns=None, x_axis=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Display a timeseries chart from Billboard.

Tags
Categories

Usage:

Related Pages:

<https://naver.github.io/billboard.js/demo/#Chart.LineChart>**Parameters**

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.

- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

C3 Interface

class epyk.interfaces.graphs.CompChartsC3.C3(ui)

area(record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None)

Display an area line chart from C3.

Tags

Categories

Usage:

Related Pages:

https://c3js.org/samples/chart_step.html

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

area_step(record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None)

Display a area step line chart from C3.

Tags

Categories

Usage:

Related Pages:

https://c3js.org/samples/chart_step.html

Parameters

- **record** – Optional. The Python list of dictionaries

- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

bar(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Display a bar chart from C3.

Tags

Categories

Usage:

Related Pages:

https://c3js.org/samples/chart_bar.html

Parameters

- **record** – Optional. The Python list of dictionaries.
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Optional. A flag to set the component performance storage.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).

donut(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Display a donut chart from C3.

Tags

Categories

Usage:

Related Pages:

https://c3js.org/samples/chart_donut.html

Parameters

- **record** – Optional. The Python list of dictionaries.
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Optional. A flag to set the component performance storage.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).

gauge(*value: float = 0, text: str = "", profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*)

Display a gauge chart from C3.

Tags

Categories

Usage:

Related Pages:

https://c3js.org/samples/chart_gauge.html

Parameters

- **value** – Optional. The value.
- **text** – Optional. The gauge text.
- **profile** – Optional. A flag to set the component performance storage.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).

hbar(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Display a horizontal bar chart from C3.

Tags

Categories

Usage:

Related Pages:

https://c3js.org/samples/axes_rotated.html

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

line(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Display a line chart from C3.

Tags

Categories

Usage:

Related Pages:

<https://c3js.org/reference.html#line-connectNull>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

pie(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Display a pie chart from C3.

Tags

Categories

Usage:

Related Pages:

https://c3js.org/samples/chart_pie.html <https://c3js.org/reference.html#pie-label-show>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

plot(*record=None, y=None, x=None, kind: str = 'line', profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Tags

Categories

Usage:

Related Pages:

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x** – Optional. The column corresponding to a key in the dictionaries in the record
- **kind** – Optional. The chart type
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

scatter(*record=None, y_columns=None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Display a Scatter chart from C3.

Tags

Categories

Usage:

Related Pages:

https://c3js.org/samples/axes_rotated.html

Parameters

- **record** – Optional. The Python list of dictionaries

- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

spline(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Display a spline line chart from C3.

Tags

Categories

Usage:

Related Pages:

https://c3js.org/samples/chart_spline.html

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

stanford(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, epoch_col=None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Tags

Categories

Related Pages:

https://c3js.org/samples/chart_stanford.html

Usage:


```

:param record: Optional. The Python list of dictionaries.
:param y_columns: Optional. The columns corresponding to keys in the
↳ dictionaries in the record
:param x_axis: Optional. The column corresponding to a key in the dictionaries
↳ in the record.
:param epoch_col: Optional. The column corresponding to a key.
:param profile: Optional. A flag to set the component performance storage.
:param width: Optional. A tuple with the integer for the component width and
↳ its unit.
:param height: Optional. A tuple with the integer for the component height and
↳ its unit.
:param options: Optional. Specific Python options available for this component.
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side).

```

step(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Display a step line chart from C3.

Tags

Categories

Usage:

Related Pages:

https://c3js.org/samples/chart_step.html

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

timeseries(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*)

Display a timeseries chart from C3.

Tags

Categories

Usage:

Related Pages:

<https://c3js.org/samples/timeseries.html>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

Canvas Interface

class epyk.interfaces.graphs.CompChartsCanvas.**Canvas**(*ui*)

new(*height: Union[tuple, int, str] = (400, 'px'), width: Union[tuple, int, str] = (100, '%'), profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

The HTML <canvas> tag is used to draw graphics, on the fly, via scripting (usually JavaScript).

However, the <canvas> element has no drawing abilities of its own (it is only a container for graphics) - you must use a script to actually draw the graphics.

The getContext() method returns an object that provides methods and properties for drawing on the canvas.

This reference will cover the properties and methods of the getContext(“2d”) object, which can be used to draw text, lines, boxes, circles, and more - on the canvas

Tags

Categories

Related Pages:

https://www.w3schools.com/tags/ref_canvas.asp

Usage:

```
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
:param profile: Optional. A flag to set the component performance storage
:param options: Optional. Specific Python options available for this component
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
```

CompChartCss Interface

class epyk.interfaces.graphs.CompChartsChartCss.**CompChartCss**(ui)

area(record=None, y_columns=None, x_axis=None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None) → ChartCssBarArea

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

bar(record=None, y_columns=None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None) → ChartCssBar

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

line(record=None, y_columns=None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None) → ChartCss

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit

- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

plot(*record=None, y=None, x=None, kind: str = 'line', profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Tags

Categories

Usage:

Related Pages:

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x** – Optional. The column corresponding to a key in the dictionaries in the record
- **kind** – Optional. The chart type
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

stacked(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*) → ChartCssBarStacked

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

ChartJs Interface

class epyk.interfaces.graphs.CompChartsChartJs.**ChartJs**(ui)

area(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → ChartLine

Display a area chart from ChartJs.

Tags

Categories

Usage:

Related Pages:

<https://www.chartjs.org/samples/latest/charts/area/line-stacked.html>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

bar(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → ChartBar

Display a bar chart from ChartJs.

Tags

Categories

Usage:

Related Pages:

<https://www.chartjs.org/samples/latest/scriptable/bar.html>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)

- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

bubble(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, r_values: Optional[list] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → ChartBubble

Display a bubble chart from ChartJs.

Tags

Categories

Usage:

Related Pages:

<https://www.chartjs.org/samples/latest/scriptable/bubble.html>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **r_values** – Optional. Set the r for the points on the chart
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, '%')
- **height** – Optional. The height of the component in the page, default (330, "px")
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

custom(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → ChartExts

Display a bar chart from ChartJs.

Tags

Categories

Usage:

Related Pages:

<https://www.chartjs.org/samples/latest/scriptable/bar.html>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record

- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, ‘%’)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

donut(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)
→ ChartPie

Display a donut chart from ChartJs.

Tags

Categories

Usage:

```
page = pk.Page()
df = pd.DataFrame({'Sales': [5000, 1222, 2000], 'Other': [500, 122, 200]},
    index=['TV', 'Smartphones', 'DVD'])
chart1 = page.ui.charts.chartJs.donut(df.to_dict("records"), y_columns=['Sales'],
    x_axis="Other")
chart1.click([
    page.js.console.log(chart1.dom.active()),
    chart1.build([{"Sales": 10, "Other": "A"}, {"Sales": 15, "Other": "B"}])
])
```

Related Pages:

<https://www.chartjs.org/samples/latest/charts/doughnut.html>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, ‘%’)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

fabric(*profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → Fabric

Tags

Categories

Usage:

```
:param profile: Optional. A flag to set the component performance storage
:param width: Optional. A tuple with the integer for the component width and
↳ its unit
:param height: Optional. A tuple with the integer for the component height and
↳ its unit
:param options: Optional. Specific Python options available for this component
:param html_code: Optional. An identifier for this component (on both Python
↳ and Javascript side)
```

hbar(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → ChartHBar

Display a horizontal bar chart from ChartJs.

Tags

Categories

Usage:

Related Pages:

<https://www.chartjs.org/samples/latest/scriptable/bar.html>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, '%')
- **height** – Optional. The height of the component in the page, default (330, "px")
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

hierarchical(*record: Optional[list] = None, labels: Optional[list] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, horizontal: bool = False, kind: str = 'bar', html_code: Optional[str] = None*) → ChartHyr

Chart.js module for adding a new categorical scale which mimics a hierarchical tree.

Related Pages:

<https://github.com/sgratzl/chartjs-plugin-hierarchical>

[chartjs-plugin-hierarchical/blob/main/samples/deep_hierarchy.html](https://github.com/sgratzl/chartjs-plugin-hierarchical/blob/main/samples/deep_hierarchy.html)

<https://github.com/sgratzl/>

Parameters

- **record** –
- **labels** –
- **profile** – Optional. A flag to set the component performance storage

- **width** – Optional. The width of the component in the page, default (100, ‘%’)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **horizontal** – Optional.
- **kind** – Optional.
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

line(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → ChartLine

Display a line chart from ChartJs.

Tags

Categories

Usage:

Related Pages:

<https://www.chartjs.org/> <https://www.chartjs.org/samples/latest/scales/logarithmic/line.html>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, ‘%’)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

matrix(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → ChartMatrix

Display a matrix chart from ChartJs.

Tags

Categories

Usage:

Related Pages:

<https://github.com/kurkle/chartjs-chart-matrix> <https://chartjs-chart-matrix.pages.dev/>

Parameters

- **record** – Optional. The list of dictionaries with the input data

- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

multi(*kind: str, record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[dict] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*) → ChartBar

Display a multi chart from ChartJs.

Tags

Categories

Usage:

Related Pages:

<https://www.chartjs.org/samples/latest/charts/combo-bar-line.html>

Parameters

- **kind** – The chart type
- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

pie(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → ChartPie

Display a pie chart from ChartJs.

Tags

Categories

Usage:

```

page = pk.Page()
df = pd.DataFrame({'Sales': [5000, 1222, 2000], 'Other': [500, 122, 200]},
    ↪ index=['TV', 'Smartphones', 'DVD'])
chart1 = page.ui.charts.chartJs.pie(df.to_dict("records"), y_columns=['Sales'],
    ↪ x_axis="Other")
chart1.click([
    page.js.console.log(chart1.dom.active()),
    chart1.build([{"Sales": 10, "Other": "A"}, {"Sales": 15, "Other": "B"}])
])

```

Related Pages:

<https://www.chartjs.org/samples/latest/charts/pie.html>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. To set the profiling
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

plot(*record: Optional[list] = None, y: Optional[list] = None, x: Optional[str] = None, kind: str = 'line', profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)
→ Chart

Generic way to define ChartJs charts.

Tags

Categories

Usage:

Related Pages:

<https://www.chartjs.org/>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x** – Optional. The column corresponding to a key in the dictionaries in the record
- **kind** – Optional. The chart type
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)

- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

polar(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)
→ ChartPolar

Display a bubble chart from ChartJs.

Tags

Categories

Usage:

Related Pages:

<https://www.chartjs.org/samples/latest/charts/polar-area.html>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, '%')
- **height** – Optional. The height of the component in the page, default (330, "px")
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

radar(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)
→ ChartRadar

Display a bubble chart from ChartJs.

Tags

Categories

Usage:

Related Pages:

<https://www.chartjs.org/samples/latest/charts/radar.html>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage

- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

scatter(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → ChartScatter

Display a scatter chart from ChartJs.

Tags

Categories

Usage:

Related Pages:

<https://www.chartjs.org/samples/latest/charts/scatter/basic.html>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

set_version(*v: str*)

Change the version of the chartJs package. Use **self.page.imports.pkgs.chart_js.version** to get the current version.

Usage:

```
page.ui.charts.chartJs.set_version("2.9.4").line(languages, y_columns=['change ↩'], x_axis="name")
```

Parameters

v – The version number

step(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*)

Display a step chart from ChartJs.

Tags
Categories

Usage:

Related Pages:

<https://www.chartjs.org/samples/latest/scales/linear/step-size.html>**Parameters**

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

timeseries(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → ChartLine

Display a line chart from ChartJs.

Tags
Categories

Usage:

Related Pages:

<https://www.chartjs.org/> <https://www.chartjs.org/samples/latest/scales/logarithmic/line.html>**Parameters**

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

treemap(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, groups: Optional[list] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → ChartTreeMap

Display a treemap chart from ChartJs.

Tags

Categories

Usage:

Related Pages:

<https://chartjs-chart-treemap.pages.dev/usage.html>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **groups** – Optional. The columns corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, '%')
- **height** – Optional. The height of the component in the page, default (330, 'px')
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

wordcloud(*record: Optional[list] = None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[dict] = None, html_code: Optional[str] = None*) → ChartWordCloud

Chart.js module for charting word or tag clouds. Adding new chart type: wordCloud.

Tags

Categories

Usage:

```
fake = Faker()
rec = [{"Sales": random.randint(1, 100), "Other": fake.name()} for _ in range(40)]
chart1 = page.ui.charts.chartJs.wordcloud(rec, y_columns=['Sales'], x_axis='Other')
page.ui.button("click").click([
    chart1.build([{"Sales": 40, "Other": 30}, {"Sales": 1, "Other": 1}, {"Sales": 15, "Other": 5}]))
```

Related Pages:

<https://github.com/sgratzl/chartjs-chart-wordcloud>

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, ‘%’)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

D3 Interface

class epyk.interfaces.graphs.CompChartsD3.D3(ui)

bar(record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None)

TODO: Handle multiple series correctly.

Usage:

```
page.ui.charts.d3.bar(mockes.languages, y_columns=["rating"], x_axis="change")
```

Related Pages:

<https://gramener.github.io/d3js-playbook/barchart.html>

Parameters

- **record** – Object. Optional. The chart input data to be serialised.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

cloud(data, width=(100, '%'), height=(330, 'px'), html_code=None, options=None, profile=None, excluded_words=None)

Tags

Categories

Usage:

Related Pages:

Parameters

- **data** – Object. Optional. The chart input data to be serialised.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).
- **excluded_words** – list of words to be excluded from the display.

pie(*record=None, y=None, x=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Related Pages:

<https://gramener.github.io/d3js-playbook/barchart.html>

TODO Add events

```
.on( "click", function(d,i) { console.log(d) }).on( "mouseover", function() {
    d3.select(this).style("fill", "red")

}).on( "mouseout", function() {
    d3.select(this).style("fill", function(d,i) {options.colors[i]})

})
```

Parameters

- **record** – Object. Optional. The chart input data to be serialised.
- **y** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

scatter(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

TODO: Handle multiple series correctly.

Usage:

```
page.ui.charts.d3.scatter(mockes.languages, y_columns=["rating"], x_axis="change  
↪")
```

Related Pages:

<https://gramener.github.io/d3js-playbook/barchart.html>

Parameters

- **record** – Object. Optional. The chart input data to be serialised.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

script(*name, scripts=None, data=None, d3_version=None, dependencies=None, profile=None, options=None, width=(400, 'px'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://gramener.github.io/d3js-playbook/barchart.html>

Templates:

Parameters

- **name** – String. The module name.
- **scripts** – List. Optional. The list of scripts.
- **data** – Object. Optional. The chart input data to be serialised.
- **d3_version** – String. Optional. Required version for the underlying D3 package.
- **dependencies** – List. Optional. The required dependency files.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **options** – Dictionary. Optional. Specific Python options available for this component.

- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

svg(*profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Parameters

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

DC Interface

class epyk.interfaces.graphs.CompChartsDc.DC(*ui*)

bar(*record=None, y_columns=None, x_axis=None, title=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **title** – String. Optional. The chart title.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

bubble(*record=None, y_columns=None, x_axis=None, r_axis=None, title=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://square.github.io/crossfilter/> <https://dc-js.github.io/dc.js/> https://www.tutorialspoint.com/dcjs/dcjs_bubble_chart.htm

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **r_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **title** – String. Optional. The chart title.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

hbar(*record=None, y_columns=None, x_axis=None, title=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **title** – String. Optional. The chart title.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

line(*record=None, y_columns=None, x_axis: Optional[str] = None, title: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*)

Tags

Categories

Usage:

Related Pages:

<https://square.github.io/crossfilter/> <https://dc-js.github.io/dc.js/>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **title** – Optional. The chart title
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

pie(*record=None, y_columns=None, x_axis=None, title=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://square.github.io/crossfilter/> <https://dc-js.github.io/dc.js/>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **title** – String. Optional. The chart title.

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

plot(*record=None, y=None, x=None, kind: str = 'line', profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Tags

Categories

Usage:

```
:param record: Optional. The list of dictionaries with the input data
:param y: Optional. The columns corresponding to keys in the dictionaries in
    ↳ the record
:param x: Optional. The column corresponding to a key in the dictionaries in
    ↳ the record
:param kind: Optional. The chart type
:param profile: Optional. A flag to set the component performance storage
:param width: Optional. The width of the component in the page, default (100, '%'
    ↳)
:param height: Optional. The height of the component in the page, default (330,
    ↳ "px")
:param options: Optional. Specific Python options available for this component
:param html_code: Optional. An identifier for this component (on both Python
    ↳ and Javascript side)
```

scatter(*record=None, y_columns=None, x_axis=None, title=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://square.github.io/crossfilter/> <https://dc-js.github.io/dc.js/>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **title** – String. Optional. The chart title.

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

series(*record=None, y_columns=None, x_axis=None, series_type: str = 'line', title: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*)

Tags

Categories

Usage:

Related Pages:

<https://square.github.io/crossfilter/> <https://dc-js.github.io/dc.js/>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **series_type** – Optional.
- **title** – Optional. The chart title
- **profile** – Optional. A flag to set the component performance storage
- **options** – Optional. Specific Python options available for this component
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

set_crossfilter(*record, y_columns, x_axis, varName, extra_cols=None*)

Set a crossfilter object and add the dimensions which will be added to a chart.

Tags

Categories

Usage:

```
:param record: The Python list of dictionaries
:param y_columns: The columns corresponding to keys in the dictionaries in the_
↪record
:param x_axis: The column corresponding to a key in the dictionaries in the_
↪record
```

(continues on next page)

(continued from previous page)

```
:param varName:  
:param extra_cols: Optional.
```

step(*record=None, y_columns=None, x_axis=None, title=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **title** – String. Optional. The chart title.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

sunburst(*record=None, y_columns=None, x_axis=None, title=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://square.github.io/crossfilter/> <https://dc-js.github.io/dc.js/>

Parameters

- **record** –
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **title** – String. Optional. The chart title.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

CompChartFrappe Interface

class epyk.interfaces.graphs.CompChartsFrappe.**CompChartFrappe**(ui)

bar(record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None)
Create a bar chart from Frappe Chart libraries.

Usage:

```
c = page.ui.charts.frappe.bar(y_columns=["Value"], x_axis="Year", height=(500,
↪ "px"))
text = page.ui.input("Italy")
slider = page.ui.sliders.range(minimum=1990, maximum=2020)
page.ui.button("Click").click([
page.js.d3.csv(data_urls.DEMO_COUNTRY).filterCol("Country Name", text.dom.
↪ content).cast(["Year", "Value"]).
    filterCol("Year", slider.dom.min_select, ">").filterCol("Year", slider.dom.
↪ max_select, "<").get(
    ["data = data.slice(1)",
    c.build(pk.events.data)])
])
```

Parameters

- **record** –
- **y_columns** –
- **x_axis** –
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, '%')
- **height** – Optional. The height of the component in the page, default (330, 'px')
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

donut(record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None)

Parameters

- **record** –

- **y_columns** –
- **x_axis** –
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

heatmap(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Parameters

- **record** –
- **y_columns** –
- **x_axis** –
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

line(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Create a line chart from Frappe Chart libraries.

Usage:

```
c = page.ui.charts.frappe.line(y_columns=["Value"], x_axis="Year", height=(500,
↪ "px"))
text = page.ui.input("Italy")
slider = page.ui.sliders.range(minimum=1990, maximum=2020)
page.ui.button("Click").click([
page.js.d3.csv(data_urls.DEMO_COUNTRY).filterCol("Country Name", text.dom.
↪ content).cast(["Year", "Value"]).
    filterCol("Year", slider.dom.min_select, ">").filterCol("Year", slider.dom.
↪ max_select, "<").get(
    ["#data = data.slice(1)",
    c.build(pk.events.data)])
])
```

Parameters

- **record** –

- **y_columns** –
- **x_axis** –
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, ‘%’)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

percentage(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Usage:

```
c = page.ui.charts.frappe.percentage(y_columns=["Value"], x_axis="Year",
↪height=(500, "px"))
text = page.ui.input("Italy")
slider = page.ui.sliders.range(minimum=1990, maximum=2020)
page.ui.button("Click").click([
    page.js.fetch(data_urls.DEMO_COUNTRY).csvtoRecords().filterCol(
        "Country Name", text.dom.content).cast(["Year", "Value"]).
        filterCol("Year", slider.dom.min_select, ">").filterCol("Year", slider.dom.
↪max_select, "<").
    get([
        c.build(pk.events.data),
    ])
])
```

Parameters

- **record** –
- **y_columns** –
- **x_axis** –
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, ‘%’)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

pie(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Parameters

- **record** –

- **y_columns** –
- **x_axis** –
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, ‘%’)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

plot(record=None, y=None, x=None, kind: str = 'line', profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None)

Tags

Categories

Usage:

Related Pages:

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x** – Optional. The column corresponding to a key in the dictionaries in the record
- **kind** – Optional. The chart type
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, ‘%’)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

ChartGoogle Interface

class epyk.interfaces.graphs.CompChartsGoogle.**ChartGoogle**(ui)

area(record, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None) → ChartLine

An area chart that is rendered within the browser using SVG or VML. Displays tips when hovering over points.

Tags

Categories

Usage:

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/areachart>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

bar(*record*, *y_columns*: *Optional[list] = None*, *x_axis*: *Optional[str] = None*, *profile*: *Optional[Union[bool, dict]] = None*, *width*: *Union[tuple, int, str] = (100, '%')*, *height*: *Union[tuple, int, str] = (330, 'px')*, *options*: *Optional[Union[bool, dict]] = None*, *html_code*: *Optional[str] = None*) → ChartLine

Google bar charts are rendered in the browser using SVG or VML, whichever is appropriate for the user's browser. Like all Google charts, bar charts display tooltips when the user hovers over the data. For a vertical version of this chart, see the column chart.

Tags**Categories**

Usage:

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/barchart>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

bubble(*record*, *y_columns*: *Optional[list] = None*, *x_axis*: *Optional[str] = None*, *profile*: *Optional[Union[bool, dict]] = None*, *width*: *Union[tuple, int, str] = (100, '%')*, *height*: *Union[tuple, int, str] = (330, 'px')*, *options*: *Optional[Union[bool, dict]] = None*, *html_code*: *Optional[str] = None*) → ChartLine

A bubble chart that is rendered within the browser using SVG or VML. Displays tips when hovering over bubbles.

A bubble chart is used to visualize a data set with two to four dimensions. The first two dimensions are visualized as coordinates, the third as color and the fourth as size.

Tags
Categories

Usage:

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/bubblechart>**Parameters**

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

candlestick(*record*, *y_columns*: *Optional[list] = None*, *x_axis*: *Optional[str] = None*, *profile*: *Optional[Union[bool, dict]] = None*, *width*: *Union[tuple, int, str] = (100, '%')*, *height*: *Union[tuple, int, str] = (330, 'px')*, *options*: *Optional[Union[bool, dict]] = None*, *html_code*: *Optional[str] = None*) → ChartLine

A candlestick chart is used to show an opening and closing value overlaid on top of a total variance. Candlestick charts are often used to show stock value behavior. In this chart, items where the opening value is less than the closing value (a gain) are drawn as filled boxes, and items where the opening value is more than the closing value (a loss) are drawn as hollow boxes.

Tags
Categories

Usage:

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/candlestickchart>**Parameters**

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

column(*record*, *y_columns*: *Optional*[*list*] = *None*, *x_axis*: *Optional*[*str*] = *None*, *profile*: *Optional*[*Union*[*bool*, *dict*]] = *None*, *width*: *Union*[*tuple*, *int*, *str*] = (100, '%'), *height*: *Union*[*tuple*, *int*, *str*] = (330, 'px'), *options*: *Optional*[*Union*[*bool*, *dict*]] = *None*, *html_code*: *Optional*[*str*] = *None*) → *ChartLine*

A column chart is a vertical bar chart rendered in the browser using SVG or VML, whichever is appropriate for the user's browser. Like all Google charts, column charts display tooltips when the user hovers over the data. For a horizontal version of this chart, see the bar chart.

Tags

Categories

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/columnchart>

Usage:

```
:param record: Optional. The Python list of dictionaries
:param y_columns: Optional. The columns corresponding to keys in the
    ↪ dictionaries in the record
:param x_axis: Optional. The column corresponding to a key in the dictionaries
    ↪ in the record
:param profile: Optional. A flag to set the component performance storage
:param width: Optional. A tuple with the integer for the component width and
    ↪ its unit
:param height: Optional. A tuple with the integer for the component height and
    ↪ its unit
:param options: Optional. Specific Python options available for this component
:param html_code: Optional. An identifier for this component (on both Python
    ↪ and Javascript side)
```

donut(*record*, *y_columns*: *Optional*[*list*] = *None*, *x_axis*: *Optional*[*str*] = *None*, *profile*: *Optional*[*Union*[*bool*, *dict*]] = *None*, *width*: *Union*[*tuple*, *int*, *str*] = (100, '%'), *height*: *Union*[*tuple*, *int*, *str*] = (330, 'px'), *options*: *Optional*[*Union*[*bool*, *dict*]] = *None*, *html_code*: *Optional*[*str*] = *None*)

A donut chart that is rendered within the browser using SVG or VML. Displays tooltips when hovering over slices.

Tags

Categories

Usage:

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/piechart>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

gauge(*record*, *y_columns*: *Optional[list] = None*, *x_axis*: *Optional[str] = None*, *profile*: *Optional[Union[bool, dict]] = None*, *width*: *Union[tuple, int, str] = (100, '%')*, *height*: *Union[tuple, int, str] = (330, 'px')*, *options*: *Optional[Union[bool, dict]] = None*, *html_code*: *Optional[str] = None*) → ChartLine

A gauge with a dial, rendered within the browser using SVG or VML.

Tags

Categories

Usage:

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/gauge>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

geo(*record*, *y_columns*: *Optional[list] = None*, *x_axis*: *Optional[str] = None*, *profile*: *Optional[Union[bool, dict]] = None*, *width*: *Union[tuple, int, str] = (100, '%')*, *height*: *Union[tuple, int, str] = (330, 'px')*, *options*: *Optional[Union[bool, dict]] = None*, *html_code*: *Optional[str] = None*) → ChartLine

A geochart is a map of a country, a continent, or a region with areas identified in one of three ways:

The region mode colors whole regions, such as countries, provinces, or states. The markers mode uses circles to designate regions that are scaled according to a value that you specify. The text mode labels the regions with identifiers (e.g., “Russia” or “Asia”).

Tags

Categories

Usage:

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/geochart>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record

- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

hbar(*record*, *y_columns*: *Optional[list] = None*, *x_axis*: *Optional[str] = None*, *profile*: *Optional[Union[bool, dict]] = None*, *width*: *Union[tuple, int, str] = (100, '%')*, *height*: *Union[tuple, int, str] = (330, 'px')*, *options*: *Optional[Union[bool, dict]] = None*, *html_code*: *Optional[str] = None*) → ChartLine

Google bar charts are rendered in the browser using SVG or VML, whichever is appropriate for the user's browser. Like all Google charts, bar charts display tooltips when the user hovers over the data. For a vertical version of this chart, see the column chart.

Tags

Categories

Usage:

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/barchart>

Parameters

- **record** – Optional. The Python list of dictionaries.
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Optional. A flag to set the component performance storage.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).

histogram(*record*, *y_columns*: *Optional[list] = None*, *x_axis*: *Optional[str] = None*, *profile*: *Optional[Union[bool, dict]] = None*, *width*: *Union[tuple, int, str] = (100, '%')*, *height*: *Union[tuple, int, str] = (330, 'px')*, *options*: *Optional[Union[bool, dict]] = None*, *html_code*: *Optional[str] = None*) → ChartLine

A histogram is a chart that groups numeric data into bins, displaying the bins as segmented columns. They're used to depict the distribution of a dataset: how often values fall into ranges.

Google Charts automatically chooses the number of bins for you. All bins are equal width and have a height proportional to the number of data points in the bin. In other respects, histograms are similar to column charts.

Tags

Categories

Usage:

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/histogram>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

line(*record*, *y_columns*: *Optional[list] = None*, *x_axis*: *Optional[str] = None*, *profile*: *Optional[Union[bool, dict]] = None*, *width*: *Union[tuple, int, str] = (100, '%')*, *height*: *Union[tuple, int, str] = (330, 'px')*, *options*: *Optional[Union[bool, dict]] = None*, *html_code*: *Optional[str] = None*) → ChartLine

A line chart that is rendered within the browser using SVG or VML. Displays tooltips when hovering over points.

Tags

Categories

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/linechart>

Usage:

```
:param record: Optional. The Python list of dictionaries
:param y_columns: Optional. The columns corresponding to keys in the
↳dictionaries in the record
:param x_axis: Optional. The column corresponding to a key in the dictionaries
↳in the record
:param profile: Optional. A flag to set the component performance storage
:param width: Optional. A tuple with the integer for the component width and
↳its unit
:param height: Optional. A tuple with the integer for the component height and
↳its unit
:param options: Optional. Specific Python options available for this component
:param html_code: Optional. An identifier for this component (on both Python
↳and Javascript side)
```

pie(*record*, *y_columns*=*None*, *x_axis*: *Optional[str] = None*, *profile*: *Optional[Union[bool, dict]] = None*, *width*: *Union[tuple, int, str] = (100, '%')*, *height*: *Union[tuple, int, str] = (330, 'px')*, *options*: *Optional[Union[bool, dict]] = None*, *html_code*: *Optional[str] = None*) → ChartLine

A pie chart that is rendered within the browser using SVG or VML. Displays tooltips when hovering over slices.

Usage:

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/piechart>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

plot(*record=None, y=None, x=None, kind: str = 'line', profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*) → ChartLine

Tags

Categories

Usage:

```
:param record: Optional. The list of dictionaries with the input data
:param y: Optional. The columns corresponding to keys in the dictionaries in
→ the record
:param x: Optional. The column corresponding to a key in the dictionaries in
→ the record
:param kind: Optional. The chart type
:param profile: Optional. A flag to set the component performance storage
:param width: Optional. The width of the component in the page, default (100, '%'
→ ')
:param height: Optional. The height of the component in the page, default (330,
→ "px")
:param options: Optional. Specific Python options available for this component
:param html_code: Optional. An identifier for this component (on both Python
→ and Javascript side)
```

scatter(*record, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*) → ChartLine

Scatter charts plot points on a graph. When the user hovers over the points, tooltips are displayed with more information.

Google scatter charts are rendered within the browser using SVG or VML depending on browser capabilities.

Tags

Categories

Usage:

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/scatterchart>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

treemap(*record*, *y_columns*: *Optional[list] = None*, *x_axis*: *Optional[str] = None*, *profile*: *Optional[Union[bool, dict]] = None*, *width*: *Union[tuple, int, str] = (100, '%')*, *height*: *Union[tuple, int, str] = (330, 'px')*, *options*: *Optional[Union[bool, dict]] = None*, *html_code*: *Optional[str] = None*) → ChartLine

Tags

Categories

Usage:

Related Pages:

<https://developers.google.com/chart/interactive/docs/gallery/treemap>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

Nvd3 Interface

class epyk.interfaces.graphs.CompChartsNvd3.Nvd3(ui)

area(record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None) → ChartArea

Display an area chart from NVD3.

Tags

Categories

Usage:

Related Pages:

<http://nvd3.org/examples/discreteBar.html>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

bar(record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None) → ChartBar

Display a bars chart from NVD3.

Tags

Categories

Usage:

```
c = page.ui.charts.nvd3.bar(y_columns=["Value"], x_axis="Year", height=(500, "px",
↪))
page.ui.button("Click").click([
text = page.ui.input("Italy")
slider = page.ui.sliders.range(minimum=1990, maximum=2020)
page.js.fetch(data_urls.DEMO_COUNTRY).csvtoRecords().filterCol("Country Name", ↪
↪text.dom.content).cast(["Year", "Value"]).
    filterCol("Year", slider.dom.min_select, ">").filterCol("Year", slider.dom.
↪max_select, "<").
    get([
        c.build(pk.events.data)
```

(continues on next page)

(continued from previous page)

```

    #"console.log(row)"
  })
})

```

Related Pages:

<http://nvd3.org/examples/discreteBar.html>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

candlestick(*record, closes, highs, lows, opens, x_axis, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

```

data = page.py.requests.csv(data_urls.PLOTLY_APPLE_PRICES)
sc = page.ui.charts.nvd3.candlestick(data, closes=["AAPL.Close"], highs=["AAPL.
↪High"], lows=["AAPL.Low"],
    opens=["AAPL.Open"], x_axis='Date')

# Example using pandas_datareader
import pandas_datareader.data as pdr

btc = pdr.get_data_yahoo("BTC-USD", datetime.strptime(start, '%Y-%m-%d'), ↪
↪datetime.strptime(end, '%Y-%m-%d'))
btc.reset_index(inplace=True)
btc["Date"] = btc["Date"].dt.strftime("%Y-%m-%d")

c1 = page.ui.charts.nvd3.candlestick(
    btc.to_dict("records"), closes=["Adj Close"], highs=["High"], lows=["Low"], ↪
↪opens=["Open"], x_axis="Date")
c1.shared.y_format_number(factor=1000)
c1.shared.y_label("$")

```

Related Pages:

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.

- **closes** –
- **highs** –
- **lows** –
- **opens** –
- **x_axis** –
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

donut(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*) → ChartPie

Display a donut chart from NVD3.

Tags

Categories

Usage:

```
c = page.ui.charts.nvd3.donut(y_columns=["Value"], x_axis="Year", height=(500,
↪ "px"))
```

Related Pages:

<http://nvd3.org/examples/pie.html>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

forceDirected(*profile=None, options=None, width=(400, 'px'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

Parameters

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

gauge(*value: float, text: Optional[str] = None, total: float = 100, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*) → ChartPie

Tags

Categories

Usage:

Related Pages:

<http://nvd3.org/examples/pie.html>

Parameters

- **value** –
- **text** –
- **total** –
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

group_box(*profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

Parameters

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.

- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

hbar(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*)

Display a bars chart from NVD3.

Tags

Categories

Usage:

```
c = page.ui.charts.nvd3.hbar(y_columns=["Value"], x_axis="Year", height=(500,
↪ "px"))
page.ui.button("Click").click([
text = page.ui.input("Italy")
slider = page.ui.sliders.range(minimum=1990, maximum=2020)
page.js.fetch(data_urls.DEMO_COUNTRY).csvtoRecords().filterCol("Country Name", ↪
↪ text.dom.content).cast(["Year", "Value"]).
filterCol("Year", slider.dom.min_select, ">").filterCol("Year", slider.dom.
↪ max_select, "<").
get([
c.build(pk.events.data)
#"console.log(row)"
])
])
```

Related Pages:

<http://nvd3.org/examples/discreteBar.html>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

histo(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*) → ChartHistoBar

Display a histo chart from NVD3.

Tags

Categories

Usage:

Related Pages:

<http://nvd3.org/examples/discreteBar.html>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

line(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*) → ChartLine

Display a line chart from NVD3.

Tags

Categories

Usage:

```
c = page.ui.charts.nvd3.line(y_columns=["Value"], x_axis="Year", height=(500,
↪ "px"))
page.ui.button("Click").click([
text = page.ui.input("Italy")
slider = page.ui.sliders.range(minimum=1990, maximum=2020)
page.js.fetch(data_urls.DEMO_COUNTRY).csvtoRecords().filterCol("Country Name", ↪
↪ text.dom.content).cast(["Year", "Value"]).
    filterCol("Year", slider.dom.min_select, ">").filterCol("Year", slider.dom.
↪ max_select, "<").
    get([
        c.build(pk.events.data)
        ↪ "#console.log(row)"
    ])
])
```

Related Pages:

<http://nvd3.org/examples/line.html>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record

- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

line_cumulative(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*) → ChartCumulativeLine

Display a Cumulative line chart from NVD3.

Tags

Categories

Usage:

Related Pages:

<http://nvd3.org/examples/line.html>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

line_focus(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*) → ChartFocusLine

Display a line chart with focus from NVD3.

Tags

Categories

Usage:

Related Pages:

<http://nvd3.org/examples/line.html>

Parameters

- **record** – Optional. The Python list of dictionaries

- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

multi(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*) → ChartMultiBar

Display a multi types chart from NVD3.

Tags

Categories

Usage:

Related Pages:

<http://nvd3.org/examples/discreteBar.html>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

ohlc(*record, closes, highs, lows, opens, x_axis, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **closes** –
- **highs** –

- **lows** –
- **opens** –
- **x_axis** –
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

parallel_coordinates(*record, dimensions=None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*) → ChartParallelCoord

Tags

Categories

Usage:

Related Pages:

Parameters

- **record** – Optional. The Python list of dictionaries
- **dimensions** – Optional.
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

pie(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*) → ChartPie

Display a pie chart from NVD3.

Tags

Categories

Usage:

Related Pages:

<http://nvd3.org/examples/pie.html>

Parameters

- **record** – Optional. The Python list of dictionaries

- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

plot(*record=None, y=None, x=None, kind: str = 'line', profile: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), options: Optional[Union[bool, dict]] = None, html_code: Optional[str] = None*)

Tags

Categories

Usage:

Related Pages:

Parameters

- **record** – Optional. The list of dictionaries with the input data
- **y** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x** – Optional. The column corresponding to a key in the dictionaries in the record
- **kind** – Optional. The chart type
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

scatter(*record=None, y_columns: Optional[list] = None, x_axis: Optional[str] = None, profile: Optional[Union[bool, dict]] = None, options: Optional[Union[bool, dict]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (330, 'px'), html_code: Optional[str] = None*) → ChartScatter

Display a scatter chart from NVD3.

Tags

Categories

Usage:

Related Pages:

<http://nvd3.org/examples/line.html>

Parameters

- **record** – Optional. The Python list of dictionaries

- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

sunburst(*record*, *name*: str, *profile*: Optional[Union[bool, dict]] = None, *options*: Optional[Union[bool, dict]] = None, *width*: Union[tuple, int, str] = (100, '%'), *height*: Union[tuple, int, str] = (330, 'px'), *html_code*: Optional[str] = None) → ChartSunburst

Tags

Categories

Usage:

Related Pages:

Parameters

- **record** – Optional. The Python list of dictionaries
- **name** – Optional.
- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

timeseries(*record*=None, *y_columns*: Optional[list] = None, *x_axis*: Optional[str] = None, *profile*: Optional[Union[bool, dict]] = None, *options*: Optional[Union[bool, dict]] = None, *width*: Union[tuple, int, str] = (100, '%'), *height*: Union[tuple, int, str] = (330, 'px'), *html_code*: Optional[str] = None) → ChartHistoBar

Display a Timeseries chart from NVD3.

Tags

Categories

Usage:

Related Pages:

<http://nvd3.org/examples/discreteBar.html>

Parameters

- **record** – Optional. The Python list of dictionaries
- **y_columns** – Optional. The columns corresponding to keys in the dictionaries in the record
- **x_axis** – Optional. The column corresponding to a key in the dictionaries in the record

- **profile** – Optional. A flag to set the component performance storage
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

Plotly Interface

```
class epyk.interfaces.graphs.CompChartsPlotly.Plotly(ui)
```

Plotly2D Interface

```
class epyk.interfaces.graphs.CompChartsPlotly.Plotly2D(ui)
```

```
area(record, y_columns=None, x_axis=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None)
```

How to make a D3.js-based filled area plot in javascript. An area chart displays a solid color between the traces of a graph.

Tags

Categories

Usage:

Related Pages:

<https://plotly.com/javascript/filled-area-plots/>

Parameters

- **record** – List of dict. The Python record.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

```
bar(record=None, y_columns=None, x_axis=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None)
```

Tags

Categories

Usage:

Related Pages:

<https://plotly.com/javascript/bar-charts/>

Parameters

- **record** – List of dict. The Python record.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

```
box(record=None, y_columns=None, x_columns=None, profile=None, options=None, width=(100, '%'),
    height=(330, 'px'), html_code=None)
```

Tags

Categories

Usage:

```
:param record: List of dict. Optional. The Python list of dictionaries.
:param y_columns: List. Optional. The columns corresponding to keys in the
    ↳ dictionaries in the record.
:param x_columns: String. Optional. The column corresponding to a key in the
    ↳ dictionaries in the record.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
    ↳ performance storage.
:param width: Tuple. Optional. A tuple with the integer for the component width
    ↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
    ↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
    ↳ this component.
:param html_code: String. Optional. An identifier for this component (on both
    ↳ Python and Javascript side).
```

```
bubble(record, y_columns=None, x_axis=None, profile=None, options=None, width=(100, '%'),
    height=(330, 'px'), html_code=None)
```

How to make a D3.js-based filled area plot in javascript. An area chart displays a solid color between the traces of a graph.

Tags

Categories

Usage:

Related Pages:

<https://plotly.com/javascript/bubble-charts/>

Parameters

- **record** – List of dict. The Python record.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

candlestick(*record, closes, highs, lows, opens, x_axis, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

```
data = page.py.requests.csv(data_urls.PLOTLY_APPLE_PRICES)
sc = page.ui.charts.plotly.candlestick(
    data, closes=["AAPL.Close"], highs=["AAPL.High"], lows=["AAPL.Low"], opens=[
↪ "AAPL.Open"], x_axis='Date')
```

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **closes** –
- **highs** –
- **lows** –
- **opens** –
- **x_axis** –
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.

- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

donut(*record=None, y_columns=None, x_axis=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://plotly.com/javascript/pie-charts/>

Parameters

- **record** – List of dict. The Python record.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

gauge(*value, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

How to make a D3.js-based gauge chart in javascript.

Tags

Categories

Usage:

```
gauge = page.ui.charts.plotly.gauge(2000)
gauge.data.gauge.axis.range = [0, 5000]
```

Related Pages:

<https://plotly.com/javascript/gauge-charts/>

Parameters

- **value** –
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.

- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

group_box(*record*, *y_columns=None*, *x_axis=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(330, 'px')*, *html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://plot.ly/javascript/box-plots/>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

hbar(*record*, *y_columns=None*, *x_axis=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(330, 'px')*, *html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://plotly.com/javascript/bar-charts/>

Parameters

- **record** – List of dict. The Python record.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.

- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

histogram(*record*, *y_columns=None*, *x_columns=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(330, 'px')*, *html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://plot.ly/javascript/plotlyjs-function-reference/#common-parameters>

<https://plot.ly/javascript/>

Parameters

- **record** –
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_columns** – List. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

Return type

graph.GraphPlotly.Bar

line(*record=None*, *y_columns=None*, *x_axis=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(330, 'px')*, *html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://plot.ly/javascript/#common-parameters>

<https://plot.ly/javascript/plotlyjs-function-reference/>

Parameters

- **record** – List. The Python record.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.

- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

number(*value*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(330, 'px')*, *html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://plotly.com/javascript/indicator/>

Parameters

- **value** –
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

number_with_delta(*value*, *delta=100*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(330, 'px')*, *html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://plotly.com/javascript/indicator/>

Parameters

- **value** –
- **delta** –
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.

- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

pie(*record=None, y_columns=None, x_axis=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://plotly.com/javascript/pie-charts/>

Parameters

- **record** – List of dict. Optional. The Python records.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

scatter(*record=None, y_columns=None, x_axis=None, text_column=None, profile=None, options=None, width=(100, '%'), height=(330, 'px'), html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://plotly.com/javascript/text-and-annotations/>

Parameters

- **record** – List of dict. The Python record.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **text_column** – String. Optional.

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

scattergl(*record*, *y_columns=None*, *x_axis=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(330, 'px')*, *html_code=None*)

Tags

Categories

Usage:

Related Pages:

<https://plotly.com/javascript/line-and-scatter/>

Parameters

- **record** – List of dict. The Python record.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

scatterpolar(*record*, *r_columns=None*, *theta_axis=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(330, 'px')*, *html_code=None*)

Tags

Categories

Usage:

```
:param record: List of dict. Optional. The Python list of dictionaries.
:param r_columns:
:param theta_axis:
:param profile: Boolean | Dictionary. Optional. A flag to set the component_
↳ performance storage.
:param width: Tuple. Optional. A tuple with the integer for the component width_
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component_
```

(continues on next page)

(continued from previous page)

↪ **height** and its unit.
 :param options: Dictionary. Optional. Specific Python options available for ↪
 ↪ this component.
 :param html_code: String. Optional. An identifier for this component (on both ↪
 ↪ Python and Javascript side).

timeseries(*record*, *y_columns=None*, *x_axis=None*, *profile=None*, *options=None*, *width=(100, '%')*,
height=(330, 'px'), *html_code=None*)

Tags

Categories

Usage:

Related Pages:

[https://plot.ly/javascript/
#common-parameters](https://plot.ly/javascript/#common-parameters)

<https://plot.ly/javascript/plotlyjs-function-reference/>

Parameters

- **record** – List of dict. The Python record.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

Plotly3D Interface

class epyk.interfaces.graphs.CompChartsPlotly.**Plotly3D**(*ui*)

line(*record*, *y_columns=None*, *x_axis=None*, *z_axis=None*, *profile=None*, *options=None*, *width=(100, '%')*,
height=(500, 'px'), *html_code=None*)

Tags

Categories

Related Pages:

<https://plot.ly/javascript/3d-line-plots/>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.

- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **z_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

maps(*record*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(500, 'px')*, *html_code=None*)

Tags

Categories

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

marker(*record*, *y_columns=None*, *x_axis=None*, *z_axis=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(500, 'px')*, *html_code=None*)

Tags

Categories

Related Pages:

<https://plot.ly/javascript/3d-line-plots/>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **z_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

mesh3d(*record, intensity, x, y, z, i=None, j=None, k=None, profile=None, options=None, width=(100, '%'), height=(500, 'px'), html_code=None*)

Tags

Categories

Related Pages:

<https://plot.ly/javascript/3d-mesh/>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **intensity** –
- **x** –
- **y** –
- **z** –
- **i** –
- **j** –
- **k** –
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

ribbon(*record, y_columns=None, x_axis=None, z_axis=None, profile=None, options=None, width=(100, '%'), height=(500, 'px'), html_code=None*)

Create ribbons on the x axis.

Tags

Categories

Related Pages:

<https://plot.ly/javascript/ribbon-plots/>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **z_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

scatter(*record*, *y_columns=None*, *x_axis=None*, *z_axis=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(500, 'px')*, *html_code=None*)

Tags

Categories

Related Pages:

<https://plot.ly/javascript/3d-line-plots/>

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **z_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

surface(*record*, *y_columns=None*, *x_axis=None*, *z_axis=None*, *profile=None*, *options=None*, *width=(100, '%')*, *height=(500, 'px')*, *html_code=None*)

Tags

Categories

Parameters

- **record** – List of dict. Optional. The Python list of dictionaries.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **z_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

CompRoughViz Interface

class epyk.interfaces.graphs.CompChartsRoughViz.**CompRoughViz**(ui)

bar(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Create a roughViz bar component.

Related Pages:

<https://github.com/jwilber/roughViz>

Parameters

- **record** – List. Optional. The list of dictionaries with the input data.
- **y_columns** – List. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. The width of the component in the page, default (100, '%').
- **height** – Tuple. Optional. The height of the component in the page, default (330, "px").
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

donut(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Create a roughViz donut component.

Related Pages:

<https://github.com/jwilber/roughViz>

Parameters

- **record** – List. Optional. The list of dictionaries with the input data.
- **y_columns** – List. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. The width of the component in the page, default (100, '%').
- **height** – Tuple. Optional. The height of the component in the page, default (330, "px").
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

hbar(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Create a roughViz Horizontal bar component.

Related Pages:

<https://github.com/jwilber/roughViz>

Parameters

- **record** – List. Optional. The list of dictionaries with the input data.
- **y_columns** – List. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. The width of the component in the page, default (100, '%').
- **height** – Tuple. Optional. The height of the component in the page, default (330, "px").
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

line(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Create a roughViz line component.

Related Pages:

<https://github.com/jwilber/roughViz>

TODO: Find answer for <https://stackoverflow.com/questions/67456146/input-data-for-line-chart-in-roughviz>

Parameters

- **record** – List. Optional. The list of dictionaries with the input data.
- **y_columns** – List. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. The column corresponding to a key in the dictionaries in the record.

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. The width of the component in the page, default (100, '%').
- **height** – Tuple. Optional. The height of the component in the page, default (330, 'px').
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

pie(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Create a roughViz pie component.

Related Pages:

<https://github.com/jwilber/roughViz>

Parameters

- **record** – List. Optional. The list of dictionaries with the input data.
- **y_columns** – List. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. The width of the component in the page, default (100, '%').
- **height** – Tuple. Optional. The height of the component in the page, default (330, 'px').
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

plot(*record=None, y=None, x=None, kind='line', profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Create a roughViz chart component.

Related Pages:

<https://github.com/jwilber/roughViz>

Tags

Categories

Related Pages:

Parameters

- **record** – List. Optional. The list of dictionaries with the input data.
- **y** – List | String. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **kind** – String. Optional. The chart type.

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. The width of the component in the page, default (100, '%').
- **height** – Tuple. Optional. The height of the component in the page, default (330, "px").
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

scatter(*record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Create a roughViz scatter component.

Related Pages:

<https://github.com/jwilber/roughViz>

Parameters

- **record** – List. Optional. The list of dictionaries with the input data.
- **y_columns** – List. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. The width of the component in the page, default (100, '%').
- **height** – Tuple. Optional. The height of the component in the page, default (330, "px").
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

Sparkline Interface

class epyk.interfaces.graphs.CompChartsSparkline.**Sparkline**(*ui*)

bar(*data=None, title=None, options=None, width=(None, '%'), height=(None, 'px'), profile=False*)

Tags

Categories

Usage:

Related Pages:

<https://omnipotent.net/jquery.sparkline/#s-about>

Parameters

- **data** – String. Optional. A String corresponding to a JavaScript object.
- **title** – String. Optional. A panel title. This will be attached to the title property.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.

- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

box_plot(*data=None, title=None, options=None, width=(None, '%'), height=(None, 'px'), profile=False*)

Tags

Categories

Usage:

Related Pages:

<https://omnipotent.net/jquery.sparkline/#s-about>

Parameters

- **data** – String. Optional. A String corresponding to a JavaScript object.
- **title** – String. Optional. A panel title. This will be attached to the title property.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

bullet(*data=None, title=None, options=None, width=(None, '%'), height=(None, 'px'), profile=False*)

Tags

Categories

Usage:

Related Pages:

<https://omnipotent.net/jquery.sparkline/#s-about>

Parameters

- **data** – String. Optional. A String corresponding to a JavaScript object.
- **title** – String. Optional. A panel title. This will be attached to the title property.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

discrete(*data=None, title=None, options=None, width=(None, '%'), height=(None, 'px'), profile=False*)

Tags

Categories

Usage:

Related Pages:

<https://omnipotent.net/jquery.sparkline/#s-about>

Parameters

- **data** – String. Optional. A String corresponding to a JavaScript object.
- **title** – String. Optional. A panel title. This will be attached to the title property.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

line(data=None, title=None, options=None, width=(None, '%'), height=(None, 'px'), profile=False)

Tags

Categories

Usage:

Related Pages:

<https://omnipotent.net/jquery.sparkline/#s-about>

Parameters

- **data** – String. Optional. A String corresponding to a JavaScript object.
- **title** – String. Optional. A panel title. This will be attached to the title property.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

pie(data=None, title=None, options=None, width=(None, '%'), height=(None, 'px'), profile=False)

Tags

Categories

Usage:

Related Pages:

<https://omnipotent.net/jquery.sparkline/#s-about>

Parameters

- **data** – String. Optional. A String corresponding to a JavaScript object.
- **title** – String. Optional. A panel title. This will be attached to the title property.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.

- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

tristate(*data=None, title=None, options=None, width=(None, '%'), height=(None, 'px'), profile=False*)

Tags

Categories

Usage:

Related Pages:

<https://omnipotent.net/jquery.sparkline/#s-about>

Parameters

- **data** – String. Optional. A String corresponding to a JavaScript object.
- **title** – String. Optional. A panel title. This will be attached to the title property.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

SVG Interface

class epyk.interfaces.graphs.CompChartsSvg.**SVG**(*ui*)

arrow_left(*x1: float = 0, y1: Optional[float] = None, x2: Optional[float] = None, y2: Optional[float] = None, size: int = 10, width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
page.ui.charts.svg.arrow_left()
```

Parameters

- **x1** –
- **y1** –
- **x2** –
- **y2** –
- **size** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)

- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

arrow_right(*x1: float = 0, y1: Optional[float] = None, x2: Optional[float] = None, y2: Optional[float] = None, size: int = 10, width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
page.ui.charts.svg.arrow_right(y1=40)
```

Parameters

- **x1** –
- **y1** –
- **x2** –
- **y2** –
- **size** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

axes(*size: int = 10, width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
svg = page.ui.charts.svg.axes()
m = svg.defs().marker("circle", "0 0 10 10", 5, 5)
m.circle(5, 5, 5, 'red')
m.markerWidth(10).markerHeight(10)
p = svg.path(0, 0, from_origin=True).line_to(50, 100).horizontal_line_to(300).
↪line_to(400, 200)
p.markers(m.url)
```

Parameters

- **size** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit

- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

circle(*x: float, y: float, r: float, width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Entry point to the basic line definition in a SVG HTML Tag.

Tags

Categories

Usage:

```
page.ui.charts.svg.line(10, 30, 40, 69)
```

Related Pages:

https://www.w3schools.com/graphics/svg_line.asp

Parameters

- **x** – The x attribute defines the start of the line on the x-axis
- **y** – The y attribute defines the start of the line on the y-axis
- **r** – The r attribute defines the radius
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

ellipse(*cx: float, cy: float, rx: float, ry: float, width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

SVG Ellipse - <ellipse>.

Tags

Categories

Usage:

```
page.ui.charts.svg.ellipse(100, 100, 40, 69)
```

Related Pages:

https://www.w3schools.com/graphics/svg_ellipse.asp

Parameters

- **cx** – The cx attribute defines the x coordinate of the center of the ellipse
- **cy** – The cy attribute defines the y coordinate of the center of the ellipse
- **rx** – The rx attribute defines the horizontal radius
- **ry** – The ry attribute defines the vertical radius

- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

heart(*w: float, h: float, fill: str = 'none', width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
c = page.ui.charts.svg.heart(w=50, h=100, fill="pink")
c[0].transform("transform", "rotate", "0 100 100", "360 100 10")
```

Parameters

- **w** –
- **h** –
- **fill** –
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

line(*x1: float = 0, y1: Optional[float] = None, x2: Optional[float] = None, y2: Optional[float] = None, width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Entry point to the basic line definition in a SVG HTML Tag.

Tags

Categories

Usage:

```
page.ui.charts.svg.line(10, 30, 40, 69)
```

Related Pages:

https://www.w3schools.com/graphics/svg_line.asp

Parameters

- **x1** – The x1 attribute defines the start of the line on the x-axis
- **y1** – Optional. The y1 attribute defines the start of the line on the y-axis
- **x2** – Optional. The x2 attribute defines the end of the line on the x-axis
- **y2** – Optional. The y2 attribute defines the end of the line on the y-axis
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

new(width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'))

SVG stands for Scalable Vector Graphics.

SVG defines vector-based graphics in XML format.

Tags

Categories

Usage:

```
svg = page.ui.charts.svg.new(width=200)
svg.add_text("I love SVG!", x=0, y=15, options={"fill": 'red'})
```

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/SVG> https://www.w3schools.com/graphics/svg_intro.asp

Parameters

- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

path(x: float = 0, y: float = 0, fill: Optional[str] = 'none', origin: bool = False, bespoke_path=None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Tags

Categories

Usage:

Related Pages:

Parameters

- **x** –
- **y** –
- **fill** –
- **origin** –
- **bespoke_path** –
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.

polygon(points: List[tuple], width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Tags

Categories

Usage:

```
page.ui.charts.svg.polygone([(15, 80), (29, 50), (43, 60), (57, 30), (71, 40),
↪(85, 15)])
```

Related Pages:

https://www.w3schools.com/graphics/tryit.asp?filename=trysvg_polygon

Parameters

- **points** – The points attribute defines the list of points (pairs of coordinates) required to draw the polyline
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

polyline(*points: List[tuple], width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
page.ui.charts.svg.polyline([(15, 80), (29, 50), (43, 60), (57, 30), (71, 40),
↪(85, 15)])
```

Related Pages:

https://www.w3schools.com/graphics/svg_polyline.asp

Parameters

- **points** – The points attribute defines the list of points (pairs of coordinates) required to draw the polyline
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

rectangle(*x: float, y: float, width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'), fill: Optional[str] = None, rx: float = 0, ry: float = 0, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
:param x:
:param y:
:param width: Optional. A tuple with the integer for the component width and
```

(continues on next page)

(continued from previous page)

```

↪ its unit.
:param height: Optional. A tuple with the integer for the component height and
↪ its unit.
:param fill:
:param rx:
:param ry:
:param options: Optional. Specific Python options available for this component.
:param profile: Optional. A flag to set the component performance storage.

```

star(*fill: str = 'none', width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags**Categories**

Usage:

Related Pages:

<https://codepen.io/susanwinters/pen/WxbRJK>

Parameters

- **fill** –
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

triangle(*point1: tuple, point2: Optional[tuple] = None, point3: Optional[tuple] = None, fill: str = 'None', width: Union[tuple, int, str] = (500, 'px'), height: Union[tuple, int, str] = (300, 'px'), options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags**Categories**

Usage:

```
rptObj.ui.charts.svg.triangle((50, 100))
```

Related Pages:

https://www.w3schools.com/graphics/svg_polyline.asp

Parameters

- **point1** –
- **point2** – Optional.
- **point3** – Optional.
- **fill** – Optional.
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit

- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

VegaEmbedded Interface

class epyk.interfaces.graphs.CompChartsVega.**VegaEmbedded**(ui)

plot(record=None, y=None, x=None, kind='line', profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None)

Tags

Categories

Usage:

```
c = page.ui.charts.vega.plot(y=["Value"], x="Year", kind="point", height=(500,
↪ "px"))
text = page.ui.input("Italy")
slider = page.ui.sliders.range(minimum=1990, maximum=2020)
page.ui.button("Click").click([
page.js.d3.csv(data_urls.DEMO_COUNTRY).filterCol("Country Name", text.dom.
↪ content).cast(["Year", "Value"]).
    filterCol("Year", slider.dom.min_select, ">").filterCol("Year", slider.dom.
↪ max_select, "<").get(
    ["data = data.slice(1)",
    c.build(pk.events.data)])
])
```

Parameters

- **record** – List. Optional. The list of dictionaries with the input data.
- **y** – List | String. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **kind** – String. Optional. The chart type.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. The width of the component in the page, default (100, '%').
- **height** – Tuple. Optional. The height of the component in the page, default (330, "px").
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

Vis Interface

class epyk.interfaces.graphs.CompChartsVis.**Vis**(ui)

bar3d(record, y_columns=None, x_axis=None, z_axis=None, profile=None, width=(100, '%'), height=(400, 'px'), options=None, html_code=None)

Tags

Categories

Usage:

```
:param record: List of dict. The Python record.
:param y_columns: List. Optional. The columns corresponding to keys in the
↳ dictionaries in the record.
:param x_axis: String. Optional. The column corresponding to a key in the
↳ dictionaries in the record.
:param z_axis: String. Optional. The column corresponding to a key in the
↳ dictionaries in the record.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳ performance storage.
:param width: Tuple. Optional. A tuple with the integer for the component width
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳ this component.
:param html_code: String. Optional. An identifier for this component (on both
↳ Python and Javascript side).
```

line3d(record, y_columns=None, x_axis=None, z_axis=None, profile=None, width=(100, '%'), height=(400, 'px'), options=None, html_code=None)

Tags

Categories

Usage:

```
:param record: List of dict. The Python record.
:param y_columns: List. Optional. The columns corresponding to keys in the
↳ dictionaries in the record.
:param x_axis: String. Optional. The column corresponding to a key in the
↳ dictionaries in the record.
:param z_axis: String. Optional. The column corresponding to a key in the
↳ dictionaries in the record.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳ performance storage.
:param width: Tuple. Optional. A tuple with the integer for the component width
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳ this component.
:param html_code: String. Optional. An identifier for this component (on both
↳ Python and Javascript side).
```

scatter3d(*record*, *y_columns=None*, *x_axis=None*, *z_axis=None*, *profile=None*, *width=(100, '%')*,
height=(400, 'px'), *options=None*, *html_code=None*)

Tags

Categories

Usage:

```
:param record: List of dict. The Python record.
:param y_columns: List. Optional. The columns corresponding to keys in the
↳ dictionaries in the record.
:param x_axis: String. Optional. The column corresponding to a key in the
↳ dictionaries in the record.
:param z_axis: String. Optional. The column corresponding to a key in the
↳ dictionaries in the record.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳ performance storage.
:param width: Tuple. Optional. A tuple with the integer for the component width
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳ this component.
:param html_code: String. Optional. An identifier for this component (on both
↳ Python and Javascript side).
```

surface(*record*, *y_columns=None*, *x_axis=None*, *z_axis=None*, *profile=None*, *width=(100, '%')*,
height=(400, 'px'), *options=None*, *html_code=None*)

Tags

Categories

Usage:

```
:param record: List of dict. The Python record.
:param y_columns: List. Optional. The columns corresponding to keys in the
↳ dictionaries in the record.
:param x_axis: String. Optional. The column corresponding to a key in the
↳ dictionaries in the record.
:param z_axis: String. Optional. The column corresponding to a key in the
↳ dictionaries in the record.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳ performance storage.
:param width: Tuple. Optional. A tuple with the integer for the component width
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳ this component.
:param html_code: String. Optional. An identifier for this component (on both
↳ Python and Javascript side).
```

Vis2D Interface

class epyk.interfaces.graphs.CompChartsVis.**Vis2D**(ui)

bar(record, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None)

Tags

Categories

Usage:

Related Pages:

<http://www.chartjs.org/>

Parameters

- **record** – List of dict. The Python record.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

line(record=None, y_columns=None, x_axis=None, profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None)

Graph2d is an interactive visualization chart to draw data in a 2D graph. You can freely move and zoom in the graph by dragging and scrolling in the window.

Graph2d uses HTML DOM and SVG for rendering. This allows for flexible customization using css styling.

Tags

Categories

Usage:

Related Pages:

<http://www.chartjs.org/>
https://visjs.github.io/vis-timeline/examples/graph2d/16_bothAxisTitles.html

Parameters

- **record** – List of dict. The Python record.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.

- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

network(*profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Tags

Categories

Usage:

```
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param width: Tuple. Optional. A tuple with the integer for the component width.
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component.
↳height and its unit.
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
:param html_code: String. Optional. An identifier for this component (on both.
↳Python and Javascript side).
```

plot(*record=None, y=None, x=None, kind='line', profile=None, width=(100, '%'), height=(330, 'px'), options=None, html_code=None*)

Tags

Categories

Usage:

```
:param record: List. Optional. The list of dictionaries with the input data.
:param y: List | String. Optional. The columns corresponding to keys in the.
↳dictionaries in the record.
:param x: String. Optional. The column corresponding to a key in the.
↳dictionaries in the record.
:param kind: String. Optional. The chart type.
:param profile: Boolean | Dictionary. Optional. A flag to set the component.
↳performance storage.
:param width: Tuple. Optional. The width of the component in the page, default.
↳(100, '%').
:param height: Tuple. Optional. The height of the component in the page,
↳default (330, "px").
:param options: Dictionary. Optional. Specific Python options available for.
↳this component.
:param html_code: String. Optional. An identifier for this component (on both.
↳Python and Javascript side).
```

scatter(*record*, *y_columns=None*, *x_axis=None*, *profile=None*, *width=(100, '%')*, *height=(330, 'px')*, *options=None*, *html_code=None*)

Tags

Categories

Usage:

Related Pages:

<http://www.chartjs.org/>

Parameters

- **record** – List of dict. The Python record.
- **y_columns** – List. Optional. The columns corresponding to keys in the dictionaries in the record.
- **x_axis** – String. Optional. The column corresponding to a key in the dictionaries in the record.
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.
- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

timeline(*record=None*, *start=None*, *content=None*, *end=None*, *type=None*, *group=None*, *profile=None*, *width=(100, '%')*, *height=(330, 'px')*, *options=None*, *html_code=None*)

Tags

Categories

Usage:

Related Pages:

<http://www.chartjs.org/>

Parameters

- **record** –
- **start** –
- **content** –
- **end** –
- **type** –
- **group** –
- **profile** – Boolean | Dictionary. Optional. A flag to set the component performance storage.
- **width** – Tuple. Optional. A tuple with the integer for the component width and its unit.

- **height** – Tuple. Optional. A tuple with the integer for the component height and its unit.
- **options** – Dictionary. Optional. Specific Python options available for this component.
- **html_code** – String. Optional. An identifier for this component (on both Python and Javascript side).

Vis3D Interface

class epyk.interfaces.graphs.CompChartsVis.**Vis3D**(ui)

bar(record, y_columns=None, x_axis=None, z_axis=None, profile=None, width=(100, '%'), height=(400, 'px'), options=None, html_code=None)

Tags

Categories

Usage:

```
:param record: List of dict. The Python record.
:param y_columns: List. Optional. The columns corresponding to keys in the_
↳ dictionaries in the record.
:param x_axis: String. Optional. The column corresponding to a key in the_
↳ dictionaries in the record.
:param z_axis: String. Optional. The column corresponding to a key in the_
↳ dictionaries in the record.
:param profile: Boolean | Dictionary. Optional. A flag to set the component_
↳ performance storage.
:param width: Tuple. Optional. A tuple with the integer for the component width_
↳ and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component_
↳ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for_
↳ this component.
:param html_code: String. Optional. An identifier for this component (on both_
↳ Python and Javascript side).
```

line(record, y_columns=None, x_axis=None, z_axis=None, profile=None, width=(100, '%'), height=(400, 'px'), options=None, html_code=None)

Tags

Categories

Usage:

```
:param record: List of dict. The Python record.
:param y_columns: List. Optional. The columns corresponding to keys in the_
↳ dictionaries in the record.
:param x_axis: String. Optional. The column corresponding to a key in the_
↳ dictionaries in the record.
:param z_axis: String. Optional. The column corresponding to a key in the_
↳ dictionaries in the record.
:param profile: Boolean | Dictionary. Optional. A flag to set the component_
↳ performance storage.
:param width: Tuple. Optional. A tuple with the integer for the component width_
↳ and its unit.
```

(continues on next page)

(continued from previous page)

```

↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳this component.
:param html_code: String. Optional. An identifier for this component (on both
↳Python and Javascript side).

```

```

scatter(record, y_columns=None, x_axis=None, z_axis=None, profile=None, width=(100, '%'),
        height=(400, 'px'), options=None, html_code=None)

```

Tags**Categories**

Usage:

```

:param record: List of dict. The Python record.
:param y_columns: List. Optional. The columns corresponding to keys in the
↳dictionaries in the record.
:param x_axis: String. Optional. The column corresponding to a key in the
↳dictionaries in the record.
:param z_axis: The column corresponding to a key in the dictionaries in the
↳record.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳performance storage.
:param width: Tuple. Optional. A tuple with the integer for the component width
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component
↳height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↳this component.
:param html_code: String. Optional. An identifier for this component (on both
↳Python and Javascript side).

```

```

surface(record, y_columns=None, x_axis=None, z_axis=None, profile=None, width=(100, '%'),
        height=(400, 'px'), options=None, html_code=None)

```

Tags**Categories**

Usage:

```

:param record: List of dict. The Python record.
:param y_columns: List. Optional. The columns corresponding to keys in the
↳dictionaries in the record.
:param x_axis: String. Optional. The column corresponding to a key in the
↳dictionaries in the record.
:param z_axis: String. Optional. The column corresponding to a key in the
↳dictionaries in the record.
:param profile: Boolean | Dictionary. Optional. A flag to set the component
↳performance storage.
:param width: Tuple. Optional. A tuple with the integer for the component width
↳and its unit.
:param height: Tuple. Optional. A tuple with the integer for the component

```

(continues on next page)

(continued from previous page)

```

↪ height and its unit.
:param options: Dictionary. Optional. Specific Python options available for
↪ this component.
:param html_code: String. Optional. An identifier for this component (on both
↪ Python and Javascript side).

```

tables Interface

AgGrid Interface

class epyk.interfaces.tables.CompAgGrid.**AgGrid**(ui)

AG Grid components module.

table(records: Optional[list] = None, cols: Optional[list] = None, rows: Optional[list] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (300, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None) → Table

Create a generic Angular Grid table.

Tags

Categories

Usage:

```

import epyk as pk
from epyk.mocks import urls as data_urls

page = pk.Page()
data = page.py.requests.csv(data_urls.AIRPORT_TRAFFIC)
table = page.ui.tables.aggrids.table(data)
table.options.paginationPageSize = 10
table.options.rowSelection = "single"

```

Parameters

- **records** – Optional. The list of dictionaries with the input data.
- **cols** – Optional. The list of key from the record to be used as columns in the table.
- **rows** – Optional. The list of key from the record to be used as rows in the table.
- **width** – Optional. A tuple with the integer for the component width and its unit.
- **height** – Optional. A tuple with the integer for the component height and its unit.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

Datatables Interface

class epyk.interfaces.tables.CompDatatable.**Datatables**(ui)

table(records=None, cols: Optional[list] = None, rows: Optional[list] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None)

Create a generic DataTable table.

Tags

Categories

Usage:

```
dt = page.ui.tables.datatables.table(cols=["test"])
page.ui.button("Update").click([dt.build([[ "row %s" % i] for i in range(n)])])
```

Parameters

- **records** – Optional. The list of dictionaries with the input data.
- **cols** – Optional. The list of key from the record to be used as columns in the table.
- **rows** – Optional. The list of key from the record to be used as rows in the table.
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

Pivottable Interface

class epyk.interfaces.tables.CompPivot.**Pivottable**(ui)

c3(records=None, rows: Optional[list] = None, cols: Optional[list] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False)

Create an HTML Pivot table.

Tags

Categories

Usage:

Related Pages:

<https://pivottable.js.org/examples/nicolaskruchten/w86bgq9o/>

<https://react-pivottable.js.org/>

<https://jsfiddle.net/>

Parameters

- **records** –

- **rows** –
- **cols** –
- **width** – Optional. The width of the component in the page, default (100, ‘%’)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. Display a tooltip info component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

d3(records=None, rows: Optional[list] = None, cols: Optional[list] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False)

Create a HTML Pivot table.

Tags

Categories

Usage:

Related Pages:

<https://pivottable.js.org/examples/nicolaskruchten/w86bgq9o/>

<https://react-pivottable.js.org/>

<https://jsfiddle.net/>

Parameters

- **records** –
- **rows** –
- **cols** –
- **width** – Optional. The width of the component in the page, default (100, ‘%’)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. Display a tooltip info component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

heatmap(records: Optional[list] = None, rows: Optional[list] = None, cols: Optional[list] = None, values=None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False)

Create an HTML Pivot table.

Tags

Categories

Usage:

Related Pages:

<https://pivottable.js.org/examples/nicolaskruchten/w86bgq9o/>

<https://react-pivottable.js.org/>

<https://jsfiddle.net/>

Parameters

- **records** –
- **rows** –
- **cols** –
- **values** –
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. Display a tooltip info component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

pivot(*records: Optional[list] = None, rows: Optional[list] = None, cols: Optional[list] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False*)

Create an HTML Pivot table.

Tags

Categories

Usage:

Related Pages:

<https://pivottable.js.org/examples/nicolaskruchten/w86bgq9o/>

<https://react-pivottable.js.org/>

<https://jsfiddle.net/>

Parameters

- **records** –
- **rows** –
- **cols** –
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **helper** – Optional. Display a tooltip info component.
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

plotly(*records=None, rows: Optional[list] = None, cols: Optional[list] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False*)

Create an HTML Pivot table.

Tags

Categories

Usage:

Related Pages:

<https://pivottable.js.org/examples/nicolaskruchten/w86bgq9o/>

<https://react-pivottable.js.org/>

<https://jsfiddle.net/>

Parameters

- **records** –
- **rows** –
- **cols** –
- **width** – Optional. The width of the component in the page, default (100, '%')
- **height** – Optional. The height of the component in the page, default (330, 'px')
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. Display a tooltip info component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

sub_total(*records: Optional[list] = None, rows: Optional[list] = None, cols: Optional[list] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False*)

Create an HTML Pivot table.

Tags

Categories

Usage:

Related Pages:

<https://pivottable.js.org/examples/nicolaskruchten/w86bgq9o/>

<https://react-pivottable.js.org/>

<https://jsfiddle.net/>

Parameters

- **records** –
- **rows** –
- **cols** –
- **width** – Optional. The width of the component in the page, default (100, '%')
- **height** – Optional. The height of the component in the page, default (330, 'px')

- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. Display a tooltip info component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

ui(records: Optional[list] = None, rows: Optional[list] = None, cols: Optional[list] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, helper: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = False)

Create an HTML Pivot table.

Tags

Categories

Usage:

Related Pages:

<https://pivottable.js.org/examples/nicolaskruchten/w86bgq9o/>

<https://react-pivottable.js.org/>

<https://jsfiddle.net/>

Parameters

- **records** –
- **rows** –
- **cols** –
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **helper** – Optional. Display a tooltip info component
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

D3 Interface

class epyk.interfaces.tables.CompTableD3.D3(ui)

table(records: Optional[list] = None, header: Optional[list] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None)

Tags

Categories

Usage:

```
:param records: Optional. The list of dictionaries with the input data
:param header: Optional.
:param width: Optional. The width of the component in the page, default (100, '%
↳')
:param height: Optional. The height of the component in the page, default (330,
↳"px")
:param html_code: Optional. An identifier for this component (on both Python
↳and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

Google Interface

```
class epyk.interfaces.tables.CompTableGoogle.Google(ui)
```

```
table(records=None, cols: Optional[list] = None, rows: Optional[list] = None, width: Union[tuple, int, str]
      = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options:
      Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None)
```

Tags

Categories

Usage:

```
:param records: Optional. The list of dictionaries with the input data
:param cols: Optional. The list of key from the record to be used as columns in
↳the table
:param rows: Optional. The list of key from the record to be used as rows in
↳the table
:param width: Optional. The width of the component in the page, default (100, '%
↳')
:param height: Optional. The height of the component in the page, default (330,
↳"px")
:param html_code: Optional. An identifier for this component (on both Python
↳and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

Tables Interface

```
class epyk.interfaces.tables.CompTables.Tables(ui)
```

property aggrids: *AgGrid*

AG-Grid is the industry standard for JavaScript Enterprise Applications. Developers using ag-Grid are building applications that would not be possible if ag-Grid did not exist.

Tags

Categories

Usage:

Related Pages:

<https://www.ag-grid.com/javascript-grid/>

basic(*records: Optional[List[dict]] = None, cols=None, rows=None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
simple_table = page.ui.tables.basic(df.to_dict("records"), cols=["COL1"], rows=[
    "COL2"])
simple_table.add({"COL1": "Value"})
```

Parameters

- **records** – Optional. The list of dictionaries with the input data
- **cols** – Optional. The list of key from the record to be used as columns in the table
- **rows** – Optional. The list of key from the record to be used as rows in the table
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

property d3: [D3](#)

Interface to the different Tabulator configurations.

Tags

Categories

Related Pages:

<https://github.com/d3/d3/wiki/Gallery>

property datatables: [Datatables](#)

Interface to the different Datatable configurations.

Tags

Categories

Related Pages:

<https://datatables.net/>

property google: [Google](#)

Interface to the Google Table interface.

In order to use it, the Google products need to be specially enabled.

grid(*records, cols: Optional[list] = None, rows: Optional[list] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
:param records: Optional. The list of dictionaries with the input data
:param cols: Optional. The list of key from the record to be used as columns in
↳the table
:param rows: Optional. The list of key from the record to be used as rows in
↳the table
:param width: Optional. The width of the component in the page, default (100, '%
↳')
:param height: Optional. The height of the component in the page, default (330,
↳"px")
:param html_code: Optional. An identifier for this component (on both Python
↳and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

menu(table: Optional[Html] = None, height: Union[tuple, int, str] = (18, 'px'), options: Optional[Union[bool, dict]] = None, update_funcs: Optional[list] = None, post: Optional[Union[list, str]] = None, profile: Optional[Union[bool, dict]] = None, columns: Optional[dict] = None, title: Optional[Union[str, dict]] = None)

Add a standard menu on the table to trigger standard operation (add, empty, copy, download).

Tags

Categories

Usage:

```
hierarchy = page.ui.tables.tabulators.hierarchy(html_code="hierarchy")
menu = page.ui.tables.menu(hierarchy)
```

Parameters

- **table** – Optional. The HTML table component
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **update_funcs** – Optional. JavaScript functions to update the table component
- **post** – Optional. The event used to update the table
- **profile** – Optional. A flag to set the component performance storage
- **columns** – Optional.
- **title** – Optional. The title value or component

property pivots: *Pivottable*

Interface to the different Pivot Table configurations.

Tags

Categories

Related Pages:

<https://pivottable.js.org/examples/>

property plotlys: *Plotly*

Interface to the different Tabulator configurations.

Tags

Categories

Related Pages:

<http://tabulator.info/>

property tabulators: *Tabulators*

Interface to the different Tabulator configurations.

Tags

Categories

Usage:

Related Pages:

<http://tabulator.info/>

Plotly Interface

class epyk.interfaces.tables.CompTablesPlotly.**Plotly**(*ui*)

table(*records=None, cols: Optional[list] = None, rows: Optional[list] = None, header: Optional[list] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*)

Create a Plotly table.

Tags

Categories

Usage:

Related Pages:

<https://plot.ly/javascript/table-subplots/>

Parameters

- **records** – Optional. The list of dictionaries with the input data
- **cols** – Optional. The list of key from the record to be used as columns in the table
- **rows** – Optional. The list of key from the record to be used as rows in the table
- **header** – Optional.
- **width** – Optional. The width of the component in the page, default (100, “%”)
- **height** – Optional. The height of the component in the page, default (330, “px”)
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

Tabulators Interface

`class epyk.interfaces.tables.CompTabulator.Tabulators(ui)`

figures(*records=None, cols: Optional[list] = None, rows: Optional[list] = None, width: Union[int, tuple, str] = (100, '%'), height: Union[int, tuple, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

Related Pages:

<https://www.npmjs.com/package/tabulator-extensions>

Parameters

- **records** – Optional. The list of dictionaries with the input data
- **cols** – Optional. The list of key from the record to be used as columns in the table
- **rows** – Optional. The list of key from the record to be used as rows in the table
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

hierarchy(*records=None, cols: Optional[list] = None, rows: Optional[list] = None, width: Union[int, tuple, str] = (100, '%'), height: Union[int, tuple, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
:param records: Optional. The list of dictionaries with the input data
:param cols: Optional. The list of key from the record to be used as columns in
↳the table
:param rows: Optional. The list of key from the record to be used as rows in
↳the table
:param width: Optional. A tuple with the integer for the component width and
↳its unit
:param height: Optional. A tuple with the integer for the component height and
↳its unit
:param html_code: Optional. An identifier for this component (on both Python
↳and Javascript side)
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

intensity(records=None, cols: Optional[list] = None, rows: Optional[list] = None, width: Union[int, tuple, str] = (100, '%'), height: Union[int, tuple, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Tags

Categories

Usage:

Related Pages:

<https://www.npmjs.com/package/tabulator-extensions>

Parameters

- **records** – Optional. The list of dictionaries with the input data
- **cols** – Optional. The list of key from the record to be used as columns in the table
- **rows** – Optional. The list of key from the record to be used as rows in the table
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

multi(records=None, cols: Optional[list] = None, rows: Optional[list] = None, width: Union[int, tuple, str] = (100, '%'), height: Union[int, tuple, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None)

Generic Tabulator configuration to get the package plus all the add-ons for Formatters and Editors. In the basic Tabulator entry point only the ones used on the Python will be added to the JavaScript page.

This configuration will load all the external JavaScript features to allow the full customisation.

Tags

Categories

Usage:

Related Pages:

<https://www.npmjs.com/package/tabulator-extensions>

Parameters

- **records** – Optional. The list of dictionaries with the input data
- **cols** – Optional. The list of key from the record to be used as columns in the table
- **rows** – Optional. The list of key from the record to be used as rows in the table
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component

- **profile** – Optional. A flag to set the component performance storage

table(*records=None, cols: Optional[list] = None, rows: Optional[list] = None, width: Union[int, tuple, str] = (100, '%'), height: Union[int, tuple, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

```
data = [{"A": 1, "B": 2}]
table = page.ui.tables.tabulators.table(data, cols=["A"], rows=["B"])
table.on("dblclick", page.js.alert("test"), profile=False)
```

Parameters

- **records** – Optional. The list of dictionaries with the input data
- **cols** – Optional. The list of key from the record to be used as columns in the table
- **rows** – Optional. The list of key from the record to be used as rows in the table
- **width** – Optional. The width of the component in the page, default (100, '%')
- **height** – Optional. The height of the component in the page, default (330, "px")
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

trafficlights(*records=None, cols: Optional[list] = None, rows: Optional[list] = None, width: Union[int, tuple, str] = (100, '%'), height: Union[int, tuple, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = None*)

Tags

Categories

Usage:

Related Pages:

<https://www.npmjs.com/package/tabulator-extensions>

Parameters

- **records** – Optional. The list of dictionaries with the input data
- **cols** – Optional. The list of key from the record to be used as columns in the table
- **rows** – Optional. The list of key from the record to be used as rows in the table
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component

- **profile** – Optional. A flag to set the component performance storage

There are links to existing Web Framework to rely on their components:

5.8.2.1.3 Full list of 81 Interfaces:

class epyk.interfaces.Interface.Components(*page: PageModel*)

Main interface for all components.

Usage:

```
# To change the default style for components.
page.ui.components_skin = {
    "buttons.absolute": {"clear": {"css": True, "cls": True}, "css": {"color": "red"},
    ↪ 'cls': ["cssbuttonbasic"]},
    "buttons.check": {"css": {"color": "green"}},
}
```

Parameters

page – The web page object.

property animations: Animations

Bespoke CSS and / or components with effects. This could be used to animate the cursor or add a loading events.

More details on the [Animations](#) property page

asterix(*tooltip: str, family: Optional[str] = None, width: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, height: Union[tuple, int, str] = (None, 'px'), color: Optional[str] = None, align: str = 'left', options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Icon

Usage:

```
:param tooltip:
:param family:
:param width: Optional. A tuple with the integer for the component width and ↪
    ↪ its unit
:param height: Optional. A tuple with the integer for the component height and ↪
    ↪ its unit
:param html_code: Optional. An identifier for this component (on both Python ↪
    ↪ and Javascript side)
:param color:
:param align:
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

property banners: Banners

Group all the available banners.

More details on the [Banners](#) property page

Usage:

```
top = page.ui.banners.top("text")
top.style.css.font_size = '40px'
```

property bars: *NavBars*

Group all the UI components dedicated to produce Navigation bar components such as navigation bar, footer, banner...

More details on the *Bars property* page

Usage:

```
page.ui
```

bespoke(*html_cls*, *args, **kwargs)

Hook to allow the creation of bespoke component using specific configurations. Components can be self-contained in a module and rely on external packages.

Tip: Look at the `Import.extend` function in order to add external Js and CSS modules to your environment.

Usage:

```
:param html_cls: Class. The bespoke HTML component
:param args: The python attributes used in the HTML component constructor
:param kwargs: The python attributes used in the HTML component constructor
```

breadcrumb(*values=None, selected: Optional[int] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = (30, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Breadcrumb

Add Breadcrumb information to the page.

Usage:

```
bc = page.ui.breadcrumb([
    {"text": 'part 1', 'url': 'part1'},
    {"text": 'part 2', 'url': 'part2'},
    {"text": 'part 3', 'url': 'part3'},
])

# This will change the link of part 2 and part 3 and add some extra information.
↪ in the link
bc.onReady([
    bc[1].dom.setAttribute("href", http.get("type").toString().prepend("part2?")),
    bc[2].dom.setAttribute("href", http.get("type").toString().prepend("http://
↪ www.w3schools.com/").add("/howto_css_breadcrumbs.asp"))
])
```

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/breadcrumb.py>

Parameters

- **values** – Optional. The breadcrumb record definition
- **selected** – Optional. The selected item index
- **width** – Optional. A tuple with the integer for the component width and its unit

- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **profile** – Optional. A flag to set the component performance storage

property buttons: *Buttons*

Group all the UI components dedicated to produce button or checkbox.

More details on the *Buttons property* page

Usage:

```
page.ui.buttons.absolute("Click Me")
page.ui.buttons.switch({'on': "true", 'off': 'false'})
page.ui.buttons.check(label="Label")
```

property calendars: *Calendar*

Group all the component related to the time and calendar management.

Usage:

```
content = {
    "2020-07-02": {'task1': 50, 'task2': 50},
    "2020-07-03": {'task1': 100},
    "2020-07-21": {'task4': 100},
    "2020-07-22": {'task4': 100}
}
july = page.ui.calendars.days(7, content, align="center", options={"colors": {
    → "task4": 'red'}})
```

captcha(*text: str = 'Submit', width: Union[tuple, int, str] = (None, 'px'), height: Union[tuple, int, str] = (None, 'px'), options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → HtmlCaptcha

Usage:

```
page.ui.captcha()
```

Parameters

- **text** – Optional. The button content for the captcha validation
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

property charts: *Graphs*

Group all the UI components dedicated to produce charts.

Different kind of charts framework are available (ChartJs, Plotly, C3, Billboard, NVD3, DC, Vis, Frappe, Vega, Apex or even D3).

Usage:

```
from epyk.mocks import urls
page = pk.Page()
chart = page.ui.charts.chartJs.line(y_columns=["Armenia", "France", "Germany"],
↪ x_axis="year")
page.body.onLoad([
    page.js.fetch(urls.CO2_DATA).csvtoRecords().get([
        chart.build(page.data.js.record("data").filterGroup("test").pivot("country",
↪ "co2", "year", type="float"))
    ])])
```

property codes: [Code](#)

Group all the UI Components dedicated to display code fragments.

This will wrap the Javascript module codemirror.

More details on the [Codes property](#) page

Usage:

```
page.ui.codes.css(".test {color: red}")
```

Related Pages:

<https://codemirror.net/doc/manual.html>

contents(*title: str = 'Contents', top: int = 10, right: int = 10, left: Optional[int] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] = (None, 'px'), html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → ContentsTable

Add a content table to the page.

Usage:

```
menu = page.ui.contents()
menu.add(page.ui.text("Simple text"))
menu.anchor("Test", 3, "#name")
page.ui.button("Button").click([
    menu.build([{"text": 'ok', "level": 0, "anchor": "#test"}])
])

page.body.onReady([
    menu.build([
        {"anchor": '#test', 'level': 1, 'text': 'Ok'}
    ])
])
```

Templates:

https://github.com/epykure/epyk-templates/blob/master/locals/components/contents_table.py
<https://github.com/epykure/epyk-templates/blob/master/locals/components/paragraph.py>

Parameters

- **title** – Optional. The title for the content table
- **top** – Optional. The top property affects the vertical position of a positioned element
- **right** – Optional. The right property affects the horizontal position of a positioned element

- **left** – Optional. The left property affects the horizontal position of a positioned element
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side)
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

css(*css_attrs: dict*)

Change the CSS Style of the main container in the page.

Usage:

```
page.ui.css({"color": "blue"})
```

Parameters

css_attrs – The CSS attributes to be applied.

property delimiters: *Delimiter*

Shortcut property to the various delimiters styles.

Related Pages:

<https://codepen.io/ibrahimjabbari/pen/ozinB>

Usage:

property drawers: *Drawers*

Group all the UI drawers components.

Usage:

extension(*package_name: str, alias: Optional[str] = None*)

Add an extension base on it is name.

Usage:

```
:param package_name: The package name.
:param alias: Optional. The alias for the link in report.ui.
```

property fields: *Fields*

Group all the UI components dedicated to produce input items.

Those components are editable items which need to be updated by the user of the dashboard. This category will take into account TextArea, input text...

Usage:

form(*components: Optional[List[Html]] = None, helper: Optional[str] = None, method: str = 'POST', action: str = '#', label: str = 'Submit') → Form*

Creates a new empty form.

Usage:

```
f = page.ui.form()
```

Parameters

- **components** – Optional. The HTML components to be added to the HTML form
- **helper** – Optional. The value to be displayed to the helper icon
- **method** – Optional. The method used to transfer data
- **action** – Optional. The end point for submitting data
- **label** – Optional. The text on the submit button

property forms: *Forms*

Group all the Forms components dedicated to drop data.

Related Pages:

https://www.w3schools.com/html/html_forms.asp

Usage:

property geo: *Geo*

Group all the UI components dedicated to produce Trees or selection items.

Usage:

```
l = page.ui.geo.mapbox.globe()
l.load([...])
l.options.style = 'mapbox://styles/mapbox/streets-v11'

marker = l.js.marker(-0.11, 51.508)
marker2 = l.js.marker(12.65147, 55.608166, options={"color": 'black', "rotation
↪": 45})
page.body.onReady([marker, marker2])
```

property icons: *Icons*

Group all the UI components dedicated to produce icon items.

This category of component will rely on the font-awesome library for the final display.

Usage:

property images: *Images*

Group all the UI components dedicated to produce image or collection of images.

Usage:

property inputs: *Inputs*

Group all the UI components dedicated to produce input items.

Those components are editable items which need to be updated by the user of the dashboard. This category will take into account TextArea, input text...

Usage:

```
json(data: Optional[dict] = None, width: Union[tuple, int, str] = (None, '%'), height: Union[tuple, int, str] =
(100, '%'), options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None)
→ HtmlJson
```

HTML component to display a Json.

Usage::

```
from epyk.mocks import urls
```

```
page = pk.Page() records = pd.DataFrame(page.py.requests.csv(urls.DEMO_COUNTRY))
records = records[records["Year"] == "2010"] # Create a link to download data as a Json
file viewer = page.ui.json(records.to_dict()) viewer.options.hoverPreviewEnabled = True
viewer.options.hoverPreviewArrayCount = 5
```

Related Pages:

<https://github.com/mohsen1/json-formatter-js>

Parameters

- **data** (*dict*) – Optional. The Json object to be display
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

property layouts: [Layouts](#)

Group all the UI components dedicated to produce component containers.

All the items in this category are dedicated for the structure of the dashboard and they are mainly holder of other components. This will mainly rely on bootstrap for the display of the different objects in the page.

Usage:

property links: [Links](#)

Group all the UI components dedicated to produce links to another page or website.

More details on the [Links property](#) page

Usage:

```
page.ui.links.external('data', 'www.google.fr', icon="fas fa-align-center",
options={"target": "_blank"})
page.ui.layouts.new_line(2)
```

property lists: [Lists](#)

Group all the UI components dedicated to produce list or selection items.

Simple list, trees or DropDown boxes will be part of this category of items.

Usage:

loading(*text: str = 'Loading', color: Optional[Union[str, bool]] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Loading

Entry point to the loading component.

This component will create a

- label component for the text
- icon component for the loading icon

Usage:

```
:param text: Optional. The text in the component (during the loading)
:param color: Optional. The font color in the component. Default inherit
:param options: Optional. Specific Python options available for this component
:param profile: Optional. A flag to set the component performance storage
```

property media: *Media*

Group all the UI components dedicated to produce media (video and audio) items.

Plain Vanilla HTML5 components.

Usage:

```
cam = page.ui.media.camera()
page.ui.button("start").click([cam.dom.start()])
page.ui.button("play").click([cam.dom.play()])
page.ui.button("Stop").click([cam.dom.stop()])
page.ui.button("record (Start)").click([cam.dom.record()])
page.ui.button("record (Stop)").click([cam.dom.record(False)])
page.ui.button("takepicture").click([cam.dom.takepicture()])
```

Templates:

```
menu(component: Union[Html, List[Html]], title: Optional[Union[str, dict]] = None, copy: str = 'fas fa-copy',
      editable: tuple = ('fas fa-user-edit', 'fas fa-user-lock'), refresh: str = 'fas fa-redo-alt', visible: tuple =
      ('fas fa-eye-slash', 'fas fa-eye'), post: Optional[dict] = None, height: tuple = (18, 'px'), save_funcs:
      Optional[Union[List[Union[str, JsDataModel]], str]] = None, update_funcs:
      Optional[Union[List[Union[str, JsDataModel]], str]] = None, menu_items=None, options:
      Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None)
```

TODO: Improve the editable feature for Markdown.

Usage:

```
p2 = page.ui.paragraph("paragraph", options={"markdown": True})
menu2 = page.ui.texts.menu(p2, save_funcs=[
    page.js.alert(p2.dom.content)
], update_funcs=[
    p2.build("Updated paragraph")
], profile=True)
```

Parameters

- **component** –
- **title** –
- **copy** –
- **editable** –
- **refresh** –
- **visible** –
- **post** –
- **height** –
- **save_funcs** –
- **update_funcs** –

- **menu_items** –
- **options** –
- **profile** –

property menus: *Menus*

Group all the UI menus.

Usage:

property modals: *Modals*

Group all the UI components dedicated to produce modal components.

Usage:

property navigation: *Navigation*

Group all the UI components dedicated to produce navigation components such as navigation bar, footer, banner...

More details on the *Navigation property* page

Usage:

```
nav = page.ui.navigation.nav(height=(30, 'px'), title={"This is an example":
↪ "by Epyk"})
for component in ["Menu 1", "Menu 2"]:
    title = nav.add_right(component)
```

property network: *Network*

Group all the UI Components dedicated to display messaging services.

This category will group (chat, RSS streams, forum, bot ...). Those components are interactive and they would require underlying services and databases in order to fully work.

More details on the *Networks property* page

Usage:

```
page = pk.Page()
page.ui.network
```

property numbers: *Numbers*

Group all the UI components dedicated to produce Numbers components.

The items in this category will not be editable and they will only provide nice number renderings.

Usage:

property panels: *Panels*

Group all the UI panels.

Usage:

property pictos

Group all the built-in pictogram.

Usage:

property pollers: *Poller*

Group all the UI with polling feature.

More details on the *Animations property* page

Usage:

```
page.ui.select()
poller = page.ui.pollers.live(2, [
    page.js.console.log(page.js.objects.date())
], components=[page.ui.text("Updated feeds")])
```

postit(*components: Optional[List[Html]] = None, anchor: Optional[Html] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Div

Usage:

Templates:

<https://github.com/epykure/epyk-templates/blob/master/locals/components/postit.py>

Parameters

- **components** – Optional.
- **anchor** – Optional.
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

print(*text: Optional[str] = None, end: str = '\n', html_code: Optional[str] = None, options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → Text

Mimic the print function available in Python. This will create a div container with the content as a string.

This function can be also used to display Python function. Inspect module will be used in this case to get the source code.

Usage:

```
import pandas
page.ui.print('pandas: {}'.format(pandas.__version__))
```

Parameters

- **text** – Optional. The content to be displayed.
- **end** – Optional. The end of line.
- **html_code** – Optional. An identifier for this component (on both Python and Javascript side).
- **options** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

Returns

:py:class: A HTML text component <epyk.core.html.HtmlText.Text>

property pyk: *Bespoke*

Bespoke catalog of components.

Usages:

```
text = page.ui.pyk.progress.circle()
text.style.css.color = "red"
```

qrcode(data=None, width: Union[tuple, int, str] = (128, 'px'), height: Union[tuple, int, str] = (128, 'px'), options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None) → HtmlQRCode

HTML component to display a QR Code from a string.

Usage:

Related Pages:

<https://davidshimjs.github.io/qrcodejs/>

TODO: Add options

Parameters

- **data** – Optional. The value to be converted to QR Code
- **width** – Optional. A tuple with the integer for the component width and its unit
- **height** – Optional. A tuple with the integer for the component height and its unit
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

property rich: *Rich*

Group all the UI components dedicated to produce rich HTML Components.

This category will take into account very specific and bespoke components.

Usage:

```
page.ui.fields.now(label="timestamp", color="red", helper="This is the report_
↪timestamp")
page.ui.rich.delta({'number': 100, 'prevNumber': 60, 'thresold1': 100,
↪'thresold2': 50}, helper="test")
page.ui.rich.info("text")
```

property sliders: *Sliders*

Group all the UI components dedicated to produce slider items.

Those components are interactive and can be used to filter the data on other items in the dashboard. Those components are mainly relying on JQuery and JQueryUi.

More details on the *Sliders property* page

Usage:

```
slider = page.ui.slider(5)
slider.options.step = 0.01
slider.options.slide(precision=2)
```

slideshow(*components: Optional[List[Html]] = None, width: Union[tuple, int, str] = (100, '%'), height: Union[tuple, int, str] = ('auto', ''), options: Optional[Union[bool, dict]] = None, profile: Optional[Union[bool, dict]] = None*) → *SlideShow*

SlideShow component for pictures from the tiny-slider library. More details regarding this library here: <https://github.com/ganlanyuan/tiny-slider>.

Usage:

```
ss = page.ui.slideshow([page.ui.text("Great results %s" % i) for i in
    ↪range(20)])

ss.add_index_changed([
    page.js.console.log("ok"),
    page.js.console.log(ss.dom.info.indexCached),
    page.js.console.log(ss.dom.info.index),
])
```

Related Pages:

<https://github.com/ganlanyuan/tiny-slider> <http://ganlanyuan.github.io/tiny-slider/demo/>

Parameters

- **components** – Optional. With the different components
- **width** – Optional. The component width in pixel or percentage
- **height** – Optional. The component height in pixel
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

property steppers: *Steppers*

Group all the UI steppers components.

Usage:

```
s = page.ui.steppers.arrow()
s.options.column_title = "name"
btn = page.ui.button("Load")
btn.click(
    s.build([
        {"value": 'test 1', "status": 'success', 'name': "1", 'title': 'test',
    ↪ "label": 'test'},
        {"value": 'test 2', "status": 'pending', "tooltip": "processing"},
        {"value": 'test 3'},
        {"value": 'test 4'}]))
)
```

property steps: *Steppers*

Group all the UI steps components.

Usage:

property tables: *Tables*

Group all the UI components dedicated to produce tables or pivot tables.

Different kind of tables are available in the framework (Tabulator, DataTable, PivotTable or even a bespoke implementation).

Usage:

property tags: *Tags*

Group all the other tags available in HTML.

Those tags can be considered as normal HTML component, which means Js and CSS features are also available.

Usage:

property texts: *Texts*

Group all the UI components dedicated to produce text components.

The items in this category will not be editable and they will only provide nice text structure like paragraph, formatted text...

Usage:

property timelines: *Timelines*

Usage:

property titles: *Titles*

Group all the UI components dedicated to produce titles.

More details on the *Titles property* page

Usage:

```
page.ui.titles.head("test")
```

property trees: *Trees*

Group all the UI components dedicated to produce Trees or selection items.

Usage:

property vignets: *Vignets*

Group all the UI components dedicated to produce rich HTML Components.

This category will take into account very specific and bespoke components.

Usage:

```
page.ui.vignets.number(500, "Test")
page.ui.vignets.number(500, "Test 2 ", options={"url": "http://www.google.fr"})
page.ui.vignets.block({
  "text": 'This is a brand new python framework', "title": 'New Python Web↵
↵Framework',
  "button": {"text": 'Get Started', 'url': "/getStarted"}, 'color': 'green'})
```

5.8.2.2 Javascript Interface

The Page object will allow you to write plain JavaScript from the js property. It will then make available most of the common features defined in the JavaScript world.

No need to move to JavaScript, to edit some extra configuration files or even to write wrappers in Strings, this interface will provide you auto completion and links to the underlying web site to learn more about those concepts.

Each component will have a JavaScript entry point which will either redirect to this or will define a specific one in line with the external package definition. More details on the component design [a link](#).

5.8.2.2.1 Console

class `epyk.core.js.Js.JsConsole`(*page: Optional[PageModel] = None*)

This is a wrapper to the Console.

Related Pages:

<https://medium.freecodecamp.org/how-to-get-the-most-out-of-the-javascript-console-b57ca9db3e6d>

property clear

The console.clear() method clears the console.

Usage:

```
page.js.console.clear
```

Related Pages:

https://www.w3schools.com/jsref/met_console_clear.asp

Returns

The Javascript String used to clear the console (F12 in standard browsers).

property debugger

Trigger a Javascript debugger from this point. The Javascript will be stopped. It will be possible to check the process step by step in the browser using F12.

Usage:

```
page.js.console.debugger
```

Related Pages:

https://www.w3schools.com/jsref/jsref_debugger.asp

Returns

The Javascript Keyword to trigger the browser debugger.

error(*data: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)

The console.error() method writes an error message to the console.

Related Pages:

https://www.w3schools.com/jsref/met_console_error.asp

Parameters

- **data** – The Javascript fragment.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

The Javascript String used to clear the console (F12 in standard browsers)

info(*data: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)

The console.info() method writes a message to the console.

Related Pages:

https://www.w3schools.com/jsref/met_console_info.asp

Parameters

- **data** – The Javascript fragment.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

The Javascript String used to clear the console (F12 in standard browsers)

log(*data: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None, skip_data_convert: bool = False*)

The console.log() method writes a message to the console.

Usage:

```
page.js.console.log("Test")
```

Related Pages:

https://www.w3schools.com/jsref/met_console_log.asp

Parameters

- **data** – The Javascript fragment.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.
- **skip_data_convert** – Optional. Flag to specify to the framework if a Json conversion is needed.

Returns

The Javascript String used to clear the console (F12 in standard browsers)

perf(*js_code: str, label: Optional[str] = None*)

Shortcut function to display performances from a variable. The variable must be global. Namely the name should start with window.

Parameters

- **js_code** – The variable var name use to compute the performance.
- **label** – Optional. The description.

service(*msg: str, headers: Optional[dict] = None*)

Send logs to the backend.

Parameters

- **msg** – The log message to be sent to the backend.

- **headers** – the service headers.

table(*data: Union[str, JsDataModel], js_header: Optional[list] = None*) → JsFunction

The console.table() method writes a table in the console view.

Related Pages:

https://www.w3schools.com/jsref/met_console_table.asp

Parameters

- **data** – The data to fill the table with.
- **js_header** – Optional. An array containing the names of the columns to be included in the table.

Returns

The Javascript String used to clear the console (F12 in standard browsers).

time(*html_code: Union[str, JsDataModel]*) → JsNumber

The console.time() method starts a timer in the console view.

Related Pages:

https://www.w3schools.com/jsref/met_console_time.asp

Parameters

html_code – Use the label parameter to give the timer a name.

Returns

A Python Javascript Number.

timeEnd(*html_code: Union[str, JsDataModel]*)

The console.timeEnd() method ends a timer, and writes the result in the console view.

Related Pages:

https://www.w3schools.com/jsref/met_console_timeend.asp

Parameters

html_code – The name of the timer to end.

Returns

The Javascript String used to clear the console (F12 in standard browsers).

tryCatch(*js_funcs: Union[str, list], js_funcs_errs: Union[str, list] = 'console.warn(err.message)', profile: Optional[Union[bool, dict]] = False*)

Javascript Try Catch Exceptions.

Related Pages:

https://www.w3schools.com/jsref/jsref_obj_error.asp

Parameters

- **js_funcs** – The Javascript functions.
- **js_funcs_errs** – The Javascript functions.
- **profile** – Optional. A flag to set the component performance storage.

Returns

The Javascript String used to clear the console (F12 in standard browsers)

warn(*data: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)

The console.warn() method writes a warning to the console.

Related Pages:

https://www.w3schools.com/jsref/met_console_warn.asp

Parameters

- **data** – The Javascript fragment.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

The Javascript String used to clear the console (F12 in standard browsers)

5.8.2.2.2 Window

Wrapper to the Javascript Window module

Allows to save key/value pairs in a web browser. Stores the data with no expiration date The localStorage and sessionStorage properties allow to save key/value pairs in a web browser.

Related Pages:

https://www.w3schools.com/Jsref/prop_win_localstorage.asp

class epyk.core.js.JsWindow.**JsHistory**(*page: PageModel*)

Interface to the Javascript history module.

Related Pages:

https://www.w3schools.com/js/js_window_history.asp

back()

The back() method loads the previous URL in the history list.

Usage:

```
rptObj.js.window.history.back()
```

Related Pages:

https://www.w3schools.com/jsref/met_his_back.asp

Returns

The Javascript String to be added to the page

cleanState(*keys: List[str]*)

Remove all attributes which are not useful or should not be passed.

Usage:

```
btn = page.ui.button("Clean URL")
btn.click([page.js.window.history.cleanState(["date"])]])
```

Parameters

keys – The attributes keys

deleteState(*key: JsDataModel*)

Remove a specific attribute from the url

Usage:

```
btn = page.ui.button("Remove from URL")
btn.click([page.js.window.history.deleteState("date")])
```

Parameters

key – The attribute key.

forward()

The forward() method loads the next URL in the history list.

Related Pages:

https://www.w3schools.com/jsref/met_his_forward.asp

Returns

The Javascript String to be added to the page.

go(*number: Union[JsDataModel, int]*)

The go() method loads a specific URL from the history list.

Related Pages:

https://www.w3schools.com/jsref/met_his_go.asp

Parameters

number (*Union[primitives.JsDataModel, int]*) – The parameter can either be a number which goes to the URL within

the specific position (-1 goes back one page, 1 goes forward one page), or a string.

Returns

The Javascript String to be added to the page

property length

The length property returns the number of URLs in the history list of the current browser window.

Usage:

```
rptObj.js.window.history.length
```

Related Pages:

https://www.w3schools.com/jsref/prop_his_length.asp

Returns

A Number, representing the number of entries in the session history

pushState(*state*, *title*, *url*)

Pushes the given data onto the session history stack with the specified title and, if provided, URL.

Note that pushState() never causes a hashchange event to be fired, even if the new URL differs from the old URL only in its hash

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/History_API

Parameters

- **state** – The state object is a JavaScript object which is associated with the new history entry created by pushState()
- **title** – Firefox currently ignores this parameter, although it may use it in the future Passing the empty string here should be safe against future changes to the method. Alternatively, you could pass a short title for the state to which you're moving.
- **url** – The new history entry's URL is given by this parameter. Note that the browser won't attempt to load this URL after a call to pushState(),

Returns**replaceState**(*state*, *title*, *url*)

history.replaceState() operates exactly like history.pushState() except that replaceState() modifies the current history entry instead of creating a new one.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/History_API

Parameters

- **state** – The state object is a JavaScript object which is associated with the new history entry created by pushState()
- **title** – Firefox currently ignores this parameter, although it may use it in the future Passing the empty string here should be safe against future changes to the method. Alternatively, you could pass a short title for the state to which you're moving.
- **url** – The new history entry's URL is given by this parameter. Note that the browser won't attempt to load this URL after a call to pushState(),

updateState(*key*: *str*, *val*: *str*)

Wrapper function

This function is a simple wrapping function on top of the pushState history method. The purpose of this method is to make easier the update of the url whenever a component in the framework is updated.

Usage:

```
component.js.window.history.updateState(self.htmlCode, self.val)
```

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/History_API

Parameters

- **key** (*str*) – The key to be added or updated in the current URL.

- **val** (*str*) – The value to be changed to the current URL.

Returns

The Javascript String for the method.

updateStateFromComponent (*component: HtmlModel*)

Add or update the url value for the specific component to keep them in case of refresh.

Usage:

```
dt = page.ui.date(html_code="date")
input = page.ui.input(html_code="input")
dt.select([
    page.js.window.history.updateStateFromComponent(dt),
    page.js.window.history.updateStateFromComponent(input)
```

Parameters

component – The HTML component

class epyk.core.js.JsWindow.JsLocalStorage**clear()**

The clear() method removes all the Storage Object item for this domain.

The localStorage object stores data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

Usage:

```
jsObj.localStorage.clear()
```

Related Pages:

https://www.w3schools.com/jsref/met_storage_clear.asp

Returns

Void

getItem (*key: Union[JsDataModel, str]*)

Syntax for READING data from localStorage:

The localStorage object stores data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

Usage:

```
jsObj.localStorage.getItem("lastname")
```

Related Pages:

https://www.w3schools.com/jsref/met_storage_getitem.asp

Parameters

key (*Union[primitives.JsDataModel, str]*) – A String specifying the name of the key you want to get the value of.

Returns

A String, representing the value of the specified key.

key(*i: Union[JsDataModel, int]*)

The key() method returns name of the key with the specified index.

The localStorage object stores data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

Usage:

```
jsObj.localStorage.key(0)
```

Related Pages:

https://www.w3schools.com/jsref/met_storage_key.asp

Parameters

i (*Union[primitives.JsDataModel, int]*) – A Number representing the index of the key you want to get the name of.

Returns

A String, representing the name of the specified key

removeItem(*key: Union[JsDataModel, str]*)

The removeItem() method removes the specified Storage Object item.

The localStorage object stores data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

Usage:

```
jsObj.localStorage.removeItem("lastname")
```

Related Pages:

https://www.w3schools.com/jsref/met_storage_removeitem.asp

Parameters

key (*Union[primitives.JsDataModel, str]*) – A String specifying the name of the item you want to remove.

Returns

Void

setItem(*key: Union[JsDataModel, str], data: Any*)

Syntax for SAVING data to localStorage.

The localStorage object stores data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

Usage:

```
jsObj.localStorage.setItem("lastname", "test")
```

Related Pages:

https://www.w3schools.com/jsref/met_storage_setitem.asp

Parameters

- **key** (*Union[primitives.JsDataModel, str]*) – A String specifying the name of the key you want to set the value of.

- **data** (*Aby*) – A String specifying the value of the key you want to set the value of.

Returns

A String, representing the inserted value.

class `epyk.core.js.JsWindow.JsSessionStorage`

The localStorage and sessionStorage properties allow to save key/value pairs in a web browser.

The sessionStorage object stores data for only one session (the data is deleted when the browser tab is closed).

https://www.w3schools.com/Jsref/prop_win_sessionstorage.asp

clear()

Syntax for REMOVING ALL saved data from sessionStorage

The sessionStorage object stores data for only one session (the data is deleted when the browser tab is closed).

Related Pages:

https://www.w3schools.com/jsref/prop_win_sessionstorage.asp

getItem(*key: Union[JsDataModel, str]*)

Syntax for READING data from sessionStorage

The sessionStorage object stores data for only one session (the data is deleted when the browser tab is closed).

Usage:

```
jsObj.sessionStorage.getItem("lastname")
jsObj.console.log(jsObj.sessionStorage.getItem("lastname"))
```

Parameters

key (*Union[primitives.JsDataModel, str]*) –

key(*i: Union[JsDataModel, int]*)

The sessionStorage object stores data for only one session (the data is deleted when the browser tab is closed).

Parameters

i (*Union[primitives.JsDataModel, int]*) – The key number.

removeItem(*data, key: Optional[Union[JsDataModel, str]] = None, is_py_data: bool = False, js_funcs: Optional[Union[list, str]] = None*)

Syntax for REMOVING ALL saved data from sessionStorage.

The sessionStorage object stores data for only one session (the data is deleted when the browser tab is closed).

Usage:

```
jsObj.sessionStorage.removeItem("lastname")
```

Related Pages:

https://www.w3schools.com/jsref/met_storage_removeitem.asp

Parameters

- **data** –

- **key** –
- **is_py_data** (*bool*) –
- **js_funcs** (*Union[list, str]*) –

setItem(*key: Union[JsDataModel, str], data: Any*)

Syntax for SAVING data to sessionStorage.

The sessionStorage object stores data for only one session (the data is deleted when the browser tab is closed).

Usage:

```
jsObj.sessionStorage.setItem("lastname", "Smith")
jsObj.sessionStorage.setItem("lastname", jsObj.objects.get("bin")),
```

Related Pages:

https://www.w3schools.com/Jsref/prop_win_sessionstorage.asp

Parameters

- **key** (*Union[primitives.JsDataModel, str]*) – The key used to store the data in the session cache.
- **data** (*Any*) –

class epyk.core.js.JsWindow.JsUrl

createObjectURL(*data: Union[JsDataModel, str]*)

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Blob>

Parameters

data (*Union[primitives.JsDataModel, str]*) –

class epyk.core.js.JsWindow.JsWindow(*page: Optional[PageModel] = None*)

The window object represents an open window in a browser.

If a document contain frames (<iframe> tags), the browser creates one window object for the HTML document, and one additional window object for each frame.

Related Pages:: https://www.w3schools.com/Jsref/obj_window.asp

property URL

addEventListener(*event_type: Union[JsDataModel, str], js_funcs: Union[JsDataModel, str], window_id: str = 'window', profile: Optional[Union[bool, dict]] = False*)

Parameters

- **event_type** (*Union[primitives.JsDataModel, str]*) –
- **js_funcs** (*Union[primitives.JsDataModel, str]*) –
- **window_id** (*str*) – Optional. The JavaScript window object reference variable.
- **profile** (*Optional[Union[dict, bool]]*) – Optional. A flag to set the component performance storage.

alert(*data*, *js_funcs*: *Optional[Union[list, str]] = None*, *window_id*: *str = 'window'*, *skip_data_convert*: *bool = False*)

The alert() method displays an alert box with a specified message and an OK button.

Usage:

```
page.js.window.alert("Test")
page.js.alert("Test 2")
```

Related Pages:

https://www.w3schools.com/jsref/met_win_alert.asp

Parameters

- **data** – Optional. Specifies the text to display in the alert box, or an object converted into a string and displayed
- **js_funcs** (*Union[list, str]*) – A JsFnc or a list of JsFncs.
- **window_id** (*str*) – Optional. The JavaScript window object reference variable.
- **skip_data_convert** (*bool*) –

atob(*data*: *Union[str, JsDataModel]*, *js_funcs*: *Optional[Union[list, str]] = None*, *window_id*: *str = 'window'*)

Decodes a base-64 encoded string.

Usage:

```
jsObj.window.btoa("Test").setVar("bin")
jsObj.window.atob(jsObj.objects.get("bin"))
```

Related Pages:

https://www.w3schools.com/jsref/met_win_atob.asp

Parameters

- **data** (*Union[str, primitives.JsDataModel]*) – The string which has been encoded by the btoa() method.
- **js_funcs** (*Union[list, str]*) – A JsFnc or a list of JsFncs
- **window_id** (*str*) – Optional. The JavaScript window object reference variable.

btoa(*data*: *Union[str, JsDataModel]*, *js_funcs*: *Optional[Union[list, str]] = None*, *window_id*: *str = 'window'*)

Encodes a string in base-64.

Usage:

```
jsObj.window.btoa("Test").setVar("bin")
```

Related Pages:

https://www.w3schools.com/jsref/met_win_btoa.asp

Parameters

- **data** (*Union[str, primitives.JsDataModel]*) – Required. The string to be encoded.

- **js_funcs** (*Union[list, str]*) – The PyJs functions.
- **window_id** (*str*) – Optional. The JavaScript window object reference variable.

clearInterval(*var_id: str, window_id: str = 'window'*)

The clearInterval() method clears a timer set with the setInterval() method.

The ID value returned by setInterval() is used as the parameter for the clearInterval() method.

Usage:

```
jsObj.window.setInterval([jsObj.console.log(jsObj.math.random())], 500).setVar(
↪ "interval"),
jsObj.window.clearInterval(jsObj.objects.get("interval"))
```

Related Pages:

https://www.w3schools.com/jsref/met_win_clearinterval.asp

#TODO: Check if interval is unique

Parameters

- **var_id** (*str*) – A PythonJs object (JsArray, JsObject...) or reference
- **window_id** (*str*) – The JavaScript window object.

Returns

Void, The Javascript String

clearTimeout(*data, js_funcs: Optional[Union[list, str]] = None, window_id: str = 'window'*)

The clearTimeout() method clears a timer set with the setTimeout() method. The ID value returned by setTimeout() is used as the parameter for the clearTimeout() method.

Related Pages:

https://www.w3schools.com/jsref/met_win_cleartimeout.asp

Parameters

- **data** –
- **js_funcs** (*Union[list, str]*) – The PyJs functions.
- **window_id** (*str*) – The JavaScript window object.

close(*window_id: str = 'window'*)

Closes the current window.

Related Pages:

https://www.w3schools.com/jsref/met_win_close.asp

Parameters

window_id (*str*) – Optional. The JavaScript window object reference variable.

Returns

The String representing the Javascript function.

property document

Interface to the DOM object on the current window.

Returns

A Python JsDoms object wrapping the DOM Js interface.

download(*data*, *file_name*: str, *profile*: Optional[Union[bool, dict]] = False)

Download the data from a flat file.

Usage:

```
page.js.window.download(rptObj.js.window.btoa(rptObj.js.objects.get("test")),  
↪ fileName="test.txt")
```

Parameters

- **data** –
- **file_name** –
- **profile** (Optional[Union[dict, bool]]) – Optional. A flag to set the component performance storage.

Returns

Void,

property events

Property to all the events.

focus(*window_id*: str = 'window')

The focus() method sets focus to the current window

Related Pages:

https://www.w3schools.com/Jsref/met_win_focus.asp

Parameters

window_id (str) – Optional. The JavaScript window object reference variable.

Returns

Void, The Javascript String

getComputedStyle(*element*, *pseudo_element*=None, *window_id*: str = 'window')

The getComputedStyle() method gets all the actual (computed) CSS property and values of the specified element.

Related Pages:

https://www.w3schools.com/jsref/jsref_getcomputedstyle.asp

Parameters

- **element** – The element to get the computed style for.
- **pseudo_element** –
- **window_id** (str) – The JavaScript window object.

Returns

A CSSStyleDeclaration object containing CSS declaration block of the element

getSelection(*window_id*: *str* = 'window')

Returns a Selection object representing the range of text selected by the user.

Parameters

window_id (*str*) – The JavaScript window object

getVar(*var_id*: *str*, *window_id*: *str* = 'window')

Get the Javascript Variable name.

Parameters

- **var_id** (*str*) – The Variable name.
- **window_id** (*str*) – The JavaScript window object.

Returns

Return the piece of script to be added to the Javascript.

property history: *JsHistory*

Interface to the History object.

Usage:

```
dt = page.ui.date(html_code="date")
input = page.ui.input(html_code="input")
dt.select([
    page.js.window.history.updateStateFromComponent(dt),
    page.js.window.history.updateStateFromComponent(input)
```

Returns

A Python Js History object.

property innerHeight

The innerHeight property returns the height of a window's content area.

Related Pages:

https://www.w3schools.com/jsref/prop_win_innerheight.asp

Parameters

window_id (*str*) – String. Optional. The window reference.

location(*url*: *str*, *window_id*: *str* = 'window')

Change the window and open the page specify by the url.

Parameters

- **url** (*str*) – The new page url.
- **window_id** (*str*) – Optional. The JavaScript window object.

moveBy(*x*: *int*, *y*: *int*, *window_id*: *str* = 'window')

The moveBy() method moves a window a specified number of pixels relative to its current coordinates.

Related Pages:

https://www.w3schools.com/Jsref/met_win_moveby.asp

Parameters

- **x** (*int*) – The horizontal move in pixel.

- **y** (*int*) – The vertical move in pixel.
- **window_id** (*str*) – Optional. The JavaScript window object reference variable.

onBeforeUnload(*js_funcs: Union[list, str]*)

Parameters

js_funcs (*Union[list, str]*) – A JsFnc or a list of JsFncs

onPageShow(*js_funcs: Union[list, str]*)

Parameters

js_funcs (*Union[list, str]*) – The PyJs functions.

open(*url: str, name: str = '_self', specs: Optional[list] = None, replace: Optional[bool] = None, window_id: str = 'window'*)

Opens a new browser window

Related Pages:

https://www.w3schools.com/Jsref/met_win_open.asp

Parameters

- **url** (*str*) – Optional. Specifies the URL of the page to open. If no URL is specified, a new window/tab with [about:blank](#) is opened
- **name** (*str*) – Optional. Specifies the target attribute or the name of the window.
- **specs** (*list*) – Optional. A comma-separated list of items, no whitespaces.
- **replace** (*bool*) – Optional. Specifies whether the URL creates a new entry or replaces the current entry in the history list
- **window_id** (*str*) – Optional. The JavaScript window object reference variable.

postData(*data*)

Parameters

data –

print_(*window_id: str = 'window'*)

Prints the content of the current window.

Related Pages:

https://www.w3schools.com/Jsref/met_win_print.asp

Parameters

window_id (*str*) – Optional. The JavaScript window object reference variable.

Returns

Void, The Javascript String

scroll(*x: int, y: int, window_id: str = 'window'*)

The Window.scroll() method scrolls the window to a particular place in the document.

Related Pages:

<https://developer.mozilla.org/uk/docs/Web/API/Window/scroll>

Parameters

- **x** (*int*) – The pixel along the horizontal axis of the document that you want displayed in the upper left.
- **y** (*int*) – The pixel along the vertical axis of the document that you want displayed in the upper left.
- **window_id** (*str*) – Optional. The JavaScript window object reference variable.

property scrollEndPage

The scrollEndPage property indicates if the page is scrolled to the end.

Parameters

window_id (*str*) – Optional. The window reference.

property scrollMaxY

The Window.scrollMaxY read-only property returns the maximum number of pixels that the document can be scrolled vertically.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Window/scrollMaxY>

Parameters

window_id – Optional. The window reference.

property scrollPercentage

The scrollPercentage property return the percentage of the page scrolled.

Parameters

window_id (*str*) – Optional. The window reference.

scrollTo(*x: Optional[int] = None, y: Optional[int] = None, window_id: str = 'window'*)

The window.scrollTo() go to a particular point.

Parameters

- **x** (*int*) – Optional.
- **y** (*int*) – Optional.
- **window_id** (*str*) – Optional. The JavaScript window object reference variable.

scrollUp(*window_id: str = 'window'*)

Parameters

window_id (*str*) – Optional. The JavaScript window object reference variable.

property scrollY

The read-only scrollY property of the Window interface returns the number of pixels that the document is currently scrolled vertically.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Window/scrollY>

Parameters

window_id (*str*) – Optional. The window reference.

setInterval(*js_funcs*: Union[list, str], *var_id*: str, *milliseconds*: int, *window_id*: str = 'window', *set_var*: bool = True, *profile*=False, *run_on_start*: bool = False)

The setInterval() method calls a function or evaluates an expression at specified intervals (in milliseconds).

The setInterval() method will continue calling the function until clearInterval() is called, or the window is closed.

Usage:

```
jsObj.window.setInterval([jsObj.console.log(jsObj.math.random())], 5000)
```

Related Pages:

https://www.w3schools.com/jsref/met_win_setinterval.asp

#TODO: Add a control on setInterval to only have one created

Parameters

- **js_funcs** (Union[list, str]) – The function that will be executed.
- **var_id** (str) – The JavaScript variable name.
- **milliseconds** (int) – The intervals (in milliseconds) on how often to execute the code.

If the value is less than 10, the value 10 is used. :param str window_id: The JavaScript window object. :param bool set_var: Set the variable on the JavaScript side. :param bool profile: A flag to set the component performance storage. :param bool run_on_start: Flag to start the call at the start.

setTimeout(*js_funcs*: Union[list, str], *milliseconds*: int = 0, *window_id*: str = 'window', *profile*: Optional[Union[bool, dict]] = False)

The setTimeout() method calls a function or evaluates an expression after a specified number of milliseconds.

Related Pages:

https://www.w3schools.com/jsref/met_win_settimeout.asp

Parameters

- **js_funcs** – The function that will be executed.
- **milliseconds** – Optional. The number of milliseconds to wait before executing the code.
- **window_id** – Optional. The JavaScript window object.
- **profile** – Optional. Set to true to get the profile for the function on the Javascript console.

toggleInterval(*js_funcs*: Union[list, str], *var_id*: str, *milliseconds*, *window_id*: str = 'window')

Usage:

```
page.ui.button("Interval Toggle").click([
    page.js.window.toggleInterval(rptObj.js.console.log('Print called'), 'test', 400),
])
```

Parameters

- **js_funcs** (Union[list, str]) – The PyJs functions.
- **var_id** (str) – A PythonJs object (JsArray, JsObject...) or reference.

- **milliseconds** (*int*) – Optional. The number of milliseconds to wait before executing the code.
- **window_id** (*str*) – Optional. The JavaScript window object.

class epyk.core.js.JsWindow.JsWindowEvent

addClickListener(*js_funcs: Union[list, str], window_id: str = 'window', sub_events: Optional[list] = None*)

Parameters

- **js_funcs** (*Union[list, str]*) – The PyJs functions.
- **window_id** (*str*) – The window object reference.
- **sub_events** (*list*) –

addContentLoaded(*js_funcs: Union[list, str], window_id: str = 'window'*)

The DOMContentLoaded event fires when the initial HTML document has been completely loaded and parsed, without waiting for stylesheets, images, and subframes to finish loading.

Usage:

```
page.js.addOnLoad(
page.js.window.events.addContentLoaded(rptObj.js.alert("DOM fully loaded and
↪parsed")))
```

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/Window/DOMContentLoaded_event

Parameters

- **js_funcs** (*Union[list, str]*) – The PyJs functions.
- **window_id** (*str*) – The window object reference.

addEventListener(*event_type: Union[JsDataModel, str], js_funcs: Union[list, str], window_id: str = 'window', sub_events: Optional[list] = None, profile: Optional[Union[bool, dict]] = False*)

Parameters

- **event_type** (*Union[primitives.JsDataModel, str]*) –
- **js_funcs** – The PyJs functions.
- **window_id** (*str*) – The window object reference.
- **sub_events** (*list*) – List of names you want your underlying function to have as arguments.
- **profile** (*Optional[Union[dict, bool]]*) – Optional. A flag to set the component performance storage.

addScrollListener(*js_funcs: Union[list, str], window_id: str = 'window'*)

Parameters

- **js_funcs** (*Union[list, str]*) – The PyJs functions.
- **window_id** (*str*) – The window object reference.

5.8.2.2.3 Location

The location object contains information about the current URL.

The location object is part of the window object and is accessed through the window.location property.

Related Pages:

https://www.w3schools.com/jsref/obj_location.asp

class `epyk.core.js.JsLocation.JsLocation`

JavaScript Location module.

classmethod `assign(url: Union[str, JsDataModel]) → JsFunction`

The assign() method loads a new document.

Related Pages:

https://www.w3schools.com/jsref/met_loc_assign.asp

Parameters

url – Specifies the URL of the page to navigate to

classmethod `download(url: Union[str, JsDataModel], name: Union[str, JsDataModel] = 'download') → JsVoid`

Download data from the url.

Parameters

- **url** – The url of the image
- **name** – Optional. The name of the file

classmethod `getUrlFromArrays(data: Union[list, JsDataModel], delimiter: Union[str, JsDataModel] = ',', charset: str = 'utf-8', end_line: Union[str, JsDataModel] = '\r\n')`

Convert data to a URL.

Parameters

- **data** – A JavaScript array
- **delimiter** – Optional. The column delimiter
- **charset** – Optional.
- **end_line** – Optional.

classmethod `getUrlFromData(data: Union[dict, JsDataModel], options: Optional[Union[dict, JsDataModel]] = None)`

Convert data to a URL.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Blob>

Parameters

- **data** – Input data to be converted
- **options** – Optional. Blob definition properties

property hash: JsonObject

The hash property sets or returns the anchor part of a URL, including the hash sign (#).

Usage:

```
jsObj.location.hash
```

Related Pages:

https://www.w3schools.com/jsref/prop_loc_hash.asp

Returns

A String, representing the anchor part of the URL, including the hash sign (#).

property host: JsString

The host property sets or returns the hostname and port of a URL.

Usage:

```
jsObj.location.host
```

Related Pages:

https://www.w3schools.com/jsref/prop_loc_host.asp

Returns

Return the hostname and port of the current URL.

property hostname: JsString

The hostname property sets or returns the hostname of a URL.

Usage:

```
page.location.hostname
```

Related Pages:

https://www.w3schools.com/jsref/obj_location.asp

Returns

Return the hostname property.

classmethod href(href: Optional[Union[JsDataModel, str]] = None, secured: bool = False) → JsonObject

The href property sets or returns the entire URL of the current component.

Usage:

```
page.js.location.href("https://www.w3schools.com/howto/howto_js_fullscreen.asp")
```

Related Pages:

https://www.w3schools.com/jsref/prop_loc_href.asp

Parameters

- **href** – Optional. Set the href property
- **secured** – Optional. The secured flag

Returns

A String, representing the entire URL of the page, including the protocol (like <http://>).

mail(*mails: List[str], subject: str, body: str*)

The mailto link when clicked opens users default email program or software. A new email page is created with “To” field containing the address of the name specified on the link by default.

Usage:

```
page.js.location.mail(["test@gmail.com"], "This is a test", "This is the email  
↪ 's content")
```

Related Pages:

http://www.tutorialspark.com/html5/HTML5_email_mailto.php

Parameters

- **mails** – The email addresses
- **subject** – The email’s subject
- **body** – The email’s content

Returns

The Javascript string.

classmethod open_new_tab(*url: Union[str, JsDataModel], name: Union[str, JsDataModel] = '_blank', specs: Optional[Union[JsDataModel, str]] = None, replace: Optional[Union[JsDataModel, str]] = None, window_id: str = 'window', data: Optional[dict] = None, secured: bool = False*) → JsFunction

Opens a new browser window in a new tab (duplicated but part of the Window module).

Usage:

```
page.js.location.open_new_tab("www.google.fr")
```

Related Pages:

https://www.w3schools.com/Jsref/met_win_open.asp

Parameters

- **url** – Optional. Specifies the URL of the page to open. If no URL is specified, a new window/tab with <about:blank> is opened
- **name** – Optional. Specifies the target attribute or the name of the window. Default `_blank`
- **specs** – Optional. A comma-separated list of items, no whitespaces
- **replace** – Optional. Specifies whether the URL creates a new entry or replaces the current entry in the history list
- **window_id** – Optional. The JavaScript window object
- **data** – Optional. The url parameters
- **secured** – Optional. The secure flag

property origin: JsString

The origin property returns the protocol, hostname and port number of a URL.

Usage:

```
page.js.location.origin + page.js.location.pathname
```

Related Pages:

https://www.w3schools.com/jsref/prop_loc_origin.asp

Returns

A String, representing the protocol (including `://`), the domain name (or IP address) and port number (including the colon sign `:`) of the URL. For URL's using the "file:" protocol, the return value differs between browser)

property pathname: JsString

The hostname property sets or returns the hostname of a URL.

Usage:

```
jsObj.location.pathname
```

Related Pages:

https://www.w3schools.com/jsref/obj_location.asp

Returns

Return the pathname property.

property port: JsString

The port property sets or returns the port number the server uses for a URL.

Related Pages:

https://www.w3schools.com/jsref/prop_loc_port.asp

Returns

A String, representing the port number of a URL.

classmethod postTo(url: str, data: dict, method: str = 'POST', target: str = '_blank')

This method will create an internal form and submit the response exactly like a post of a form to another page.

Related Pages:

https://www.w3schools.com/jsref/dom_obj_form.asp

Parameters

- **url** – The target url
- **data** – The url parameters
- **method** – Optional. The method used to send the data. Default POST
- **target** – Optional. Target method to access the new page

classmethod reload(*force_get: bool = False*)

The reload() method is used to reload the current document.

The reload() method does the same as the reload button in your browser.

Related Pages:

https://www.w3schools.com/jsref/met_loc_reload.asp

Parameters

force_get – Optional. Specifies the type of reloading: false - Default. Reloads the current page from the cache true - Reloads the current page from the server

classmethod replace(*url: Union[str, JsDataModel], secured: bool = False*) → JsFunction

The replace() method replaces the current document with a new one.

The difference between this method and assign(), is that replace() removes the URL of the current document from the document history, meaning that it is not possible to use the “back” button to navigate back to the original document.

Related Pages:

https://www.w3schools.com/jsref/met_loc_replace.asp

Parameters

- **url** – Specifies the URL of the page to navigate to
- **secured** – Optional. If the http is missing. This will be used to fix the url

property search: JsString

The search property sets or returns the querystring part of a URL, including the question mark (?).

Related Pages:

https://www.w3schools.com/jsref/prop_loc_search.asp

Returns

A String, representing the querystring part of a URL, including the question mark (?).

property urlSearchParams: URLSearchParams

The URLSearchParams() constructor creates and returns a new URLSearchParams object.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams/URLSearchParams>

class epyk.core.js.JsLocation.URLSearchParams(*query: str*)

add(*component: HtmlModel*)

append(*key: Union[str, JsDataModel], value: Any*)

Append a key, value to the url parameter object.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams>

Parameters

- **key** – The url parameter

- **value** – The value to be appended to the URL

delete(*key*)

The delete() method of the URLSearchParams interface deletes the given search parameter and all its associated values, from the list of all search parameters.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams/delete>

Parameters

key –

get(*key: str, default: Optional[Any] = None*)

Get the value of a request parameter in the url.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams>

Parameters

- **key** – The url parameter
- **default** – Optional. The default value

getAll(*key: Union[str, JsDataModel]*)

Get all the values of a request parameter in the url.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams>

Parameters

key – The url parameter

has(*key: Union[str, JsDataModel]*)

Check if a given parameter is in the url.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams>

Parameters

key – The url parameter

set(*key: str, value: Any*)

Set the value of a request parameter in the url.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams>

Parameters

- **key** – The url parameter
- **value** – The value to set

5.8.2.2.4 Shortcut and Features

This will provide common features like:

- math
- navigator
- performance
- web socket
- primitives

5.8.2.2.5 Technical Documentation

The below section will provide the technical documentation of the base class for JavaScript.

```
class epyk.core.js.Js.JsBase(page: Optional[PageModel] = None, component: Optional[HtmlModel] = None)
```

property accounting

Shortcut to accounting properties.

Usages:

```
page.js.accounting.add_to_imports()
```

Related Pages:

<http://openexchangerates.github.io/accounting.js/>

activeElement()

The activeElement property returns the currently focused element in the document.

Related Pages:

https://www.w3schools.com/jsref/prop_document_activeelement.asp

Returns

A reference to the element object in the document that has focus.

property body: JsDoms

Get the DOM object.

This will return the object. It will not create any variable.

property breadcrumb: JsBreadCrumb

Create an internal Breadcrumb to keep track of the user journey within your page.

Related Pages:

https://www.w3schools.com/howto/howto_css_breadcrumbs.asp

Returns

A Python breadcrumb object.

clipboard(*data: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)

Copy the full URL to the clipboard.

Related Pages:

<https://isabelcastillo.com/hidden-input-javascript>

Parameters

- **data** – The Javascript expression
- **js_conv_func** – Optional. A specific JavaScript data conversion function

createAttribute(*attribute_name*)

The createAttribute() method creates an attribute with the specified name, and returns the attribute as an Attr object.

Related Pages:

https://www.w3schools.com/jsref/met_document_createattribute.asp

Parameters

attribute_name – The name of the attribute you want to create.

Returns

A Node object, representing the created attribute.

createElement(*tag_name: str, js_code: Optional[str] = None, set_var: bool = True, dom_id: Optional[str] = None*)

The createElement() method creates an Element Node with the specified name.

Related Pages:

https://www.w3schools.com/jsref/met_document_createelement.asp

Parameters

- **tag_name** – The name of the element you want to create.
- **js_code** – The variable name to be set. Default random name.
- **set_var** – Optional. Create a variable for the new object. Default True.
- **dom_id** – Optional. The Dom ID reference for the object.

createEvent(*event_type: str*)

The createEvent() method creates an event object.

The event can be of any legal event type, and must be initialized before use.

Related Pages:

https://www.w3schools.com/jsref/event_createevent.asp

Parameters

event_type – A String that specifies the type of the event.

Returns

An Event object

```
static createTextNode(text: Optional[Union[JsDataModel, str]] = None, js_conv_func:
    Optional[Union[list, str]] = None) → JsObject
```

The createTextNode() method creates a Text Node with the specified text.

Related Pages:

https://www.w3schools.com/jsref/met_document_createtextnode.asp

Parameters

- **text** – Optional. The text of the Text node.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

A Text Node object with the created Text Node.

```
custom(data: Union[str, JsDataModel], key: Optional[str] = None, is_py_data: bool = False, js_func:
    Optional[Union[list, str]] = None)
```

Allow the definition of bespoke javascript strings.

Parameters

- **data** – A String corresponding to a JavaScript object
- **key** – Optional. A key reference in the JavaScript object
- **is_py_data** – Optional. Specify if the data is in Python and should be jsonify first
- **js_func** – Optional. Javascript functions

```
customFile(filename: str, path: Optional[str] = None, module_type: str = 'text/javascript', absolute_path:
    bool = False, requirements: Optional[list] = None, randomize: bool = False, authorize: bool =
    False)
```

This will load your local javascript file when the report will be built. Then you will be able to use the new features in the different Javascript wrappers.

Usage:

```
page.js.customFile("test.js", r"C:
```

older")

param filename

The filename

param path

Optional. The file path

param module_type

Optional. The module type

param absolute_path

Optional. If path is None this flag will map to the current main path

param requirements

Optional. The list of required packages

param randomize

Optional. Add random suffix to the module to avoid browser caching

param authorize

Optional. Add to the restricted list of packages

return

The Js Object to allow the chaining.

customText(*text: str*)

Javascript fragment added at the beginning of the page. This will be called before any function in the framework.

Parameters

text – The Javascript fragment

Returns

self to allow the chaining.

property d3

D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG, and CSS.

D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

Related Pages:

<https://d3js.org/>

property data: JsData

Get wrapped JavaScript data structures.

decodeURIComponent(*url_enc: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)
→ JsObject

The decodeURIComponent() function decodes a URI component.

Related Pages:

https://www.w3schools.com/jsref/jsref_decodeuricomponent.asp

Parameters

- **url_enc** – The URI to be decoded
- **js_conv_func** – Optional. A specific JavaScript data conversion function

Returns

A String, representing the decoded URI.

delay(*js_funcs: Union[list, str], seconds: int = 0, window_id: str = 'window', profile: Optional[Union[bool, dict]] = False*)

Add a wrapper on top of the setTimeout.

Usage:

```
page.js.delay([text.build("Change the value")], 5)
```

Parameters

- **js_funcs** – The function that will be executed
- **seconds** – Optional. The number of seconds to wait before executing the code

- **window_id** – Optional. The JavaScript window object
- **profile** – Optional. Set to true to get the profile for the function on the Javascript console

delete(url: Union[str, JsDataModel], data: Optional[dict] = None, js_code: str = 'response', is_json: bool = True, components: Optional[List[Union[Tuple[HtmlModel, str], HtmlModel]]] = None, profile: Optional[Union[bool, dict]] = None, headers: Optional[dict] = None, asynchronous: bool = False, stringify: bool = True) → XMLHttpRequest

Create a DELETE HTTP request.

Related Pages:

<https://pythonise.com/series/learning-flask/flask-http-methods>

Parameters

- **url** – The url path of the HTTP request
- **data** – Optional. Corresponding to a JavaScript object
- **js_code** – Optional. The variable name created in the Javascript (default response)
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. This will add the component value to the request object
- **profile** – Optional. A flag to set the component performance storage
- **headers** – Optional. The request headers
- **asynchronous** – Async flag: true (asynchronous) or false (synchronous)
- **stringify** – Optional. Stringify the request data for json exchange

property documentElement

Document.documentElement returns the Element that is the root element of the document (for example, the <html> element for HTML documents).

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Document/documentElement>

encodeURIComponent(uri: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None) → JsObject

The encodeURIComponent() function encodes a URI component.

Related Pages:

https://www.w3schools.com/jsref/jsref_encodeuricomponent.asp

Parameters

- **uri** – The URI to be encoded
- **js_conv_func** – Optional. A specific JavaScript data conversion function

Returns

A String, representing the encoded URI.

eval(*data: Union[JsDataModel, str]*, *js_conv_func: Optional[Union[list, str]] = None*)

The eval() function evaluates JavaScript code represented as a string.

Warning: Executing JavaScript from a string is an enormous security risk. It is far too easy for a bad actor to run arbitrary code when you use eval(). See Never use eval()!, below.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/eval

Parameters

- **data** – Data to be evaluated
- **js_conv_func** – Optional. A specific JavaScript data conversion function

execCommand(*command: str*, *show_ui: bool*, *value: str*)

The execCommand() method executes the specified command for the selected part of an editable section.

Related Pages:

https://www.w3schools.com/jsref/met_document_execcommand.asp

:param command:. Specifies the name of the command to execute on the selected section. :param show_ui: specifies if the UI should be shown or not. :param value: Some commands need a value to be completed.

Returns

A Boolean, false if the command is not supported, otherwise true.

extendProto(*py_class: Any*, *func_name: str*, *js_funcs: Union[str, list]*, *pmts: Optional[dict] = None*, *profile: Optional[Union[bool, dict]] = False*)

Javascript Framework extension.

Hook in the base class to allow the definition of specific function to add extra primitive features. Usual this function should be used in a wrapper function with the same name in order to have a coherent bridge between Python and Javascript.

Related Pages:

https://www.w3schools.com/js/js_object_prototypes.asp

Parameters

- **py_class** – PyJs class name
- **func_name** – The Javascript function name
- **js_funcs** – Javascript functions
- **pmts** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

Returns

The Js Object to allow the chaining.

fetch(*url: str*, *options: Optional[dict] = None*, *profile: Optional[Union[bool, dict]] = False*, *async_await: bool = False*) → JsPromise

The Fetch API provides a JavaScript interface for accessing and manipulating parts of the HTTP pipeline, such as requests and responses.

Usage:

```
page.ui.button("Click").click([
    page.js.fetch("test", {"method": "POST"}).then([
        page.js.console.log(pk.events.response)
    ])
])
```

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Parameters

- **url** – The target url
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **async_await** – Optional.

property fncs: JsRegisteredFunctions

Property to the predefined Javascript functions.

Returns

The predefined functions.

for_(*js_fncs: Optional[Union[list, str]] = None, step: int = 1, start: int = 0, end: int = 10, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*) → JsFor

Shortcut to a for loop.

Usage:

```
js_for = page.js.for_(end=30)
js_for.fncs([page.js.console.log(js_for.i)])
```

Related Pages:

https://www.w3schools.com/js/js_loop_for.asp

Parameters

- **js_fncs** – Javascript functions
- **step** – Optional. The value to increment. Default 1
- **start** – Optional. The first index in the for loop
- **end** – Optional. The last index in the for loop
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

get(*url: Union[str, JsDataModel], data: Optional[dict] = None, js_code: str = 'response', is_json: bool = True, components: Optional[Union[Tuple[HtmlModel, str], List[HtmlModel]]] = None, headers: Optional[dict] = None, asynchronous: bool = False, stringify: bool = True*) → XMLHttpRequest

Create a GET HTTP request.

Usage:

```
inputs = page.ui.input("")
btn = page.ui.button("Click").click([
    page.js.get("/test", {"fegeg": "efefe", "ok": inputs.dom.content},
components=({"input", inputs}))
])
```

Parameters

- **url** – The url path of the HTTP request
- **data** – Optional. A String corresponding to a JavaScript object
- **js_code** – Optional. The variable name created in the Javascript (default response)
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. This will add the component value to the request object
- **headers** – Optional. The request headers
- **asynchronous** – Async flag: true (asynchronous) or false (synchronous)

static getElementById(*id_name: Union[str, JsDataModel]*, *js_conv_func: Optional[Union[list, str]] = None*)

The getElementById() method returns the element that has the ID attribute with the specified value.

Related Pages:

https://www.w3schools.com/jsref/met_document_getelementbyid.asp

Parameters

- **id_name** – The ID attribute's value of the element you want to get.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

An Element Object, representing an element with the specified ID. Returns null if no elements with

the specified ID exists

static getElementsByClassName(*cls_name: str*) → JsDoms

The getElementsByClassName() method returns a collection of all elements in the document with the specified class name, as a NodeList object.

Related Pages:

https://www.w3schools.com/jsref/met_document_getelementsbyclassname.asp

Parameters

- **cls_name** – The class name of the elements you want to get.

Returns

A NodeList object, representing a collection of elements with the specified class name. The elements in the returned collection are sorted as they appear in the source code.

static **getElementsByName**(*name: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*) → JsDomsList

The `getElementsByName()` method returns a collection of all elements in the document with the specified name (the value of the name attribute), as a `NodeList` object.

The `NodeList` object represents a collection of nodes. The nodes can be accessed by index numbers. The index starts at 0.

Related Pages:

https://www.w3schools.com/jsref/met_doc_getelementsbyname.asp

Parameters

- **name** – The name attribute value of the element you want to access/manipulate.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

A `NodeList` object, representing a collection of elements with the specified name. The elements in the returned collection are sorted as they appear in the source code.

static **getElementsByTagName**(*tag_name: Union[str, JsDataModel], i: int = 0, js_conv_func: Optional[Union[list, str]] = None*) → JsDoms

The `getElementsByTagName()` method returns a collection of an elements's child elements with the specified tag name, as a `NodeList` object.

The `NodeList` object represents a collection of nodes. The nodes can be accessed by index numbers. The index starts at 0.

Related Pages:

https://www.w3schools.com/jsref/met_element_getelementsbytagname.asp

Parameters

- **tag_name** – The tag name of the child elements you want to get
- **i** – Optional. The index of the element
- **js_conv_func** – Optional. A specific JavaScript data conversion function

getVar(*js_code: Union[str, JsDataModel], var_type: str = 'var'*) → JsObject

Get the Javascript Variable name.

Parameters

- **js_code** – The Variable name
- **var_type** – Optional. The scope of the variable

Returns

Return the piece of script to be added to the Javascript.

if_(*condition: Union[str, list, bool], js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = False*)

Conditional statements are used to perform different actions based on different conditions.

Usage:

```
page.js.if_(icon.icon.dom.content == "fas fa-lock-open cssicon", [
    page.js.console.log(icon.icon.dom.content),
])
```

Related Pages:

https://www.w3schools.com/js/js_if_else.asp

Parameters

- **condition** – The Javascript condition. Can be a JsBoolean object
- **js_funcs** – Optional. The Javascript functions
- **profile** – Optional. A flag to set the component performance storage

import_css(*script: str*)

Add a CSS file on the fly from a JavaScript event.

Related Pages:

<https://stackoverflow.com/questions/19844545/replacing-css-file-on-the-fly-and-apply-the-new-style-to-the-page>

Parameters

script – A script name with a CSS extension.

import_js(*script: str, js_funcs: Union[str, list], profile: Optional[Union[bool, dict]] = None, self_contained: bool = False*)

Add a Javascript module and then run function once it is loaded.

Related Pages:

<https://cleverbeagle.com/blog/articles/tutorial-how-to-load-third-party-scripts-dynamically-in-javascript>

Parameters

- **script** – A script name. A Js extension.
- **js_funcs** – The Javascript functions.
- **profile** – Optional. A flag to set the component performance storage.
- **self_contained** – Optional. A flag to specify where the import will be done.

info(*data: Union[str, JsDataModel], css_style: Optional[dict] = None, icon: str = 'fas fa-spinner fa-spin', seconds: int = 10000*)

Display a message.

Related Pages:

<https://fontawesome.com/how-to-use/on-the-web/styling/animating-icons>

Parameters

- **data** – A String corresponding to a JavaScript object
- **css_style** – Optional. The CSS attributes to be added to the HTML component
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **seconds** – Optional. The number of second the info will be visible

intersectionObserver(*js_code*: str, *callback*: Optional[Union[List[Union[str, JsDataModel]], str]] = None, *options*: Optional[dict] = None, *observe_once*: bool = False, *profile*: Optional[Union[bool, dict]] = None) → IntersectionObserver

Parameters

- **js_code** – The PyJs functions.
- **callback** – JavaScript functions called by the intersectionObserver.
- **options** – intersectionObserver options.
- **observe_once** – A flag to remove the observable once callbacks run.
- **profile** – Option to perform profiling logs in the browser console.

property **jquery**

jQuery is a fast, small, and feature-rich JavaScript library.

It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Usage:

```
btn = page.ui.button("Click")
btn.js.jquery.on("click", [
    page.js.alert("It works"),
    btn.js.jquery.after('<div style="background-color:yellow"> New div </div>'),
])
```

Related Pages:

<https://jquery.com/>

property **keydown**: **KeyCode**

The onkeydown event occurs when the user is pressing a key (on the keyboard).

Related Pages:

https://www.w3schools.com/jsref/event_onkeydown.asp

property **keypress**: **KeyCode**

The onkeypress event occurs when the user presses a key (on the keyboard).

Related Pages:

https://www.w3schools.com/jsref/event_onkeypress.asp

property **keyup**: **KeyCode**

The onkeypress event occurs when the user presses a key (on the keyboard).

Related Pages:

https://www.w3schools.com/jsref/event_onkeypress.asp

property **location**: **JsLocation**

Property to the Javascript Location functions.

Usage:

```
page.ui.text("Test").click([
  page.js.location.open_new_tab(page.js.location.getUrlFromArrays([
    ["AAA", "BBB"], ["111", "222"]]), end_line="
```

“”))

Related Pages:

https://www.w3schools.com/jsref/obj_location.asp

mail(*mails*, *subject=None*, *body=None*, *cc=None*, *bcc=None*)

Create an email.

Related Pages:

<https://www.w3docs.com/snippets/html/how-to-create-mailto-links.html>

Parameters

- **mails** –
- **subject** –
- **body** –
- **bcc** –

property **mediaRecorder**: **MediaRecorder**

The MediaRecorder interface of the MediaStream Recording API provides functionality to easily record media. It is created using the MediaRecorder() constructor.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder>

property **moment**

Parse, validate, manipulate, and display dates and times in JavaScript.

Usage:

```
page.js.moment.new("2021-08-05", varName="momentTime"),
page.js.console.log(page.js.moment.var("momentTime").weekYear(1998)),
page.js.console.log(page.js.moment.var("momentTime").weekYear()),
page.js.console.log(page.js.moment.new("2021-08-05")),
```

Related Pages:

<https://momentjs.com/> <https://github.com/you-dont-need/You-Dont-Need-Momentjs>

property **msg**: **Msg**

Shortcut to predefined temporary messages displayed to the UI.

navigateTo(*url: Union[str, JsDataModel]*, *options: Optional[dict] = None*)

Navigators to another URL like NodeJs.

Usage:

```
icon.click([self.context.page.js.navigateTo(url)])
```

Related Pages:

<https://redfin.github.io/react-server/annotated-src/navigateTo.html>

Parameters

- **url** – The target url.
- **options** – Optional. The property of the location object.

property navigator: JsNavigator

The information from the navigator object can often be misleading, and should not be used to detect browser versions because:

- Different browsers can use the same name.
- The navigator data can be changed by the browser owner.
- Some browsers misidentify themselves to bypass site tests.
- Browsers cannot report new operating systems, released later than the browser.

not_(*data, js_conv_func: Optional[Union[list, str]] = None*) → JsFunction

Add the Symbol (!) for the boolean negation. This feature is also available directly to any JsBoolean objects.

Usage:

```
jsObj.not_(jsObj.objects.boolean.get("weekend"))
```

Related Pages:

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Op%C3%A9rateurs/Op%C3%A9rateurs_logiques

Parameters

- **data** – A String corresponding to a JavaScript object
- **js_conv_func** – Optional. A specific JavaScript data conversion function

Returns

The Javascript fragment string.

number(*data, js_code: Optional[str] = None, set_var: bool = False, is_py_data: bool = True*) → JsNumber

Shortcut to the Javascript Number primitives.

Parameters

- **data** – The String data.
- **js_code** – Optional. The specific name to be used for this JavaScript String.
- **set_var** – Optional. Set a variable. Default False.
- **is_py_data** – Optional. Specify the type of data.

object(*data, js_code: Optional[str] = None, set_var: bool = False, is_py_data: bool = True*) → JsObject

Shortcut to the Javascript Object primitives.

Parameters

- **data** – The String data.
- **js_code** – Optional. The specific name to be used for this JavaScript String.
- **set_var** – Optional. Set a variable. Default False.
- **is_py_data** – Optional. Specify the type of data.

property objects: JsObjects

Interface to the main Javascript Classes and Primitives.

onReady(*js_funcs*: Union[*str*, *list*], *profile*: Optional[Union[*bool*, *dict*]] = *False*)

The ready event occurs when the body DOM (document object model) has been loaded.

Related Pages:

https://www.w3schools.com/jquery/event_ready.asp

Parameters

- **js_funcs** – The Javascript functions to be added to this section
- **profile** – Optional. A flag to set the component performance storage

static parseDate(*value*: *str*) → JsNumber

The parse() method parses a date string and returns the number of milliseconds between the date string and midnight of January 1, 1970.

Related Pages:

https://www.w3schools.com/jsref/jsref_parse.asp

Parameters

value – A string representing a date.

Returns

Number. Representing the milliseconds between the specified date-time and midnight January 1, 1970.

static parseFloat(*value*: *str*) → JsNumber

The parseFloat() function parses a string and returns a floating point number.

Related Pages:

https://www.w3schools.com/jsref/jsref_parseint.asp

Parameters

value – The string to be parsed.

Returns

A Number. If the first character cannot be converted to a number, NaN is returned.

static parseInt(*value*: *str*)

The parseInt() function parses a string and returns an integer.

Related Pages:

https://www.w3schools.com/jsref/jsref_parseint.asp

Parameters

value – The string to be parsed.

Returns

A Number. If the first character cannot be converted to a number, NaN is returned.

patch(url: Union[str, JsDataModel], data: Optional[dict] = None, js_code: str = 'response', is_json: bool = True, components: Optional[List[Union[Tuple[HtmlModel, str], HtmlModel]]] = None, profile: Optional[Union[bool, dict]] = None, headers: Optional[dict] = None, asynchronous: bool = False, stringify: bool = True) → XMLHttpRequest

Create a PATH HTTP request.

Related Pages:

<https://pythonise.com/series/learning-flask/flask-http-methods>

Parameters

- **url** – The url path of the HTTP request
- **data** – Optional. Corresponding to a JavaScript object
- **js_code** – Optional. The variable name created in the Javascript (default response)
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. This will add the component value to the request object
- **profile** – Optional. A flag to set the component performance storage
- **headers** – Optional. The request headers
- **asynchronous** – Async flag: true (asynchronous) or false (synchronous)
- **stringify** – Optional. Stringify the request data for json exchange

post(url: Union[str, JsDataModel], data: Optional[dict] = None, js_code: str = 'response', is_json: bool = True, components: Optional[List[Union[Tuple[HtmlModel, str], HtmlModel]]] = None, profile: Optional[Union[bool, dict]] = None, headers: Optional[dict] = None, asynchronous: bool = False, stringify: bool = True) → XMLHttpRequest

Create a POST HTTP request.

Related Pages:

<https://pythonise.com/series/learning-flask/flask-http-methods>

Parameters

- **url** – The url path of the HTTP request
- **data** – Optional. Corresponding to a JavaScript object
- **js_code** – Optional. The variable name created in the Javascript (default response)
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. This will add the component value to the request object
- **profile** – Optional. A flag to set the component performance storage
- **headers** – Optional. The request headers
- **asynchronous** – Async flag: true (asynchronous) or false (synchronous)
- **stringify** – Optional. Stringify the request data for json exchange

print(content: Union[str, JsDataModel], timer: int = 1000, css_attrs: Optional[dict] = None)

Print a temporary message.

Parameters

- **content** – The content of the popup.
- **timer** – Optional. The time the popup will be displayed.
- **css_attrs** – Optional. The CSS attributes for the popup.

profile(*type: Union[str, JsDataModel], html_code: str, mark: Union[str, JsDataModel], records_count: Optional[int] = None*)

Parameters

- **type** – The type of profile tag.
- **html_code** – The HTML component ID.
- **mark** – The mark reference.
- **records_count** – Optional. The records count.

put(*url: Union[str, JsDataModel], data: Optional[dict] = None, js_code: str = 'response', is_json: bool = True, components: Optional[List[Union[Tuple[HtmlModel, str], HtmlModel]]] = None, profile: Optional[Union[bool, dict]] = None, headers: Optional[dict] = None, asynchronous: bool = False, stringify: bool = True*) → XMLHttpRequest

Create a PUT HTTP request.

Related Pages:

<https://pythonise.com/series/learning-flask/flask-http-methods>

Parameters

- **url** – The url path of the HTTP request
- **data** – Optional. Corresponding to a JavaScript object
- **js_code** – Optional. The variable name created in the Javascript (default response)
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. This will add the component value to the request object
- **profile** – Optional. A flag to set the component performance storage
- **headers** – Optional. The request headers
- **asynchronous** – Async flag: true (asynchronous) or false (synchronous)
- **stringify** – Optional. Stringify the request data for json exchange

querySelector(*selector: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)

The querySelector() method returns the first element that matches a specified CSS selector(s) in the document.

Related Pages:

https://www.w3schools.com/jsref/met_document_queryselector.asp

Parameters

- **selector** – CSS selectors.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

querySelectorAll (*selector: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)
→ JsDomsList

The querySelectorAll() method returns all elements in the document that matches a specified CSS selector(s), as a static NodeList object.

Related Pages:

https://www.w3schools.com/jsref/met_document_queryselectorall.asp

Parameters

- **selector** – CSS selectors.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

queueMicrotask (*js_funcs: Union[List[Union[str, JsDataModel]], str], profile: Optional[Union[bool, dict]] = None*)

The queueMicrotask() method, which is exposed on the Window or Worker interface, queues a microtask to be executed at a safe time prior to control returning to the browser's event loop.

Usage:

```
page.body.onReady([
    page.js.queueMicrotask([page.js.alert("ok")])
])
```

Related Pages:

<https://developer.mozilla.org/fr/docs/Web/API/queueMicrotask>

Parameters

- **js_funcs** – The Javascript function definition
- **profile** – Optional. A flag to set the component performance storage

registerFunction (*func_name: str, js_funcs: Union[str, list], args: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*)

Javascript Framework extension.

Register a predefined Javascript function. This is only dedicated to specific Javascript transformation functions.

Parameters

- **func_name** – The function name.
- **js_funcs** – The Javascript function definition.
- **args** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

request_http (*method_type: str, url: str, js_code: str = 'response', is_json: bool = True, components: Optional[List[HtmlModel]] = None*) → XMLHttpRequest

All modern browsers have a built-in XMLHttpRequest object to request data from a server.

Related Pages:

https://www.w3schools.com/xml/xml_http.asp

Usage:

```

page.js.request_http("ajax", "POST", "https://api.cdnjs.com/libraries").
  ↪setHeaders(header).onSuccess([
page.js.alert(rptObj.js.objects.request.get("ajax").responseText)]).
  ↪send(encodeURIComponent={"search": 'ractive'})

```

Parameters

- **method_type** – The method of the HTTP Request
- **url** – The url path of the HTTP request
- **js_code** – Optional. The variable name created in the Javascript
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. A list of HTML objects values to be passed in the request

Return type

JsObjects.XMLHttpRequest

request_rpc(*js_code: str, method_type: Union[str, JsDataModel], fnc: Callable, url: str, extra_params: Optional[Union[dict, JsDataModel]] = None*) → XMLHttpRequest

Internal RPC to trigger services.

Parameters

- **js_code** – The variable name created in the Javascript.
- **method_type** – The method type
- **fnc** – Python function.
- **url** – The service url
- **extra_params** – Optional.

Return type

JsObjects.XMLHttpRequest

rest(*method: str, url: Union[str, JsDataModel], data: Optional[dict] = None, js_code: str = 'response', is_json: bool = True, components: Optional[List[Union[Tuple[HtmlModel, str], HtmlModel]]] = None, profile: Optional[Union[bool, dict]] = None, headers: Optional[dict] = None, asynchronous: bool = False, stringify: bool = True*) → XMLHttpRequest

Create a POST HTTP request.

Parameters

- **method** – The REST method used
- **url** – The url path of the HTTP request
- **data** – Optional. Corresponding to a JavaScript object
- **js_code** – Optional. The variable name created in the Javascript (default response)
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. This will add the component value to the request object
- **profile** – Optional. A flag to set the component performance storage
- **headers** – Optional. The request headers
- **asynchronous** – Optional. Async flag: true (asynchronous) or false (synchronous)

- **stringify** – Optional. Stringify the request data for json exchange

return_(data: str) → JsFunction

Javascript return keyword.

Parameters

data – The Javascript expression.

property samples: Samples

JavaScript feature to provide sample of data for a test/demo.

Usage:

```
page.js.samples.months(count_=7)
page.js.samples.numbers(count_=7, min_=-100, max_=100)
```

property screen

The screen object contains information about the visitor's screen.

Related Pages:

https://www.w3schools.com/jsref/obj_screen.asp

serverSentEvent(html_code: Optional[str] = None) → ServerSentEvent

SSE is a native HTML5 feature that allows the server to keep the HTTP connection open and push data changes to the client. Server-sent Streaming is really ideal for server-push notifications, device monitoring and all other tasks that do not require real-time push back from the client.

Related Pages:

<https://medium.com/code-zen/python-generator-and-html-server-sent-events-3cdf14140e56>
https://www.w3schools.com/html/html5_serversentevents.asp https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events

Parameters

html_code – The EventSource id (variable name) on the JavaScript side

socketio(html_code: Optional[str] = None)

This object must be created on the Python side.

The various function will be the one generating the Javascript string. This is just a Python wrapper on top of the library.

Related Pages:

https://www.tutorialspoint.com/socket.io/socket.io_event_handling.htm

Parameters

html_code – Optional. The WebSocket id (variable name) on the JavaScript side

speechRecognition(js_code: str) → SpeechRecognition

The SpeechRecognition interface of the Web Speech API is the controller interface for the recognition service; this also handles the SpeechRecognitionEvent sent from the recognition service.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition>

Usage:

```

page = pk.Page()
rec = page.js.speechRecognition("reco")

test = page.ui.button("Start recording")
test.click([rec.start()])

page.ui.input(html_code="test")

rec.speechend([rec.stop()])
rec.onresult([pk.js_callback("UpdateComponent(transcript, confidence)", page.
↪js.console.log("Done"))])
page.body.onReady([page.js.import_js("test_fnc.js", [], self_contained=True), ↪
↪rec])

# in the module test_fnc.js
function ProcessData(transcript, confidence){
    console.log(transcript); return (transcript == 'hello')}

function UpdateComponent(transcript, confidence){
    var expr = transcript.split(" ");
    if (expr[0] === "put"){document.getElementById(expr[3]).value = expr[1]}}

```

Parameters

js_code – The variable name for the speech recognition object

string(*data*, *js_code*: *Optional[str] = None*, *set_var*: *bool = False*, *is_py_data*: *bool = True*) → JsString

Shortcut to the Javascript String primitives.

Parameters

- **data** – The String data.
- **js_code** – Optional. The specific name to be used for this JavaScript String.
- **set_var** – Optional. Set a variable. Default False.
- **is_py_data** – Optional. Specify the type of data.

switch(*variable*: *Union[str, JsDataModel, HtmlModel]*, *js_conv_func*: *Optional[Union[list, str]] = None*) → JsSwitch

switch statement is used to perform different actions based on different conditions.

Related Pages:

https://www.w3schools.com/js/js_switch.asp

Parameters

- **variable** – Variable on which we will apply the switch
- **js_conv_func** – Optional. A specific JavaScript data conversion function

static title(*text*: *Optional[Union[JsDataModel, str]] = None*, *js_conv_func*: *Optional[Union[list, str]] = None*)

The title property sets or returns the title of the current document (the text inside the HTML title element).

Related Pages:

https://www.w3schools.com/jsref/prop_doc_title.asp

Parameters

- **text** – Optional. Representing the title of the document.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

static typeof(*data: str, var_type: Optional[str] = None*)

The typeof function.

Related Pages:

https://www.w3schools.com/js/js_datatypes.asp

Parameters

- **data** – A String corresponding to a JavaScript object
- **var_type** – Optional. The type of object

property viewHeight

Return the current View port height visible in the browser.

websocket(*html_code: Optional[str] = None, secured: bool = False*)

WebSocket client applications use the WebSocket API to communicate with WebSocket servers using the WebSocket protocol.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications <https://javascript.info/websocket>

Parameters

- **html_code** – Optional. The WebSocket id (variable name) on the JavaScript side
- **secured** – Optional. To define the right protocol for the WebSocket connection we or wss

while_(*condition: Union[str, list], js_funcs: Union[list, str], options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*) → JsWhile

The while loop loops through a block of code as long as a specified condition is true.

Related Pages:

https://www.w3schools.com/js/js_loop_while.asp

Parameters

- **condition** – The JavaScript condition
- **js_funcs** – Javascript functions
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

worker(*html_code: Optional[str] = None, server: bool = False*)

A web worker is a JavaScript running in the background, without affecting the performance of the page.

Related Pages:

https://www.w3schools.com/html/html5_webworkers.asp

Parameters

- **html_code** – Optional. The WebSocket id (variable name) on the JavaScript side
- **server** – Optional. Specify if the page is running on a server

writeln(*value: str*)

The writeln() method is identical to the document.write() method, with the addition of writing a newline character after each statement.

Related Pages:

https://www.w3schools.com/jsref/met_doc_writeln.asp

Parameters

value – What to write to the output stream. Multiple arguments can be listed and they will be appended to the document in order of occurrence

Returns

No return value

5.8.2.3 Outputs Interface

• Browsers:

- CodePen
- Stackblitz
- JsFiddle
- w3CTryIt

• Files:

- Jupyter
- JupyterLab
- HTML page
- HTML, JavaScript and CSS files

• Web:

class `epyk.core.py.PyOuts.OutBrowsers`(*context*)

codepen(*path: Optional[str] = None, target: str = '_blank', open_browser: bool = True*)

Update the Html launcher and send the data to Codepen. URL used: <https://codepen.io/pen/define/>

Usage:

```
page = Report()
page.ui.text("This is a text")
page.outs.browser.codepen()
```

Related Pages:

<https://www.debuggex.com/cheatsheet/regex/python>

Parameters

- **path** – Optional. Output path in which the static files will be generated.
- **target** – Optional. Load the data in a new tab in the browser.
- **open_browser** – Optional. Flag to open the browser automatically.

Returns

The output launcher full file name.

stackblitz(*path: Optional[str] = None, target: str = '_blank', open_browser: bool = True*)

Create an output to be compatible with stackblitz.

Usage:

```
page = Report()
page.ui.text("This is a text")
page.outs.codepen()
```

Related Pages:

<https://stackblitz.com/docs>

Parameters

- **path** – Optional. Output path in which the static files will be generated.
- **target** – Optional. Not used. Load the data in a new tab in the browser.
- **open_browser** – Optional. Flag to open the browser automatically.

class epyk.core.py.PyOuts.**PyOuts**(*page: Optional[PageModel] = None, options: Optional[dict] = None*)

property browser

This module will require the package web browser. It will allow outputs to be created directly in the web pages (without using intermediary text files).

codepen(*path: Optional[str] = None, name: Optional[str] = None*)

Produce files which will be compatible with codepen.

Usage:

```
page = Report()
page.ui.text("This is a text")
page.outs.codepen()
```

Related Pages:

<https://codepen.io/>

Parameters

- **path** – Optional. The path in which the output files will be created.
- **name** – Optional. The filename without the extension.

TODO Try to add the prefill <https://blog.codepen.io/documentation/api/prefill/>

Returns

The file path

html()

Function to get the result HTML page fragments from all the HTML components.

Usage:

```
page = Report()
page.ui.text("This is a text")
page.outs.html()
```

html_file(*path: Optional[str] = None, name: Optional[str] = None, options: Optional[dict] = None, print_paths: bool = False*)

Function used to generate a static HTML page for the report.

Usage:

```
page = Report()
page.ui.text("This is a text")
page.outs.html_file()

# To generate multiple files using local packages
page.imports.static_url = "C:\\epyks\\statics"
page.outs.html_file(name="test.html", options={"split": True, "minify": True,
↪ "static_path": page.imports.static_url})
```

Parameters

- **path** – Optional. The path in which the output files will be created
- **name** – Optional. The filename without the extension
- **print_paths** – Optional. Print the page for the created file
- **options** – Optional.

Returns

The file full path.

jsfiddle(*path: Optional[str] = None, name: Optional[str] = None, framework: str = 'jsfiddle'*)

Produce files which can be copied directly to <https://jsfiddle.net> in order to test the results and perform changes.

The output is always in a sub-directory jsfiddle.

Usage:

```
page = Report()
page.ui.text("This is a text")
page.outs.codepen()
```

Related Pages:

<https://jsfiddle.net/>

Parameters

- **path** – Optional. The path in which the output files will be created.
- **name** – Optional. The filename without the extension.
- **framework** – optional. The framework in which the result page will be used.

Returns

The file path

jupyter(*verbose: bool = False, requireJs: Optional[dict] = None, closure: bool = True, requirejs_path: Optional[dict] = None, requirejs_func: Optional[dict] = None*)

For a display of the report in Jupyter. Thanks to this function some packages will not be imported to not conflict with the existing ones.

Usage:

```
page = Report()
page.ui.text("This is a text")
page.outs.jupyter()
```

Related Pages:

<https://jupyter.org/>

Parameters

- **verbose** – Optional. Get the excluded packages.
- **requireJs** – Optional. The requirements overrides from the apps property.
- **closure** – Optional.
- **requirejs_path** – Optional.
- **requirejs_func** – Optional.

Returns

The output object with the function `_repr_html_`

jupyterlab()

For a display of the report in JupyterLab. Thanks to this function some packages will not be imported to not conflict with the existing ones.

Usage:

```
page = Report()
page.ui.text("This is a text")
page.outs.jupyterlab()
```

Related Pages:

<https://jupyter.org/>

markdown_file(*path: Optional[str] = None, name: Optional[str] = None*)

Writes a Markdown file from the report object.

Parameters

- **path** – The path in which the output files will be created.
- **name** – The filename without the extension.

Returns

The file path

publish(*server: str, app_path: str, selector: str, name: Optional[str] = None, module: Optional[str] = None, target_folder: str = 'apps', auto_route: bool = False*)

Publish the HTML page to a distant web server.

Usage:

```
page = Report()
page.ui.text("This is a text")
```

Parameters

- **server** – The webserver type (angular, react, vue, node, deno).
- **app_path** – The webserver path.
- **selector** –
- **name** – Optional. The application name in the webserver.
- **module** –
- **target_folder** –
- **auto_route** –

w3cTryIt(*path: Optional[str] = None, name: Optional[str] = None*)

This will produce everything in a single page which can be directly copied to the try editor in w3C website.

Usage:

```
page = Report()
page.ui.text("This is a text")
page.outs.w3cTryIt()
```

Related Pages:

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_basic

Parameters

- **path** – Optional. The path in which the output files will be created.
- **name** – Optional. The filename without the extension.

web()

Return the complete page structure to allow the various web framework to split the code accordingly. Fragments will then be used by the various framework to create the corresponding pages.

5.8.2.4 Python Interface

5.8.2.4.1 Dates

class `epyk.core.py.PyDates.PyDates`(*src: Optional[PageModel] = None*)

Common module for managing dates.

This module is a light wrapper on top of datetime in order to perform basic operations on dates. This will also standardise the date format to YYYY-MM-DD in the Python layer to simplify the conversion to the Javascript

All the tests in this module are using doctest.

property cob

Returns the last close of business date.

In this property the parameter weekdays is forced to True.

Usage:

```
page.py.dates.cob
```

Returns

A string date

date_from_alias(*alias: str, from_date: Optional[str] = None*)

Return the date corresponding to an alias code like T, T-N, M...

Usage:

```
>>> PyDates().date_from_alias("T", "2019-08-08")
'2019-08-07'
```

Parameters

- **alias** – The alias of the operation (T-3, M-2...)
- **from_date** – Optional. The start date from which the time operation is applied. Today by default

Returns

The converted date or a list of dates.

static date_from_excel(*xl_date: int*)

Convert an Excel date to a standard date format YYYY-MM-DD.

Usage:

```
>>> PyDates().date_from_excel(39448)
'2008-01-01'
```

Related Pages:

<https://support.office.com/en-gb/article/date-function-e36c0c8c-4104-49da-ab83-82328b832349?ui=en-US&rs=en-GB&ad=GB>

Parameters

xl_date – A date in the Excel format.

Returns

The date as a String in the common format YYYY-MM-DD

static delta(*from_dt: str, to_dt: str, format_dt: str = '%Y-%m-%d'*)**Parameters**

- **from_dt** –
- **to_dt** –
- **format_dt** –

static elapsed(*delta_time*, *with_time*: *bool* = *False*)

Get the time between two dates. This function will only format the result of a delta time object.

TODO: Fix this method

Parameters

- **delta_time** – *delta_time*. The delta time between two python dates.
- **with_time** (*bool*) – Optional. A flag to mention if the time should be computed.

static from_timestamp(*timestamp*: *int*, *offset*: *int* = 0, *reference*: *int* = 60, *factor*: *int* = 1000)

The default value will be given considering the GMT time.

Usage:

```
timestamp_s = page.py.dates.from_timestamp(1573074335010, 0)
```

Parameters

- **timestamp** – The timestamp in milliseconds.
- **offset** – Optional. The time zone.
- **reference** – Optional. The reference shift in minutes.
- **factor** –

Returns

The server timestamp string

property month_end

Returns the last month end date.

In this property the parameter weekdays is forced to True.

Usage:

```
page.py.dates.month_end
```

Returns

A string date

month_ends(*from_dt*: *str*, *to_dt*: *str*, *weekdays*: *bool* = *True*)

Return the list of end of month dates between two dates.

Usage:

```
>>> PyDates().month_ends("2019-01-01", "2019-06-05", False)
['2019-01-31', '2019-02-28', '2019-03-31', '2019-04-30', '2019-05-31']
```

Parameters

- **from_dt** – The start date in format YYYY-MM-DD.
- **to_dt** – The end date in format YYYY-MM-DD.
- **weekdays** – Optional. remove the weekends from the potential dates (take the day before). Default True.

Returns

A list of dates.

property months

Returns the list of month end dates from the beginning of the year.

In this property the parameter weekdays is forced to True.

Usage:

```
page.py.dates.months
```

Returns

A list of String dates

property now

dd.

Usage:

```
PyDates().now
```

Related Pages:

<https://docs.python.org/2/library/datetime.html>

Returns

Return a string timestamp

Type

Return the current timestamp in a format YYYY-MM-DD HH

Type

mm

static path(*with_time: bool = False*)

Return a predefined format for date in a file path. Using this method will ensure a consistency in the naming convention of the various files in the project.

Parameters

with_time (*bool*) – Optional. Specify if the time should be added to the path.

property quarters

Return the list of quarter dates since the beginning of the year.

In this property the parameter weekdays is forced to True.

Usage:

```
page.py.dates.quarters
```

Returns

A list of String dates

range_dates(*to_dt: str, from_dt: Optional[str] = None, weekdays: bool = True*)

Get the list of dates between two dates.

The date should be two string dates in the format YYYY-MM-DD. The resulting range of date will always be increasing

Usage:

```
>>> PyDates().range_dates("2019-01-01", "2019-01-11")
['2019-01-11', '2019-01-10', '2019-01-09', '2019-01-08', '2019-01-07', '2019-01-04', '2019-01-03', '2019-01-02', '2019-01-01']
```

Parameters

- **from_dt** – The start date in format YYYY-MM-DD.
- **to_dt** – Optional. The end date in format YYYY-MM-DD.
- **weekdays** – Optional. Remove the weekends from the potential dates (take the day before). Default True

Returns

A list of dates.

static to_server_time(*timestamp: str, offset: int = 0, reference: int = 60*)

Return the converted timestamp to be stored in the database. This conversion will be based on the offset coming from the UI to convert to common time.

Usage:

```
>>> PyDates().to_server_time("2019-08-20 20:04:10", 2)
'2019-08-20 21:06:10'
```

Parameters

- **timestamp** – The client timestamp.
- **offset** – Optional. The client offset time to be applied before storage in hour.
- **reference** – Optional. The reference time used on the server side.

Returns

The server timestamp string

static to_user_time(*timestamp: str, offset: int, reference: int = 60*)

Return the converted timestamp to be returned to the user. This is converting a stored timestamp to a user one.

Usage:

```
>>> PyDates().to_user_time('2019-08-20 21:06:10', 2)
'2019-08-20 20:04:10'
```

Parameters

- **timestamp** – The server timestamp.
- **offset** – The client offset time to be applied before storage.
- **reference** – Optional. The reference time used on the server side (default 20).

Returns

The client timestamp string

property today

Return a String date in a format YYYY-MM-DD.

Even if within the property python date object are used, this function will always return a string date in a specific format to guarantee and simplify the compatibility between languages within the components.

Usage:

`PyDates().today`

Related Pages:

<https://docs.python.org/2/library/datetime.html>

Returns

A string date in the format YYYY-MM-DD

5.8.2.4.2 Mails

```
class epyk.core.py.PyMails.Email(sender, recipients, subject, content, attachments=None, headers=None,
                                  mimetype='text/plain', charset='utf8')
```

to_mime()

Return a MIME representation of this object - used internally

5.8.2.4.3 Markdown

```
class epyk.core.py.PyMarkdown.MarkDown(page: PageModel)
```

all(data: str)

TODO: Improve this function.

Parameters

data – The data to be parsed.

```
resolve(data: str, css_attrs: Optional[dict] = None)
```

Convert a string to a markdown file.

Parameters

- **data** – Data to be converted.
- **css_attrs** – Optional. The CSS Style to be applied to the component.

```
classmethod translate(data: str)
```

Parameters

data –

5.8.2.4.4 NPM

class epyk.core.py.PyNpm.**Packages**

classmethod **descriptions**(*verbose: bool = True*)

Get all the packages and the short description from NPM.

Parameters

verbose – Optional. Display version details (default True).

classmethod **repositories**(*verbose: bool = True*)

Get the repositories used to retrieve the external packages.

This is a collaborative framework so do not hesitate to contact the author of those packages with ideas or even things to fix. It is important to encourage this open source community and to be part of modules improvements.

Usage:

```
repos = PyNpm.Packages.repositories()
```

Parameters

verbose – Optional. Display version details (default True).

classmethod **versions**(*verbose: bool = True*)

Get the current latest version of all the package in NPM. This could help on maintaining the internal framework up to date with the improvements.

It is important to align with the new version in order to benefit from the community hard work !

Parameters

verbose – Optional. Display version details (default True).

5.8.2.4.5 ReST

class epyk.core.py.PyRest.**PyRest**(*page: Optional[PageModel] = None*)

static **csv**(*url: str, delimiter: str = ',', encoding: str = 'utf-8', with_header: bool = True, store_location: Optional[str] = None, quotechar: str = '"'*)

Retrieve tabular data from an external REST service.

Parameters

- **url** – The url with the data to request.
- **delimiter** – Optional. The line delimiter.
- **encoding** – Optional. The encoding format.
- **with_header** – Optional. A flag to mention if the header is available. (it will be used for the keys)
- **store_location** – Optional. The temp folder to cache the data locally. False will cancel the temps data retrieval
- **quotechar** – Optional.

get(url: str, data=None, encoding: str = 'utf-8', headers: Optional[dict] = None, proxy: Optional[dict] = None)

Run an external REST call using the GET method.

This should be used to retrieve data from external services. If data should be extracted using an existing internal service the method query is better as it will embedded the security aspects

Usage:

```
page.py.requests.get("https://api.cdnjs.com/libraries")
pyrest.get(r"https://jsonplaceholder.typicode.com/todos/1")
```

Related Pages:

<https://docs.python.org/2/howto/urllib2.html>

Parameters

- **url** – Should be a string containing a valid URL
- **data** – Optional. Must be an object specifying additional data to send to the server, or None if no such data is needed
- **encoding** – Optional. The encoding of this request (defaults to 'utf-8'). This encoding will be used to percent-encode the URL and to convert the body to str (if given as unicode)
- **headers** – Optional. Should be a dictionary, and will be treated as if add_header() was called with each key and value as arguments
- **proxy** – Optional.

Returns

Return a Python object by default

http_server(port: int = 5000, service_name: str = "")

Start a local server for all the services. This should be at the end of the script in order to allow the services debug.

Parameters

- **port** –
- **service_name** –

json(url: str, encoding: str = 'utf-8', store_location: Optional[str] = None)

Retrieve Json data from an external REST service.

Parameters

- **url** – The url with the data to request.
- **encoding** – Optional. The encoding format.
- **store_location** – Optional. String. The temp folder to cache the data locally.

post(url: str, data=None, encoding: str = 'utf-8', headers: Optional[dict] = None, proxy: Optional[dict] = None)

Run a external REST call using the POST method.

This should be used to retrieve data from external services. If data should be extracted using an existing internal service the method query is better as it will embedded the security aspects

Usage:

```
page.py.requests.post("https://jsonplaceholder.typicode.com/todos/1")
```

Related Pages:

<https://docs.python.org/2/howto/urllib2.html>

Parameters

- **url** – The external URL to the REST service
- **data** – Optional. Must be an object specifying additional data to send to the server, or None if no such data is needed
- **encoding** – Optional. The encoding of this request (defaults to 'utf-8'). This encoding will be used to percent-encode the URL and to convert the body to str (if given as unicode)
- **headers** – Optional. Should be a dictionary, and will be treated as if add_header() was called with each key and value as arguments
- **proxy** – Optional.

Returns

The content of the REST call as a String

proxy(*username: str, password: str, proxy_host: str, proxy_port: int, protocols: Optional[list] = None*)

Set the proxy connexions for the Python requests.

Parameters

- **username** – The username.
- **password** – The user password.
- **proxy_host** – The proxy server hostname.
- **proxy_port** – The proxy server port.
- **protocols** – Protocols for the proxy. Default [(`'http'`, `'http://'`), (`'https'`, `'https://'`)]

static request(*url: str, data=None, method: Optional[str] = None, encoding: str = 'utf-8', headers: Optional[dict] = None, unverifiable: bool = False, proxy: Optional[dict] = None*)

Run a external REST call using a specific method (PUT, DELETE, OPTIONS, HEAD, PUT, PATCH...).

This should be used to retrieve data from external services. If data should be extracted using an existing internal service the method query is better as it will embedded the security aspects

Usage:

```
json.loads(PyRest().request(r"https://jsonplaceholder.typicode.com/todos/1",
↪method="GET"))
```

Related Pages:

<https://2.python-requests.org/en/master/api/> <https://docs.python.org/3/library/urllib.request.html>

Parameters

- **url** – Should be a string containing a valid URL
- **method** – Optional. Must be an object specifying additional data to send to the server, or None if no such data is needed

- **data** – Optional. Must be an object specifying additional data to send to the server, or None if no such data is needed
- **encoding** – String. Optional. the encoding of this request (defaults to 'utf-8'). This encoding will be used to percent-encode the URL and to convert the body to str (if given as unicode)
- **headers** – Optional. Should be a dictionary, and will be treated as if `add_header()` was called with each key and value as arguments
- **unverifiable** – Optional. Should indicate whether the request is unverifiable, as defined by RFC 2965
- **proxy** – Optional.

Returns

The content of the REST call as a String

```
static webscrapping(url: str, data=None, encoding: str = 'utf-8', headers: Optional[dict] = None,  
                    unverifiable: bool = False, proxy: Optional[dict] = None)
```

Create a REST request with the appropriate header to mimic a browser GET request.

Usage:

```
PyRest().webscrapping(r"https://doc.scrapy.org/en/latest/topics/request-  
↪response.html")
```

Parameters

- **url** – Should be a string containing a valid URL
- **data** – Optional. Must be an object specifying additional data to send to the server, or None if no such data is needed
- **encoding** – Optional. the encoding of this request (defaults to 'utf-8'). This encoding will be used to percent-encode the URL and to convert the body to str (if given as unicode)
- **headers** – Optional. Should be a dictionary, and will be treated as if `add_header()` was called with each key and value as arguments
- **unverifiable** – Optional. Should indicate whether the request is unverifiable, as defined by RFC 2965
- **proxy** – Optional.

Returns

The HTML content of the REST call as a String

5.8.2.4.6 Database

```
class epyk.core.py.PySql.SqlConn(family, database=None, filename=None, model_path=None, reset=False,  
                                migrate=True, tables_scope=None, **kwargs)
```

```
column(table_name, column_name)
```

Return a sqlalchemy column object. This can be useful in the where clauses

Usage:

```
select('table').where([db.column("table", 'column') == 'X'])
```

Parameters

- **table_name** – The database table name
- **column_name** – The column name

Returns

Python column object

columns(*table_name*)

Return the list of columns defined in the selected database.

Usage:

```
page.db().columns("table_name")
```

Parameters

table_name –

Returns

A python object with the list of tables

commit()

Commit the current transaction.

This will save the results in the database

Returns

The SQL object

property count

Return the number of records.

Usage:

```
print(db.select("table").count)
```

Returns

The number of records.

data(*limit=None*)

Returns the results of the select statement previously instantiated in a pandas dataframe.

Usage:

```
rptObj.db().getData()
```

Parameters

limit – Optional. The number of records to be returned.

Returns

A pandas dataframe

delete(*table_name*)

Create a SQL delete SQL statement.

Usage:

```
page.db().delete('table1')
```

Related Pages:

<https://docs.sqlalchemy.org/en/13/core/dml.html>

Parameters

table_name – Optional. The database table name.

Returns

self

distinct(*columns=None*)**Parameters**

columns – the list of columns.

Returns

self for the chaining.

drop(*table_name, validate=True*)

Delete the table from the database. The pre check can be disabled and the table will be automatically created again when the report will be triggered again. No extra function to create a table in the framework this is done by the SQL framework itself

Usage:

```
page.db().drop('test')
page.db().drop('test', withCheck=False)
```

Related Pages:

<https://docs.sqlalchemy.org/en/13/core/connections.html#sqlalchemy.engine.ResultProxy>

Parameters

- **table_name** – The database table name
- **validate** – Boolean, Validation check before dropping the table

Returns

Python column object

execute()

Execute the current SQL query.

Related Pages:

<https://docs.sqlalchemy.org/en/13/core/connections.html#sqlalchemy.engine.ResultProxy>

Returns

The SQL Result proxy

first(*items=False*)

Return only the first items from the SQL query.

Usage:

```
print(db.first())
print(db.first(items=True))
```

Parameters

items – Return a dictionary or a list of data.

Returns

None or the first record.

force_create()

Force the creation of the database in the given project.

Returns

The python Sql object

get_last_id(*table_name*)

Return the table last primary key ID. This will return an error if the table does not have a primary key defined in its schema.

Usage:

```
db.get_last_id("table")
```

Parameters

table_name – The table name.

Returns

Return the last row ID or -1

insert(*table_name, records, commit=False, col_user_name=None, clean_rec=False, getIdCol=False*)

insert a list of records to a table.

Usage:

```
db.insert('table1', [{'name': 'test'}], commit=True)
db.insert("table", {"user_name": "Test", "data": "test"}, commit=True)
```

Related Pages:

<https://docs.sqlalchemy.org/en/13/core/dml.html> <https://docs.sqlalchemy.org/en/13/core/dml.html>

Parameters

- **table_name** – The database table name.
- **records** – The list of dictionaries with the data to inserts.
- **commit** – Optional. Boolean to commit the insert. Set to False by default.
- **col_user_name** –
- **clean_rec** – Optional. Remove the key in the dictionaries which are not related to the table. Set to False.

- **getIdCol** –

Returns

The python object itself

limit(*n*)

Limit the number of records returned.

Parameters

n – Integer, the number of records.

Returns

The SQL Query object.

load_data_file(*filename*, *path*, *reset=False*, *new_tables=None*)

Load a python sql file to the local database. This will only add records and then commit the changes.

Those data should not be sensitive ones if they are store and committed to the folder.

Parameters

- **filename** –
- **path** –
- **reset** –
- **new_tables** –

Returns

The Python SQL object.

load_schema(*filename=None*, *model_path=None*, *reset=False*)

Function that takes care of initialising the DB. Please note that some column names are prohibited such as `lst_mod_dt`.

Parameters

- **filename** – Optional. The python module used to get the database schema.
- **model_path** – Optional. The python path with the model.
- **reset** – Optional. Flag to reset the database. This will empty the tables.

property records

Return the records.

Usage:

```
for rec in db.select("table").records:
    print(rec)
```

Returns

A iterator for the SQL result

select(*table_name=None*, *columns=None*)

Create a SQL statement.

Usage:

```
page.db().select(["worldcup_teams"])
```

Documentation

<http://docs.sqlalchemy.org/en/latest/core/selectable.html> <http://docs.sqlalchemy.org/en/latest/core/sqlelement.html>

Parameters

- **table_name** – String. Optional. The database table name.
- **columns** – String. Optional. The list of columns.

Returns

self

table(*table_name*)

Return a sqlalchemy table object. This can be useful in the where clauses.

Usage:

```
db.table('table1')
```

Parameters

table_name – The table name.

Returns

Python table object

table_clone(*old_table*, *new_table*, *mapping=None*)

Helps to migrate between two tables. The mapping argument is used in case the column names differ between the two tables.

Parameters

- **old_table** – A SQLAlchemy table class in the current database model
- **new_table** – A SQLAlchemy table class
- **mapping** – Optional. A dictionary for the column names.

Returns

self for the chaining.

table_create(*table_name*, *table_def*, *reset=False*)

Create a table in the database.

Usage:

```
db = rptObj.db(database=r"newTest.db")
tableDef = [sqlalchemy.Column('environment', sqlalchemy.String, nullable=False),
            sqlalchemy.Column('report', sqlalchemy.String, nullable=False),]
db.createTable('newTable', tableDef)
```

Parameters

- **table_name** – The table name.
- **table_def** –
- **reset** –

table_create_from_file(*filename*, *table_name*, *records=None*, *path=None*, *reset=False*, *commit=True*)

Usage:

```
df = page.py.file(htmlCode=r"IBRD_Balance_Sheet__FY2010.csv").read()
db = page.py.db(database=r"newTest.db").forceCreate()
dbObj.createTable('myschema.py', 'mytable', records=df)
```

Parameters

- **filename** –
- **table_name** –
- **records** –
- **path** –
- **reset** –
- **commit** –

Returns

The Python SQL object.

table_empty(*table_name*)

This function will empty an existing table.

Usage:

```
db.emptyTable('test')
```

Parameters

table_name – A string with the databale name.

Returns

self

table_migrate(*from_table*, *to_table*)

Copy data from one table to another.

Parameters

- **from_table** – The table name.
- **to_table** – The destination table name.

Returns

The Python SQL object

property tables

Return the list of tables defined in the selected database

Usage:

```
page.db().tables()
```

Returns

A python object with the list of tables

update(*table_name*, *values*)

Create a delete SQL statement.

Usage:

```
page.db().update('table1', {db.column('test', 'name'): 'user'})
```

Documentation

<http://docs.sqlalchemy.org/en/latest/core/selectable.html> <http://docs.sqlalchemy.org/en/latest/core/sqlelement.html> <https://docs.sqlalchemy.org/en/13/core/dml.html>

Parameters

- **table_name** – The table name.
- **values** –

Returns

self for the chaining.

where(*stmts*)

Add a where clause to the SQLAlchemy query.

Usage:

```
db.select().where([db.column("table", 'column') == 'X'])
```

Parameters

stmts – The SQL where statement.

Returns

The python object itself

class epyk.core.py.PySql.SqlConnOdbc(*database=None*, ***kwargs*)

Connector to Access databases. This connector will allow you to create, store and retrieve data from any MS Access Database. This will return the SQL database object. It will be possible to reuse the same syntax to then interact with it.

This would need the ODBC driver available here: <https://www.microsoft.com/en-us/download/confirmation.aspx?id=13255>

property tables

return:

class epyk.core.py.PySql.SqlConnNeo4j(*host=None*, *port=None*, *usr=None*, *pwd=None*)**alias**(*aliases*)

Defines a set of aliases that will appear as WITH a, b, c, d as count(id) The aliases argument will be defined as follows: ['a', 'b', 'c', {'d': 'count(id)'}]

clear()

Clears all nodes and edges from the Database

compose(*query*)

Simply joins the query clauses all together

foreach(*conditions*)

link(*labels=""*, *attr=None*, *direction='from'*)

Adds the edge definition to the query

node(*name=""*, *labels=None*, *attr=None*)

Adds the node pattern to the query

5.8.2.4.7 Cryptography

class `epyk.core.py.PyCrypto.PyCrypto`(*src: Optional[PageModel] = None*)

classmethod `b64encode`(*text: str*, *salt: Optional[str] = None*)

Usage:

```
PyCrypto.b64encode("Test")
```

Related Pages:

<https://cryptography.io/en/latest/hazmat/primitives/key-derivation-functions/>

Parameters

- **text** (*str*) – The text to be encrypted.
- **salt** (*Optional[str]*) – Optional. The salt used for the encryption (default None).

classmethod `cryptKeyPairs`(*msg*, *key1*, *key2*)

Usage:

```
:param msg:  
:param key1:  
:param key2:
```

decrypt(*encrypted: str*, *token: Optional[str] = None*, *salt: Optional[str] = None*, *label: str = ""*)

This function will use the two keys in order to decrypt the data. In case of failure this function will raise an exception.

Usage:

```
PyCrypto().decrypt(encrypted)
```

Parameters

- **encrypted** (*str*) – The encrypted data.
- **token** (*Optional[str]*) – Optional. The token used to encrypt the data.
- **salt** (*Optional[str]*) – Optional. The salt id.
- **label** (*str*) – Optional. A label used to store the reference in the log file.

Returns

A string with the decrypted data.

classmethod `decryptKeyPairs`(*encrypted*, *key1*, *key2*)

Usage:

```
:param encrypted: The encrypted data
:param key1:
:param key2:
```

encrypt (*data: str, token: Optional[str] = None, salt: Optional[str] = None*)

This function will use the cryptography to ensure a secured encryption of the different credential and private data.

This can be also used to protect data from the report. In order to ensure the right privacy please do not store the token and the salt in the framework.

Parameters

- **data** (*str*) – The data to be encrypted.
- **token** (*Optional[str]*) – Optional. The token used to encrypt the data.
- **salt** (*Optional[str]*) – Optional. The salt id.

Returns

The encrypted data with the salt used.

property getId

Return a unique token.

Usage:

```
PyCrypto.getId
```

Related Pages:

<https://docs.python.org/2/library/uuid.html>

Returns

A unique ID (based on the timestamp)

property key

Return a Fernet key.

Usage:

```
PyCrypto().key
```

Related Pages:

<https://cryptography.io/en/latest/fernet/>

5.8.2.4.8 Extension

class `epyk.core.py.PyExt.PyExt` (*page: PageModel*)

property crypto: *PyCrypto*

Property to the internal cryptography module.

This will rely on the package cryptography. This should be added to the python environment before using it. This package can be installed using the usual pip install function.

Related Pages:

<https://pypi.org/project/cryptography/> <https://cryptography.io/en/latest/>

Return type

PyCrypto.PyCrypto

property dates: *PyDates*

This is a simple wrapper to the datetime Python module.

No external package is required to use this interface.

Usage:

```
page.py.dates.today()
```

Returns

A PyDate object

Return type

PyDates.PyDates

static `encode_html`(*text: str, encoding: str = 'utf-8'*)

Parameters

- **text** – a text to encode with HTML special symbols.
- **encoding** – Optional. teh encoding type.

static `format_money`(*text: float, digits: int = 0, thousand_sep: str = ',', decimal_sep: str = '.', symbol: str = '£', format: str = '%s%v'*)

Parameters

- **text** –
- **digits** – Optional.
- **thousand_sep** – Optional.
- **decimal_sep** – Optional.
- **symbol** – Optional.
- **format** – Optional.

static `format_number`(*value: float, digits: int = 0, thousand_sep: str = ',', decimal_sep: str = '.'*)

Parameters

- **value** –
- **digits** –
- **thousand_sep** –
- **decimal_sep** –

property geo: *PyGeo*

Property to some predefined Geolocation functions.

Return type

PyGeo.PyGeo

import_lib(*lib_name: str, folder: str = 'libs', report_name: Optional[str] = None, path: Optional[str] = None*)

Import dynamically a python module.

Usage:

```
page.py.import_lib("test.py", folder="tables", path=r"filePath")
```

Parameters

- **lib_name** – The python module name.
- **folder** – Optional. The internal folder with the libraries to be imported.
- **report_name** – Optional. the report name in which the library is defined. Default current folder.
- **path** – Optional. the path to be added to the classpath.

Returns

The imported Python module.

import_package(*package: str, sub_module: Optional[str] = None*)

Install the external Python package. This can automatically install it from the Python Index online repository is missing.

Usage:

```
>>> PyExt().import_package("sqlalchemy").__name__
'sqlalchemy'
```

Parameters

- **package** – The Python Package Name.
- **sub_module** – Optional. The sub module or class within the package.

Returns

The installed Python module.

property markdown: [*Markdown*](#)

Property to the Markdown String conversion.

Return type

[*PyMarkdown.MarkDown*](#)

property requests: [*PyRest*](#)

This is a simple wrapper to the internal Python modules to run REST calls.

No external package is required to use this interface.

Returns

A PyRest object.

Return type

[*PyRest.PyRest*](#)

5.8.2.4.9 Geo

class epyk.core.py.PyGeo.**PyGeo**(page: *Optional[PageModel]* = None)

static distance(lat1: float, lon1: float, lat2: float, lon2: float, unit: str = 'km')

Calculate the great circle distance between two points on the earth (specified in decimal degrees) 3956

Related Pages:

https://en.wikipedia.org/wiki/Haversine_formula

Parameters

- **lat1** – Float.
- **lon1** – Float.
- **lat2** – Float.
- **lon2** – Float.
- **unit** – String. mi / km. Default km

5.8.2.4.10 Hash

class epyk.core.py.PyHash.**SipHash**(c: float = 2, d: float = 4)

Generate a unique hash ID from the given string. This is supposed to be unique with a minimum expectation of collisions. This module is only in charge of producing the hash ID and the potential collisions should be monitored in the environment by the users.

Related Pages:

<https://pypi.org/project/siphash/>

hashId(text: str)

Produce a unique ID for a given string. This can be used to replace the internal numbers.

Usage:

```
>>> SipHash().hashId("Test")
3169529217224722230
```

Related Pages:

<https://github.com/bozhu/siphash-python>

Parameters

- **text** – String. The String to be hashed.

5.8.2.5 Data Interface

Data is key for Visualisation so this is why Epyk provides ways to retrieve / connect the data.

5.8.2.5.1 Data Transformers

Those will transform the data to fit the format expected by the various containers. Basically this is the format expected in the HTML components in the `__init__` or in the `build` method.

Those functions can be used from the page object from `page.data` or directly from the module by using `pk`.

This module should be moved to a dedicated package in future releases to make the use on the backend side lighter.

5.8.2.5.2 Interface Documentation

class `epyk.core.data.Data.DataJs`(*page: PageModel*)

list(*js_code: str, data*)

Transform a Python object to a JavaScript list.

Parameters

- **js_code** (*str*) – The Javascript variable name.
- **data** – Object passed to the Javascript layer.

number(*js_code: str, value*)

Transform a Python number to a JavaScript one.

Parameters

- **js_code** – The Javascript variable name.
- **value** – Object passed to the Javascript layer.

object(*js_code: str, value: float*)

Transform a Python object to a JavaScript object.

Parameters

- **js_code** – The Javascript variable name.
- **value** – Object passed to the Javascript layer.

record(*js_code: Optional[str] = None, data=None*)

Interface to transform Python records to Javascript objects. This will allow interactivity of the various HTML components.

Usage:

```
:param js_code: Optional. The Javascript variable name.
:param data: Object passed to the Javascript layer.
```

Return type

`DataCore.DataGlobal`

server(hostname: str, port: int = 8080)

Configuration data for server interaction. This will only help on centralising the configuration in the final page.

Parameters

- **hostname** – The server hostname.
- **port** – Optional. The server port.

Return type

DataCore.ServerConfig

class epyk.core.data.Data.DataSrc(page: Optional[PageModel] = None)

property bb

Interface to Billboard data transformation.

This will convert Python object to input data for Billboard charts.

Return type

DataPy.C3

property c3

Interface to C3 data transformation.

This will convert Python object to input data for C3 charts.

Return type

DataPy.C3

property chartJs

Interface to chartJs data transformation.

This will convert Python object to input data for chartJs charts.

Return type

DataPy.ChartJs

property db

Interface to the internal database wrapper.

Return type

DataDb.DataDb

from_cache(code: str, is_secured: bool = False, report_name: Optional[str] = None)

Loads data from a cached files.

Parameters

- **code** – The code for the data.
- **is_secured** – Optional, boolean to set if the file should be secured. Default False.
- **report_name** – Optional. the environment in which cache are stored. Default current one.

Returns

Return the data

from_file(filename, isSecured=False, report_name=None)

Return the file.

Parameters

- **filename** – The filename.

- **isSecured** – Optional. Check if the file is secured or not.
- **report_name** – Optional. The environment with the file.

Returns

The file object

from_get(url, data=None, code=None)

from_source(http_data, file_name, func_name='getData', report_name=None, folder='sources', path=None)

Returns data from a internal data service defined in the sources folder.

Parameters

- **http_data** – The input data for the service
- **file_name** – The service file name
- **func_ame** – Optional, the function name in the service. Default getData
- **report_name** – Optional, the report name. Default the current one
- **folder** – Optional, the folder with the services. Default sources
- **path** – Optional, the path to be added to the python system path

property google

Interface to Google data transformation.

This will convert Python object to input data for Google charts.

Return type

DataPy.Google

grpc(service_name, path, module, host='localhost', port=50051)

Interface to a GRPC server.

Usage:

```
grpc = page.data.grpc(serviceName="GreeterStub", module="helloworld_pb2_grpc",
    path="")
data = grpc.imp("helloworld_pb2").HelloRequest(name="Test")
print(grpc.request("SayHello", data))
```

Related Pages:

<https://grpc.io/docs/tutorials/basic/python/> <https://grpc.io/docs/quickstart/python.html>

Parameters

- **service_name** – The Service name (the class name in the python module)
- **path** – The path with the GRPC features
- **module** – The python module name for the service
- **host** – The service host name (e.g localhost)
- **port** – The service port

Returns

A GRPC wrapped object

Return type

DataGrpc.DataGrpc

property js

Interface to standard JavaScript transformation.

Return type*DataJs***property nvd3**

Interface to NVD3 data transformation.

This will convert Python object to input data for NVD3 charts.

Return type

DataPy.NVD3

pdf(filename, path=None)

Read a pdf file

This will require an external module PyPDF2.

Usage:

```
data = page.data.pdf("document.pdf", r"")
data.getPage(0)
```

Related Pages:

<https://www.geeksforgeeks.org/working-with-pdf-files-in-python/>**Parameters**

- **filename** – The pdf file name
- **path** – The file path

Returns

A pdf object from PyPDF2

property plotly

Interface to Plotly data transformation.

This will convert Python object to input data for Plotly charts.

Return type

DataPy.Plotly

rest(url, data=None, method=None, encoding='utf-8', headers=None, unverifiable=False, proxy=None)

Interface to a REST server.

Test with a online server can be done here <https://jsonplaceholder.typicode.com/>

Usage:

```
page.data.rest("https://jsonplaceholder.typicode.com/posts/1", method="PUT")
```

Related Pages:

<https://jsonrpcclient.readthedocs.io/en/latest/api.html>**Parameters**

- **url** – The REST service url
- **data** – The input data for the service

rpc(url, data=None, headers=None, is_secured=False)

Interface to a RPC server.

This is using the external python package jsonrpcclient (<https://jsonrpcclient.readthedocs.io/en/latest/>)

Related Pages:

https://en.wikipedia.org/wiki/Rapid_control_prototyping <https://gurujsonrpc.appspot.com/>
<https://jsonrpcclient.readthedocs.io/en/latest/>

Parameters

- **url** – The RPC service url
- **data** – The input data for the service

rss(url, proxy=None, method='GET')

Entry point to retrieve RSS feeds.

This module will require beautifulsoup4 as external package

Usage:

```
xml_soup = rptObj.data.rss("http://feeds.reuters.com/reuters/businessNews")
for title in xml_soup.findAll('title'):
    print(title)
```

Related Pages:

<https://pypi.org/project/beautifulsoup4/>

Parameters

- **url** – The url of the html page
- **method** – Optional, The request method. Default method GET

Returns

A xml object

save_cache(data, code, is_secured: bool = False, if_missing: bool = True)

Temporary files are saved in a pickle manner in order to avoid having to parse those files again.

Parameters

- **data** – The data to be saved.
- **code** – The code for the data.
- **is_secured** – Optional. boolean to set if the file should be secured. Default False.
- **if_missing** – Optional. boolean to set the fact that caches are only saved if missing.

soap(wsdl)

Interface to a SOAP server.

This function will require an external python package zeep to use SOAP

Usage:

```
soap = page.data.soap("http://www.soapclient.com/xml/soapresponder.wsdl")
soap.Method1('Zeep', 'is cool')
```

Related Pages:

<https://en.wikipedia.org/wiki/SOAP> <https://python-zeep.readthedocs.io/en/master/>

Parameters

wsdl – The wsdl service url

Return type

zeep.service

Returns

The SOAP services

socket(*data*, *host*='localhost', *port*=5000, *encoding*='utf-8')

Parameters

- **data** – The input data for the service
- **host** – The service host name (e.g localhost)
- **port** – The service port
- **encoding** –

property vis

Interface to Vis data transformation.

This will convert Python object to input data for Vis charts.

Return type

DataPy.Vis

webscrapping(*url*, *parser*='html.parser', *proxy*=None, *method*=None)

Entry point to retrieve data from any website.

This module will require beautifulsoup4 as external package

Usage:

```
page.data.webscrapping("https://www.w3schools.com/colors/default.asp")
xml_soup.findAll('title')
```

Related Pages:

<https://pypi.org/project/beautifulsoup4/>

Parameters

- **url** – The url of the html page
- **parser** – The output data parser
- **proxy** –
- **method** –

Returns

A xml object

5.8.2.6 Web Framework Interfaces

5.8.2.7 Example

```
import epyk as pk

page = pk.Page()
page.ui.title("This is a title")
paragraph = page.ui.texts.paragraph("This is a paragraph")
paragraph.click([
    page.js.console("This is a log on the JavaScript side")
])
page.outs.html()
```

5.8.2.8 Technical Documentation

class epyk.core.Page.**Report**(inputs: Optional[dict] = None, script: Optional[str] = None)

Main entry point for any web UI.

This class will store all the HTML components, JavaScript fragments and CSS definition in order to then render a rich web page.

This will allow Python to access the components before the JavaScript on the fly computation to change then according to the input data.

This class will also interface with plain Vanilla JavaScript feature to allow the design and definition of events and / or interactions. Most of the Web documentation in this framework is either coming from w3School or from the various external packages.

property apps: **AppRoute**

Change the report to a web application.

This will add extra features available on the target framework. For example this HTML page can be transformed to an Angular app, a React App or a Vue one.

Usage:

```
page = Report()
```

property auth: **Auth**

Auth interface to allow easy sign-in pages.

Usage:

```
page = Report()
page.auth.
```

Related Pages:

<https://developers.google.com/identity/sign-in/web/sign-in>

Returns

Python Auth Object.

property body: Body

Property that returns the Body element of the HTML page.

Usage:

```
page = Report()
page.body.onReady([page.js.alert("Loading started")])
```

property css: Catalog

Returns the set of CSS Classes for the HTML report.

Usage:

```
page = Report()
page.css.
```

property data: DataSrc

Python internal data source management.

This can be extended by inheriting from this `epyk.core.data.DataSrc.DataSrc` and adding extra entry points.

Usage:

```
page = Report()
```

Returns

The framework available data source

dumps(result: dict)

Function used to dump the data before being sent to the Javascript layer.

This function relies on `json.dumps` with a special encoder in order to work with Numpy array and Pandas data structures.

As NaN is not valid on the Json side those object are not allowed during the dump. It is advised to use `fillna()` in your script before returning the data to the framework to avoid this issue.

Usage:

```
page = Report()
page.dumps(result)
```

Related Pages:

<https://docs.python.org/2/library/json.html>

Parameters

result – The python dictionary or data structure

Returns

The serialised data

property entities: Entities

Shortcut to the HTML Entities.

Those can be added in string in order to improve the render of a text.

Usage:

```
page = Report()
page.ui.text(page.entities.non_breaking_space)
```

Related Pages:

https://www.w3schools.com/html/html_entities.asp

framework(*name: str*)

Flag to change the way code is transpiled in order to fit with the destination framework.

By default the code transpiled will be used from a browser in plain Vanilla Js but this will be extended to then be compatible with other framework in order to simplify the path to production and the collaboration between teams.

Many framework will be compatible like React, Angular, Vue but also some features will be exposed to Kotlin for mobile generation.

This work is still in progress.

Usage:

```
page = Report()
page.ui.text("This is an example")
page.framework("Vue")
```

Parameters

name – The destination framework for the page

get_components(*html_codes: Union[list, str]*)

Retrieve the components based on their ID.

This should be used when the htmlCode is defined for a component.

Usage:

```
page = Report()
page.ui.button(htmlCode="Button")
but = page.get_components(["Button"])
```

Parameters

html_codes – The reference of the HTML components loaded on the page

property headers: Header

Property to the HTML page header.

Usage:

```
page = pk.Page()
page.headers.meta.viewport({"width": "device-width"})

# Use the default DEV icon.
page.headers.dev()
```

property icons: IconModel

Change the icons framework used in the page. Defaults.py in the CSS module is to change the framework for all the page generated by the framework.

Usage:

```
page = pk.Page()
page.icons.family = "bootstrap-icons"
icons = page.ui.menus.icons([
    "bi-1-circle-fill",
    "bi-search-heart-fill",
    "bi-x-circle-fill",
])
```

property imports: *ImportManager*

Return the report/import_manager, which allows to import automatically packages for certain components to run.

By default the imports are retrieved online from CDNJS paths. This can be changed by loading the packages locally and switching off the online mode.

Usage:

```
page = Report()
page.imports.setVersion(page.imports.pkgs.popper_js.alias, "1.00.0")
```

property js: *JsBase*

Go to the Javascript section. Property to get all the JavaScript features.

Most of the standard modules will be available in order to add event and interaction to the Js transpiled.

Usage:

```
page = Report()
page.js.console.log("test")

page.js.accounting.add_to_imports()
page.js.moment.add_to_imports()
```

Related Pages:

<https://www.w3schools.com/js/default.asp>

node_modules(*path: str, alias: Optional[str] = None, install: bool = False, update: bool = False*)

Usage:

```
:param path: Optional. The nodeJs path.
:param alias: Optional.
:param install: Optional.
:param update: Optional.
```

property outs: *PyOuts*

Link to the possible output formats for a page.

This will transpile the Python code to web artifacts. Those outputs are standard outputs files in web development.

The property framework should be used to link to other web framework.

Usage:

```
page = Report()
page.ui.text("This is an example")
page.outs.html()
```

property properties

Property to the different Page properties JavaScript and CSS.

property py: *PyExt*

Python external module section.

Those are pre-defined Python function to simplify the use of the various components.

Usage:

```
page = Report()
page.py.dates.today()
```

Related Pages:

<https://www.w3schools.com/js/default.asp>

register(*ext_components: list*)

This function allows you to register external Components (namely coming from Pyk Reports) by registering them you this will engrave the object within your report.

The example below will add obj1 and obj2 from an external pyk report previously required, then create a div and then add obj3 from an external file.

Usage:

```
page = Report()
page.register([obj1, obj2])
page.ui.div('this is a div')
page.register(obj3)
```

Parameters

ext_components – The external components to be added

property root__script: str

Return the name of the script creating the Page object.

property skins: Skins

Add a special skin to the page.

This could be used for special event or season during the year (Christmas for example).

Usage:

```
page = pk.Page()
page.ui.text("Hello World !")
page.skins.rains()
page.outs.html_file(name="test", print_paths=True)
```

property symbols: Symboles

Shortcut to the HTML symbols.

Those can be added in string in order to improve the render of a text.

Usage:

```
page = Report()
page.ui.text(page.symbols.shapes.BLACK_SQUARE)
```

Related Pages:

https://www.w3schools.com/html/html_symbols.asp https://www.w3schools.com/charsets/ref_utf_math.asp

property theme: `<module 'epyk.core.css.themes.Theme' from
'../epyk/core/css/themes/Theme.py'>`

Return the currently used report/theme for the report.

Usage:

```
page = Report()
page.theme = themes.ThemeBlue.Blue
```

property ui: *Components*

User Interface section.

All the components which can be used in the dashboard to display the data. Within this object different categories of items can be used like (list, simple text, charts...).

Usage:

```
page = Report()
page.ui.text("This is a text")
```

Related Pages:

<https://www.w3schools.com/html/default.asp>

property web: *WebComponents*

User Interface section.

All the components which can be used in the dashboard to display the data. Within this object different categories of items can be used like (list, simple text, charts...).

Usage:

```
page = Report()
page.web
```

Related Pages:

<https://www.w3schools.com/html/default.asp>

5.8.3 Themes

5.8.4 HTML Built-Ins

5.8.5 CSS Built-Ins

- CSS Properties.
- CSS Classes (Catalog).
- Effects.
- Themes.

5.8.5.1 Core Modules

5.8.5.1.1 CSS Class

class epyk.core.css.styles.GrpCls.**ClassPage**(*component: Optional[HtmlModel] = None, page: Optional[PageModel] = None*)

property add_classes: **Catalog**

Property to get access to the catalog of CSS classes to be added to the HTML class tag component.

contenteditable() → CssPageContentEditable

Set the border color of the editable content according to the selected theme.

Related Pages:

https://www.w3schools.com/howto/howto_css_contenteditable_border.asp

property css: **Body**

Property to the underlying CSS definition to be added to the style HTML tag of a component.

property css_class: **CatalogDiv**

The internal class used to put a custom Style to this object.

Only 1 CSS class can be added to an HTML object.

custom_class(*css_attrs: dict, classname: Optional[str] = None, selector: Optional[str] = None, is_class: bool = True, important: bool = False*) → dict

This will create dynamic CSS class which will not be added to any component.

The class definition can then be reused in multiple components. The CSS style of the body can only be done using predefined classes or inline CSS.

TODO: Enable the important for nested css_attrs.

Usage:

```
page.body.style.custom_class(css_attrs={"_attrs": {"fill": 'red'}}, classname=
↪ 'nvd3.nv-pie .nv-pie-title')
```

Parameters

- **css_attrs** – Nested dictionary with the different attributes
- **classname** – Optional. The classname in the CSS definition

- **selector** – Optional. The class selector (if it is not a classname using . but a strict definition)
- **is_class** – Optional. Automatically transform the name to a CSS class definition by adding a dot
- **important** – Optional. Specify if the style is important

property defaults

The Default CSS Attributes in the framework.

property define_classes: Catalog

Property to get access to the catalog of CSS classes to be loaded in the page.

Those classes will not be automatically added to any HTML tag and they need to be added manually.

fit_screen_height(*margin_size: Optional[int] = None*)

Parameters

margin_size – Optional.

get_classes()

Returns the list of Internal and bespoke classes to be added to the class HTML table on the component.

get_classes_css()

Attach the predefined styles for the scrollbar and selection then return all the classes.

property globals: GlobalStyle

Reference for all the global setting in the page.

This should be changed in order to be the proxy to the Default CSS settings in the framework. Changing this should only impact the report default settings.

TODO: Extend to more than the font

property moz_selection: CssWebkitMozSelection

Selection predefined style (background color based on the selected theme).

Related Pages:

https://www.w3schools.com/howto/howto_css_text_selection.asp

property scrollbar_webkit: CssWebkitScrollbar

Scrollbars predefined styles.

property scrollbar_webkit_thumb: CssWebkitScrollbarThumb

Scrollbars predefined styles.

property scrollbar_webkit_track

Scrollbars predefined styles.

property selection: CssWebkitSelection

Selection predefined style (background color based on the selected theme).

Related Pages:

https://www.w3schools.com/howto/howto_css_text_selection.asp

5.8.5.1.2 CSS Style

```
class epyk.core.css.styles.classes.CssStyle.Style(page: PageModel, css_ovrs: Optional[dict] =
None, selector_ovrs: Optional[dict] = None,
html_id: Optional[str] = None, component:
Optional[HtmlModel] = None)
```

property active: Data

Selects the active link.

Related Pages:

https://www.w3schools.com/cssref/sel_active.asp

property after: Data

Insert something after the content of each <p> element.

Related Pages:

https://www.w3schools.com/cssref/sel_after.asp

```
animation(name=None, attrs=None, duration=2, delay=None, iteration='infinite', timing_fnc=None,
effect=None, fill_mode=None)
```

The @keyframes rule specifies the animation code.

The animation is created by gradually changing from one set of CSS classes to another.

Usage:

```
page.ui.button("Ok").style.css_class.animation('test', {
    "from": {"border-color": "white"},
    "to": {"border-color": "red"},
})
```

Related Pages:

https://www.w3schools.com/cssref/css3_pr_animation-keyframes.asp https://www.w3schools.com/css/css3_animations.asp

Parameters

- **effect** – Effect Class.
- **name** – String. Required. Defines the name of the animation.
- **attrs** – String. Required. Percentage of the animation duration.
- **duration** –
- **delay** –
- **iteration** –
- **timing_fnc** –
- **fill_mode** – String Specify the fill mode (whether the style should go back to its original position or

something else etc...

property before: Data

Insert something before the content of each <p> element.

Related Pages:

https://www.w3schools.com/cssref/sel_before.asp

property checked: Data

Selects every checked <input> element.

Related Pages:

https://www.w3schools.com/cssref/sel_checked.asp

css(key: Union[str, dict], value: Optional[str] = None, important: bool = False, change: bool = True)

Add a CSS attribute to a class.

Parameters

- **key** (Union[str, dict]) – The CSS attribute.
- **value** (str) – Optional. The CSS value.
- **important** (bool) – Optional. The level of priority for this attribute.
- **change** (bool) – Optional. A flag to specify the state of the CSS class.

customize()

Define for child classes to define some CSS attributes. This parent class is just there to define the structure of all the child ones, it is not used directly.

property disabled: Data

Selects every disabled <input> element.

Related Pages:

https://www.w3schools.com/cssref/sel_disabled.asp

property empty: Data

Selects every <p> element that has no children (including text nodes).

Related Pages:

https://www.w3schools.com/cssref/sel_empty.asp

property enabled: Data

Selects every enabled <input> element.

Related Pages:

https://www.w3schools.com/cssref/sel_enabled.asp

property focus: Data

Selects the input element which has focus.

Related Pages:

https://www.w3schools.com/cssref/sel_focus.asp

get_ref() → str

Get the style class reference.

property has_changed: bool

Set an internal flag to specify if the class has changed from the creation in the framework.

If the state of the class has changed, the class will not be generic anymore so it will change the CSS class reference by adding the Component id in order to make it specific.

This is a trick in order to be able to change common CSS classes for a specific component without impacting the other ones in the page.

property hover: Data

Selects links on mouse over.

Related Pages:

https://www.w3schools.com/cssref/sel_hover.asp

property invalid: Data

Selects all input elements with an invalid value.

Related Pages:

https://www.w3schools.com/cssref/sel_invalid.asp

keyframes(*name: str, attrs: dict, effects=None, change: bool = True*)

The @keyframes rule specifies the animation code.

The animation is created by gradually changing from one set of CSS styles to another.

Usage:

```
page.style.keyframes("test", {
    "50%": {"transform": "scale(1.5, 1.5)", "opacity": 0},
    "99%": {"transform": "scale(0.001, 0.001)", "opacity": 0},
    "100%": {"transform": "scale(0.001, 0.001)", "opacity": 1},
})
```

Related Pages:

https://www.w3schools.com/cssref/css3_pr_animation-keyframes.asp

Parameters

- **effects** – Effect Class.
- **name** (*str*) – Defines the name of the animation.
- **attrs** (*dict*) – Percentage of the animation duration.
- **change** (*bool*) – Optional. A flag to specify the state of the CSS class.

media(*attrs: dict, rule=None, media_type=None, media_feature=None, change=True, this_class=False*)

The @media is used in media queries to apply different styles for different media types/devices.

Usage:

```
page.style.media({"body": {"background-color": "lightblue"}}, "only", "screen",
    {'and': [{'height': '100px'}, {'min-width': '600px'}]})
```

The first key of the attributes can be an Epyk html object.

Related Pages:

https://www.w3schools.com/cssref/css3_pr_mediaquery.asp

Parameters

- **attrs** (*dict*) – Percentage of the animation duration.
- **rule** – String. Optional. not or only or and see documentation for more info.
- **media_type** – String. Optional. the media to which the rule will need to be applied.
- **media_feature** – String. Optional. Media features provide more specific details to media queries.
- **change** – Boolean. Optional. A flag to specify the state of the CSS class.
- **this_class** – Boolean. Optional. Specify if this should be applied to this class only.

transition(*attribute: str, duration: int = 2, delay: Optional[int] = None, iteration=None, timing_fnc: Optional[str] = None*)

Parameters

- **attribute** (*str*) –
- **duration** (*int*) – Optional. The duration of the transition effect.
- **delay** (*int*) – Optional. The time delay before starting the transition.
- **iteration** (*int*) – Optional. The count of iteration.
- **timing_fnc** (*str*) – Optional. The timing function from (“ease”, “linear”, “ease-in”, “ease-out”, “ease-in-out”).

property valid: Data

Selects all input elements with a valid value.

Related Pages:

https://www.w3schools.com/cssref/sel_valid.asp

property visited: Data

Selects all visited links.

Related Pages:

https://www.w3schools.com/cssref/sel_visited.asp

property webkit_slider_thumb: Data

5.8.5.1.3 Colors

class epyk.core.css.Colors.HexColors

5.8.6 Javascript Built-Ins

Most of the JavaScript features are wrapped in Epyk in order to be able to use the auto completion to write JavaScript code. By design this framework is not dedicated to write thousand of lines of JavaScript. It has been implemented in order to help link components and modules with the Python object.

This module can be split into 4 main items:

- JavaScript core features.
- JavaScript Primitives.
- JavaScript HTML wrappers.
- JavaScript Packages wrappers.

5.8.6.1 Core Modules

```
class epyk.core.js.Js.JsBase(page: Optional[PageModel] = None, component: Optional[HtmlModel] = None)
```

property accounting

Shortcut to accounting properties.

Usages:

```
page.js.accounting.add_to_imports()
```

Related Pages:

<http://openexchangerates.github.io/accounting.js/>

activeElement()

The activeElement property returns the currently focused element in the document.

Related Pages:

https://www.w3schools.com/jsref/prop_document_activeelement.asp

Returns

A reference to the element object in the document that has focus.

property body: JsDoms

Get the DOM object.

This will return the object. It will not create any variable.

property breadcrumb: JsBreadcrumb

Create an internal Breadcrumb to keep track of the user journey within your page.

Related Pages:

https://www.w3schools.com/howto/howto_css_breadcrumbs.asp

Returns

A Python breadcrumb object.

clipboard(*data: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)

Copy the full URL to the clipboard.

Related Pages:

<https://isabelcastillo.com/hidden-input-javascript>

Parameters

- **data** – The Javascript expression
- **js_conv_func** – Optional. A specific JavaScript data conversion function

createAttribute(*attribute_name*)

The createAttribute() method creates an attribute with the specified name, and returns the attribute as an Attr object.

Related Pages:

https://www.w3schools.com/jsref/met_document_createattribute.asp

Parameters

attribute_name – The name of the attribute you want to create.

Returns

A Node object, representing the created attribute.

createElement(*tag_name: str, js_code: Optional[str] = None, set_var: bool = True, dom_id: Optional[str] = None*)

The createElement() method creates an Element Node with the specified name.

Related Pages:

https://www.w3schools.com/jsref/met_document_createelement.asp

Parameters

- **tag_name** – The name of the element you want to create.
- **js_code** – The variable name to be set. Default random name.
- **set_var** – Optional. Create a variable for the new object. Default True.
- **dom_id** – Optional. The Dom ID reference for the object.

createEvent(*event_type: str*)

The createEvent() method creates an event object.

The event can be of any legal event type, and must be initialized before use.

Related Pages:

https://www.w3schools.com/jsref/event_createevent.asp

Parameters

event_type – A String that specifies the type of the event.

Returns

An Event object

static createTextNode(*text: Optional[Union[JsDataModel, str]] = None, js_conv_func: Optional[Union[list, str]] = None*) → JsObject

The createTextNode() method creates a Text Node with the specified text.

Related Pages:

https://www.w3schools.com/jsref/met_document_createtextnode.asp

Parameters

- **text** – Optional. The text of the Text node.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

A Text Node object with the created Text Node.

custom(*data: Union[str, JsDataModel], key: Optional[str] = None, is_py_data: bool = False, js_func: Optional[Union[list, str]] = None*)

Allow the definition of bespoke javascript strings.

Parameters

- **data** – A String corresponding to a JavaScript object
- **key** – Optional. A key reference in the JavaScript object
- **is_py_data** – Optional. Specify if the data is in Python and should be jsonify first
- **js_func** – Optional. Javascript functions

customFile(*filename: str, path: Optional[str] = None, module_type: str = 'text/javascript', absolute_path: bool = False, requirements: Optional[list] = None, randomize: bool = False, authorize: bool = False*)

This will load your local javascript file when the report will be built. Then you will be able to use the new features in the different Javascript wrappers.

Usage:

```
page.js.customFile("test.js", r"C:
```

older")

param filename

The filename

param path

Optional. The file path

param module_type

Optional. The module type

param absolute_path

Optional. If path is None this flag will map to the current main path

param requirements

Optional. The list of required packages

param randomize

Optional. Add random suffix to the module to avoid browser caching

param authorize

Optional. Add to the restricted list of packages

return

The Js Object to allow the chaining.

customText(*text: str*)

Javascript fragment added at the beginning of the page. This will be called before any function in the framework.

Parameters

text – The Javascript fragment

Returns

self to allow the chaining.

property d3

D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG, and CSS.

D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

Related Pages:

<https://d3js.org/>

property data: JsData

Get wrapped JavaScript data structures.

decodeURIComponent(*url_enc: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)
→ JsObject

The decodeURIComponent() function decodes a URI component.

Related Pages:

https://www.w3schools.com/jsref/jsref_decodeuricomponent.asp

Parameters

- **url_enc** – The URI to be decoded
- **js_conv_func** – Optional. A specific JavaScript data conversion function

Returns

A String, representing the decoded URI.

delay(*js_funcs: Union[list, str], seconds: int = 0, window_id: str = 'window', profile: Optional[Union[bool, dict]] = False*)

Add a wrapper on top of the setTimeout.

Usage:

```
page.js.delay([text.build("Change the value")], 5)
```

Parameters

- **js_funcs** – The function that will be executed
- **seconds** – Optional. The number of seconds to wait before executing the code

- **window_id** – Optional. The JavaScript window object
- **profile** – Optional. Set to true to get the profile for the function on the Javascript console

delete(*url: Union[str, JsDataModel], data: Optional[dict] = None, js_code: str = 'response', is_json: bool = True, components: Optional[List[Union[Tuple[HtmlModel, str], HtmlModel]]] = None, profile: Optional[Union[bool, dict]] = None, headers: Optional[dict] = None, asynchronous: bool = False, stringify: bool = True*) → XMLHttpRequest

Create a DELETE HTTP request.

Related Pages:

<https://pythonise.com/series/learning-flask/flask-http-methods>

Parameters

- **url** – The url path of the HTTP request
- **data** – Optional. Corresponding to a JavaScript object
- **js_code** – Optional. The variable name created in the Javascript (default response)
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. This will add the component value to the request object
- **profile** – Optional. A flag to set the component performance storage
- **headers** – Optional. The request headers
- **asynchronous** – Async flag: true (asynchronous) or false (synchronous)
- **stringify** – Optional. Stringify the request data for json exchange

property documentElement

Document.documentElement returns the Element that is the root element of the document (for example, the <html> element for HTML documents).

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Document/documentElement>

encodeURIComponent(*uri: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*) → JsObject

The encodeURIComponent() function encodes a URI component.

Related Pages:

https://www.w3schools.com/jsref/jsref_encodeuricomponent.asp

Parameters

- **uri** – The URI to be encoded
- **js_conv_func** – Optional. A specific JavaScript data conversion function

Returns

A String, representing the encoded URI.

eval(*data: Union[JsDataModel, str], js_conv_func: Optional[Union[list, str]] = None*)

The eval() function evaluates JavaScript code represented as a string.

Warning: Executing JavaScript from a string is an enormous security risk. It is far too easy for a bad actor to run arbitrary code when you use eval(). See Never use eval()!, below.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/eval

Parameters

- **data** – Data to be evaluated
- **js_conv_func** – Optional. A specific JavaScript data conversion function

execCommand(*command: str, show_ui: bool, value: str*)

The execCommand() method executes the specified command for the selected part of an editable section.

Related Pages:

https://www.w3schools.com/jsref/met_document_execcommand.asp

:param command:. Specifies the name of the command to execute on the selected section. :param show_ui: specifies if the UI should be shown or not. :param value: Some commands need a value to be completed.

Returns

A Boolean, false if the command is not supported, otherwise true.

extendProto(*py_class: Any, func_name: str, js_funcs: Union[str, list], pmts: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*)

Javascript Framework extension.

Hook in the base class to allow the definition of specific function to add extra primitive features. Usual this function should be used in a wrapper function with the same name in order to have a coherent bridge between Python and Javascript.

Related Pages:

https://www.w3schools.com/js/js_object_prototypes.asp

Parameters

- **py_class** – PyJs class name
- **func_name** – The Javascript function name
- **js_funcs** – Javascript functions
- **pmts** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

Returns

The Js Object to allow the chaining.

fetch(*url: str, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False, async_await: bool = False*) → JsPromise

The Fetch API provides a JavaScript interface for accessing and manipulating parts of the HTTP pipeline, such as requests and responses.

Usage:

```
page.ui.button("Click").click([
    page.js.fetch("test", {"method": "POST"}).then([
        page.js.console.log(pk.events.response)
    ])
])
```

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Parameters

- **url** – The target url
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage
- **async_await** – Optional.

property fncs: JsRegisteredFunctions

Property to the predefined Javascript functions.

Returns

The predefined functions.

for_(*js_funcs: Optional[Union[list, str]] = None, step: int = 1, start: int = 0, end: int = 10, options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*) → JsFor

Shortcut to a for loop.

Usage:

```
js_for = page.js.for_(end=30)
js_for.fncs([page.js.console.log(js_for.i)])
```

Related Pages:

https://www.w3schools.com/js/js_loop_for.asp

Parameters

- **js_funcs** – Javascript functions
- **step** – Optional. The value to increment. Default 1
- **start** – Optional. The first index in the for loop
- **end** – Optional. The last index in the for loop
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

get(*url: Union[str, JsDataModel], data: Optional[dict] = None, js_code: str = 'response', is_json: bool = True, components: Optional[Union[Tuple[HtmlModel, str], List[HtmlModel]]] = None, headers: Optional[dict] = None, asynchronous: bool = False, stringify: bool = True*) → XMLHttpRequest

Create a GET HTTP request.

Usage:

```
inputs = page.ui.input("")
btn = page.ui.button("Click").click([
    page.js.get("/test", {"fegeg": "efefe", "ok": inputs.dom.content}, {
components=({"input", inputs}))
])
```

Parameters

- **url** – The url path of the HTTP request
- **data** – Optional. A String corresponding to a JavaScript object
- **js_code** – Optional. The variable name created in the Javascript (default response)
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. This will add the component value to the request object
- **headers** – Optional. The request headers
- **asynchronous** – Async flag: true (asynchronous) or false (synchronous)

static getElementById(*id_name: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)

The getElementById() method returns the element that has the ID attribute with the specified value.

Related Pages:

https://www.w3schools.com/jsref/met_document_getelementbyid.asp

Parameters

- **id_name** – The ID attribute's value of the element you want to get.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

An Element Object, representing an element with the specified ID. Returns null if no elements with

the specified ID exists

static getElementsByClassName(*cls_name: str*) → JsDoms

The getElementsByClassName() method returns a collection of all elements in the document with the specified class name, as a NodeList object.

Related Pages:

https://www.w3schools.com/jsref/met_document_getelementsbyclassname.asp

Parameters

cls_name – The class name of the elements you want to get.

Returns

A NodeList object, representing a collection of elements with the specified class name. The elements in the returned collection are sorted as they appear in the source code.

static `getElementsByName`(*name*: Union[str, JsDataModel], *js_conv_func*: Optional[Union[list, str]] = None) → JsDomsList

The `getElementsByName()` method returns a collection of all elements in the document with the specified name (the value of the name attribute), as a `NodeList` object.

The `NodeList` object represents a collection of nodes. The nodes can be accessed by index numbers. The index starts at 0.

Related Pages:

https://www.w3schools.com/jsref/met_doc_getelementsbyname.asp

Parameters

- **name** – The name attribute value of the element you want to access/manipulate.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

A `NodeList` object, representing a collection of elements with the specified name. The elements in the returned collection are sorted as they appear in the source code.

static `getElementsByTagName`(*tag_name*: Union[str, JsDataModel], *i*: int = 0, *js_conv_func*: Optional[Union[list, str]] = None) → JsDoms

The `getElementsByTagName()` method returns a collection of an element's child elements with the specified tag name, as a `NodeList` object.

The `NodeList` object represents a collection of nodes. The nodes can be accessed by index numbers. The index starts at 0.

Related Pages:

https://www.w3schools.com/jsref/met_element_getelementsbytagname.asp

Parameters

- **tag_name** – The tag name of the child elements you want to get
- **i** – Optional. The index of the element
- **js_conv_func** – Optional. A specific JavaScript data conversion function

getVar(*js_code*: Union[str, JsDataModel], *var_type*: str = 'var') → JsObject

Get the Javascript Variable name.

Parameters

- **js_code** – The Variable name
- **var_type** – Optional. The scope of the variable

Returns

Return the piece of script to be added to the Javascript.

if_(*condition*: Union[str, list, bool], *js_funcs*: Union[list, str], *profile*: Optional[Union[bool, dict]] = False)

Conditional statements are used to perform different actions based on different conditions.

Usage:

```
page.js.if_(icon.icon.dom.content == "fas fa-lock-open cssicon", [
    page.js.console.log(icon.icon.dom.content),
])
```

Related Pages:

https://www.w3schools.com/js/js_if_else.asp

Parameters

- **condition** – The Javascript condition. Can be a JsBoolean object
- **js_funcs** – Optional. The Javascript functions
- **profile** – Optional. A flag to set the component performance storage

import_css(*script: str*)

Add a CSS file on the fly from a JavaScript event.

Related Pages:

<https://stackoverflow.com/questions/19844545/replacing-css-file-on-the-fly-and-apply-the-new-style-to-the-page>

Parameters

script – A script name with a CSS extension.

import_js(*script: str, js_funcs: Union[str, list], profile: Optional[Union[bool, dict]] = None, self_contained: bool = False*)

Add a Javascript module and then run function once it is loaded.

Related Pages:

<https://cleverbeagle.com/blog/articles/tutorial-how-to-load-third-party-scripts-dynamically-in-javascript>

Parameters

- **script** – A script name. A Js extension.
- **js_funcs** – The Javascript functions.
- **profile** – Optional. A flag to set the component performance storage.
- **self_contained** – Optional. A flag to specify where the import will be done.

info(*data: Union[str, JsDataModel], css_style: Optional[dict] = None, icon: str = 'fas fa-spinner fa-spin', seconds: int = 10000*)

Display a message.

Related Pages:

<https://fontawesome.com/how-to-use/on-the-web/styling/animating-icons>

Parameters

- **data** – A String corresponding to a JavaScript object
- **css_style** – Optional. The CSS attributes to be added to the HTML component
- **icon** – Optional. A string with the value of the icon to display from font-awesome
- **seconds** – Optional. The number of second the info will be visible

intersectionObserver(*js_code*: str, *callback*: Optional[Union[List[Union[str, JsDataModel]], str]] = None, *options*: Optional[dict] = None, *observe_once*: bool = False, *profile*: Optional[Union[bool, dict]] = None) → IntersectionObserver

Parameters

- **js_code** – The PyJs functions.
- **callback** – JavaScript functions called by the intersectionObserver.
- **options** – intersectionObserver options.
- **observe_once** – A flag to remove the observable once callbacks run.
- **profile** – Option to perform profiling logs in the browser console.

property jquery

jQuery is a fast, small, and feature-rich JavaScript library.

It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Usage:

```
btn = page.ui.button("Click")
btn.js.jquery.on("click", [
    page.js.alert("It works"),
    btn.js.jquery.after('<div style="background-color:yellow"> New div </div>'),
])
```

Related Pages:

<https://jquery.com/>

property keydown: KeyCode

The onkeydown event occurs when the user is pressing a key (on the keyboard).

Related Pages:

https://www.w3schools.com/jsref/event_onkeydown.asp

property keypress: KeyCode

The onkeypress event occurs when the user presses a key (on the keyboard).

Related Pages:

https://www.w3schools.com/jsref/event_onkeypress.asp

property keyup: KeyCode

The onkeypress event occurs when the user presses a key (on the keyboard).

Related Pages:

https://www.w3schools.com/jsref/event_onkeypress.asp

property location: JsLocation

Property to the Javascript Location functions.

Usage:

```
page.ui.text("Test").click([
  page.js.location.open_new_tab(page.js.location.getUrlFromArrays([
    ["AAA", "BBB"], ["111", "222"]]), end_line="
```

```
“”))])
```

Related Pages:

https://www.w3schools.com/jsref/obj_location.asp

mail(*mails*, *subject=None*, *body=None*, *cc=None*, *bcc=None*)

Create an email.

Related Pages:

<https://www.w3docs.com/snippets/html/how-to-create-mailto-links.html>

Parameters

- **mails** –
- **subject** –
- **body** –
- **bcc** –

property **mediaRecorder**: **MediaRecorder**

The MediaRecorder interface of the MediaStream Recording API provides functionality to easily record media. It is created using the MediaRecorder() constructor.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder>

property **moment**

Parse, validate, manipulate, and display dates and times in JavaScript.

Usage:

```
page.js.moment.new("2021-08-05", varName="momentTime"),
page.js.console.log(page.js.moment.var("momentTime").weekYear(1998)),
page.js.console.log(page.js.moment.var("momentTime").weekYear()),
page.js.console.log(page.js.moment.new("2021-08-05")),
```

Related Pages:

<https://momentjs.com/> <https://github.com/you-dont-need/You-Dont-Need-Momentjs>

property **msg**: **Msg**

Shortcut to predefined temporary messages displayed to the UI.

navigateTo(*url: Union[str, JsDataModel]*, *options: Optional[dict] = None*)

Navigators to another URL like NodeJs.

Usage:

```
icon.click([self.context.page.js.navigateTo(url)])
```

Related Pages:

<https://redfin.github.io/react-server/annotated-src/navigateTo.html>

Parameters

- **url** – The target url.
- **options** – Optional. The property of the location object.

property navigator: JsNavigator

The information from the navigator object can often be misleading, and should not be used to detect browser versions because:

- Different browsers can use the same name.
- The navigator data can be changed by the browser owner.
- Some browsers misidentify themselves to bypass site tests.
- Browsers cannot report new operating systems, released later than the browser.

not_(*data*, *js_conv_func*: *Optional[Union[list, str]] = None*) → JsFunction

Add the Symbol (!) for the boolean negation. This feature is also available directly to any JsBoolean objects.

Usage:

```
jsObj.not_(jsObj.objects.boolean.get("weekend"))
```

Related Pages:

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Op%C3%A9rateurs/Op%C3%A9rateurs_logiques

Parameters

- **data** – A String corresponding to a JavaScript object
- **js_conv_func** – Optional. A specific JavaScript data conversion function

Returns

The Javascript fragment string.

number(*data*, *js_code*: *Optional[str] = None*, *set_var*: *bool = False*, *is_py_data*: *bool = True*) → JsNumber

Shortcut to the Javascript Number primitives.

Parameters

- **data** – The String data.
- **js_code** – Optional. The specific name to be used for this JavaScript String.
- **set_var** – Optional. Set a variable. Default False.
- **is_py_data** – Optional. Specify the type of data.

object(*data*, *js_code*: *Optional[str] = None*, *set_var*: *bool = False*, *is_py_data*: *bool = True*) → JsObject

Shortcut to the Javascript Object primitives.

Parameters

- **data** – The String data.
- **js_code** – Optional. The specific name to be used for this JavaScript String.
- **set_var** – Optional. Set a variable. Default False.
- **is_py_data** – Optional. Specify the type of data.

property objects: JsObjects

Interface to the main Javascript Classes and Primitives.

onReady(*js_funcs*: Union[str, list], *profile*: Optional[Union[bool, dict]] = False)

The ready event occurs when the body DOM (document object model) has been loaded.

Related Pages:

https://www.w3schools.com/jquery/event_ready.asp

Parameters

- **js_funcs** – The Javascript functions to be added to this section
- **profile** – Optional. A flag to set the component performance storage

static parseDate(*value*: str) → JsNumber

The parse() method parses a date string and returns the number of milliseconds between the date string and midnight of January 1, 1970.

Related Pages:

https://www.w3schools.com/jsref/jsref_parse.asp

Parameters

value – A string representing a date.

Returns

Number. Representing the milliseconds between the specified date-time and midnight January 1, 1970.

static parseFloat(*value*: str) → JsNumber

The parseFloat() function parses a string and returns a floating point number.

Related Pages:

https://www.w3schools.com/jsref/jsref_parseint.asp

Parameters

value – The string to be parsed.

Returns

A Number. If the first character cannot be converted to a number, NaN is returned.

static parseInt(*value*: str)

The parseInt() function parses a string and returns an integer.

Related Pages:

https://www.w3schools.com/jsref/jsref_parseint.asp

Parameters

value – The string to be parsed.

Returns

A Number. If the first character cannot be converted to a number, NaN is returned.

patch(url: Union[str, JsDataModel], data: Optional[dict] = None, js_code: str = 'response', is_json: bool = True, components: Optional[List[Union[Tuple[HtmlModel, str], HtmlModel]]] = None, profile: Optional[Union[bool, dict]] = None, headers: Optional[dict] = None, asynchronous: bool = False, stringify: bool = True) → XMLHttpRequest

Create a PATH HTTP request.

Related Pages:

<https://pythonise.com/series/learning-flask/flask-http-methods>

Parameters

- **url** – The url path of the HTTP request
- **data** – Optional. Corresponding to a JavaScript object
- **js_code** – Optional. The variable name created in the Javascript (default response)
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. This will add the component value to the request object
- **profile** – Optional. A flag to set the component performance storage
- **headers** – Optional. The request headers
- **asynchronous** – Async flag: true (asynchronous) or false (synchronous)
- **stringify** – Optional. Stringify the request data for json exchange

post(url: Union[str, JsDataModel], data: Optional[dict] = None, js_code: str = 'response', is_json: bool = True, components: Optional[List[Union[Tuple[HtmlModel, str], HtmlModel]]] = None, profile: Optional[Union[bool, dict]] = None, headers: Optional[dict] = None, asynchronous: bool = False, stringify: bool = True) → XMLHttpRequest

Create a POST HTTP request.

Related Pages:

<https://pythonise.com/series/learning-flask/flask-http-methods>

Parameters

- **url** – The url path of the HTTP request
- **data** – Optional. Corresponding to a JavaScript object
- **js_code** – Optional. The variable name created in the Javascript (default response)
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. This will add the component value to the request object
- **profile** – Optional. A flag to set the component performance storage
- **headers** – Optional. The request headers
- **asynchronous** – Async flag: true (asynchronous) or false (synchronous)
- **stringify** – Optional. Stringify the request data for json exchange

print(content: Union[str, JsDataModel], timer: int = 1000, css_attrs: Optional[dict] = None)

Print a temporary message.

Parameters

- **content** – The content of the popup.
- **timer** – Optional. The time the popup will be displayed.
- **css_attrs** – Optional. The CSS attributes for the popup.

profile(*type: Union[str, JsDataModel], html_code: str, mark: Union[str, JsDataModel], records_count: Optional[int] = None*)

Parameters

- **type** – The type of profile tag.
- **html_code** – The HTML component ID.
- **mark** – The mark reference.
- **records_count** – Optional. The records count.

put(*url: Union[str, JsDataModel], data: Optional[dict] = None, js_code: str = 'response', is_json: bool = True, components: Optional[List[Union[Tuple[HtmlModel, str], HtmlModel]]] = None, profile: Optional[Union[bool, dict]] = None, headers: Optional[dict] = None, asynchronous: bool = False, stringify: bool = True*) → XMLHttpRequest

Create a PUT HTTP request.

Related Pages:

<https://pythonise.com/series/learning-flask/flask-http-methods>

Parameters

- **url** – The url path of the HTTP request
- **data** – Optional. Corresponding to a JavaScript object
- **js_code** – Optional. The variable name created in the Javascript (default response)
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. This will add the component value to the request object
- **profile** – Optional. A flag to set the component performance storage
- **headers** – Optional. The request headers
- **asynchronous** – Async flag: true (asynchronous) or false (synchronous)
- **stringify** – Optional. Stringify the request data for json exchange

querySelector(*selector: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)

The querySelector() method returns the first element that matches a specified CSS selector(s) in the document.

Related Pages:

https://www.w3schools.com/jsref/met_document_queryselector.asp

Parameters

- **selector** – CSS selectors.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

querySelectorAll (*selector: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)
→ JsDomsList

The querySelectorAll() method returns all elements in the document that matches a specified CSS selector(s), as a static NodeList object.

Related Pages:

https://www.w3schools.com/jsref/met_document_queryselectorall.asp

Parameters

- **selector** – CSS selectors.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

queueMicrotask (*js_funcs: Union[List[Union[str, JsDataModel]], str], profile: Optional[Union[bool, dict]] = None*)

The queueMicrotask() method, which is exposed on the Window or Worker interface, queues a microtask to be executed at a safe time prior to control returning to the browser's event loop.

Usage:

```
page.body.onReady([
    page.js.queueMicrotask([page.js.alert("ok")])
])
```

Related Pages:

<https://developer.mozilla.org/fr/docs/Web/API/queueMicrotask>

Parameters

- **js_funcs** – The Javascript function definition
- **profile** – Optional. A flag to set the component performance storage

registerFunction (*func_name: str, js_funcs: Union[str, list], args: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*)

Javascript Framework extension.

Register a predefined Javascript function. This is only dedicated to specific Javascript transformation functions.

Parameters

- **func_name** – The function name.
- **js_funcs** – The Javascript function definition.
- **args** – Optional. Specific Python options available for this component.
- **profile** – Optional. A flag to set the component performance storage.

request_http (*method_type: str, url: str, js_code: str = 'response', is_json: bool = True, components: Optional[List[HtmlModel]] = None*) → XMLHttpRequest

All modern browsers have a built-in XMLHttpRequest object to request data from a server.

Related Pages:

https://www.w3schools.com/xml/xml_http.asp

Usage:

```
page.js.request_http("ajax", "POST", "https://api.cdnjs.com/libraries").
  ↪setHeaders(header).onSuccess([
page.js.alert(rptObj.js.objects.request.get("ajax").responseText)]).
  ↪send(encodeURIData={"search": 'ractive'})
```

Parameters

- **method_type** – The method of the HTTP Request
- **url** – The url path of the HTTP request
- **js_code** – Optional. The variable name created in the Javascript
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. A list of HTML objects values to be passed in the request

Return type

JsObjects.XMLHttpRequest

request_rpc(*js_code: str, method_type: Union[str, JsDataModel], fnc: Callable, url: str, extra_params: Optional[Union[dict, JsDataModel]] = None*) → XMLHttpRequest

Internal RPC to trigger services.

Parameters

- **js_code** – The variable name created in the Javascript.
- **method_type** – The method type
- **fnc** – Python function.
- **url** – The service url
- **extra_params** – Optional.

Return type

JsObjects.XMLHttpRequest

rest(*method: str, url: Union[str, JsDataModel], data: Optional[dict] = None, js_code: str = 'response', is_json: bool = True, components: Optional[List[Union[Tuple[HtmlModel, str], HtmlModel]]] = None, profile: Optional[Union[bool, dict]] = None, headers: Optional[dict] = None, asynchronous: bool = False, stringify: bool = True*) → XMLHttpRequest

Create a POST HTTP request.

Parameters

- **method** – The REST method used
- **url** – The url path of the HTTP request
- **data** – Optional. Corresponding to a JavaScript object
- **js_code** – Optional. The variable name created in the Javascript (default response)
- **is_json** – Optional. Specify the type of object passed
- **components** – Optional. This will add the component value to the request object
- **profile** – Optional. A flag to set the component performance storage
- **headers** – Optional. The request headers
- **asynchronous** – Optional. Async flag: true (asynchronous) or false (synchronous)

- **stringify** – Optional. Stringify the request data for json exchange

return_(data: str) → JsFunction

Javascript return keyword.

Parameters

data – The Javascript expression.

property samples: Samples

JavaScript feature to provide sample of data for a test/demo.

Usage:

```
page.js.samples.months(count_=7)
page.js.samples.numbers(count_=7, min_=-100, max_=100)
```

property screen

The screen object contains information about the visitor's screen.

Related Pages:

https://www.w3schools.com/jsref/obj_screen.asp

serverSentEvent(html_code: Optional[str] = None) → ServerSentEvent

SSE is a native HTML5 feature that allows the server to keep the HTTP connection open and push data changes to the client. Server-sent Streaming is really ideal for server-push notifications, device monitoring and all other tasks that do not require real-time push back from the client.

Related Pages:

<https://medium.com/code-zen/python-generator-and-html-server-sent-events-3cdf14140e56>
https://www.w3schools.com/html/html5_serversentevents.asp https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events

Parameters

html_code – The EventSource id (variable name) on the JavaScript side

socketio(html_code: Optional[str] = None)

This object must be created on the Python side.

The various function will be the one generating the Javascript string. This is just a Python wrapper on top of the library.

Related Pages:

https://www.tutorialspoint.com/socket.io/socket.io_event_handling.htm

Parameters

html_code – Optional. The WebSocket id (variable name) on the JavaScript side

speechRecognition(js_code: str) → SpeechRecognition

The SpeechRecognition interface of the Web Speech API is the controller interface for the recognition service; this also handles the SpeechRecognitionEvent sent from the recognition service.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition>

Usage:

```

page = pk.Page()
rec = page.js.speechRecognition("reco")

test = page.ui.button("Start recording")
test.click([rec.start()])

page.ui.input(html_code="test")

rec.speechend([rec.stop()])
rec.onresult([pk.js_callback("UpdateComponent(transcript, confidence)", page.
↪js.console.log("Done"))])
page.body.onReady([page.js.import_js("test_fnc.js", [], self_contained=True), ↪
↪rec])

# in the module test_fnc.js
function ProcessData(transcript, confidence){
    console.log(transcript); return (transcript == 'hello')}

function UpdateComponent(transcript, confidence){
    var expr = transcript.split(" ");
    if (expr[0] === "put"){document.getElementById(expr[3]).value = expr[1]}}

```

Parameters

js_code – The variable name for the speech recognition object

string(*data*, *js_code*: *Optional[str] = None*, *set_var*: *bool = False*, *is_py_data*: *bool = True*) → JsString

Shortcut to the Javascript String primitives.

Parameters

- **data** – The String data.
- **js_code** – Optional. The specific name to be used for this JavaScript String.
- **set_var** – Optional. Set a variable. Default False.
- **is_py_data** – Optional. Specify the type of data.

switch(*variable*: *Union[str, JsDataModel, HtmlModel]*, *js_conv_func*: *Optional[Union[list, str]] = None*) → JsSwitch

switch statement is used to perform different actions based on different conditions.

Related Pages:

https://www.w3schools.com/js/js_switch.asp

Parameters

- **variable** – Variable on which we will apply the switch
- **js_conv_func** – Optional. A specific JavaScript data conversion function

static title(*text*: *Optional[Union[JsDataModel, str]] = None*, *js_conv_func*: *Optional[Union[list, str]] = None*)

The title property sets or returns the title of the current document (the text inside the HTML title element).

Related Pages:

https://www.w3schools.com/jsref/prop_doc_title.asp

Parameters

- **text** – Optional. Representing the title of the document.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

static typeof(*data: str, var_type: Optional[str] = None*)

The typeof function.

Related Pages:

https://www.w3schools.com/js/js_datatypes.asp

Parameters

- **data** – A String corresponding to a JavaScript object
- **var_type** – Optional. The type of object

property viewHeight

Return the current View port height visible in the browser.

websocket(*html_code: Optional[str] = None, secured: bool = False*)

WebSocket client applications use the WebSocket API to communicate with WebSocket servers using the WebSocket protocol.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications <https://javascript.info/websocket>

Parameters

- **html_code** – Optional. The WebSocket id (variable name) on the JavaScript side
- **secured** – Optional. To define the right protocol for the WebSocket connection we or wss

while_(*condition: Union[str, list], js_funcs: Union[list, str], options: Optional[dict] = None, profile: Optional[Union[bool, dict]] = False*) → JsWhile

The while loop loops through a block of code as long as a specified condition is true.

Related Pages:

https://www.w3schools.com/js/js_loop_while.asp

Parameters

- **condition** – The JavaScript condition
- **js_funcs** – Javascript functions
- **options** – Optional. Specific Python options available for this component
- **profile** – Optional. A flag to set the component performance storage

worker(*html_code: Optional[str] = None, server: bool = False*)

A web worker is a JavaScript running in the background, without affecting the performance of the page.

Related Pages:

https://www.w3schools.com/html/html5_webworkers.asp

Parameters

- **html_code** – Optional. The WebSocket id (variable name) on the JavaScript side
- **server** – Optional. Specify if the page is running on a server

writeln(*value: str*)

The writeln() method is identical to the document.write() method, with the addition of writing a newline character after each statement.

Related Pages:

https://www.w3schools.com/jsref/met_doc_writeln.asp

Parameters

value – What to write to the output stream. Multiple arguments can be listed and they will be appended to the document in order of occurrence

Returns

No return value

5.8.6.2 JavaScript features

5.8.6.2.1 Console

class `epyk.core.js.Js.JsConsole`(*page: Optional[PageModel] = None*)

This is a wrapper to the Console.

Related Pages:

<https://medium.freecodecamp.org/how-to-get-the-most-out-of-the-javascript-console-b57ca9db3e6d>

property `clear`

The `console.clear()` method clears the console.

Usage:

```
page.js.console.clear
```

Related Pages:

https://www.w3schools.com/jsref/met_console_clear.asp

Returns

The Javascript String used to clear the console (F12 in standard browsers).

property `debugger`

Trigger a Javascript debugger from this point. The Javascript will be stopped. It will be possible to check the process step by step in the browser using F12.

Usage:

```
page.js.console.debugger
```

Related Pages:

https://www.w3schools.com/jsref/jsref_debugger.asp

Returns

The Javascript Keyword to trigger the browser debugger.

error(*data: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)

The console.error() method writes an error message to the console.

Related Pages:

https://www.w3schools.com/jsref/met_console_error.asp

Parameters

- **data** – The Javascript fragment.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

The Javascript String used to clear the console (F12 in standard browsers)

info(*data: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)

The console.info() method writes a message to the console.

Related Pages:

https://www.w3schools.com/jsref/met_console_info.asp

Parameters

- **data** – The Javascript fragment.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

The Javascript String used to clear the console (F12 in standard browsers)

log(*data: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None, skip_data_convert: bool = False*)

The console.log() method writes a message to the console.

Usage:

```
page.js.console.log("Test")
```

Related Pages:

https://www.w3schools.com/jsref/met_console_log.asp

Parameters

- **data** – The Javascript fragment.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.
- **skip_data_convert** – Optional. Flag to specify to the framework if a Json conversion is needed.

Returns

The Javascript String used to clear the console (F12 in standard browsers)

perf(*js_code: str, label: Optional[str] = None*)

Shortcut function to display performances from a variable. The variable must be global. Namely the name should start with window.

Parameters

- **js_code** – The variable var name use to compute the performance.
- **label** – Optional. The description.

service(*msg: str, headers: Optional[dict] = None*)

Send logs to the backend.

Parameters

- **msg** – The log message to be sent to the backend.
- **headers** – the service headers.

table(*data: Union[str, JsDataModel], js_header: Optional[list] = None*) → JsFunction

The console.table() method writes a table in the console view.

Related Pages:

https://www.w3schools.com/jsref/met_console_table.asp

Parameters

- **data** – The data to fill the table with.
- **js_header** – Optional. An array containing the names of the columns to be included in the table.

Returns

The Javascript String used to clear the console (F12 in standard browsers).

time(*html_code: Union[str, JsDataModel]*) → JsNumber

The console.time() method starts a timer in the console view.

Related Pages:

https://www.w3schools.com/jsref/met_console_time.asp

Parameters

html_code – Use the label parameter to give the timer a name.

Returns

A Python Javascript Number.

timeEnd(*html_code: Union[str, JsDataModel]*)

The console.timeEnd() method ends a timer, and writes the result in the console view.

Related Pages:

https://www.w3schools.com/jsref/met_console_timeend.asp

Parameters

html_code – The name of the timer to end.

Returns

The Javascript String used to clear the console (F12 in standard browsers).

tryCatch(*js_funcs*: Union[str, list], *js_funcs_errs*: Union[str, list] = 'console.warn(err.message)', *profile*: Optional[Union[bool, dict]] = False)

Javascript Try Catch Exceptions.

Related Pages:

https://www.w3schools.com/jsref/jsref_obj_error.asp

Parameters

- **js_funcs** – The Javascript functions.
- **js_funcs_errs** – The Javascript functions.
- **profile** – Optional. A flag to set the component performance storage.

Returns

The Javascript String used to clear the console (F12 in standard browsers)

warn(*data*: Union[str, JsDataModel], *js_conv_func*: Optional[Union[list, str]] = None)

The console.warn() method writes a warning to the console.

Related Pages:

https://www.w3schools.com/jsref/met_console_warn.asp

Parameters

- **data** – The Javascript fragment.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

The Javascript String used to clear the console (F12 in standard browsers)

5.8.6.2.2 Json

class epyk.core.js.Js.JsJson

Wrapper around the Javascript Json module.

This wrapper will only wrapper the different functions available in the underlying library. The documentation can be found in each function or are available on the Javascript Official documentation.

Related Pages:

https://www.w3schools.com/js/js_json_intro.asp

parse(*data*: Union[str, JsDataModel], *js_result_func*: Optional[str] = None, *js_conv_func*: Optional[Union[list, str]] = None)

Parses a JSON string and returns a JavaScript object.

Related Pages:

https://www.w3schools.com/js/js_json_parse.asp

https://www.w3schools.com/jsref/jsref_parse_json.asp

Parameters

- **data** – A String corresponding to a JavaScript object.

- **js_result_func** – Optional. A function used to transform the result. The function is called for each item. Any nested objects are transformed before the parent.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

The Javascript string method

stringify(*data: Union[str, JsDataModel], replacer=None, space: int = 0, js_conv_func: Optional[Union[list, str]] = None*)

The JSON.stringify() method converts JavaScript objects into strings.

Related Pages:

https://www.w3schools.com/js/js_json_stringify.asp

Parameters

- **data** – The value to convert to a string.
- **replacer** – Optional. Either a function or an array used to transform the result. The replacer is called for each item.
- **space** – Optional. Either a String or a Number. A string to be used as white space (max 10 characters), or a Number, from 0 to 10, to indicate how many space characters to use as white space.
- **js_conv_func** – Optional. A specific JavaScript data conversion function.

Returns

The Javascript string method.

5.8.6.2.3 BreadCrumb

class `epyk.core.js.Js.JsBreadCrumb`(*src: Optional[PageModel] = None*)

add(*key: str, data: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)

Add an entry to the Javascript breadcrumb dictionary.

Parameters

- **key** – The key in the Breadcrumb dictionary
- **data** – A String corresponding to a JavaScript object
- **js_conv_func** – Optional. A specific JavaScript data conversion function

get(*key: Optional[str] = None*)

returns the object stored in the breadcrumb dictionary.

Parameters

key – Optional. The key in the Breadcrumb dictionary

Returns

A Python object.

hash(*data: Union[str, JsDataModel], js_conv_func: Optional[Union[list, str]] = None*)

Add an anchor to the URL after the hashtag.

Related Pages:

https://www.w3schools.com/jsref/prop_loc_hash.asp

Parameters

- **data** – A String corresponding to a JavaScript object
- **js_conv_func** – Optional. A specific JavaScript data conversion function

property url

Get the full URL.

5.8.6.2.4 Screen

class epyk.core.js.Js.JsScreen

property availHeight: JsNumber

The availHeight property returns the height of the user's screen, in pixels, minus interface features like the Windows Task bar.

Related Pages:

https://www.w3schools.com/jsref/prop_screen_availheight.asp

property availWidth: JsNumber

The availWidth property returns the width of the user's screen, in pixels, minus interface features like the Windows Task bar.

Related Pages:

https://www.w3schools.com/jsref/prop_screen_availwidth.asp

property colorDepth: JsNumber

The colorDepth property returns the bit depth of the color palette for displaying images (in bits per pixel).

Related Pages:

https://www.w3schools.com/jsref/prop_screen_colordepth.asp

property height: JsNumber

The height property returns the total height of the user's screen, in pixels.

Related Pages:

https://www.w3schools.com/jsref/prop_screen_height.asp

property pixelDepth: JsNumber

The pixelDepth property returns the color resolution (in bits per pixel) of the visitor's screen.

Related Pages:

https://www.w3schools.com/jsref/prop_screen_pixeldepth.asp

property width: JsNumber

The width property returns the total width of the user's screen, in pixels.

Related Pages:

https://www.w3schools.com/jsref/prop_screen_width.asp

5.8.6.2.5 Maths

class epyk.core.js.Js.JsMaths.JsMaths

Wrapper for the Javascript Math module

Related Pages:

https://www.w3schools.com/jsref/jsref_obj_math.asp

property E: JsNumber

The E property returns the Euler's number and the base of natural logarithms, approximately 2.718.

Usage:

```
page.js.math.E
```

Related Pages:

https://www.w3schools.com/jsref/jsref_e.asp

Returns

Returns Euler's number (approx. 2.718)

property LN10

The LN10 property returns the natural logarithm of 10, approximately 2.302.

Usage:

```
jsObj.math.LN10
```

Related Pages:

https://www.w3schools.com/jsref/jsref_ln10.asp

Returns

Returns the natural logarithm of 10 (approx. 2.302)

property LN2

The LN2 property returns the natural logarithm of 2, approximately 0.693.

Usage:

```
jsObj.math.LN2
```

Related Pages:

https://www.w3schools.com/jsref/jsref_ln2.asp

Returns

Returns the natural logarithm of 2 (approx. 0.693)

property LOG2E

The LOG2E property returns the base-2 logarithm of E, approximately 1.442

Usage:

```
jsObj.math.LOG2E
```


Related Pages:

https://www.w3schools.com/jsref/jsref_log2e.asp

Returns

Returns the base-2 logarithm of E (approx. 1.442)

property PI

The PI property returns the ratio of a circle's area to the square of its radius, approximately 3.14.

Related Pages:

https://www.w3schools.com/jsref/jsref_pi.asp

property SQRT1_2

The SQRT1_2 property returns the square root of 1/2, approximately 0.707.

Usage:

```
jsObj.math.SQRT1_2
```

Related Pages:

https://www.w3schools.com/jsref/jsref_sqrt1_2.asp

Returns

Returns the square root of 1/2 (approx. 0.707)

property SQRT2: `<module 'epyk.core.js.primitives.JsNumber' from
'../epyk/core/js/primitives/JsNumber.py'>`

The SQRT2 property returns the square root of 2, approximately 1.414.

Usage:

```
jsObj.math.SQRT2
```

Related Pages:

https://www.w3schools.com/jsref/jsref_sqrt2.asp

Returns

Returns the square root of 2 (approx. 1.414)

abs(*number*: *Union[float, JsDataModel]*)

The abs() method returns the absolute value of a number.

Related Pages:

https://www.w3schools.com/jsref/jsref_abs.asp

Parameters

number (*Union[infloatt, primitives.JsDataModel]*) – A number.

Returns

Returns the absolute value of x.

ceil(*number*: *Union[float, JsDataModel]*)

The ceil() method rounds a number UPWARDS to the nearest integer, and returns the result.

Usage:

```
jsObj.math.ceil(jsObj.objects.number.get("MyNumber"))
```

Related Pages:

https://www.w3schools.com/jsref/jsref_ceil.asp

Parameters

number (*Union[float, primitives.JsDataModel]*) – The number you want to round.

Returns

Returns x, rounded upwards to the nearest integer.

cos(*number*: *Union[float, JsDataModel]*)

The acos() method returns the cosine of a number as a value value between 0 and PI radians.

Related Pages:

https://www.w3schools.com/jsref/jsref_cos.asp

Parameters

number (*Union[float, primitives.JsDataModel]*) – Returns the cosine of x (x is in radians).

Returns

A Number, from -1 to 1, representing the cosine of an angle, or NaN if the value is empty.

exp(*number*: *Union[float, JsDataModel]*)

The exp() method returns the value of Ex, where E is Euler's number (approximately 2.7183) and x is the number passed to it.

Related Pages:

https://www.w3schools.com/jsref/jsref_exp.asp

Parameters

number (*Union[float, primitives.JsDataModel]*) – Number. Required. A number,

Returns

Returns the value of exponential of x,

floor(*number*: *Union[float, JsDataModel]*)

The floor() method rounds a number DOWNWARDS to the nearest integer, and returns the result.

Usage:

```
jsObj.math.floor(13.566)
```

Related Pages:

https://www.w3schools.com/jsref/jsref_floor.asp

Parameters

number (*Union[float, primitives.JsDataModel]*) – Required. The number you want to round.

Returns

A Number, representing the nearest integer when rounding downwards

log(*number*: *Union[float, JsDataModel]*)

The log() method returns the natural logarithm (base E) of a number.

Related Pages:

https://www.w3schools.com/jsref/jsref_log.asp

Parameters

number (*Union[float, primitives.JsDataModel]*) – Number. Required. A number.

Returns

Returns the natural logarithm (base E) of x.

max(**args*)

The max() method returns the number with the highest value.

Usage:

```
jsObj.math.max(10, 45, 100, -3, 56)
```

Related Pages:

https://www.w3schools.com/jsref/jsref_max.asp
calculate-the-max-min-value-from-an-array/

<https://www.jstips.co/en/javascript/>

Parameters

args – Optional. One or more numbers to compare.

Returns

A Number, representing the highest number of the arguments, or -Infinity if no arguments are given, or NaN

if one or more arguments are not numbers

min(**args*)

The min() method returns the number with the lowest value.

Usage:

```
jsObj.math.min(10, 45, 100, -3, 56)
```

Related Pages:

https://www.w3schools.com/jsref/jsref_min.asp

Parameters

args – Optional. One or more numbers to compare.

Returns

A Number, representing the lowest number of the arguments, or Infinity

if no arguments are given, or NaN if one or more arguments are not numbers

static pow(*number*: Union[JsDataModel, float], *power*: Union[JsDataModel, int])

The pow() method returns the value of x to the power of y (xy).

Usage:

```
jsObj.objects.number.new(23.6, varName="MyNumber")
jsObj.math.pow(jsObj.objects.number.get("MyNumber"), 2)
```

Related Pages:

https://www.w3schools.com/jsref/jsref_pow.asp

Parameters

- **number** (Union[float, primitives.JsDataModel]) – The base.
- **power** (Union[int, primitives.JsDataModel]) – The exponent.

Returns

Returns the value of x to the power of y.

random(*min_val*: Union[int, JsDataModel] = 0, *max_val*: Union[int, JsDataModel] = 1)

Math.random() returns a random number between 0 (inclusive), and 1 (exclusive):

Usage:

```
page.js.math.random()
jsObj.math.random(10, 100)
```

Related Pages:

https://www.w3schools.com/js/js_random.asp

Parameters

- **min_val** (Union[int, primitives.JsDataModel]) – Optional The minimum value for the random function.
- **max_val** (Union[int, primitives.JsDataModel]) – Optional The maximum value for the random function.

Returns

A Number, representing a number from 0 up to but not including 1.

round(*number*: Union[float, JsDataModel])

The round() method rounds a number to the nearest integer.

Note: 2.49 will be rounded down (2), and 2.5 will be rounded up (3).

Usage:

```
jsObj.objects.number.new(23.6, varName="MyNumber")
jsObj.math.round(jsObj.objects.number.get("MyNumber"))
```

Related Pages:

https://www.w3schools.com/jsref/jsref_round.asp

Parameters

- **number** (Union[float, primitives.JsDataModel]) – The number to be rounded.

Returns

Rounds x to the nearest integer.

sin(*number*: *Union[float, JsDataModel]*)

The sin() method returns the sinus of a number as a value value between 0 and PI radians.

Related Pages:

https://www.w3schools.com/jsref/jsref_sin.asp

Parameters

number (*Union[float, primitives.JsDataModel]*) – Returns the sinus of x (x is in radians).

Returns

Number. from -1 to 1, representing the sine of an angle, or NaN if the value is empty.

sqrt(*number*: *Union[float, JsDataModel]*)

The sqrt() method returns the square root of a number.

Usage:

```
jsObj.objects.number.new(23.6, varName="MyNumber")
jsObj.math.sqrt(jsObj.objects.number.get("MyNumber"))
```

Related Pages:

https://www.w3schools.com/jsref/jsref_sqrt.asp

Parameters

number (*Union[float, primitives.JsDataModel]*) – A number.

Returns

A Number. If x is a negative number, NaN is returned.

trunc(*number*: *Union[float, JsDataModel]*)

The trunc() method returns the integer part of a number.

Usage:

```
page.js.math.trunc(rptObj.js.math.SQRT2)
```

Related Pages:

https://www.w3schools.com/jsref/jsref_trunc.asp

Parameters

number (*Union[float, primitives.JsDataModel]*) – Number. Required. A number.

Returns

Returns the integer part of a number (x).

5.8.6.2.6 Location

class epyk.core.js.Js.JsLocation.**JsLocation**

JavaScript Location module.

classmethod **assign**(*url: Union[str, JsDataModel]*) → JsFunction

The assign() method loads a new document.

Related Pages:

https://www.w3schools.com/jsref/met_loc_assign.asp

Parameters

url – Specifies the URL of the page to navigate to

classmethod **download**(*url: Union[str, JsDataModel], name: Union[str, JsDataModel] = 'download'*) → JsVoid

Download data from the url.

Parameters

- **url** – The url of the image
- **name** – Optional. The name of the file

classmethod **getUrlFromArrays**(*data: Union[list, JsDataModel], delimiter: Union[str, JsDataModel] = ',' , charset: str = 'utf-8', end_line: Union[str, JsDataModel] = '\r\n'*)

Convert data to a URL.

Parameters

- **data** – A JavaScript array
- **delimiter** – Optional. The column delimiter
- **charset** – Optional.
- **end_line** – Optional.

classmethod **getUrlFromData**(*data: Union[dict, JsDataModel], options: Optional[Union[dict, JsDataModel]] = None*)

Convert data to a URL.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Blob>

Parameters

- **data** – Input data to be converted
- **options** – Optional. Blob definition properties

property hash: JsObject

The hash property sets or returns the anchor part of a URL, including the hash sign (#).

Usage:

<code>jsObj.location.hash</code>

Related Pages:

https://www.w3schools.com/jsref/prop_loc_hash.asp

Returns

A String, representing the anchor part of the URL, including the hash sign (#).

property host: JsString

The host property sets or returns the hostname and port of a URL.

Usage:

```
jsObj.location.host
```

Related Pages:

https://www.w3schools.com/jsref/prop_loc_host.asp

Returns

Return the hostname and port of the current URL.

property hostname: JsString

The hostname property sets or returns the hostname of a URL.

Usage:

```
page.location.hostname
```

Related Pages:

https://www.w3schools.com/jsref/obj_location.asp

Returns

Return the hostname property.

classmethod href(*href: Optional[Union[JsDataModel, str]] = None, secured: bool = False*) → JsObject

The href property sets or returns the entire URL of the current component.

Usage:

```
page.js.location.href("https://www.w3schools.com/howto/howto_js_fullscreen.asp")
```

Related Pages:

https://www.w3schools.com/jsref/prop_loc_href.asp

Parameters

- **href** – Optional. Set the href property
- **secured** – Optional. The secured flag

Returns

A String, representing the entire URL of the page, including the protocol (like <http://>).

mail(*mails: List[str], subject: str, body: str*)

The mailto link when clicked opens users default email program or software. A new email page is created with “To” field containing the address of the name specified on the link by default.

Usage:

```
page.js.location.mail(["test@gmail.com"], "This is a test", "This is the email  
↪ 's content")
```

Related Pages:

http://www.tutorialspark.com/html5/HTML5_email_mailto.php

Parameters

- **mails** – The email addresses
- **subject** – The email's subject
- **body** – The email's content

Returns

The Javascript string.

classmethod open_new_tab(url: Union[str, JsDataModel], name: Union[str, JsDataModel] = '_blank', specs: Optional[Union[JsDataModel, str]] = None, replace: Optional[Union[JsDataModel, str]] = None, window_id: str = 'window', data: Optional[dict] = None, secured: bool = False) → JsFunction

Opens a new browser window in a new tab (duplicated but part of the Window module).

Usage:

```
page.js.location.open_new_tab("www.google.fr")
```

Related Pages:

https://www.w3schools.com/Jsref/met_win_open.asp

Parameters

- **url** – Optional. Specifies the URL of the page to open. If no URL is specified, a new window/tab with `about:blank` is opened
- **name** – Optional. Specifies the target attribute or the name of the window. Default `_blank`
- **specs** – Optional. A comma-separated list of items, no whitespaces
- **replace** – Optional. Specifies whether the URL creates a new entry or replaces the current entry in the history list
- **window_id** – Optional. The JavaScript window object
- **data** – Optional. The url parameters
- **secured** – Optional. The secure flag

property origin: JsString

The origin property returns the protocol, hostname and port number of a URL.

Usage:

```
page.js.location.origin + page.js.location.pathname
```

Related Pages:

https://www.w3schools.com/jsref/prop_loc_origin.asp

Returns

A String, representing the protocol (including `://`), the domain name (or IP address) and port number (including the colon sign `:`) of the URL. For URL's using the "file:" protocol, the return value differs between browser)

property pathname: JsString

The hostname property sets or returns the hostname of a URL.

Usage:

```
jsObj.location.pathname
```

Related Pages:

https://www.w3schools.com/jsref/obj_location.asp

Returns

Return the pathname property.

property port: JsString

The port property sets or returns the port number the server uses for a URL.

Related Pages:

https://www.w3schools.com/jsref/prop_loc_port.asp

Returns

A String, representing the port number of a URL.

classmethod postTo(url: str, data: dict, method: str = 'POST', target: str = '_blank')

This method will create an internal form and submit the response exactly like a post of a form to another page.

Related Pages:

https://www.w3schools.com/jsref/dom_obj_form.asp

Parameters

- **url** – The target url
- **data** – The url parameters
- **method** – Optional. The method used to send the data. Default POST
- **target** – Optional. Target method to access the new page

classmethod reload(force_get: bool = False)

The reload() method is used to reload the current document.

The reload() method does the same as the reload button in your browser.

Related Pages:

https://www.w3schools.com/jsref/met_loc_reload.asp

Parameters

force_get – Optional. Specifies the type of reloading: false - Default. Reloads the current page from the cache true - Reloads the current page from the server

classmethod `replace(url: Union[str, JsDataModel], secured: bool = False) → JsFunction`

The `replace()` method replaces the current document with a new one.

The difference between this method and `assign()`, is that `replace()` removes the URL of the current document from the document history, meaning that it is not possible to use the “back” button to navigate back to the original document.

Related Pages:

https://www.w3schools.com/jsref/met_loc_replace.asp

Parameters

- **url** – Specifies the URL of the page to navigate to
- **secured** – Optional. If the http is missing. This will be used to fix the url

property `search: JsString`

The `search` property sets or returns the querystring part of a URL, including the question mark (?).

Related Pages:

https://www.w3schools.com/jsref/prop_loc_search.asp

Returns

A String, representing the querystring part of a URL, including the question mark (?).

property `urlSearchParams: URLSearchParams`

The `URLSearchParams()` constructor creates and returns a new `URLSearchParams` object.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams/URLSearchParams>

class `epyk.core.js.Js.JsLocation.URLSearchParams(query: str)`

append(*key: Union[str, JsDataModel], value: Any*)

Append a key, value to the url parameter object.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams>

Parameters

- **key** – The url parameter
- **value** – The value to be appended to the URL

delete(*key*)

The `delete()` method of the `URLSearchParams` interface deletes the given search parameter and all its associated values, from the list of all search parameters.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams/delete>

Parameters

key –

get(key: str, default: Optional[Any] = None)

Get the value of a request parameter in the url.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams>

Parameters

- **key** – The url parameter
- **default** – Optional. The default value

getAll(key: Union[str, JsDataModel])

Get all the values of a request parameter in the url.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams>

Parameters

- key** – The url parameter

has(key: Union[str, JsDataModel])

Check if a given parameter is in the url.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams>

Parameters

- key** – The url parameter

set(key: str, value: Any)

Set the value of a request parameter in the url.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams>

Parameters

- **key** – The url parameter
- **value** – The value to set

5.8.6.2.7 Navigator

5.8.6.2.8 Window

`epyk.core.js.Js.JsWindow`

alias of <module 'epyk.core.js.JsWindow' from './epyk/core/js/JsWindow.py'>

5.8.6.2.9 WebWorkers

Web Workers are a simple means for web content to run scripts in background threads. The worker thread can perform tasks without interfering with the user interface.

It is possible to create and to use web worker with Epyk. To do so it is possible to use them in a dedicated page or in a Jupyter Notebook:

```
w2 = page.js.worker()
w2.connect(content='')
self.addEventListener('message', function(e) {
    var data = e.data; console.log(data);
    switch (data.cmd) {
        case 'add':
            self.postMessage('Result: ' + (data.value1 + data.value2 + data.value3)); break;
        case 'mult':
            self.postMessage('Result: ' + (data.value1 * data.value2 * data.value3)); break;
        case 'stop':
            self.postMessage('WORKER STOPPED: ' + data.msg + '. (buttons will no longer work)');
            self.close(); break;
        default:
            self.postMessage('Unknown command: ' + data.msg);
    };
}, false);
''')

slider = page.ui.slider()
number = page.ui.fields.number()

div = page.ui.div()
page.ui.button("Add").click([w2.postMessage({'cmd': 'add', 'value1': 2},
components=[(slider, "value2"), (number, "value3")])])
page.ui.button("Mult").click([w2.postMessage({'cmd': 'mult', 'value1': 5},
components=[(slider, "value2"), (number, "value3")])])

page.ui.button("Stop worker").click([w2.postMessage({'cmd': 'stop'})])
```

More details on the web workers are available in the functions documentation.

5.8.6.2.10 WebSocket

class epyk.core.js.Js.JsWebSocket.**WebSocket**(*html_code: Optional[str] = None, src: Optional[Union[str, PageModel]] = None, secured: bool = False*)

close(*code: int = 1000, reason: Optional[Union[JsDataModel, str]] = None*)

When you've finished using the WebSocket connection, call the WebSocket method close()

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications

Parameters

- **code** (*int*) – Optional. The HTTP code to be sent to the server for the closure
- **reason** (*Optional[str]*) – Optional. The message to be sent to the server for the closure

connect (*url: Optional[str] = None, port: Optional[int] = None, protocol: Optional[Union[list, str]] = None, from_config=None*)

In order to communicate using the WebSocket protocol, you need to create a WebSocket object; this will automatically attempt to open the connection to the server.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications

Parameters

- **url** (*str*) – The URL to which to connect; this should be the URL to which the WebSocket server will respond. This should use the URL scheme wss://, although some software may allow you to use the insecure ws:// for local connections.
- **port** (*Optional[int]*) – The application port number.
- **protocol** (*Union[list, str]*) – Either a single protocol string or an array of protocol strings.
- **from_config** –

property http_codes

To get connection state, additionally there's socket.readyState property with values:

Related Pages:

<https://tools.ietf.org/html/rfc6455#section-7.4.1>

property message

Fired when data is received through a WebSocket. Also available via the onmessage property.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/WebSocket/message_event

onclose (*js_funcs: Optional[Union[list, str]], profile: Optional[Union[bool, dict]] = None*)

Fired when a connection with a WebSocket is closed. Also available via the onclose property.

Related Pages:

<https://javascript.info/websocket>

Parameters

- **js_funcs** (*Optional[Union[list, str]]*) – Javascript functions.
- **profile** (*Optional[Union[dict, bool]]*) – Optional. A flag to set the component performance storage.

onerror (*js_funcs: Optional[Union[list, str]], profile: Optional[Union[bool, dict]] = None*)

Fired when a connection with a WebSocket has been closed because of an error, such as when some data couldn't be sent. Also available via the onerror property.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications

Parameters

- **js_funcs** (*Optional[Union[list, str]]*) – Javascript functions.
- **profile** (*Optional[Union[dict, bool]]*) – Optional. A flag to set the component performance storage.

onmessage(*js_funcs: Optional[Union[list, str]], profile: Optional[Union[bool, dict]] = None*)

Fired when data is received through a WebSocket. Also available via the onmessage property.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications

Parameters

- **js_funcs** (*Optional[Union[list, str]]*) – Javascript functions.
- **profile** (*Optional[Union[dict, bool]]*) – Optional. A flag to set the component performance storage.

onopen(*js_funcs: Optional[Union[list, str]], profile: Optional[Union[bool, dict]] = None*)

Fired when a connection with a WebSocket is opened. Also available via the onopen property.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications

Parameters

- **js_funcs** (*Optional[Union[list, str]]*) – Javascript functions.
- **profile** (*Optional[Union[dict, bool]]*) – Optional. A flag to set the component performance storage.

property readyState

The WebSocket.readyState read-only property returns the current state of the WebSocket connection.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/WebSocket/readyState>

receive(*js_funcs: Optional[Union[list, str]], profile: Optional[Union[bool, dict]] = None*)

Fired when data is received through a WebSocket. Also available via the onmessage property.

Parameters

- **js_funcs** (*Optional[Union[list, str]]*) – Javascript functions.
- **profile** (*Optional[Union[dict, bool]]*) – Optional. A flag to set the component performance storage.

reconnect()

send(*data*)

Basic way to send a text message to the server.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications

Parameters

data – String. The message to be sent

sendText(*components: List[HtmlModel], attrs: Optional[dict] = None*)

Send a complex message from components.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications

Parameters

- **components** (*List[primitives.HtmlModel]*) – The list of HTML components (it will get the dom.content automatically)
- **attrs** (*dict*) – Optional. Attach some static attributes to the request

property states

To get connection state, additionally there's socket.readyState property with values:

Related Pages:

<https://javascript.info/websocket>

5.8.6.2.11 Performance

class epyk.core.js.Js.JsPerformance.**JsPerformance**(*page: Optional[PageModel] = None*)

add_profiling(*js_funcs: Optional[Union[list, str]]*)

Wrap the Javascript functions with function to asset on the execution time.

Usage:

```
page.js.performance.add_profiling(fncs['content'])
```

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Performance/now>

Parameters

js_funcs (*Optional[Union[list, str]]*) – The Javascript functions.

Returns

The profile variable name

clearMarks(*name: Optional[str] = None*)

The clearMarks() method removes the named mark from the browser's performance entry buffer. If the method is called with no arguments, all performance entries with an entry type of "mark" will be removed from the performance entry buffer.

Usage:

```
performance.clearMarks("a")
```

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Performance/clearMarks>

Parameters

name (*Optional[str]*) – Optional. The mark name.

Returns

Void, the String for the Javascript side

clearMeasures(*name: Optional[str] = None*)

The clearMeasures() method removes the named measure from the browser's performance entry buffer. If the method is called with no arguments, all performance entries with an entry type of "measure" will be removed from the performance entry buffer.

Usage:

```
performance.clearMeasures("a");
```

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Performance/clearMeasures>

Parameters

name (*Optional[str]*) – Optional. The name of the mark to be cleared.

Returns

Void, the String for the Javascript side

clearResourceTimings()

The clearResourceTimings() method removes all performance entries with an entryType of "resource" from the browser's performance data buffer and sets the size of the performance data buffer to zero. To set the size of the browser's performance data buffer, use the Performance.setResourceTimingBufferSize() method.

Usage:

```
performance.clearResourceTimings()
```

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Performance/clearResourceTimings>

Returns

This method has no return value, but only the String for the Javascript side

getEntries()

The `getEntries()` method returns a list of all `PerformanceEntry` objects for the page. The list's members (entries) can be created by making performance marks or measures (for example by calling the `mark()` method) at explicit points in time. If you are only interested in performance entries of certain types or that have certain names, see `getEntriesByType()` and `getEntriesByName()`.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Performance/getEntries>

Returns

An array of `PerformanceEntry` objects

getEntriesByName(*name: Union[JsDataModel, str]*, *entry_type: Optional[str] = None*)

The `getEntriesByName()` method returns a list of `PerformanceEntry` objects for the given name and type. The list's members (entries) can be created by making performance marks or measures (for example by calling the `mark()` method) at explicit points in time.

Usage:

```
performance.getEntriesByName("Begin", "mark")
```

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Performance/getEntriesByName>

Parameters

- **name** (*Union[primitives.JsDataModel, str]*) – The name of the entry to retrieve.
- **entry_type** (*Optional[str]*) – Optional. The type of entry to retrieve such as “mark”.

Returns

A list of `PerformanceEntry` objects that have the specified name and type

getEntriesByType(*entry_type: str*)

The `getEntriesByType()` method returns a list of `PerformanceEntry` objects for a given type. The list's members (entries) can be created by making performance marks or measures (for example by calling the `mark()` method) at explicit points in time.

Usage:

```
performance.getEntriesByType("mark")
```

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Performance/getEntriesByType>

Parameters

entry_type (*str*) – The type of entry to retrieve such as “mark”.

Returns

A list of `PerformanceEntry` objects that have the specified type.

mark(*name: Union[JsDataModel, str]*)

The `mark()` method creates a timestamp in the browser's performance entry buffer with the given name. The application defined timestamp can be retrieved by one of the Performance interface's `getEntries*()` methods (`getEntries()`, `getEntriesByName()` or `getEntriesByType()`).

Usage:

```
performance.mark("a")
```

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Performance/mark>

Parameters

name (*Union[primitives JsDataModel, str]*) – A DOMString representing the name of the mark.

Returns

Void, The String for the Javascript side.

measure (*name: Union[JsDataModel, str], start_mark: Optional[str] = None, end_mark: Optional[str] = None*)

The measure() method creates a named timestamp in the browser's performance entry buffer between marks, the navigation start time, or the current time.

When measuring between two marks, there is a start mark and end mark, respectively. The named timestamp is referred to as a measure.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Performance/measure>

Parameters

- **name** (*str*) – A DOMString representing the name of the measure.
- **start_mark** (*Optional[str]*) – Optional. A DOMString representing the name of the measure's starting mark.
- **end_mark** (*Optional[str]*) – Optional, A DOMString representing the name of the measure's ending mark.

Returns

Void, The String for the Javascript side

property now

The performance.now() method returns a DOMHighResTimeStamp, measured in milliseconds.

Usage:

```
var t0 = performance.now();
```

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Performance/now>

Returns

A Javascript Number

setResourceTimingBufferSize (*max_size: int*)

The setResourceTimingBufferSize() method sets the browser's resource timing buffer size to the specified number of "resource" performance entry type objects.

Usage:

```
performance.setResourceTimingBufferSize(maxSize)
```

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Performance/setResourceTimingBufferSize>

Parameters

max_size (*int*) – The buffer maximum size.

Returns

Void, the String for the Javascript side.

toJSON()

The toJSON() method of the Performance interface is a standard serializer: it returns a JSON representation of the performance object's properties.

Usage:

```
performance.toJSON()
```

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/API/Performance/toJSON>

Returns

A JSON object that is the serialization of the Performance object.

5.8.7 Standards

5.8.7.1 Header Module

class epyk.core.html.Header.**Header**(*page: Optional[PageModel] = None*)

add_code(*code: str, attrs: Optional[dict] = None*)

Add a JavaScript tag to the HTML page.

The code will be added in a script tag.

Parameters

- **code** – The JavaScript code to be added to the page
- **attrs** – optional. The various attributes to be added to the script tag

add_script(*src: str, attrs: Optional[dict] = None*)

Add a JavaScript tag to the HTML page.

The script will be added in a script tag.

Parameters

- **src** – The script path added to the page
- **attrs** – optional. The various attributes to be added to the script tag

base(url: str)

Specify a dedicated path for the relative paths in the page.

Basically the images will use this path as base if present in the page.

Related Pages:

https://www.w3schools.com/tags/tag_base.asp

Parameters

url – The url path

dev(icon: Optional[str] = None)

Change the tab icon to highlight this page is still in dev mode.

Usage:

```
page = Report()
page.headers.dev()
```

Parameters

icon – Optional. The url path of the icon

favicon(url: str, rel: str = 'icon', sizes: Optional[str] = None, img_type: Optional[str] = None)

The <link> tag defines a link between a document and an external resource.

The <link> tag is used to link to external style sheets.

Usage:

```
rptObj.headers.favicon('https://github.com/favicon.ico')
```

Related Pages:

<https://developer.mozilla.org/fr/docs/Web/HTML/Element/link> https://www.w3schools.com/tags/tag_link.asp

Parameters

- **url** – The url full path
- **rel** – Optional
- **sizes** – Optional
- **img_type** – Optional

property icons

Property to defined / add more icons to the page header. Some browsers (like Safari or Opera) could require specify tags in the page.

property links

The various HTML page header links.

Related Pages:

https://www.w3schools.com/jsref/dom_obj_link.asp <https://developer.mozilla.org/fr/docs/Web/HTML/Element/link>

property meta: Meta

Property to the Meta data dictionary for the HTML page.

Metadata is data (information) about data.

The <meta> tag provides metadata about the HTML document. Metadata will not be displayed on the page, but will be machine parsable.

Usage:

```
page.headers.meta
```

Related Pages:

https://www.w3schools.com/tags/tag_meta.asp

title(value: str)

The <title> tag is required in all HTML documents and it defines the title of the document.

You can NOT have more than one <title> element in an HTML document.

Usage:

```
page.headers.title("title")
```

Related Pages:

https://www.w3schools.com/tags/tag_title.asp

Parameters

value – The title value for the page

class epyk.core.html.Header.Icons(header)**apple_touch_icon(url: str, sizes: Optional[str] = None)**

Related Pages:

https://developer.apple.com/library/archive/documentation/AppleApplications/Reference/SafariWebContent/ConfiguringWebApplications/ConfiguringWebApplications.html#//apple_ref/doc/uid/TP40002051-CH3-SW4

Parameters

- **url** – The path for the svg file
- **sizes** – Optional. The size for 25x25

apple_touch_startup_icon(url: str, sizes: Optional[str] = None)

Related Pages:

https://developer.apple.com/library/archive/documentation/AppleApplications/Reference/SafariWebContent/ConfiguringWebApplications/ConfiguringWebApplications.html#//apple_ref/doc/uid/TP40002051-CH3-SW4

Parameters

- **url** – The path for the svg file
- **sizes** – Optional. The size for 25x25

gif(*url: str, sizes: Optional[str] = None*)

Related Pages:

https://www.w3schools.com/tags/att_link_sizes.asp

Parameters

- **url** – The path for the gif
- **sizes** – Optional. The size in a format 25x25

icon(*url: str, sizes: Optional[str] = None, img_type: Optional[str] = None*)

Set the icon for the page.

Related Pages:

https://developer.mozilla.org/fr/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types

Parameters

- **url** – The url of the icon on the server
- **sizes** – Optional. The size of the icon
- **img_type** – Optional. The type of picture

svg(*url: str, sizes: Optional[str] = None*)

Related Pages:

https://www.w3schools.com/tags/att_link_sizes.asp

Parameters

- **url** – The path for the svg file
- **sizes** – Optional. The size for 25x25

class epyk.core.html.Header.**Links**(*header*)

alternative(*href: str, file_type: str = 'text/css', media: Optional[str] = None*)

Specifying alternative style sheets in a web page provides a way for users to see multiple versions of a page, based on their needs or preferences.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link>

Parameters

- **href** – The link path for the stylesheet page
- **file_type** – Optional. The type of the href tag
- **media** – Optional. This resource will then only be loaded if the media condition is true

dns_prefetch(*href: str, media: Optional[str] = None, cross_origin: bool = False*)

The dns-prefetch keyword for the rel attribute of the <link> element is a hint to browsers that the user is likely to need resources from the target resource's origin, and therefore the browser can likely improve the user experience by preemptively performing DNS resolution for that origin.

Related Pages:

<https://www.keycdn.com/blog/resource-hints> https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types/dns-prefetch

Parameters

- **href** – The link path for the stylesheet page
- **media** – Optional. This resource will then only be loaded if the media condition is true
- **cross_origin** – Optional. Specify to the browser to enable the cross origin to get resource from different website.

icon(*href: str, cross_origin: bool = False*)

Defines a resource for representing the page in the user interface, usually an icon (auditory or visual). In the browser, it is usually referred to as the favicon.

If there are multiple `<link rel="icon">`s, the browser uses their media, type, and sizes attributes to select the most appropriate icon. If several icons are equally appropriate, the last one is used. If the most appropriate icon is later found to be inappropriate, for example because it uses an unsupported format, the browser proceeds to the next-most appropriate, and so on.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types

Parameters

- **href** – The link path for the stylesheet page
- **cross_origin** – Optional. Specify to the browser to enable the cross origin to get resource from different

website.

imports(*href: str, file_type: str = "", media: Optional[str] = None, rel: str = 'import'*)

HTML Imports is intended to be the packaging mechanism for web components, but you can also use HTML Imports by itself.

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/Web_Components/HTML_Imports

Parameters

- **href** – The link path for the stylesheet page
- **file_type** – Optional. The type of the href tag
- **media** – Optional. This resource will then only be loaded if the media condition is true
- **rel** – Optional. This attribute names a relationship of the linked document to the current document

manifest(*href: str, file_type: str = "", media: Optional[str] = None*)

The manifest keyword for the rel attribute of the `<link>` element indicates that the target resource is a Web app manifest.

Web app manifests are deployed in your HTML pages using a `<link>` element in the `<head>` of a document:

Related Pages:

https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types/manifest <https://developer.mozilla.org/en-US/docs/Web/Manifest>

Parameters

- **href** – The link path for the stylesheet page
- **file_type** – Optional. The type of the href tag
- **media** – Optional. This resource will then only be loaded if the media condition is true

pingback(*href: str, cross_origin: bool = False*)

Pingbacks (also known as trackbacks) are a form of automated comment for a page or post, created when another WordPress blog links to that page or post.

Related Pages:

<https://www.keycdn.com/blog/resource-hints> <https://wordpress.stackexchange.com/questions/116079/what-is-rel-pingback-and-what-is-the-use-of-this-in-my-website>

Parameters

- **href** – The link path for the stylesheet page
- **cross_origin** – Optional. Specify to the browser to enable the cross origin to get resource from

different website.

preconnect(*href: str, media: Optional[str] = None, cross_origin: bool = False*)

The preconnect directive allows the browser to setup early connections before an HTTP request is actually sent to the server. This includes DNS lookups, TLS negotiations, TCP handshakes. This in turn eliminates roundtrip latency and saves time for users.

Related Pages:

<https://www.keycdn.com/blog/resource-hints> https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types/preconnect

Parameters

- **href** – The link path for the stylesheet page
- **media** – Optional. This resource will then only be loaded if the media condition is true
- **cross_origin** – Optional. Specify to the browser to enable the cross origin to get resource from different website.

prefetch(*href: str, media: Optional[str] = None, cross_origin: bool = False*)

The prefetch keyword for the rel attribute of the <link> element is a hint to browsers that the user is likely to need the target resource for future navigations, and therefore the browser can likely improve the user experience by preemptively fetching and caching the resource.

Related Pages:

<https://www.keycdn.com/blog/resource-hints> https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types/prefetch

Parameters

- **href** – The link path for the stylesheet page

- **media** – Optional. This resource will then only be loaded if the media condition is true
- **cross_origin** – Optional. Specify to the browser to enable the cross origin to get resource from different website.

preload(*href: str, file_type: str = "", media: Optional[str] = None, cross_origin: bool = False*)

The preload keyword for the rel attribute of the <link> element indicates the user is highly likely to require the target resource for the current navigation, and therefore the browser must preemptively fetch and cache the resource.

Related Pages:

<https://www.keycdn.com/blog/resource-hints> https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types/preload

Parameters

- **href** – The link path for the stylesheet page
- **file_type** – Optional. The type of the href tag
- **media** – Optional. This resource will then only be loaded if the media condition is true
- **cross_origin** – Optional. Specify to the browser to enable the cross origin to get resource from different website.

prerender(*href: str, media: Optional[str] = None, cross_origin: bool = False*)

The prerender keyword for the rel attribute of the <link> element is a hint to browsers that the user might need the target resource for the next navigation, and therefore the browser can likely improve the user experience by preemptively fetching and processing the resource — for example, by fetching its subresources or performing some rendering in the background offscreen.

Related Pages:

<https://www.keycdn.com/blog/resource-hints> https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types/prerender

Parameters

- **href** – The link path for the stylesheet page
- **media** – Optional. This resource will then only be loaded if the media condition is true
- **cross_origin** – Optional. Specify to the browser to enable the cross origin to get resource from different website.

shortlink(*href: str*)

Some websites create short links to make sharing links via instant messaging easier.

Related Pages:

<https://www.keycdn.com/blog/resource-hints>

Parameters

- href** – The url path

stylesheet(*href: str, file_type: str = 'text/css', media: Optional[str] = None, rel: str = 'stylesheet', cross_origin: bool = False*)

Link the page to a style sheet.

Related Pages:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link>

Parameters

- **href** – The link path for the stylesheet page
- **file_type** – Optional. The type of the href tag
- **media** – Optional. This resource will then only be loaded if the media condition is true
- **rel** – Optional. This attribute names a relationship of the linked document to the current document
- **cross_origin** – Optional. Specify to the browser to enable the cross origin to get resource from different website.

class `epyk.core.html.Header.Meta`(*page: PageModel*)

author(*name: str*)

Define the author of the page.

Usage:

```
page.headers.meta.author('epykure')
```

Related Pages:

https://www.w3schools.com/tags/att_meta_name.asp

Parameters

name – The author name

charset(*value: str = 'utf-8'*)

Define the character set used.

Usage:

```
page.headers.meta.charset("test")
```

Related Pages:

https://www.w3schools.com/tags/tag_meta.asp

https://www.w3schools.com/tags/att_meta_charset.asp

Parameters

value – Optional. The charset encoding

custom(*name: str, content: str*)

Bespoke function to add other meta tags.

Usage:

```
page.headers.meta.custom('language', 'python')
```

Related Pages:

https://www.w3schools.com/tags/att_meta_name.asp

Parameters

- **name** – The name for the meta tag
- **content** – The value of the meta tag

description(*value: str*)

Define a description of your web page.

Usage:

```
page.headers.meta.description('This is a description')
```

Related Pages:

https://www.w3schools.com/html/html_head.asp

Parameters

value – The report description

http_equiv(*name: str, content: str*)

Bespoke function to add other http-equiv tags to the meta section.

Usage:

```
rptObj.headers.meta.http_equiv('language', 'python')
```

Related Pages:

https://www.w3schools.com/tags/att_meta_http_equiv.asp

Parameters

- **name** – The name for the meta tag
- **content** – The value of the meta tag

keywords(*content: Union[list, str]*)

Define keywords for search engine.

Usage:

```
page.headers.meta.keywords(['python', 'javascript'])
```

Related Pages:

https://www.w3schools.com/html/html_head.asp

Parameters

content – The keyword data

refresh(*time: int*)

Refresh document every X seconds.

Usage:

```
page.headers.meta.refresh(10)
```

Related Pages:

https://www.w3schools.com/tags/tag_meta.asp

Parameters

time – A time in second

viewport (*attrs: Optional[dict] = None*)

Setting the viewport to make your website look good on all devices.

Usage:

```
page.headers.meta.viewport({"width": "device-width"})
```

Related Pages:

https://www.w3schools.com/html/html_head.asp https://www.w3schools.com/tags/tag_meta.asp

Parameters

attrs – Optional. The view port attributes

5.8.7.2 Aria Module

class `epyk.core.html.Aria.Aria` (*component: HtmlModel*)

property atomic

Indicates whether assistive technologies will present all, or only parts of, the changed region based on the change notifications defined by the aria-relevant attribute.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-atomic>

property autocomplete

Indicates whether inputting text could trigger display of one or more predictions of the user's intended value for an input and specifies how predictions would be presented if they are made.

The aria-autocomplete property describes the type of interaction model a textbox, searchbox, or combobox employs when dynamically helping users complete text input. It distinguishes between two models: the inline model (`aria-autocomplete="inline"`) that presents a value completion prediction inside the text input and the list model (`aria-autocomplete="list"`) that presents a collection of possible values in a separate element that pops up adjacent to the text input. It is possible for an input to offer both models at the same time (`aria-autocomplete="both"`).

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-autocomplete>

property busy

Indicates an element is being modified and that assistive technologies MAY want to wait until the modifications are complete before exposing them to the user.

The default value of aria-busy is false for all elements. When aria-busy is true for an element, assistive technologies MAY ignore changes to content owned by that element and then process all changes made during the busy period as a single, atomic update when aria-busy becomes false.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-busy>

property checked

Indicates the current “checked” state of checkboxes, radio buttons, and other widgets. See related `aria-pressed` and `aria-selected`.

The `aria-checked` attribute indicates whether the element is checked (true), unchecked (false), or represents a group of other elements that have a mixture of checked and unchecked values (mixed). Most inputs only support values of true and false, but the mixed value is supported by certain tri-state inputs such as a checkbox or `menuitemcheckbox`.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-checked>

property colcount

Defines the total number of columns in a table, grid, or treegrid. See related `aria-colindex`.

If all of the columns are present in the DOM, it is not necessary to set this attribute as the user agent can automatically calculate the total number of columns. However, if only a portion of the columns is present in the DOM at a given moment, this attribute is needed to provide an explicit indication of the number of columns in the full table.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-colcount>

property colindex

Defines an element’s column index or position with respect to the total number of columns within a table, grid, or treegrid. See related `aria-colcount` and `aria-colspan`.

If all of the columns are present in the DOM, it is not necessary to set this attribute as the user agent can automatically calculate the column index of each cell or gridcell. However, if only a portion of the columns is present in the DOM at a given moment, this attribute is needed to provide an explicit indication of the column of each cell or gridcell with respect to the full table.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-colindex>

property colspan

Defines the number of columns spanned by a cell or gridcell within a table, grid, or treegrid. See related `aria-colindex` and `aria-rowspan`.

This attribute is intended for cells and gridcells which are not contained in a native table. When defining the column span of cells or gridcells in a native table, authors **SHOULD** use the host language’s attribute instead of `aria-colspan`. If `aria-colspan` is used on an element for which the host language provides an equivalent attribute, user agents **MUST** ignore the value of `aria-colspan` and instead expose the value of the host language’s attribute to assistive technologies.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-colspan>

property controls

Identifies the element (or elements) whose contents or presence are controlled by the current element. See related `aria-owns`.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-controls>

property current

Indicates the element that represents the current item within a container or set of related elements.

The `aria-current` attribute is an enumerated type. Any value not included in the list of allowed values **SHOULD** be treated by assistive technologies as if the value `true` had been provided. If the attribute is not present or its value is an empty string or undefined, the default value of `false` applies and the `aria-current` state **MUST NOT** be exposed by user agents or assistive technologies.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-current>

custom(*key*, *val*)**Parameters**

- **key** –
- **val** –

property describedby

Identifies the element (or elements) that describes the object. See related `aria-labelledby`.

The `aria-labelledby` attribute is similar to the `aria-describedby` in that both reference other elements to calculate a text alternative, but a label should be concise, where a description is intended to provide more verbose information.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-describedby>

property details

Identifies the element that provides a detailed, extended description for the object. See related `aria-describedby`.

The `aria-details` attribute references a single element that provides more detailed information than would normally be provided by `aria-describedby`. It enables assistive technologies to make users aware of the availability of an extended description as well as navigate to it. Authors **SHOULD** ensure the element referenced by `aria-details` is visible to all users.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-details>

property disabled

Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. See related `aria-hidden` and `aria-readonly`.

For example, irrelevant options in a radio group may be disabled. Disabled elements might not receive focus from the tab order. For some disabled elements, applications might choose not to support navigation to descendants. In addition to setting the `aria-disabled` attribute, authors **SHOULD** change the appearance (grayed out, etc.) to indicate that the item has been disabled.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-disabled>

property errormessage

Identifies the element that provides an error message for the object. See related `aria-invalid` and `aria-describedby`.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-errormessage>

property expanded

Indicates whether the element, or another grouping element it controls, is currently expanded or collapsed.

For example, this indicates whether a portion of a tree is expanded or collapsed. In other instances, this may be applied to page sections to mark expandable and collapsible regions that are flexible for managing content density. Simplifying the user interface by collapsing sections may improve usability for all, including those with cognitive or developmental disabilities.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-expanded>

property flowto

Identifies the next element (or elements) in an alternate reading order of content which, at the user's discretion, allows assistive technology to override the general default of reading in document source order.

When aria-flowto has a single IDREF, it allows assistive technologies to, at the user's request, forego normal document reading order and go to the targeted object. However, when aria-flowto is provided with multiple IDREFS, assistive technologies SHOULD present the referenced elements as path choices.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-flowto>

get(key: str, dfl=None)

Get the value of a custom aria.

Parameters

- **key** – The key definition of the aria.
- **dfl** – The default value.

property haspopup

Indicates the availability and type of interactive popup element, such as menu or dialog, that can be triggered by an element.

A popup element usually appears as a block of content that is on top of other content. Authors MUST ensure that the role of the element that serves as the container for the popup content is menu, listbox, tree, grid, or dialog, and that the value of aria-haspopup matches the role of the popup container.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-haspopup>

property hidden

Indicates whether the element is exposed to an accessibility API. See related aria-disabled.

User agents determine an element's hidden status based on whether it is rendered, and the rendering is usually controlled by CSS. For example, an element whose display property is set to none is not rendered. An element is considered hidden if it, or any of its ancestors are not rendered or have their aria-hidden attribute value set to true.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-hidden>

property invalid

Indicates the entered value does not conform to the format expected by the application. See related aria-errormessage.

If the value is computed to be invalid or out-of-range, the application author **SHOULD** set this attribute to true. User agents **SHOULD** inform the user of the error. Application authors **SHOULD** provide suggestions for corrections if they are known.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-invalid>

property shortcuts

Indicates keyboard shortcuts that an author has implemented to activate or give focus to an element.

The value of the aria-keyshortcuts attribute is a space-delimited list of keyboard shortcuts that can be pressed to activate a command or textbox widget. The keys defined in the shortcuts represent the physical keys pressed and not the actual characters generated. Each keyboard shortcut consists of one or more tokens delimited by the plus sign (“+”) representing zero or more modifier keys and exactly one non-modifier key that must be pressed simultaneously to activate the given shortcut.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-keyshortcuts>

property label

Defines a string value that labels the current element. See related aria-labelledby.

The purpose of aria-label is the same as that of aria-labelledby. It provides the user with a recognizable name of the object. The most common accessibility API mapping for a label is the accessible name property.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-label>

property labelledby

Identifies the element (or elements) that labels the current element. See related aria-describedby.

The purpose of aria-labelledby is the same as that of aria-label. It provides the user with a recognizable name of the object. The most common accessibility API mapping for a label is the accessible name property.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-label>

property level

Defines the hierarchical level of an element within a structure.

This can be applied inside trees to tree items, to headings inside a document, to nested grids, nested tablists and to other structural items that may appear inside a container or participate in an ownership hierarchy. The value for aria-level is an integer greater than or equal to 1.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-level>

property live

Indicates that an element will be updated, and describes the types of updates the user agents, assistive technologies, and user can expect from the live region.

The values of this attribute are expressed in degrees of importance. When regions are specified as polite, assistive technologies will notify users of updates but generally do not interrupt the current task, and updates take low priority. When regions are specified as assertive, assistive technologies will immediately notify the user, and could potentially clear the speech queue of previous updates.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-level>

property modal

Indicates whether an element is modal when displayed.

The aria-modal attribute is used to indicate that the presence of a “modal” element precludes usage of other content on the page. For example, when a modal dialog is displayed, it is expected that the user’s interaction is limited to the contents of the dialog, until the modal dialog loses focus or is no longer displayed.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-modal>

property multiline

Indicates whether a text box accepts multiple lines of input or only a single line.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-multiline>

property multiselectable

Indicates that the user may select more than one item from the current selectable descendants.

Authors **SHOULD** ensure that selected descendants have the aria-selected attribute set to true, and selectable descendant have the aria-selected attribute set to false. Authors **SHOULD NOT** use the aria-selected attribute on descendants that are not selectable.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-multiselectable>

property orientation

Indicates whether the element’s orientation is horizontal, vertical, or unknown/ambiguous.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-multiselectable>

property owns

Identifies an element (or elements) in order to define a visual, functional, or contextual parent/child relationship between DOM elements where the DOM hierarchy cannot be used to represent the relationship. See related aria-controls.

The value of the aria-owns attribute is a space-separated list of IDREFS that reference one or more elements in the document by ID. The reason for adding aria-owns is to expose a parent/child contextual relationship to assistive technologies that is otherwise impossible to infer from the DOM.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-owns>

property placeholder

Defines a short hint (a word or short phrase) intended to aid the user with data entry when the control has no value. A hint could be a sample value or a brief description of the expected format.

Authors **SHOULD NOT** use aria-placeholder instead of a label as their purposes are different: The label indicates what kind of information is expected. The placeholder text is a hint about the expected value. See related aria-labelledby and aria-label.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-placeholder>

property posinset

Defines an element's number or position in the current set of listitems or treeitems. Not required if all elements in the set are present in the DOM. See related aria-setsizes.

If all items in a set are present in the document structure, it is not necessary to set this attribute, as the user agent can automatically calculate the set size and position for each item. However, if only a portion of the set is present in the document structure at a given moment, this property is needed to provide an explicit indication of an element's position.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-posinset>

property pressed

Indicates the current “pressed” state of toggle buttons. See related aria-checked and aria-selected.

Toggle buttons require a full press-and-release cycle to change their value. Activating it once changes the value to true, and activating it another time changes the value back to false. A value of mixed means that the values of more than one item controlled by the button do not all share the same value. Examples of mixed-state buttons are described in WAI-ARIA Authoring Practices [wai-aria-practices-1.1]. If the attribute is not present, the button is not a toggle button.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-pressed>

property readonly

Indicates that the element is not editable, but is otherwise operable. See related aria-disabled.

This means the user can read but not set the value of the widget. Readonly elements are relevant to the user, and application authors SHOULD NOT restrict navigation to the element or its focusable descendants. Other actions such as copying the value of the element are also supported. This is in contrast to disabled elements, to which applications might not allow user navigation to descendants.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-readonly>

property relevant

Indicates what notifications the user agent will trigger when the accessibility tree within a live region is modified. See related aria-atomic.

The attribute is represented as a space delimited list of the following values: additions, removals, text; or a single catch-all value all.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-relevant>

property role

Example of roles Roles: button, checkbox, menuitem, menuitemcheckbox, menuitemradio, option, radio, switch, tab or treeitem

property roledescription

Defines a human-readable, author-localized description for the role of an element.

Some assistive technologies, such as screen readers, present the role of an element as part of the user experience. Such assistive technologies typically localize the name of the role, and they may customize it as well. Users of these assistive technologies depend on the presentation of the role name, such as “region,” “button,” or “slider,” for an understanding of the purpose of the element and, if it is a widget, how to interact with it.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-roledescription>

property rowindex

Defines an element's row index or position with respect to the total number of rows within a table, grid, or treegrid. See related `aria-rowcount` and `aria-rowspan`.

If all of the rows are present in the DOM, it is not necessary to set this attribute as the user agent can automatically calculate the index of each row. However, if only a portion of the rows is present in the DOM at a given moment, this attribute is needed to provide an explicit indication of each row's position with respect to the full table.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-rowindex>

property rowspan

Defines the number of rows spanned by a cell or gridcell within a table, grid, or treegrid. See related `aria-rowindex` and `aria-colspan`.

This attribute is intended for cells and gridcells which are not contained in a native table. When defining the row span of cells or gridcells in a native table, authors **SHOULD** use the host language's attribute instead of `aria-rowspan`. If `aria-rowspan` is used on an element for which the host language provides an equivalent attribute, user agents **MUST** ignore the value of `aria-rowspan` and instead expose the value of the host language's attribute to assistive technologies.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-rowspan>

property selected

Indicates the current "selected" state of various widgets. See related `aria-checked` and `aria-pressed`.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-selected>

set(*arias*)

Set multiple aria properties

Parameters

arias –

property setsize

Defines the number of items in the current set of listitems or treeitems. Not required if all elements in the set are present in the DOM. See related `aria-posinset`.

This property is marked on the members of a set, not the container element that collects the members of the set. To orient the user by saying an element is "item X out of Y," the assistive technologies would use X equal to the `aria-posinset` attribute and Y equal to the `aria-setsize` attribute.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-setsize>

property sort

Defines the maximum allowed value for a range widget.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-sort>

property valuemax

Defines the maximum allowed value for a range widget.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-valuemax>

property valuemin

Defines the minimum allowed value for a range widget.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-valuemin>

property valuenow

Defines the current value for a range widget. See related aria-valuetext.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-valuenow>

property valuetext

Defines the human-readable text alternative of aria-valuenow for a range widget.

Related Pages:

<https://www.w3.org/TR/wai-aria-1.1/#aria-valuetext>

5.8.7.3 KeyCodes Module

```
class epyk.core.html.KeyCodes.KeyCode(component: Optional[HtmlModel] = None, source_event:  
Optional[str] = None, page=None)
```

```
alt(js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event:  
Optional[str] = None)
```

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

```
any(js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, source_event: Optional[str] =  
None)
```

Trigger event for any keycodes.

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **source_event** – Optional. The source component for the event

```
backspace(js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False,  
source_event: Optional[str] = None)
```

Parameters

- **js_funcs** – Javascript functions

- **profile** – Optional. A flag to set the component performance storage
- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

control(*js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event: Optional[str] = None*)

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

custom(*rule: str, js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, source_event: Optional[str] = None*)

Parameters

- **rule** – Bespoke keys combination
- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **source_event** – Optional. The source component for the event

delete(*js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event: Optional[str] = None*)

Keycode 46, the sup key

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

down(*js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event: Optional[str] = None*)

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

enter(*js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event: Optional[str] = None*)

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage

- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

escape(*js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event: Optional[str] = None*)

Keycode 27, the escape key.

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

get_event()

Return the complete definition for the key event.

key(*key_code: int, js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event: Optional[str] = None*)

Parameters

- **key_code** – The key code
- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

left(*js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event: Optional[str] = None*)

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

right(*js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event: Optional[str] = None*)

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

save(*js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, source_event: Optional[str] = None*)

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **source_event** – Optional. The source component for the event

shift(*js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event: Optional[str] = None*)

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

shift_with(*key: str, js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, source_event: Optional[str] = None*)

Parameters

- **key** (*str*) –
- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **source_event** – Optional. The source component for the event

space(*js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event: Optional[str] = None*)

Add an event on the space key.

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

tab(*js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event: Optional[str] = None*)

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage
- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

up(*js_funcs: Union[list, str], profile: Optional[Union[bool, dict]] = None, reset: bool = False, source_event: Optional[str] = None*)

Parameters

- **js_funcs** – Javascript functions
- **profile** – Optional. A flag to set the component performance storage

- **reset** – Optional. Flag to reset the events sequence
- **source_event** – Optional. The source component for the event

5.8.8 Entities

5.8.8.1 SymbArrows Module

5.8.8.2 SymbCurrencies Module

5.8.9 Symboles

5.8.9.1 EntHtml4 Module

5.8.9.2 EntHtml5_A Module

5.8.9.3 EntHtml5_B Module

5.8.9.4 EntHtml5_C Module

5.8.9.5 EntHtml5_D Module

5.9 External resources

5.9.1 Python Packages

Those packages are not included in the library as we wanted Epyk to be generic and with no dependency those as useful but not mandatory.

This section will illustrate how to link those to Epyk components.

5.9.1.1 Libraries

5.9.1.1.1 Pandas

Note: Most of the components are expecting serializable objects so this is way we are using list of dictionary as a common

input source for components.

If you are using pandas in your script you can easily convert it to a valid input data by doing the below for example:

```
df = pd.DataFrame({
    "Animal": ["Mouse", "Rabbit", "Dog", "Tiger", "Elefant", "Wale"],
    "Weight [g]": [19, 3000, 40000, 200000, 6000000, 50000000]})

line = page.ui.charts.apex.line(df.to_dict(orient="records"), y_columns=["Weight [g]"],
    ↪x_axis="Animal")
```

This will create a records object using the method `to_dict(orient="records")`

5.9.1.1.2 pandas_datareader

Note: Most of the components are expecting serializable objects so this is way we are using list of dictionary as a common

input source for components.

If you are using pandas_datareader in your script you can easily convert it to a valid input data by doing the below for example:

```
start = pd.to_datetime('2020-02-04')
end = pd.to_datetime('today')

tesla_df = data.DataReader('TSLA', 'yahoo', start, end)

columns = ['Close', 'Open', 'Volume']
records = []
for rec in tesla_df[columns].to_records():
    records.append(dict(zip(['Date'] + columns, rec)))
    records[-1]['Date'] = pd.to_datetime(records[-1]['Date']).strftime('%Y-%m-%d')
```

This will create a records object using the method to_records()

5.9.1.1.3 Matplotlib

- Documentation: <https://matplotlib.org/>
- pypi alias: matplotlib
- Package Type: Python

Using a simple image component:

```
import epyk as pk

import numpy as np
import matplotlib.pyplot as plt

page = pk.Page()

x = np.arange(0, 15, 0.1)
y = np.sin(x)
plt.plot(x, y)

img1 = page.ui.img(width=(50, "%"))
img1.from_plot(plt)
img1.style.css.display = "inline-block"
```

Using animated image:

```
img1 = page.ui.images.animated("", text="This is a comment", title="Title", width=(49, "%  
↪"), url="#")  
img1.from_plot(plt)  
img1.style.css.borders()  
img1.style.css.display = "inline-block"
```

Using the carousel:

```
page = pk.Page()  
  
import numpy as np  
import matplotlib.pyplot as plt  
  
carousel = page.ui.images.carousel()  
  
for i in range(10):  
    x = np.arange(0, i * 5, 0.1)  
    y = np.sin(x)  
    plt.plot(x, y)  
    carousel.add_plot(plt, width=(220, 'px'))  
  
carousel.set_nav_dots()
```

Using an external package slideshow:

```
page = pk.Page()  
  
import numpy as np  
import matplotlib.pyplot as plt  
  
slideshow = page.ui.slideshow()  
  
for i in range(10):  
    x = np.arange(0, i * 5, 0.1)  
    y = np.sin(x)  
    plt.plot(x, y)  
    slideshow.add_plot(plt, width=(220, 'px'))
```

5.9.1.2 Servers

5.9.2 JavaScript Packages

The full and up to date list of packages in Epyk is available in the section *Supported Libraries and Frameworks*

This section will illustrate how to use them and also how to learn their API.

5.9.2.1 Libraries

5.9.2.1.1 Slideshow

- Documentation: <https://github.com/ganlanyuan/tiny-slider>
- NPM alias: `tiny-slider`
- Package Type: JavaScript

The package slideshow from Tiny slider is available in the framework by using the following component:

```
ss = page.ui.slideshow([page.ui.text("Great results %s" % i) for i in range(20)])
```

The above will display a slide object with simple text elements.

Each of the functions and properties, when they are wrapping an core API function, will point from the doc string in the framework to its external documentation. Do not hesitate to have a look at it and either to challenge the library to get new features or to check with us if something does not seem correct.

Options

Then exactly in the same way it is mentioned in the slideshow documentation it is possible to add options to this component. All the options are available from the `component.options` property:

```
ss.options.autoplay = False
ss.options.items = 4
```

The above will disable the autoplay and change the number of items per page. More information in the package documentation available from there Github repository.

Events

All the component events are available from the component `.js` property. Basically by adding the below it will go to the next element when a button is clicked:

```
btn = page.ui.button("Click")
btn.click([
    ss.js.next()
])
```

Note: The `js` property must be added to a JavaScript event to be rendered as string to the page during the transpiling. Any functions called directly from Python will rendered string which will not be captured in the page object.

Style

This component will have some predefined CSS class in order to change the nav bar and the button but it is possible as any other component to change the style by using the `style.css` property.

5.9.2.1.2 Json Formatter

- Documentation: <https://github.com/mohsen1/json-formatter-js>
 - NPM alias: `json-formatter-js`
 - Package Type: JavaScript
-

The package `json-formatter-js` is available in the framework by using the following component:

```
js_formatter = page.ui.json({"foo": 42})
```

The above will display a tree object as shown here. <https://azimi.me/json-formatter-js/>

Each of the functions and properties, when they are wrapping an core API function, will point from the doc string in the framework to its external documentation. Do not hesitate to have a look at it and either to challenge the library to get new features or to check with us if something does not seem correct.

Options

Then exactly in the same way it is mentioned in the documentation it is possible to add options to this component from the Python. All the options are available from the `component.options` property:

```
js_formatter.options.hoverPreviewEnabled = False
```

Events

All the component events are available from the component `.js` property. Basically by adding the below it will open the tree up to n level:

```
btn = page.ui.button("Click")
btn.click([
    js_formatter.js.openAtDepth(3)
])
```

Note: The `js` property must be added to a JavaScript event to be rendered as string to the page during the transpiling. Any functions called directly from Python will rendered string which will not be captured in the page object.

Style

This component will have some predefined CSS class in order to change the nav bar and the button but it is possible as any other component to change the style by using the `style.css` property.

The below will add a border to the component:

```
js_formatter.style.css.border = "1px solid black"
```

5.9.2.1.3 Font awesome

- Documentation: <https://fontawesome.com/>
- NPM alias: font-awesome
- Package Type: JavaScript

The package Font-Awesome is available in the framework by using the following component. It is the default package used for the various icons in the framework:

```
import epyk as pk
from epyk.core.js.packages import JsFontAwesome

page = pk.Page()
icon = page.ui.icons.next()
icon.click([
    page.js.alert("This is a test")
])

icon = page.ui.icon("fab fa-python")
icon.pulse()

icon = page.ui.icon(JsFontAwesome.ICON_CARET_SQUARE_O_UP)
```

It is possible to use the module JsFontAwesome to get all the available icons on the IDE using the autocompletion.

Events

All the component events are available from the component `.js` property. Basically by adding the below it will open the tree up to n level:

```
icon = page.ui.icons.next()
icon.click([page.js.alert("ok")])
```

Note: The `js` property must be added to a JavaScript event to be rendered as string to the page during the transpiling. Any functions called directly from Python will be rendered as string which will not be captured in the page object.

Style

This component will have some predefined CSS class in order to change the nav bar and the button but it is possible as any other component to change the style by using the `style.css` property.

The below will add a border to the component:

```
icon.style.css.color = "1px solid black"
```

5.9.2.1.4 Mathjax

- Documentation: <https://www.mathjax.org>
- NPM alias: `mathjax`
- Package Type: JavaScript

The package **mathjax** is available in the framework by using the following component:

```
formulas = page.ui.texts.formula("$$ E=mc^2 $$", helper="This is a formula")
```

The above will display a maths formulas. More details available on the website: <https://www.mathjax.org> The content is expecting a LaTeX format.

Style

This component will have some predefined CSS class in order to change the nav bar and the button but it is possible as any other component to change the style by using the `style.css` property.

The below will add a border to the component:

```
formulas.style.css.color = "green"
```

5.9.2.1.5 JQuery UI

- Documentation: <https://github.com/jquery/jqueryui.com>
- NPM alias: `jqueryui`
- Package Type: JavaScript

This package is a set of UI components which can be used on top of JQuery in order to create input for Dashboards.

DatePicker

To add a data picker:

```
import epyk as pk

page = pk.Page()
date = page.ui.date()
page.outs.jupyter(closure=False)
```

Event

Get the selected value:

```
page.ui.button("Get Value").click([
    page.js.alert(date.dom.content)
])
```

Or when it has changed:

```
date.select([
    page.js.alert(date.dom.content)
])
```

Slider

To add a slider:

```
import epyk as pk

page = pk.Page()
page.ui.slider()
page.outs.jupyter(closure=False)
```

Event

To get the content when the value has changed:

```
slider.change([
    page.js.alert(slider.dom.content)
])
```

Autocompletion

To get toe autocompletion based on an external data source:

```
items = page.ui.inputs.autocomplete(placeholder="select a language and press enter",  
↵options={"select": True})  
selected = page.ui.text()  
items.enter([  
    selected.build(items.dom.content)  
)  
page.js.customFile("FR.js", r"http://pypl.github.io/PYPL")  
page.body.onReady([items.js.source(pk.js_std.var("graphData")[0])])
```

or from an hard coded source using the component options:

```
items = page.ui.inputs.autocomplete(placeholder="select a language and press enter",  
↵options={"select": True})  
items.options.source = values
```

Events

Use components in an external service call:

```
page.js.post("url", components=[slider, date]).onSuccess([  
    ...  
)
```

5.9.2.1.6 PivotTable.js

- Documentation: <https://pivottable.js.org/examples/>
- NPM alias: `pivottable`
- Package Type: JavaScript

The package **pivottable** is available in the framework by using the following component:

```
languages = [  
    {"name": 'C', 'type': 'code', 'rating': 17.07, 'change': 12.82},  
    {"name": 'Java', 'type': 'code', 'rating': 16.28, 'change': 0.28},  
]  
  
tb = page.ui.tables.pivot(languages, ['name'], ['type'])
```

The above will display a pivot table. More details available on the website: <https://github.com/nicolaskruchten/pivottable/wiki>

Options

It is possible to use aggregator and renderers directly in the framework. The below will use the sum aggregator on the column rating:

```
tb = page.ui.tables.pivot(languages, ['name'], ['type'])
tb.aggregators.sum('rating')
```

More bespoke ones are also available:

```
tb.aggregators.diffAbsolute('change', 'rating')
```

And it is possible to create custom ones:

```
tb.aggregators.quick("change", "Abs Change", "+= Math.abs(col1)")
```

Different renders are available in the property `page.ui.tables.pivots`.

And all pivotables options are available `tb.options`. A detailed documentation is available for each property:

```
@property
def exclusions(self):
    """
    Description:
    -----
    Object whose keys are attribute names and values are arrays of attribute values which
    denote records to exclude
    from rendering; used to prepopulate the filter menus that appear on double-click.

    Related Pages:

    https://github.com/nicolaskruchten/pivortable/wiki/Parameters
    """
    return self._config_get({})
```

Style

This component will have some predefined CSS class in order to change the nav bar and the button but it is possible as any other component to change the style by using the `style.css` property.

5.9.2.2 UI Frameworks

5.9.2.3 Web Frameworks

5.10 Library extensions

Anything in Epyk can be customized. The style and the final page layout can be fully changed.

5.10.1 Styles & configurations

Add CSS Inline to a type of components:

```
page.ui.components_skin = {
    "title": {"css": {"color": "#A89A37"}},
    "layouts.hr": {"css": {"background-color": "#f0db4f", "margin-bottom": "10px"}},
    "button": {"css": {"background": "#323330", "color": "#f0db4f", "border-color": "#323330"}}}
```

This will automatically add the a TestClass to any component button:

```
page = pk.Page()
page.ui.components_skin = {
    "button": {"cls": ["TestClass"]}
}
page.ui.button("Test")
```

Also the framework is using some internal names for the CSS classes based on the internal Python classes.

For example, the below base classes used for button:

```
class CssButtonBasic(CssStyle.Style):
    _attrs = {'font-weight': 'bold', 'padding': '2px 20px', 'margin': '2px 0 2px 0',
        'text-decoration': 'none',
        'border-radius': '4px', 'white-space': 'nowrap', 'display': 'inline-block',
        'line-height': '30px',
        '-webkit-appearance': 'none', '-moz-appearance': 'none'}
    _hover = {'text-decoration': 'none', 'cursor': 'pointer'}
    _focus = {'outline': 0}
    _disabled = {'cursor': 'none'}
```

will be referenced with the below names in the CSS section:

```
.cssbuttonbasic {font-weight: bold ;padding: 2px 20px ;margin: 2px 0 2px 0 ;text-
    decoration: none ;border-radius: 4px ;white-space: nowrap ;display: inline-block ;line-
    height: 30px ;-webkit-appearance: none ;-moz-appearance: none ;border: 1px solid
    #f4f9fc ;color: inherit ;background-color: #FFFFFF ;}
.cssbuttonbasic:hover {text-decoration: none ;cursor: pointer ;background-color: #f4f9fc
    !IMPORTANT ;color: #607d8b !IMPORTANT ;border: 1px solid #607d8b !IMPORTANT ;}
.cssbuttonbasic:focus {outline: 0 ;}
.cssbuttonbasic:disabled {cursor: none ;background-color: #263238 ;color: #455a64 ;font-
    style: italic ;}
```

It is possible to remove or rename this by using shortcut `rename_css_cls` as shown below:

```
pk.rename_css_cls({"cssbuttonbasic": "cssNewName"})
```

To get more details on CSS configurations and the way it is managed for components and the pages use the below lines.

5.10.1.1 Component style

Any component in Epyk can be fully changed and the style can be entirely updated. First to mention that everything done in this library has been achieved thanks to the good quality of the tutorials in [w3schools](#)

By default, all the components come with predefined CSS styles and CSS classes. Usually the CSS properties are set in the component functions available in the Interface section.

For example for the `page.ui.buttons.colored`:

```
@html.Html.css_skin()
def colored(self, text="", icon=None, color=None, width=(None, "%"), height=(None, "px",
↪"), align="left",
            html_code=None, tooltip=None, profile=None, options=None):

    component = self.button(text, icon, width, height, align, html_code, tooltip,
↪profile, options)
    component.style.css.background = color or self.page.theme.colors[-1]
    component.style.css.border = "1px solid %s" % (color or self.page.theme.colors[-1])
    component.style.css.color = self.page.theme.colors[0]
    component.style.css.margin_top = 5
    component.style.css.margin_bottom = 5
    component.style.css.padding_left = 10
    component.style.css.padding_right = 10
    return component
```

All CSS inline properties are available in any components from the `component.style.css` property

5.10.1.1.1 Using CSS inline

The above example can be changed by using the below lines of code for example to make it fixed to the page:

```
button = page.ui.buttons.colored("Test")
button.style.css.color = "yellow" # Change the text color
button.style.css.position = "fixed"
button.style.css.bottom = 10 # Default will use px
button.style.css.right = 10 # Default will use px
```

This will then render the page the HTML tag:

```
<button data-count="0" id="button_2561249251968" style="font-size:14px;margin:0;
↪padding:0px;padding-left:10px;padding-right:10px;line-height:23px;background:#263238;
↪border:1px solid #263238;color:yellow;margin-top:5px;margin-bottom:5px;position:fixed;
↪bottom:0px;right:0px" class="cssbuttonbasic">Test</button>
```

5.10.1.1.2 Using CSS Inline object

If it possible to do the same thing by using the `CssInline` shortcut:

```
import epyk as pk

inline = pk.CssInline()
inline.color = "yellow"
inline.position = "fixed"
inline.bottom = 0
inline.right = 0

page = pk.Page()

button = page.ui.buttons.colored("Test")
button.css(inline.to_dict())
```

5.10.1.1.3 Using CSS classes

It is also possible to move away from `inline` and to use CSS classes instead. CSS classes from a `CssInline` object can be done:

```
inline = pk.CssInline()
inline.background_color = "yellow"
inline.position = "fixed"
inline.bottom = 0
inline.right = 0
inline.important(["background_color"])

myClass = inline.to_class("MyClass")

page = pk.Page()

button = page.ui.buttons.colored("Test")
button.style.classList['main'].add(myClass)
```

It is also possible to create an internal CSS class using the below. In this example if it possible to add more information to the CSS class as it is generated from a `inline` structure:

```
from epyk.core.css.styles.classes import CssStyle

class CssHoverColor(CssStyle.Style):
    _attrs = {'color': 'blue', 'cursor': 'pointer'}
    _hover = {'color': 'orange'}

div1 = page.ui.div("This is a text")
# Attach the class to the component
div1.style.add_classes.custom(CssHoverColor)
```

5.10.1.1.4 Using external CSS

In order to facilitate the use of external data Epyk has multiple ways to integrate external styles.

From text

It is also possible to add a bespoke CSS text manually and then to add your CSS class to the component:

```
page.properties.css.add_text('''
.MyClass {
    ...
}
''')

button = page.ui.buttons.colored("Test")
button.attr["class"].add("MyClass")
```

From file

But it can also be done using a CSS file. This time it is required to register the file:

```
page.css.customFile("animate.min.css", path="https://cdnjs.cloudflare.com/ajax/libs/
↪animate.css/3.7.2")

button = page.ui.buttons.colored("Test")
button.attr["class"].add("MyClass")
```

Note: Do not forget that Epyk is a collaborative library so do not hesitate to share your improvements to ensure other people will benefit from your knowledge.

5.10.1.2 Derived component

The notion of derived component or configuration is heavily used in the framework to provide multiple predefined options to the user.

Thus instead of always using a standard button and then having to update the style every time:

```
class Buttons:

    def button(self, text="", icon=None, width=(None, "%"), height=(None, "px"), align=
↪"left", html_code=None,
        tooltip=None, profile=None, options=None):

        ...

    def large(self, text="", icon=None, width=(None, "%"), height=(None, "px"), align=
↪"left", html_code=None,
        tooltip=None, profile=None, options=None):

        ...
```

(continues on next page)

(continued from previous page)

```
def normal(self, text="", icon=None, width=(None, "%"), height=(None, "px"), align=
↪ "left", html_code=None,
    tooltip=None, profile=None, options=None):
    ...

def run(self, text="", width=(None, "%"), height=(None, "px"), align="left", html_
↪ code=None, tooltip=None,
    profile=None, options=None):
```

Those configurations are coming from bespoke CSS style which have been added as shortcut to assist the creation of rich web page. This can be seen as a toolbox with shortcut to help the web development.

As a general rule derived component have always the same signature to make easy the migration from one to another within the same component category.

Note: Do not hesitate to propose your configuration to be added as derived components to the framework.

5.10.1.3 Package extension

In progress

5.10.1.4 Page skins

There is also a notion of skin to the page to add nice backgrounds to your page. All the skins are accessible from the `page.skins` property

Currently few skins are available:

```
def rains(self, width=(100, '%'), height=(100, '%'), options=None, profile=None)
def winter(self, width=(100, '%'), height=(100, '%'), options=None, profile=None)
def matrix(self, width=(100, '%'), height=(100, '%'), options=None, profile=None)
def fireworks(self, width=(100, '%'), height=(100, '%'), options=None, profile=None)
def lights(self, width=(100, '%'), height=(100, '%'), options=None, profile=None)
```

For example the below lines will add a Matrix style to your page:

```
page = pk.Page()

page.skins.matrix()
page.ui.div("This is a text")
```

By transpiling the file with the below command you get a nice HTML page with the code inside:

```
epyk.exe transpile -s=N
```

This could be a easy and nice way to adapt and change your web site based on the season :).

5.11 Bugs & ToDo

For the ones interested in participating there is a list of improvements to be done to the framework. Those bugs are all defined in the code and they are either known bugs or extensions which need to be added.

5.11.1 Style

5.11.1.1 Global CSS

- Extend the `.globals` in **GrpCls.py** property to update more than the font and few properties.
- Improve `custom_class` in **GrpCls.py** to propagate the important attribute
- Improve way colors are defined for Charts in **Theme.py**.

5.11.2 Data

- Create a dedicated data core package with the input data definition for the components.
- Full revamping of the module `PyMarkdown.py`.

5.11.3 JavaScript

Obviously some work is still needed to fully wrap all the external packages. There are some but do not hesitate to add more issues there: <https://github.com/epykure/epyk-ui/issues>, we will try to tackle them)

5.11.4 Packages

5.11.4.1 PivotTable

- Fix the column initial selections for custom aggregators.

5.11.4.2 Sparklines

- Fix display tooltips in Jupyter
- Add event and tooltip style

PYTHON MODULE INDEX

e

`epyk.core.cli.cli_export`, [67](#)
`epyk.core.cli.cli_npm`, [68](#)
`epyk.core.cli.cli_project`, [70](#)
`epyk.core.data.Data.DataPy`, [568](#)
`epyk.core.js.JsLocation`, [510](#)
`epyk.core.js.JsWindow`, [495](#)
`epyk.core.py.PyOuts`, [537](#)

A

- `a()` (*epyk.interfaces.components.CompTags.Tags method*), 283
- `abbr()` (*epyk.interfaces.components.CompTags.Tags method*), 283
- `abs()` (*epyk.core.js.Js.JsMaths.JsMaths method*), 609
- `absolute()` (*epyk.interfaces.components.CompButtons.Buttons method*), 72
- `absolute()` (*epyk.interfaces.components.CompTexts.Texts method*), 302
- `accentuate()` (*epyk.interfaces.components.CompLayouts.Layouts method*), 182
- `accounting` (*epyk.core.js.Js.JsBase property*), 516, 581
- `acknowledge()` (*epyk.interfaces.components.CompModals.Modals method*), 221
- `active` (*epyk.core.css.styles.classes.CssStyle.Style property*), 577
- `activeElement()` (*epyk.core.js.Js.JsBase method*), 516, 581
- `add()` (*epyk.core.js.Imports.ImportManager method*), 56
- `add()` (*epyk.core.js.Js.JsBreadCrumb method*), 606
- `add()` (*epyk.core.js.JsLocation.URLSearchParams method*), 514
- `add()` (*in module epyk.core.cli.cli_project*), 70
- `add_classes` (*epyk.core.css.styles.GrpCls.ClassPage property*), 575
- `add_code()` (*epyk.core.html.Header.Header method*), 627
- `add_map()` (*epyk.interfaces.geo.CompGeoJqV.JqueryVertorMap method*), 338
- `add_profiling()` (*epyk.core.js.Js.JsPerformance.JsPerformance method*), 623
- `add_script()` (*epyk.core.html.Header.Header method*), 627
- `addClickListener()` (*epyk.core.js.JsWindow.JsWindowEvent method*), 509
- `addContentLoaded()` (*epyk.core.js.JsWindow.JsWindowEvent method*), 509
- `addEventListener()` (*epyk.core.js.JsWindow.JsWindow method*), 501
- `addEventListener()` (*epyk.core.js.JsWindow.JsWindowEvent method*), 509
- `addPackage()` (*epyk.core.js.Imports.ImportManager method*), 56
- `addScrollListener()` (*epyk.core.js.JsWindow.JsWindowEvent method*), 509
- `adv_text()` (*epyk.interfaces.components.CompRich.Rich method*), 270
- `africa()` (*epyk.interfaces.geo.CompGeoJqV.JqueryVertorMap method*), 338
- `africa()` (*epyk.interfaces.geo.CompGeoPlotly.PlotlyBubble method*), 345
- `africa()` (*epyk.interfaces.geo.CompGeoPlotly.PlotlyChoropleth method*), 349
- `after` (*epyk.core.css.styles.classes.CssStyle.Style property*), 577
- `agenda()` (*epyk.interfaces.components.CompCalendars.Calendar method*), 93
- `AgGrid` (*class in epyk.interfaces.tables.CompAgGrid*), 466
- `aggrids` (*epyk.interfaces.tables.CompTables.Tables property*), 472
- `alert()` (*epyk.core.js.JsWindow.JsWindow method*), 501
- `alert()` (*epyk.interfaces.components.CompNetwork.Network method*), 244
- `alert()` (*epyk.interfaces.components.CompTexts.Texts method*), 302
- `alias()` (*epyk.core.py.PySql.SqlConnNeo4j method*), 557
- `all()` (*epyk.core.py.PyMarkdown.MarkDown method*), 546
- `alpha()` (*epyk.interfaces.components.CompLists.Lists method*), 198
- `alt()` (*epyk.core.html.KeyCodes.KeyCode method*), 644
- `alternative()` (*epyk.core.html.Header.Links method*), 630
- `angular()` (*in module epyk.core.cli.cli_export*), 67
- `angular()` (*in module epyk.core.cli.cli_npm*), 68
- `angular_parser()` (*in module epyk.core.cli.cli_npm*), 68
- `animated()` (*epyk.interfaces.components.CompImages.Images method*), 158
- `animation()` (*epyk.core.css.styles.classes.CssStyle.Style*

- method), 577
- animations (epyk.interfaces.Interface.Components property), 479
- any() (epyk.core.html.KeyCodes.KeyCode method), 644
- apex (epyk.interfaces.graphs.CompCharts.Chart2d property), 352
- apex (epyk.interfaces.graphs.CompCharts.Graphs property), 356
- ApexChart (class in epyk.interfaces.graphs.CompChartsApex), 362
- app() (in module epyk.core.cli.cli_project), 71
- append() (epyk.core.js.Js.JsLocation.URLSearchParams method), 618
- append() (epyk.core.js.Js.JsLocation.URLSearchParams method), 514
- apple_touch_icon() (epyk.core.html.Header.Icons method), 629
- apple_touch_startup_icon() (epyk.core.html.Header.Icons method), 629
- apps (epyk.core.Page.Report property), 569
- area() (epyk.interfaces.graphs.CompChartsApex.ApexChart method), 362
- area() (epyk.interfaces.graphs.CompChartsBillboard.Billboard method), 369
- area() (epyk.interfaces.graphs.CompChartsC3.C3 method), 379
- area() (epyk.interfaces.graphs.CompChartsChartCss.CompChart method), 387
- area() (epyk.interfaces.graphs.CompChartsChartJs.Chart method), 389
- area() (epyk.interfaces.graphs.CompChartsGoogle.Chart method), 412
- area() (epyk.interfaces.graphs.CompChartsNvd3.Nvd3 method), 421
- area() (epyk.interfaces.graphs.CompChartsPlotly.Plotly2D method), 432
- area_step() (epyk.interfaces.graphs.CompChartsBillboard.Billboard method), 370
- area_step() (epyk.interfaces.graphs.CompChartsC3.C3 method), 379
- Aria (class in epyk.core.html.Aria), 636
- arrow() (epyk.interfaces.components.CompPictos.Pictogram method), 266
- arrow_left() (epyk.interfaces.graphs.CompChartsSvg.SVG method), 451
- arrow_right() (epyk.interfaces.graphs.CompChartsSvg.SVG method), 452
- arrows_down() (epyk.interfaces.components.CompPanels.Panels method), 258
- arrows_up() (epyk.interfaces.components.CompPanels.Panels method), 258
- asia() (epyk.interfaces.geo.CompGeoJqV.JqueryVectorMap method), 338
- asia() (epyk.interfaces.geo.CompGeoPlotly.PlotlyBubble method), 345
- asia() (epyk.interfaces.geo.CompGeoPlotly.PlotlyChoropleth method), 349
- aside() (epyk.interfaces.components.CompTags.Tags method), 284
- assign() (epyk.core.js.Js.JsLocation.JsLocation class method), 614
- assign() (epyk.core.js.Js.JsLocation.JsLocation class method), 510
- assistant() (epyk.interfaces.components.CompNetwork.Network method), 244
- asterix() (epyk.interfaces.Interface.Components method), 479
- atob() (epyk.core.js.Js.JsWindow.JsWindow method), 502
- atomic (epyk.core.html.Aria.Aria property), 636
- audio() (epyk.interfaces.components.CompMedia.Media method), 211
- australia() (epyk.interfaces.geo.CompGeoJqV.JqueryVectorMap method), 339
- auth (epyk.core.Page.Report property), 569
- author() (epyk.core.html.Header.Meta method), 634
- autocomplete (epyk.core.html.Aria.Aria property), 636
- autocomplete() (epyk.interfaces.components.CompFields.Fields method), 106
- autocomplete() (epyk.interfaces.components.CompInputs.Inputs method), 169
- availableHeight (epyk.core.js.Js.JsScreen property), 607
- availWidth (epyk.core.js.Js.JsScreen property), 607
- avatar() (epyk.interfaces.components.CompIcons.Icons method), 130
- avatar() (epyk.interfaces.components.CompImages.Images method), 158
- awesome() (epyk.interfaces.components.CompIcons.Icons method), 130
- axes() (epyk.interfaces.graphs.CompChartsSvg.SVG method), 452
- ## B
- b() (epyk.interfaces.components.CompTags.Tags method), 284
- b64encode() (epyk.core.py.PyCrypto.PyCrypto class method), 558
- back() (epyk.core.js.Js.JsWindow.JsHistory method), 495
- background() (epyk.interfaces.components.CompImages.Images method), 159
- background() (epyk.interfaces.components.CompVignets.Vignets method), 325
- backspace() (epyk.core.html.KeyCodes.KeyCode method), 644
- badge() (epyk.interfaces.components.CompIcons.Icons method), 131
- badge() (epyk.interfaces.components.CompImages.Images method), 160

- `badges()` (`epyk.interfaces.components.CompLists.Lists` method), 198
- `banner()` (`epyk.interfaces.components.CompNavigation.Navigation` method), 235
- `Banners` (class in `epyk.interfaces.components.CompNavigation`), 227
- `banners` (`epyk.interfaces.Interface.Components` property), 479
- `bar()` (`epyk.interfaces.components.CompIcons.Icons` method), 132
- `bar()` (`epyk.interfaces.components.CompMenus.Menus` method), 213
- `bar()` (`epyk.interfaces.components.CompNavigation.Navigation` method), 236
- `bar()` (`epyk.interfaces.graphs.CompChartsApex.ApexCharts` method), 362
- `bar()` (`epyk.interfaces.graphs.CompChartsBillboard.Billboard` method), 370
- `bar()` (`epyk.interfaces.graphs.CompChartsC3.C3` method), 380
- `bar()` (`epyk.interfaces.graphs.CompChartsChartCss.CompChartsChartCss` method), 387
- `bar()` (`epyk.interfaces.graphs.CompChartsChartJs.ChartJs` method), 389
- `bar()` (`epyk.interfaces.graphs.CompChartsD3.D3` method), 400
- `bar()` (`epyk.interfaces.graphs.CompChartsDc.DC` method), 403
- `bar()` (`epyk.interfaces.graphs.CompChartsFrappe.CompChartsFrappe` method), 409
- `bar()` (`epyk.interfaces.graphs.CompChartsGoogle.ChartGoogle` method), 413
- `bar()` (`epyk.interfaces.graphs.CompChartsNvd3.Nvd3` method), 421
- `bar()` (`epyk.interfaces.graphs.CompChartsPlotly.Plotly2D` method), 432
- `bar()` (`epyk.interfaces.graphs.CompChartsRoughViz.CompChartsRoughViz` method), 445
- `bar()` (`epyk.interfaces.graphs.CompChartsSparkline.Sparkline` method), 448
- `bar()` (`epyk.interfaces.graphs.CompChartsVis.Vis2D` method), 461
- `bar()` (`epyk.interfaces.graphs.CompChartsVis.Vis3D` method), 464
- `bar3d()` (`epyk.interfaces.graphs.CompChartsVis.Vis` method), 459
- `bars` (`epyk.interfaces.Interface.Components` property), 480
- `base()` (`epyk.core.html.Header.Header` method), 627
- `basic()` (`epyk.interfaces.tables.CompTables.Tables` method), 473
- `bb` (`epyk.core.data.Data.DataSrc` property), 564
- `bb` (`epyk.interfaces.graphs.CompCharts.Graphs` property), 356
- `bdi()` (`epyk.interfaces.components.CompTags.Tags` method), 285
- `body()` (`epyk.interfaces.components.CompTags.Tags` method), 285
- `before` (`epyk.core.css.styles.classes.CssStyle.Style` property), 577
- `bespoke()` (`epyk.interfaces.Interface.Components` method), 480
- `Billboard` (class in `epyk.interfaces.graphs.CompChartsBillboard`), 369
- `billboard` (`epyk.interfaces.graphs.CompCharts.Chart2d` property), 353
- `billboard` (`epyk.interfaces.graphs.CompCharts.Graphs` property), 356
- `block()` (`epyk.interfaces.components.CompTexts.Texts` method), 303
- `block()` (`epyk.interfaces.components.CompVignets.Vignets` method), 325
- `blockquote()` (`epyk.interfaces.components.CompTexts.Texts` method), 304
- `body` (`epyk.core.js.Js.JsBase` property), 516, 581
- `body` (`epyk.core.Page.Report` property), 569
- `bold()` (`epyk.interfaces.components.CompTitles.Titles` method), 316
- `book()` (`epyk.interfaces.components.CompTexts.TextReferences` method), 300
- `bot()` (`epyk.interfaces.components.CompNetwork.Network` method), 245
- `bottom` (`epyk.interfaces.components.CompMenus.Menus` method), 214
- `bottom` (`epyk.interfaces.components.CompNavigation.Banners` method), 227
- `box()` (`epyk.interfaces.components.CompLists.Lists` method), 199
- `box()` (`epyk.interfaces.graphs.CompChartsPlotly.Plotly2D` method), 433
- `boxplot()` (`epyk.interfaces.graphs.CompChartsSparkline.Sparkline` method), 449
- `boxes()` (`epyk.interfaces.components.CompPanels.Panels` method), 259
- `br()` (`epyk.interfaces.components.CompLayouts.Layouts` method), 182
- `brackets()` (`epyk.interfaces.components.CompLists.Lists` method), 199
- `breadcrumb` (`epyk.core.js.Js.JsBase` property), 516, 581
- `breadcrumb()` (`epyk.interfaces.Interface.Components` method), 480
- `browser` (`epyk.core.py.PyOuts.PyOuts` property), 538
- `bs` (`epyk.interfaces.Interface.WebComponents` property), 63
- `btoa()` (`epyk.core.js.JsWindow.JsWindow` method), 502
- `bubble()` (`epyk.interfaces.components.CompVignets.Vignets` method), 326
- `bubble()` (`epyk.interfaces.geo.CompGeoPlotly.PlotlyBubble` method), 326

- method*), 346
- `bubble()` (*epyk.interfaces.graphs.CompChartsApex.ApexChart* *method*), 363
- `bubble()` (*epyk.interfaces.graphs.CompChartsBillboard.Billboard* *method*), 371
- `bubble()` (*epyk.interfaces.graphs.CompChartsChartJs.ChartJs* *method*), 390
- `bubble()` (*epyk.interfaces.graphs.CompChartsDc.DC* *method*), 403
- `bubble()` (*epyk.interfaces.graphs.CompChartsGoogle.ChartGoogle* *method*), 413
- `bubble()` (*epyk.interfaces.graphs.CompChartsPlotly.Plotly2D* *method*), 433
- `BubbleMaps` (*class in epyk.interfaces.geo.CompGeoChartJs*), 333
- `bubbles` (*epyk.interfaces.geo.CompGeoPlotly.Plotly* *property*), 343
- `bullet()` (*epyk.interfaces.graphs.CompChartsSparkline.Sparkline* *method*), 449
- `busy` (*epyk.core.html.Aria.Aria* *property*), 636
- `button()` (*epyk.interfaces.components.CompButtons.Buttons* *method*), 73
- `button()` (*epyk.interfaces.components.CompLinks.Links* *method*), 194
- `button()` (*epyk.interfaces.components.CompMenus.Menus* *method*), 214
- `button()` (*epyk.interfaces.components.CompTexts.Texts* *method*), 304
- `Buttons` (*class in epyk.interfaces.components.CompButtons.Buttons*), 72
- `buttons` (*epyk.interfaces.Interface.Components* *property*), 481
- `buttons()` (*epyk.interfaces.components.CompMenus.Menus* *method*), 215
- C**
- `C3` (*class in epyk.interfaces.graphs.CompChartsC3*), 379
- `c3` (*epyk.core.data.Data.DataSrc* *property*), 564
- `c3` (*epyk.interfaces.graphs.CompCharts.Chart2d* *property*), 353
- `c3` (*epyk.interfaces.graphs.CompCharts.Graphs* *property*), 356
- `c3()` (*epyk.interfaces.tables.CompPivot.Pivottable* *method*), 467
- `Calendar` (*class in epyk.interfaces.components.CompCalendar*), 93
- `calendars` (*epyk.interfaces.Interface.Components* *property*), 481
- `camera()` (*epyk.interfaces.components.CompMedia.Media* *method*), 211
- `cancel()` (*epyk.interfaces.components.CompButtons.Buttons* *method*), 74
- `candlestick()` (*epyk.interfaces.graphs.CompChartsGoogle.ChartGoogle* *method*), 414
- `candlestick()` (*epyk.interfaces.graphs.CompChartsNvd3.Nvd3* *method*), 422
- `candlestick()` (*epyk.interfaces.graphs.CompChartsPlotly.Plotly2D* *method*), 434
- `Canvas` (*class in epyk.interfaces.graphs.CompChartsCanvas*), 386
- `canvas` (*epyk.interfaces.graphs.CompCharts.Chart2d* *property*), 353
- `canvas` (*epyk.interfaces.graphs.CompCharts.Graphs* *property*), 357
- `captcha()` (*epyk.interfaces.Interface.Components* *method*), 481
- `caption()` (*epyk.interfaces.components.CompTitles.Titles* *method*), 316
- `capture()` (*epyk.interfaces.components.CompIcons.Icons* *method*), 132
- `carousel()` (*epyk.interfaces.components.CompImages.Images* *method*), 161
- `categories()` (*epyk.interfaces.components.CompFields.Timelines* *method*), 124
- `category()` (*epyk.interfaces.components.CompTitles.Titles* *method*), 317
- `ceil()` (*epyk.core.js.Js.JsMaths.JsMaths* *method*), 609
- `cell()` (*epyk.interfaces.components.CompInputs.Inputs* *method*), 169
- `centered()` (*epyk.interfaces.components.CompLayouts.Layouts* *method*), 183
- `charset()` (*epyk.core.html.Header.Meta* *method*), 634
- `Chart2d` (*class in epyk.interfaces.graphs.CompCharts*), 352
- `Chart3d` (*class in epyk.interfaces.graphs.CompCharts*), 355
- `chartCss` (*epyk.interfaces.graphs.CompCharts.Graphs* *property*), 357
- `ChartGoogle` (*class in epyk.interfaces.graphs.CompChartsGoogle*), 412
- `ChartJs` (*class in epyk.interfaces.geo.CompGeoChartJs*), 334
- `ChartJs` (*class in epyk.interfaces.graphs.CompChartsChartJs*), 389
- `chartJs` (*epyk.core.data.Data.DataSrc* *property*), 564
- `chartJs` (*epyk.interfaces.geo.CompGeo.Geo* *property*), 331
- `chartJs` (*epyk.interfaces.graphs.CompCharts.Chart2d* *property*), 353
- `chartJs` (*epyk.interfaces.graphs.CompCharts.Graphs* *property*), 357
- `charts` (*epyk.interfaces.Interface.Components* *property*), 481
- `chat()` (*epyk.interfaces.components.CompNetwork.Network* *method*), 245
- `check()` (*epyk.interfaces.components.CompButtons.Buttons* *method*), 75

checkbox() (*epyk.interfaces.components.CompFields.Fields method*), 134
checkbox() (*epyk.interfaces.components.CompInputs.Inputs method*), 170
checkboxes() (*epyk.interfaces.components.CompButtons.Buttons method*), 75
checked (*epyk.core.css.styles.classes.CssStyle.Style property*), 578
checked (*epyk.core.html.Aria.Aria property*), 637
checks() (*epyk.interfaces.components.CompLists.Lists method*), 200
chips() (*epyk.interfaces.components.CompLists.Lists method*), 200
chorolet() (*epyk.interfaces.geo.CompGeoPlotly.Plotly method*), 343
Choropleth (*class in epyk.interfaces.geo.CompGeoChartJs*), 334
choropleths (*epyk.interfaces.geo.CompGeoPlotly.Plotly property*), 343
circle() (*epyk.interfaces.graphs.CompChartsSvg.SVG method*), 453
circular() (*epyk.interfaces.components.CompImages.Images method*), 161
cite() (*epyk.interfaces.components.CompTags.Tags method*), 286
ClassPage (*class in epyk.core.css.styles.GrpCls*), 575
cleanImports() (*epyk.core.js.Imports.ImportManager method*), 57
cleanState() (*epyk.core.js.JsWindow.JsHistory method*), 495
clear (*epyk.core.js.Js.JsConsole property*), 492, 602
clear() (*epyk.core.js.JsWindow.JsLocalStorage method*), 498
clear() (*epyk.core.js.JsWindow.JsSessionStorage method*), 500
clear() (*epyk.core.py.PySql.SqlConnNeo4j method*), 557
clear() (*epyk.interfaces.components.CompButtons.Buttons method*), 76
clear() (*epyk.interfaces.components.CompIcons.Icons method*), 133
clearInterval() (*epyk.core.js.JsWindow.JsWindow method*), 503
clearMarks() (*epyk.core.js.Js.JsPerformance.JsPerformance method*), 623
clearMeasures() (*epyk.core.js.Js.JsPerformance.JsPerformance method*), 624
clearResourceTimings() (*epyk.core.js.Js.JsPerformance.JsPerformance method*), 624
clearTimeout() (*epyk.core.js.JsWindow.JsWindow method*), 503
clipboard() (*epyk.core.js.Js.JsBase method*), 516, 581
clock() (*epyk.interfaces.components.CompIcons.Icons method*), 134
close() (*epyk.core.js.Js.JsWebSocket.WebSocket method*), 620
close() (*epyk.core.js.JsWindow.JsWindow method*), 503
close() (*epyk.interfaces.graphs.CompChartsD3.D3 method*), 400
clr (*epyk.interfaces.Interface.WebComponents property*), 63
cob (*epyk.core.py.PyDates.PyDates property*), 541
cob() (*epyk.interfaces.components.CompFields.Fields method*), 107
Code (*class in epyk.interfaces.components.CompCodes*), 98
code() (*epyk.interfaces.components.CompCodes.Code method*), 98
code() (*epyk.interfaces.components.CompTexts.Texts method*), 305
codepen() (*epyk.core.py.PyOuts.OutBrowsers method*), 537
codepen() (*epyk.core.py.PyOuts.PyOuts method*), 538
codes (*epyk.interfaces.Interface.Components property*), 482
col() (*epyk.interfaces.components.CompLayouts.Layouts method*), 183
col() (*epyk.interfaces.components.CompTexts.Texts method*), 305
colcount (*epyk.core.html.Aria.Aria property*), 637
colindex (*epyk.core.html.Aria.Aria property*), 637
collapse() (*epyk.interfaces.components.CompIcons.Toggles method*), 156
color() (*epyk.interfaces.components.CompImages.Images method*), 162
color() (*epyk.interfaces.components.CompRich.Rich method*), 270
colorDepth (*epyk.core.js.Js.JsScreen property*), 607
colored() (*epyk.interfaces.components.CompButtons.Buttons method*), 77
colored() (*epyk.interfaces.components.CompLinks.Links method*), 194
colspan (*epyk.core.html.Aria.Aria property*), 637
column() (*epyk.core.py.PySql.SqlConn method*), 550
column() (*epyk.interfaces.graphs.CompChartsGoogle.ChartGoogle method*), 414
column_date() (*epyk.interfaces.components.CompFields.Fields method*), 108
column_text() (*epyk.interfaces.components.CompFields.Fields method*), 108
columns() (*epyk.core.py.PySql.SqlConn method*), 551
columns() (*epyk.interfaces.components.CompLayouts.Layouts method*), 184
comment() (*epyk.interfaces.components.CompTags.Tags method*), 286
comments() (*epyk.interfaces.components.CompNetwork.Network method*), 245

`commit()` (*epyk.core.py.PySql.SqlConn* method), 551
`compass()` (*epyk.interfaces.components.CompPictos.Pictos* method), 266
`CompChartCss` (class in *epyk.interfaces.graphs.CompChartsChartCss*), 387
`CompChartFrappe` (class in *epyk.interfaces.graphs.CompChartsFrappe*), 409
`compile()` (in module *epyk.core.cli.cli_project*), 71
`Components` (class in *epyk.interfaces.Interface*), 479
`compose()` (*epyk.core.py.PySql.SqlConnNeo4j* method), 557
`composite()` (*epyk.interfaces.components.CompRich.Rich* method), 270
`CompRoughViz` (class in *epyk.interfaces.graphs.CompChartsRoughViz*), 445
`connect()` (*epyk.core.js.Js.JsWebSocket.WebSocket* method), 621
`console()` (*epyk.interfaces.components.CompRich.Rich* method), 271
`contact_us()` (*epyk.interfaces.components.CompNavigation.Banners* method), 228
`container()` (*epyk.interfaces.components.CompImages.Images* method), 162
`contenteditable()` (*epyk.core.css.styles.GrpCls.ClassPage* method), 575
`contents()` (*epyk.interfaces.Interface.Components* method), 482
`contextual()` (*epyk.interfaces.components.CompMenus.Menus* method), 215
`control()` (*epyk.core.html.KeyCodes.KeyCode* method), 645
`controls` (*epyk.core.html.Aria.Aria* property), 637
`cookies()` (*epyk.interfaces.components.CompNavigation.Banners* method), 228
`corner()` (*epyk.interfaces.components.CompNavigation.Banners* method), 229
`cos()` (*epyk.core.js.Js.JsMaths.JsMaths* method), 610
`count` (*epyk.core.py.PySql.SqlConn* property), 551
`countdown()` (*epyk.interfaces.components.CompRich.Rich* method), 271
`country()` (*epyk.interfaces.geo.CompGeoChartJs.Choropleth* method), 334
`createAttribute()` (*epyk.core.js.Js.JsBase* method), 517, 582
`createElement()` (*epyk.core.js.Js.JsBase* method), 517, 582
`createEvent()` (*epyk.core.js.Js.JsBase* method), 517, 582
`createObjectURL()` (*epyk.core.js.Js.Window.JsUrl* method), 501
`createTextNode()` (*epyk.core.js.Js.JsBase* static method), 517, 582
`cryptoKeyPairs()` (*epyk.core.py.PyCrypto.PyCrypto* class method), 558
`crypto` (*epyk.core.py.PyExt.PyExt* property), 559
`css` (*epyk.core.css.styles.GrpCls.ClassPage* property), 575
`css` (*epyk.core.Page.Report* property), 570
`css()` (*epyk.core.css.styles.classes.CssStyle.Style* method), 578
`css()` (*epyk.interfaces.components.CompCodes.Code* method), 98
`css()` (*epyk.interfaces.Interface.Components* method), 483
`css_class` (*epyk.core.css.styles.GrpCls.ClassPage* property), 575
`cssGetAll()` (*epyk.core.js.Imports.ImportManager* method), 57
`cssResolve()` (*epyk.core.js.Imports.ImportManager* method), 57
`cssURLs()` (*epyk.core.js.Imports.ImportManager* method), 57
`csv()` (*epyk.core.py.PyRest.PyRest* static method), 547
`current` (*epyk.core.html.Aria.Aria* property), 637
`current()` (*epyk.interfaces.geo.CompGeoGoogle.GeoGoogle* method), 337
`custom()` (*epyk.core.html.Aria.Aria* method), 638
`custom()` (*epyk.core.html.Header.Meta* method), 634
`custom()` (*epyk.core.html.KeyCodes.KeyCode* method), 645
`custom()` (*epyk.core.js.Js.JsBase* method), 518, 583
`custom()` (*epyk.interfaces.graphs.CompChartsChartJs.ChartJs* method), 390
`custom_class()` (*epyk.core.css.styles.GrpCls.ClassPage* method), 575
`customFile()` (*epyk.core.js.Js.JsBase* method), 518, 583
`customize()` (*epyk.core.css.styles.classes.CssStyle.Style* method), 578
`customText()` (*epyk.core.js.Js.JsBase* method), 519, 584

D

`D3` (class in *epyk.interfaces.geo.CompGeoD3*), 336
`D3` (class in *epyk.interfaces.graphs.CompChartsD3*), 400
`D3` (class in *epyk.interfaces.tables.CompTableD3*), 471
`d3` (*epyk.core.js.Js.JsBase* property), 519, 584
`d3` (*epyk.interfaces.geo.CompGeo.Geo* property), 331
`d3` (*epyk.interfaces.graphs.CompCharts.Chart2d* property), 354
`d3` (*epyk.interfaces.graphs.CompCharts.Graphs* property), 357
`d3` (*epyk.interfaces.tables.CompTables.Tables* property), 473

d3() (epyk.interfaces.tables.CompPivot.Pivottable method), 468
 d_date() (epyk.interfaces.components.CompInputs.Inputs method), 170
 d_int() (epyk.interfaces.components.CompInputs.Inputs method), 171
 d_radio() (epyk.interfaces.components.CompInputs.Inputs method), 171
 d_range() (epyk.interfaces.components.CompInputs.Inputs method), 172
 d_search() (epyk.interfaces.components.CompInputs.Inputs method), 172
 d_text() (epyk.interfaces.components.CompInputs.Inputs method), 173
 d_time() (epyk.interfaces.components.CompInputs.Inputs method), 173
 danger() (epyk.interfaces.components.CompIcons.Icons method), 134
 danger() (epyk.interfaces.components.CompNetwork.Network method), 246
 dark() (epyk.interfaces.components.CompNavigation.Navigation method), 234
 dashed() (epyk.interfaces.components.CompLayouts.Delimiters method), 181
 data (epyk.core.js.Js.JsBase property), 519, 584
 data (epyk.core.Page.Report property), 570
 data() (epyk.core.py.PySql.SqlConn method), 551
 data() (epyk.interfaces.components.CompButtons.Buttons method), 78
 data() (epyk.interfaces.components.CompLinks.Links method), 195
 DataJs (class in epyk.core.data.Data), 563
 DataSrc (class in epyk.core.data.Data), 564
 Datatables (class in epyk.interfaces.tables.CompDatatables), 467
 datatables (epyk.interfaces.tables.CompTables.Tables property), 473
 date() (epyk.interfaces.components.CompFields.Fields method), 108
 date() (epyk.interfaces.components.CompForms.Forms method), 127
 date() (epyk.interfaces.components.CompIcons.Icons method), 135
 date() (epyk.interfaces.components.CompSliders.Sliders method), 278
 date() (epyk.interfaces.components.CompTexts.Texts method), 306
 date_from_alias() (epyk.core.py.PyDates.PyDates method), 542
 date_from_excel() (epyk.core.py.PyDates.PyDates static method), 542
 date_range() (epyk.interfaces.components.CompSliders.Sliders method), 279
 dates (epyk.core.py.PyExt.PyExt property), 560
 dates() (epyk.interfaces.components.CompForms.Forms method), 128
 days() (epyk.interfaces.components.CompCalendars.Calendar method), 93
 days() (epyk.interfaces.components.CompFields.Fields method), 109
 db (epyk.core.data.Data.DataSrc property), 564
 Dc (class in epyk.interfaces.geo.CompGeoDc), 336
 DC (class in epyk.interfaces.graphs.CompChartsDc), 403
 dc (epyk.interfaces.graphs.CompCharts.Chart2d property), 354
 dc (epyk.interfaces.graphs.CompCharts.Graphs property), 357
 debugger (epyk.core.js.Js.JsConsole property), 492, 602
 decodeURIComponent() (epyk.core.js.Js.JsBase method), 519, 584
 decrypt() (epyk.core.py.PyCrypto.PyCrypto method), 558
 decryptKeyPairs() (epyk.core.py.PyCrypto.PyCrypto class method), 558
 Defaults (epyk.core.css.styles.GrpCls.ClassPage property), 576
 define_classes (epyk.core.css.styles.GrpCls.ClassPage property), 576
 delay() (epyk.core.js.Js.JsBase method), 519, 584
 delete() (epyk.core.html.KeyCodes.KeyCode method), 645
 delete() (epyk.core.js.Js.JsBase method), 520, 585
 delete() (epyk.core.js.Js.JsLocation.URLSearchParams method), 618
 delete() (epyk.core.js.JsLocation.URLSearchParams method), 515
 delete() (epyk.core.py.PySql.SqlConn method), 551
 delete() (epyk.interfaces.components.CompIcons.Icons method), 135
 delete() (epyk.interfaces.components.CompTags.Tags method), 286
 deleteState() (epyk.core.js.JsWindow.JsHistory method), 496
 Delimiter (class in epyk.interfaces.components.CompLayouts), 181
 delimiters (epyk.interfaces.Interface.Components property), 483
 delta() (epyk.core.py.PyDates.PyDates static method), 542
 delta() (epyk.interfaces.components.CompRich.Rich method), 272
 demo() (in module epyk.core.cli.cli_export), 67
 density() (epyk.interfaces.geo.CompGeoPlotly.Plotly method), 344
 describedby (epyk.core.html.Aria.Aria property), 638
 description() (epyk.core.html.Header.Meta method), 635
 descriptions() (epyk.core.py.PyNpm.Packages class

- method), 547
- details (epyk.core.html.Aria.Aria property), 638
- dev() (epyk.core.html.Header.Header method), 628
- dfn() (epyk.interfaces.components.CompTags.Tags method), 287
- dialog() (epyk.interfaces.components.CompModals.Modals method), 221
- dialogs() (epyk.interfaces.components.CompLayouts.Layouts method), 184
- digits() (epyk.interfaces.components.CompNumbers.Numbers method), 251
- disabled (epyk.core.css.styles.classes.CssStyle.Style property), 578
- disabled (epyk.core.html.Aria.Aria property), 638
- disc() (epyk.interfaces.components.CompLists.Lists method), 201
- disclaimer() (epyk.interfaces.components.CompModals.Modals method), 222
- disclaimer() (epyk.interfaces.components.CompNavigation.Banners method), 229
- discrete() (epyk.interfaces.graphs.CompChartsSparkline.Sparkline method), 449
- distance() (epyk.core.py.PyGeo.PyGeo static method), 562
- distinct() (epyk.core.py.PySql.SqlConn method), 552
- div() (epyk.interfaces.components.CompLayouts.Layouts method), 185
- divisor() (epyk.interfaces.components.CompMenus.Menus method), 216
- dns_prefetch() (epyk.core.html.Header.Links method), 630
- document (epyk.core.js.JsWindow.JsWindow property), 503
- documentElement (epyk.core.js.Js.JsBase property), 520, 585
- dollar() (epyk.interfaces.components.CompNumbers.Numbers method), 252
- donut() (epyk.interfaces.graphs.CompChartsApex.ApexChart method), 363
- donut() (epyk.interfaces.graphs.CompChartsBillboard.Billboard method), 371
- donut() (epyk.interfaces.graphs.CompChartsC3.C3 method), 380
- donut() (epyk.interfaces.graphs.CompChartsChartJs.ChartJs method), 391
- donut() (epyk.interfaces.graphs.CompChartsFrappe.CompChartFrappe method), 409
- donut() (epyk.interfaces.graphs.CompChartsGoogle.ChartGoogle method), 415
- donut() (epyk.interfaces.graphs.CompChartsNvd3.Nvd3 method), 423
- donut() (epyk.interfaces.graphs.CompChartsPlotly.Plotly2D method), 435
- donut() (epyk.interfaces.graphs.CompChartsRoughViz.CompRoughViz method), 445
- dots() (epyk.interfaces.components.CompNavigation.Navigation method), 236
- dotted() (epyk.interfaces.components.CompLayouts.Delimiter method), 181
- double() (epyk.interfaces.components.CompLayouts.Delimiter method), 181
- down() (epyk.core.html.KeyCodes.KeyCode method), 645
- down() (epyk.interfaces.components.CompNavigation.Navigation method), 237
- download() (epyk.core.js.Js.JsLocation.JsLocation class method), 614
- download() (epyk.core.js.JsLocation.JsLocation class method), 510
- download() (epyk.core.js.JsWindow.JsWindow method), 504
- download() (epyk.interfaces.components.CompIcons.Icons method), 136
- download() (epyk.interfaces.components.CompLinks.Links method), 195
- download() (epyk.interfaces.components.CompNetwork.Network method), 247
- drawer() (epyk.interfaces.components.CompDrawers.Drawers method), 103
- Drawers (class in epyk.interfaces.components.CompDrawers), 103
- drawers (epyk.interfaces.Interface.Components property), 483
- drop() (epyk.core.py.PySql.SqlConn method), 552
- drop() (epyk.interfaces.components.CompLists.Lists method), 202
- dropdown() (epyk.interfaces.components.CompLists.Lists method), 202
- dropdown() (epyk.interfaces.components.CompTrees.Trees method), 322
- dropfile() (epyk.interfaces.components.CompNetwork.Network method), 247
- dumps() (epyk.core.Page.Report method), 570
- ## E
- E (epyk.core.js.Js.JsMaths.JsMaths property), 608
- edit() (epyk.interfaces.components.CompIcons.Icons method), 136
- editor() (epyk.interfaces.components.CompInputs.Inputs method), 174
- elapsed() (epyk.core.py.PyDates.PyDates static method), 542
- elapsed() (epyk.interfaces.components.CompRich.Rich method), 273
- ellipse() (epyk.interfaces.graphs.CompChartsSvg.SVG method), 453
- em() (epyk.interfaces.components.CompTags.Tags method), 287

- Email (class in *epyk.core.py.PyMails*), 546
- emoji() (*epyk.interfaces.components.CompImages.Images* method), 163
- empty (*epyk.core.css.styles.classes.CssStyle.Style* property), 578
- enabled (*epyk.core.css.styles.classes.CssStyle.Style* property), 578
- encode_html() (*epyk.core.py.PyExt.PyExt* static method), 560
- encodeURIComponent() (*epyk.core.js.Js.JsBase* method), 520, 585
- encrypt() (*epyk.core.py.PyCrypto.PyCrypto* method), 559
- enter() (*epyk.core.html.KeyCodes.KeyCode* method), 645
- entities (*epyk.core.Page.Report* property), 570
- epyk() (*epyk.interfaces.components.CompIcons.Icons* method), 137
- epyk() (*epyk.interfaces.components.CompImages.Images* method), 163
- epyk.core.cli.cli_export module, 67
- epyk.core.cli.cli_npm module, 68
- epyk.core.cli.cli_project module, 70
- epyk.core.data.Data.DataPy module, 568
- epyk.core.js.JsLocation module, 510
- epyk.core.js.JsWindow module, 495
- epyk.core.py.PyOuts module, 537
- error() (*epyk.core.js.Js.JsConsole* method), 492, 603
- error() (*epyk.interfaces.components.CompIcons.Icons* method), 137
- error() (*epyk.interfaces.components.CompModals.Modals* method), 222
- errorMessage (*epyk.core.html.Aria.Aria* property), 638
- escape() (*epyk.core.html.KeyCodes.KeyCode* method), 646
- euro() (*epyk.interfaces.components.CompNumbers.Numbers* method), 252
- europe() (*epyk.interfaces.geo.CompGeoJqV.JqueryVetorMap* method), 339
- europe() (*epyk.interfaces.geo.CompGeoPlotly.PlotlyBubble* method), 346
- europe() (*epyk.interfaces.geo.CompGeoPlotly.PlotlyChoropleth* method), 350
- eval() (*epyk.core.js.Js.JsBase* method), 520, 585
- events (*epyk.core.js.JsWindow.JsWindow* property), 504
- evr (*epyk.interfaces.Interface.WebComponents* property), 64
- excel() (*epyk.interfaces.components.CompIcons.Icons* method), 138
- execCommand() (*epyk.core.js.Js.JsBase* method), 521, 586
- execute() (*epyk.core.py.PySql.SqlConn* method), 552
- exp() (*epyk.core.js.Js.JsMaths.JsMaths* method), 610
- expanded (*epyk.core.html.Aria.Aria* property), 639
- extend() (*epyk.core.js.Imports.ImportManager* method), 58
- extendProto() (*epyk.core.js.Js.JsBase* method), 521, 586
- extension() (*epyk.interfaces.Interface.Components* method), 483
- external() (*epyk.interfaces.components.CompLinks.Links* method), 196
- ## F
- fabric() (*epyk.interfaces.graphs.CompChartsChartJs.ChartJs* method), 391
- facebook() (*epyk.interfaces.components.CompIcons.Icons* method), 138
- faq() (*epyk.interfaces.components.CompPictos.Pictogram* method), 266
- favicon() (*epyk.core.html.Header.Header* method), 628
- fetch() (*epyk.core.js.Js.JsBase* method), 521, 586
- Fields (class in *epyk.interfaces.components.CompFields*), 106
- fields (*epyk.interfaces.Interface.Components* property), 483
- fieldset() (*epyk.interfaces.components.CompTexts.Texts* method), 307
- figcaption() (*epyk.interfaces.components.CompTags.Tags* method), 288
- figure() (*epyk.interfaces.components.CompImages.Images* method), 164
- figures() (*epyk.interfaces.tables.CompTabulator.Tabulators* method), 476
- file() (*epyk.interfaces.components.CompFields.Fields* method), 110
- file() (*epyk.interfaces.components.CompInputs.Inputs* method), 174
- filter() (*epyk.interfaces.components.CompButtons.Buttons* method), 79
- filters() (*epyk.interfaces.components.CompFields.Fields* method), 110
- filters() (*epyk.interfaces.components.CompInputs.Inputs* method), 175
- filters() (*epyk.interfaces.components.CompPanels.Panels* method), 259
- first() (*epyk.core.py.PySql.SqlConn* method), 552
- fit_screen_height() (*epyk.core.css.styles.GrpCls.ClassPage* method), 576

- fixed() (epyk.interfaces.components.CompIcons.Icons method), 139
- fixed() (epyk.interfaces.components.CompNavigation.Navigation method), 234
- flam() (epyk.interfaces.components.CompPictos.Pictogram method), 267
- floor() (epyk.core.js.Js.JsMaths.JsMaths method), 610
- flowto (epyk.core.html.Aria.Aria property), 639
- fluent() (epyk.interfaces.components.CompIcons.Icons method), 139
- fncs (epyk.core.js.Js.JsBase property), 522, 587
- focus (epyk.core.css.styles.classes.CssStyle.Style property), 578
- focus() (epyk.core.js.Js.JsWindow.JsWindow method), 504
- folder() (epyk.interfaces.components.CompTrees.Trees method), 323
- follow() (epyk.interfaces.components.CompNavigation.Banners method), 230
- footer() (epyk.interfaces.components.CompNavigation.Navigation method), 237
- for_() (epyk.core.js.Js.JsBase method), 522, 587
- force_create() (epyk.core.py.PySql.SqlConn method), 553
- forceDirected() (epyk.interfaces.graphs.CompChartsNvd3.Nvd3 method), 423
- foreach() (epyk.core.py.PySql.SqlConnNeo4j method), 557
- forecast() (epyk.interfaces.components.CompCalendars.Calendar method), 94
- form() (epyk.interfaces.components.CompLayouts.Layouts method), 186
- form() (epyk.interfaces.Interface.Components method), 483
- format_money() (epyk.core.py.PyExt.PyExt static method), 560
- format_number() (epyk.core.py.PyExt.PyExt static method), 560
- Forms (class in epyk.interfaces.components.CompForms), 127
- forms (epyk.interfaces.Interface.Components property), 484
- forms() (epyk.interfaces.components.CompModals.Modals method), 223
- formula() (epyk.interfaces.components.CompTexts.Texts method), 307
- forward() (epyk.core.js.Js.JsWindow.JsHistory method), 496
- framework() (epyk.core.Page.Report method), 571
- france() (epyk.interfaces.geo.CompGeoJqV.JqueryVectorMap method), 340
- frappe (epyk.interfaces.graphs.CompCharts.Graphs property), 358
- from_cache() (epyk.core.data.Data.DataSrc method), 564
- from_file() (epyk.core.data.Data.DataSrc method), 564
- from_get() (epyk.core.data.Data.DataSrc method), 565
- from_source() (epyk.core.data.Data.DataSrc method), 565
- from_timestamp() (epyk.core.py.PyDates.PyDates static method), 543
- ftw (epyk.interfaces.Interface.WebComponents property), 64
- ## G
- gallery() (epyk.interfaces.components.CompIcons.Icons method), 140
- gallery() (epyk.interfaces.components.CompImages.Images method), 164
- gauge() (epyk.interfaces.graphs.CompChartsApex.ApexChart method), 364
- gauge() (epyk.interfaces.graphs.CompChartsBillboard.Billboard method), 372
- gauge() (epyk.interfaces.graphs.CompChartsC3.C3 method), 381
- gauge() (epyk.interfaces.graphs.CompChartsGoogle.ChartGoogle method), 416
- gauge() (epyk.interfaces.graphs.CompChartsNvd3.Nvd3 method), 424
- gauge() (epyk.interfaces.graphs.CompChartsPlotly.Plotly2D method), 435
- Geo (class in epyk.interfaces.geo.CompGeo), 331
- geo (epyk.core.py.PyExt.PyExt property), 560
- geo (epyk.interfaces.Interface.Components property), 484
- geo() (epyk.interfaces.graphs.CompChartsGoogle.ChartGoogle method), 416
- GeoGoogle (class in epyk.interfaces.geo.CompGeoGoogle), 337
- GeoLeaflet (class in epyk.interfaces.geo.CompGeoLeaflet), 343
- germany() (epyk.interfaces.geo.CompGeoJqV.JqueryVectorMap method), 340
- get() (epyk.core.html.Aria.Aria method), 639
- get() (epyk.core.js.Js.JsBase method), 522, 587
- get() (epyk.core.js.Js.JsBreadcrumb method), 606
- get() (epyk.core.js.Js.JsLocation.URLSearchParams method), 618
- get() (epyk.core.js.Js.JsLocation.URLSearchParams method), 515
- get() (epyk.core.py.PyRest.PyRest method), 547
- get_classes() (epyk.core.css.styles.GrpCls.ClassPage method), 576
- get_classes_css() (epyk.core.css.styles.GrpCls.ClassPage method), 576
- get_components() (epyk.core.Page.Report method), 571

get_event() (epyk.core.html.KeyCodes.KeyCode method), 646
 get_last_id() (epyk.core.py.PySql.SqlConn method), 553
 get_ref() (epyk.core.css.styles.classes.CssStyle.Style method), 578
 getAll() (epyk.core.js.Js.JsLocation.URLSearchParams method), 619
 getAll() (epyk.core.js.Js.JsLocation.URLSearchParams method), 515
 getComputedStyle() (epyk.core.js.Js.JsWindow.JsWindow method), 504
 getElementById() (epyk.core.js.Js.JsBase static method), 523, 588
 getElementsByClassName() (epyk.core.js.Js.JsBase static method), 523, 588
 getElementsByName() (epyk.core.js.Js.JsBase static method), 523, 588
 getElementsByTagName() (epyk.core.js.Js.JsBase static method), 524, 589
 getEntries() (epyk.core.js.Js.JsPerformance.JsPerformance method), 624
 getEntriesByName() (epyk.core.js.Js.JsPerformance.JsPerformance method), 186
 getEntriesByType() (epyk.core.js.Js.JsPerformance.JsPerformance method), 473
 getFiles() (epyk.core.js.Imports.ImportManager method), 58
 getFullPackage() (epyk.core.js.Imports.ImportManager method), 58
 getId (epyk.core.py.PyCrypto.PyCrypto property), 559
 getItem() (epyk.core.js.Js.JsWindow.JsLocalStorage method), 498
 getItem() (epyk.core.js.Js.JsWindow.JsSessionStorage method), 500
 getModules() (epyk.core.js.Imports.ImportManager method), 58
 getReq() (epyk.core.js.Imports.ImportManager method), 59
 getSelection() (epyk.core.js.Js.JsWindow.JsWindow method), 504
 getUrlFromArrays() (epyk.core.js.Js.JsLocation.JsLocation class method), 614
 getUrlFromArrays() (epyk.core.js.Js.JsLocation.JsLocation class method), 510
 getUrlFromData() (epyk.core.js.Js.JsLocation.JsLocation class method), 614
 getUrlFromData() (epyk.core.js.Js.JsLocation.JsLocation class method), 510
 getVar() (epyk.core.js.Js.JsBase method), 524, 589
 getVar() (epyk.core.js.Js.JsWindow.JsWindow method), 505
 gif() (epyk.core.html.Header.Icons method), 629
 github() (epyk.interfaces.components.CompIcons.Icons method), 140
 github() (epyk.interfaces.components.CompTexts.TextReferences method), 301
 globals (epyk.core.css.styles.GrpCls.ClassPage property), 576
 go() (epyk.core.js.Js.JsWindow.JsHistory method), 496
 Google (class in epyk.interfaces.tables.CompTableGoogle), 472
 google (epyk.core.data.Data.DataSrc property), 565
 google (epyk.interfaces.geo.CompGeo.Geo property), 331
 google (epyk.interfaces.graphs.CompCharts.Graphs property), 358
 google (epyk.interfaces.tables.CompTables.Tables property), 473
 google() (epyk.interfaces.components.CompCalendars.Calendar method), 94
 google_products() (epyk.core.js.Imports.ImportManager method), 59
 Graphs (class in epyk.interfaces.graphs.CompCharts), 356
 grid() (epyk.interfaces.components.CompLayouts.Layouts method), 186
 grid() (epyk.interfaces.tables.CompTables.Tables method), 473
 group_box() (epyk.interfaces.graphs.CompChartsNvd3.Nvd3 method), 424
 group_box() (epyk.interfaces.graphs.CompChartsPlotly.Plotly2D method), 436
 groups() (epyk.interfaces.components.CompLists.Lists method), 203
 grpc() (epyk.core.data.Data.DataSrc method), 565
H
 h1() (epyk.interfaces.components.CompTags.Tags method), 288
 h2() (epyk.interfaces.components.CompTags.Tags method), 288
 h3() (epyk.interfaces.components.CompTags.Tags method), 289
 hamburger() (epyk.interfaces.components.CompIcons.Icons method), 141
 hamburger() (epyk.interfaces.components.CompPanels.Panels method), 260
 has() (epyk.core.js.Js.JsLocation.URLSearchParams method), 619
 has() (epyk.core.js.Js.JsLocation.URLSearchParams method), 515
 has_changed (epyk.core.css.styles.classes.CssStyle.Style property), 578
 hash (epyk.core.js.Js.JsLocation.JsLocation property), 614
 hash (epyk.core.js.Js.JsLocation.JsLocation property), 510
 hash() (epyk.core.js.Js.JsBreadCrumb method), 606

- [hashId\(\)](#) (epyk.core.py.PyHash.SipHash method), 562
[haspopup](#) (epyk.core.html.Aria.Aria property), 639
[hbar\(\)](#) (epyk.interfaces.graphs.CompChartsApex.ApexChart method), 364
[hbar\(\)](#) (epyk.interfaces.graphs.CompChartsBillboard.Billboard method), 372
[hbar\(\)](#) (epyk.interfaces.graphs.CompChartsC3.C3 method), 381
[hbar\(\)](#) (epyk.interfaces.graphs.CompChartsChartJs.Chart method), 392
[hbar\(\)](#) (epyk.interfaces.graphs.CompChartsDc.DC method), 404
[hbar\(\)](#) (epyk.interfaces.graphs.CompChartsGoogle.Chart method), 417
[hbar\(\)](#) (epyk.interfaces.graphs.CompChartsNvd3.Nvd3 method), 425
[hbar\(\)](#) (epyk.interfaces.graphs.CompChartsPlotly.Plotly2D method), 436
[hbar\(\)](#) (epyk.interfaces.graphs.CompChartsRoughViz.Comp method), 446
[head\(\)](#) (epyk.interfaces.components.CompTitles.Titles method), 317
[Header](#) (class in epyk.core.html.Header), 627
[header\(\)](#) (epyk.interfaces.components.CompLayouts.Layouts method), 187
[headers](#) (epyk.core.Page.Report property), 571
[headline\(\)](#) (epyk.interfaces.components.CompTitles.Titles method), 318
[heart\(\)](#) (epyk.interfaces.graphs.CompChartsSvg.SVG method), 454
[heatmap\(\)](#) (epyk.interfaces.graphs.CompChartsApex.ApexChart method), 365
[heatmap\(\)](#) (epyk.interfaces.graphs.CompChartsFrappe.Comp method), 410
[heatmap\(\)](#) (epyk.interfaces.tables.CompPivot.Pivottable method), 468
[height](#) (epyk.core.js.Js.JsScreen property), 607
[HexColors](#) (class in epyk.core.css.Colors), 580
[hidden](#) (epyk.core.html.Aria.Aria property), 639
[hidden\(\)](#) (epyk.interfaces.components.CompFields.Fields method), 111
[hidden\(\)](#) (epyk.interfaces.components.CompInputs.Inputs method), 175
[hierarchical\(\)](#) (epyk.interfaces.graphs.CompChartsChartJs.Chart method), 392
[hierarchy\(\)](#) (epyk.interfaces.tables.CompTabulator.Tabulator method), 476
[highlights\(\)](#) (epyk.interfaces.components.CompTexts.Text method), 308
[histo\(\)](#) (epyk.interfaces.graphs.CompChartsNvd3.Nvd3 method), 425
[histogram\(\)](#) (epyk.interfaces.graphs.CompChartsGoogle.Chart method), 417
[histogram\(\)](#) (epyk.interfaces.graphs.CompChartsPlotly.Plotly2D method), 437
[history](#) (epyk.core.js.JsWindow.JsWindow property), 505
[hn\(\)](#) (epyk.interfaces.components.CompTags.Tags method), 289
[host](#) (epyk.core.js.Js.JsLocation.JsLocation property), 615
[host](#) (epyk.core.js.JsLocation.JsLocation property), 511
[hostname](#) (epyk.core.js.Js.JsLocation.JsLocation property), 615
[hostname](#) (epyk.core.js.JsLocation.JsLocation property), 511
[hover](#) (epyk.core.css.styles.classes.CssStyle.Style property), 579
[hr\(\)](#) (epyk.interfaces.components.CompLayouts.Layouts method), 187
[href\(\)](#) (epyk.core.js.Js.JsLocation.JsLocation class method), 615
[href\(\)](#) (epyk.core.js.JsLocation.JsLocation class method), 511
[html\(\)](#) (epyk.core.py.PyOuts.PyOuts method), 538
[html\(\)](#) (in module epyk.core.cli.cli_export), 67
[html_file\(\)](#) (epyk.core.py.PyOuts.PyOuts method), 539
[http_codes](#) (epyk.core.js.Js.JsWebSocket.WebSocket property), 621
[http_equiv\(\)](#) (epyk.core.html.Header.Meta method), 635
[http_server\(\)](#) (epyk.core.py.PyRest.PyRest method), 548
[icon\(\)](#) (epyk.core.html.Header.Icons method), 630
[icon\(\)](#) (epyk.core.html.Header.Links method), 631
[icon\(\)](#) (epyk.interfaces.components.CompImages.Images method), 165
[icon\(\)](#) (epyk.interfaces.components.CompModals.Modals method), 223
[Icons](#) (class in epyk.core.html.Header), 629
[Icons](#) (class in epyk.interfaces.components.CompIcons), 130
[icons](#) (epyk.core.html.Header.Header property), 628
[icons](#) (epyk.core.Page.Report property), 571
[icons](#) (epyk.interfaces.Interface.Components property), 484
[icons\(\)](#) (epyk.interfaces.components.CompLayouts.Layouts method), 188
[icons\(\)](#) (epyk.interfaces.components.CompLists.Lists method), 203
[icons](#) (epyk.interfaces.components.CompMenus.Menus method), 216
[id\(\)](#) (epyk.core.js.Js.JsBase method), 524, 589

- [iframe\(\)](#) (*epyk.interfaces.components.CompLayouts.Layouts* method), 129
[method](#)), 189
[image\(\)](#) (*epyk.interfaces.components.CompVignets.Vignets* method), 326
[Images](#) (class in *epyk.interfaces.components.CompImages*), 158
[images](#) (*epyk.interfaces.Interface.Components* property), 484
[images\(\)](#) (*epyk.interfaces.components.CompMenus.Menus* method), 217
[img\(\)](#) (*epyk.interfaces.components.CompImages.Images* method), 166
[import_css\(\)](#) (*epyk.core.js.Js.JsBase* method), 525, 590
[import_js\(\)](#) (*epyk.core.js.Js.JsBase* method), 525, 590
[import_lib\(\)](#) (*epyk.core.py.PyExt.PyExt* method), 560
[import_package\(\)](#) (*epyk.core.py.PyExt.PyExt* method), 561
[important\(\)](#) (*epyk.interfaces.components.CompButtons.Buttons* method), 79
[ImportManager](#) (class in *epyk.core.js.Imports*), 56
[imports](#) (*epyk.core.Page.Report* property), 572
[imports\(\)](#) (*epyk.core.html.Header.Links* method), 631
[impression\(\)](#) (*epyk.interfaces.components.CompNetwork.Network* method), 248
[indices\(\)](#) (*epyk.interfaces.components.CompNavigation.Navigation* method), 238
[info\(\)](#) (*epyk.core.js.Js.JsBase* method), 525, 590
[info\(\)](#) (*epyk.core.js.Js.JsConsole* method), 493, 603
[info\(\)](#) (*epyk.interfaces.components.CompIcons.Icons* method), 141
[info\(\)](#) (*epyk.interfaces.components.CompModals.Modals* method), 224
[info\(\)](#) (*epyk.interfaces.components.CompNavigation.Banners* method), 230
[info\(\)](#) (*epyk.interfaces.components.CompNetwork.Network* method), 248
[info\(\)](#) (*epyk.interfaces.components.CompRich.Rich* method), 274
[inline\(\)](#) (*epyk.interfaces.components.CompLayouts.Layouts* method), 189
[innerHeight](#) (*epyk.core.js.JsWindow.JsWindow* property), 505
[input\(\)](#) (*epyk.interfaces.components.CompFields.Fields* method), 112
[input\(\)](#) (*epyk.interfaces.components.CompForms.Forms* method), 128
[input\(\)](#) (*epyk.interfaces.components.CompInputs.Inputs* method), 176
[Inputs](#) (class in *epyk.interfaces.components.CompInputs*), 169
[inputs](#) (*epyk.interfaces.Interface.Components* property), 484
[inputs\(\)](#) (*epyk.interfaces.components.CompForms.Forms* method), 129
[inputs\(\)](#) (*epyk.interfaces.components.CompTrees.Trees* method), 323
[ins\(\)](#) (*epyk.interfaces.components.CompTags.Tags* method), 290
[insert\(\)](#) (*epyk.core.py.PySql.SqlConn* method), 553
[instagram\(\)](#) (*epyk.interfaces.components.CompIcons.Icons* method), 141
[install\(\)](#) (in module *epyk.core.cli.cli_npm*), 68
[install_all\(\)](#) (in module *epyk.core.cli.cli_npm*), 69
[integer\(\)](#) (*epyk.interfaces.components.CompFields.Fields* method), 112
[intensity\(\)](#) (*epyk.interfaces.tables.CompTabulator.Tabulators* method), 476
[intersectionObserver\(\)](#) (*epyk.core.js.Js.JsBase* method), 525, 590
[invalid](#) (*epyk.core.css.styles.classes.CssStyle.Style* property), 579
[invalid](#) (*epyk.core.html.Aria.Aria* property), 639
[issues\(\)](#) (*epyk.interfaces.components.CompFields.Timelines* method), 125
[item\(\)](#) (*epyk.interfaces.components.CompLists.Lists* method), 204
[items\(\)](#) (*epyk.interfaces.components.CompLists.Lists* method), 204
- ## J
- [javascript\(\)](#) (*epyk.interfaces.components.CompCodes.Code* method), 99
[jquery](#) (*epyk.core.js.Js.JsBase* property), 526, 591
[JqueryVertorMap](#) (class in *epyk.interfaces.geo.CompGeoJqV*), 338
[jqv](#) (*epyk.interfaces.geo.CompGeo.Geo* property), 331
[js](#) (*epyk.core.data.Data.DataSrc* property), 566
[js](#) (*epyk.core.Page.Report* property), 572
[JsBase](#) (class in *epyk.core.js.Js*), 516, 581
[JsBreadCrumb](#) (class in *epyk.core.js.Js*), 606
[JsConsole](#) (class in *epyk.core.js.Js*), 492, 602
[jsfiddle\(\)](#) (*epyk.core.py.PyOuts.PyOuts* method), 539
[jsGetAll\(\)](#) (*epyk.core.js.Imports.ImportManager* method), 60
[JsHistory](#) (class in *epyk.core.js.JsWindow*), 495
[JsJson](#) (class in *epyk.core.js.Js*), 605
[JsLocalStorage](#) (class in *epyk.core.js.JsWindow*), 498
[JsLocation](#) (class in *epyk.core.js.Js.JsLocation*), 614
[JsLocation](#) (class in *epyk.core.js.JsLocation*), 510
[JsMaths](#) (class in *epyk.core.js.Js.JsMaths*), 608
[json\(\)](#) (*epyk.core.py.PyRest.PyRest* method), 548
[json\(\)](#) (*epyk.interfaces.Interface.Components* method), 484
[JsPerformance](#) (class in *epyk.core.js.Js.JsPerformance*), 623

- jsResolve() (*epyk.core.js.Imports.ImportManager* method), 60
 JsScreen (class in *epyk.core.js.Js*), 607
 JsSessionStorage (class in *epyk.core.js.JsWindow*), 500
 JsUrl (class in *epyk.core.js.JsWindow*), 501
 jsURLs() (*epyk.core.js.Imports.ImportManager* method), 60
 JsWindow (class in *epyk.core.js.JsWindow*), 501
 JsWindow (in module *epyk.core.js.Js*), 619
 JsWindowEvent (class in *epyk.core.js.JsWindow*), 509
 jupyter() (*epyk.core.py.PyOuts.PyOuts* method), 540
 jupyterlab() (*epyk.core.py.PyOuts.PyOuts* method), 540
- ## K
- kbd() (*epyk.interfaces.components.CompTags.Tags* method), 291
 key (*epyk.core.py.PyCrypto.PyCrypto* property), 559
 key() (*epyk.core.html.KeyCodes.KeyCode* method), 646
 key() (*epyk.core.js.JsWindow.JsLocalStorage* method), 499
 key() (*epyk.core.js.JsWindow.JsSessionStorage* method), 500
 KeyCode (class in *epyk.core.html.KeyCodes*), 644
 keydown (*epyk.core.js.Js.JsBase* property), 526, 591
 keyframes() (*epyk.core.css.styles.classes.CssStyle.Style* method), 579
 keypress (*epyk.core.js.Js.JsBase* property), 526, 591
 keyshortcuts (*epyk.core.html.Aria.Aria* property), 640
 keyup (*epyk.core.js.Js.JsBase* property), 526, 591
 keywords() (*epyk.core.html.Header.Meta* method), 635
- ## L
- label (*epyk.core.html.Aria.Aria* property), 640
 label() (*epyk.interfaces.components.CompInputs.Inputs* method), 176
 label() (*epyk.interfaces.components.CompTags.Tags* method), 291
 label() (*epyk.interfaces.components.CompTexts.Texts* method), 309
 labelledby (*epyk.core.html.Aria.Aria* property), 640
 large() (*epyk.interfaces.components.CompButtons.Buttons* method), 80
 large() (*epyk.interfaces.components.CompIcons.Icons* method), 142
 Layouts (class in *epyk.interfaces.components.CompLayouts*), 182
 layouts (*epyk.interfaces.Interface.Components* property), 485
 leaflet (*epyk.interfaces.geo.CompGeo.Geo* property), 331
 left() (*epyk.core.html.KeyCodes.KeyCode* method), 646
 left() (*epyk.interfaces.components.CompDrawers.Drawers* method), 103
 left() (*epyk.interfaces.components.CompInputs.Inputs* method), 177
 left() (*epyk.interfaces.components.CompPanels.Slidings* method), 265
 legend() (*epyk.interfaces.components.CompCalendars.Calendar* method), 95
 length (*epyk.core.js.JsWindow.JsHistory* property), 496
 level (*epyk.core.html.Aria.Aria* property), 640
 light() (*epyk.interfaces.components.CompRich.Rich* method), 274
 limit() (*epyk.core.py.PySql.SqlConn* method), 554
 line() (*epyk.interfaces.components.CompLayouts.Delimiter* method), 182
 line() (*epyk.interfaces.graphs.CompChartsApex.ApexChart* method), 365
 line() (*epyk.interfaces.graphs.CompChartsBillboard.Billboard* method), 373
 line() (*epyk.interfaces.graphs.CompChartsC3.C3* method), 382
 line() (*epyk.interfaces.graphs.CompChartsChartCss.CompChartCss* method), 387
 line() (*epyk.interfaces.graphs.CompChartsChartJs.ChartJs* method), 393
 line() (*epyk.interfaces.graphs.CompChartsDc.DC* method), 405
 line() (*epyk.interfaces.graphs.CompChartsFrappe.CompChartFrappe* method), 410
 line() (*epyk.interfaces.graphs.CompChartsGoogle.ChartGoogle* method), 418
 line() (*epyk.interfaces.graphs.CompChartsNvd3.Nvd3* method), 426
 line() (*epyk.interfaces.graphs.CompChartsPlotly.Plotly2D* method), 437
 line() (*epyk.interfaces.graphs.CompChartsPlotly.Plotly3D* method), 441
 line() (*epyk.interfaces.graphs.CompChartsRoughViz.CompRoughViz* method), 446
 line() (*epyk.interfaces.graphs.CompChartsSparkline.Sparkline* method), 450
 line() (*epyk.interfaces.graphs.CompChartsSvg.SVG* method), 454
 line() (*epyk.interfaces.graphs.CompChartsVis.Vis2D* method), 461
 line() (*epyk.interfaces.graphs.CompChartsVis.Vis3D* method), 464
 line3d() (*epyk.interfaces.graphs.CompChartsVis.Vis* method), 459
 line_cumulative() (*epyk.interfaces.graphs.CompChartsNvd3.Nvd3* method), 427
 line_focus() (*epyk.interfaces.graphs.CompChartsNvd3.Nvd3* method), 427
 line_range() (*epyk.interfaces.graphs.CompChartsBillboard.Billboard*

- method), 374
- link() (epyk.core.py.PySql.SqlConnNeo4j method), 557
- link() (epyk.interfaces.components.CompLinks.Links method), 196
- linkedIn() (epyk.interfaces.components.CompIcons.Icons method), 142
- Links (class in epyk.core.html.Header), 630
- Links (class in epyk.interfaces.components.CompLinks), 194
- links (epyk.core.html.Header.Header property), 628
- links (epyk.interfaces.Interface.Components property), 485
- list() (epyk.core.data.Data.DataJs method), 563
- list() (epyk.interfaces.components.CompLists.Lists method), 205
- Lists (class in epyk.interfaces.components.CompLists), 198
- lists (epyk.interfaces.Interface.Components property), 485
- live (epyk.core.html.Aria.Aria property), 640
- live() (epyk.interfaces.components.CompButtons.Buttons method), 81
- live() (epyk.interfaces.components.CompPollers.Poller method), 269
- LN10 (epyk.core.js.Js.JsMaths.JsMaths property), 608
- LN2 (epyk.core.js.Js.JsMaths.JsMaths property), 608
- load_data_file() (epyk.core.py.PySql.SqlConn method), 554
- load_schema() (epyk.core.py.PySql.SqlConn method), 554
- loading() (epyk.interfaces.components.CompModals.Modals method), 225
- loading() (epyk.interfaces.Interface.Components method), 485
- locals() (epyk.core.js.Imports.ImportManager method), 60
- location (epyk.core.js.Js.JsBase property), 526, 591
- location() (epyk.core.js.Js.Window.JsWindow method), 505
- lock() (epyk.interfaces.components.CompIcons.Toggles method), 157
- log() (epyk.core.js.Js.JsConsole method), 493, 603
- log() (epyk.core.js.Js.JsMaths.JsMaths method), 611
- LOG2E (epyk.core.js.Js.JsMaths.JsMaths property), 608
- logo() (epyk.interfaces.components.CompImages.Images method), 166
- lookup() (epyk.interfaces.components.CompLists.Lists method), 206
- lower() (epyk.interfaces.components.CompSliders.Sliders method), 279
- mail() (epyk.core.js.Js.JsBase method), 527, 592
- mail() (epyk.core.js.Js.JsLocation.JsLocation method), 615
- mail() (epyk.core.js.Js.JsLocation.JsLocation method), 512
- mail() (epyk.interfaces.components.CompButtons.Buttons method), 81
- mail() (epyk.interfaces.components.CompIcons.Icons method), 143
- main() (in module epyk.core.cli.cli_export), 68
- main() (in module epyk.core.cli.cli_npm), 69
- main() (in module epyk.core.cli.cli_project), 71
- manifest() (epyk.core.html.Header.Links method), 631
- map() (epyk.interfaces.geo.CompGeo.JqV.JqueryVectorMap method), 340
- mapbox (epyk.interfaces.geo.CompGeo.Geo property), 331
- maps() (epyk.interfaces.geo.CompGeoGoogle.GeoGoogle method), 337
- maps() (epyk.interfaces.graphs.CompChartsPlotly.Plotly3D method), 442
- mark() (epyk.core.js.Js.JsPerformance.JsPerformance method), 625
- mark() (epyk.interfaces.components.CompTags.Tags method), 292
- MarkDown (class in epyk.core.py.PyMarkdown), 546
- markdown (epyk.core.py.PyExt.PyExt property), 561
- markdown() (epyk.interfaces.components.CompCodes.Code method), 99
- markdown() (epyk.interfaces.components.CompRich.Rich method), 275
- markdown_file() (epyk.core.py.PyOuts.PyOuts method), 540
- marker() (epyk.interfaces.graphs.CompChartsPlotly.Plotly3D method), 442
- matrix() (epyk.interfaces.graphs.CompChartsChartJs.ChartJs method), 393
- max() (epyk.core.js.Js.JsMaths.JsMaths method), 611
- mdc (epyk.interfaces.Interface.WebComponents property), 64
- measure() (epyk.core.js.Js.JsPerformance.JsPerformance method), 626
- Media (class in epyk.interfaces.components.CompMedia), 211
- media (epyk.interfaces.Interface.Components property), 486
- media() (epyk.core.css.styles.classes.CssStyle.Style method), 579
- mediaRecorder (epyk.core.js.Js.JsBase property), 527, 592
- meeting() (epyk.interfaces.components.CompFields.Timelines method), 125
- menu() (epyk.interfaces.components.CompButtons.Buttons method), 82
- menu() (epyk.interfaces.components.CompIcons.Icons

M

- method), 143
- menu() (epyk.interfaces.components.CompMenus.Menus method), 217
- menu() (epyk.interfaces.components.CompPanels.Panels method), 260
- menu() (epyk.interfaces.graphs.CompCharts.Graphs method), 358
- menu() (epyk.interfaces.Interface.Components method), 486
- menu() (epyk.interfaces.tables.CompTables.Tables method), 474
- Menus (class in epyk.interfaces.components.CompMenus), 213
- menus (epyk.interfaces.Interface.Components property), 487
- mesh3d() (epyk.interfaces.graphs.CompChartsPlotly.Plotly3D method), 443
- message (epyk.core.js.Js.JsWebSocket.WebSocket property), 621
- messenger() (epyk.interfaces.components.CompIcons.Icons method), 144
- Meta (class in epyk.core.html.Header), 634
- meta (epyk.core.html.Header.Header property), 628
- meter() (epyk.interfaces.components.CompTags.Tags method), 292
- milestone() (epyk.interfaces.components.CompFields.TimeLine method), 125
- min() (epyk.core.js.Js.JsMaths.JsMaths method), 611
- modal (epyk.core.html.Aria.Aria property), 640
- Modals (class in epyk.interfaces.components.CompModals), 221
- modals (epyk.interfaces.Interface.Components property), 487
- module
- epyk.core.cli.cli_export, 67
 - epyk.core.cli.cli_npm, 68
 - epyk.core.cli.cli_project, 70
 - epyk.core.data.Data.DataPy, 568
 - epyk.core.js.Js.Location, 510
 - epyk.core.js.Js.Window, 495
 - epyk.core.py.PyOuts, 537
- moment (epyk.core.js.Js.JsBase property), 527, 592
- money() (epyk.interfaces.components.CompNumbers.Numbers method), 253
- month_end (epyk.core.py.PyDates.PyDates property), 543
- month_ends() (epyk.core.py.PyDates.PyDates method), 543
- months (epyk.core.py.PyDates.PyDates property), 544
- months() (epyk.interfaces.components.CompCalendars.Calendar method), 96
- months() (epyk.interfaces.components.CompFields.Fields method), 113
- more() (epyk.interfaces.components.CompButtons.Buttons method), 83
- more() (epyk.interfaces.components.CompNavigation.Navigation method), 238
- move() (epyk.interfaces.components.CompNumbers.Numbers method), 254
- moveBy() (epyk.core.js.Js.Window.Js.Window method), 505
- moz_selection (epyk.core.css.styles.GrpCls.ClassPage property), 576
- msg (epyk.core.js.Js.JsBase property), 527, 592
- multi() (epyk.interfaces.components.CompDrawers.Drawers method), 104
- multi() (epyk.interfaces.graphs.CompChartsChartJs.ChartJs method), 394
- multi() (epyk.interfaces.graphs.CompChartsNvd3.Nvd3 method), 428
- multi() (epyk.interfaces.tables.CompTabulator.Tabulators method), 477
- multiline (epyk.core.html.Aria.Aria property), 641
- multiselectable (epyk.core.html.Aria.Aria property), 641
- ## N
- nav() (epyk.interfaces.components.CompNavigation.Navigation method), 239
- nav() (epyk.interfaces.components.CompPanels.Panels method), 261
- nav() (epyk.interfaces.components.CompTags.Tags method), 293
- NavBars (class in epyk.interfaces.components.CompNavigation), 234
- navigateTo() (epyk.core.js.Js.JsBase method), 527, 592
- Navigation (class in epyk.interfaces.components.CompNavigation), 235
- navigation (epyk.interfaces.Interface.Components property), 487
- navigator (epyk.core.js.Js.JsBase property), 528, 593
- Network (class in epyk.interfaces.components.CompNetwork), 244
- network (epyk.interfaces.Interface.Components property), 487
- network() (epyk.interfaces.graphs.CompChartsVis.Vis2D method), 462
- new() (epyk.interfaces.components.CompForms.Forms method), 129
- new() (epyk.interfaces.graphs.CompChartsCanvas.Canvas method), 386
- new() (epyk.interfaces.graphs.CompChartsSvg.SVG method), 455
- new() (in module epyk.core.cli.cli_project), 71
- new_line() (epyk.interfaces.components.CompLayouts.Layouts method), 189

- `news()` (*epyk.interfaces.components.CompNetwork.Network* method), 248
- `next()` (*epyk.interfaces.components.CompIcons.Icons* method), 144
- `no_handle()` (*epyk.interfaces.components.CompDrawers.Drawers* method), 104
- `no_tag()` (*epyk.interfaces.components.CompTags.Tags* method), 293
- `node()` (*epyk.core.py.PySql.SqlConnNeo4j* method), 558
- `node_modules()` (*epyk.core.Page.Report* method), 572
- `normal()` (*epyk.interfaces.components.CompButtons.Buttons* method), 83
- `north_america()` (*epyk.interfaces.geo.CompGeoJqV.JqueryVectorMap* method), 341
- `north_america()` (*epyk.interfaces.geo.CompGeoPlotly.PlotlyBubble* method), 347
- `north_america()` (*epyk.interfaces.geo.CompGeoPlotly.PlotlyChoropleth* method), 350
- `not_()` (*epyk.core.js.Js.JsBase* method), 528, 593
- `note()` (*epyk.interfaces.components.CompTexts.Texts* method), 309
- `now` (*epyk.core.js.Js.JsPerformance.JsPerformance* property), 626
- `now` (*epyk.core.py.PyDates.PyDates* property), 544
- `now()` (*epyk.interfaces.components.CompFields.Fields* method), 113
- `npm()` (in module *epyk.core.cli.cli_npm*), 69
- `number()` (*epyk.core.data.Data.DataJs* method), 563
- `number()` (*epyk.core.js.Js.JsBase* method), 528, 593
- `number()` (*epyk.interfaces.components.CompFields.Fields* method), 114
- `number()` (*epyk.interfaces.components.CompNumbers.Numbers* method), 255
- `number()` (*epyk.interfaces.components.CompTexts.Texts* method), 310
- `number()` (*epyk.interfaces.components.CompVignets.Vignets* method), 327
- `number()` (*epyk.interfaces.graphs.CompChartsPlotly.PlotlyPD* method), 438
- `number_with_delta()` (*epyk.interfaces.graphs.CompChartsPlotly.Plotly2D* method), 438
- `Numbers` (class in *epyk.interfaces.components.CompNumbers*), 251
- `numbers` (*epyk.interfaces.Interface.Components* property), 487
- `numbers()` (*epyk.interfaces.components.CompLists.Lists* method), 206
- `Nvd3` (class in *epyk.interfaces.graphs.CompChartsNvd3*), 421
- `nvd3` (*epyk.core.data.Data.DataSrc* property), 566
- `nvd3` (*epyk.interfaces.graphs.CompCharts.Chart2d* property), 354
- `nvd3` (*epyk.interfaces.graphs.CompCharts.Graphs* property), 358
- O**
- `object()` (*epyk.core.data.Data.DataJs* method), 563
- `object()` (*epyk.core.js.Js.JsBase* method), 528, 593
- `objects` (*epyk.core.js.Js.JsBase* property), 528, 593
- `ohlc()` (*epyk.interfaces.graphs.CompChartsNvd3.Nvd3* method), 428
- `ol()` (*epyk.interfaces.components.CompTags.Tags* method), 294
- `onBeforeUnload()` (*epyk.core.js.Js.Window.JsWindow* method), 506
- `onClose()` (*epyk.core.js.Js.JsWebSocket.WebSocket* method), 621
- `onError()` (*epyk.core.js.Js.JsWebSocket.WebSocket* method), 621
- `onMessage()` (*epyk.core.js.Js.JsWebSocket.WebSocket* method), 622
- `onopen()` (*epyk.core.js.Js.JsWebSocket.WebSocket* method), 622
- `onPageShow()` (*epyk.core.js.Js.Window.JsWindow* method), 506
- `onReady()` (*epyk.core.js.Js.JsBase* method), 529, 594
- `open()` (*epyk.core.js.Js.Window.JsWindow* method), 506
- `open_new_tab()` (*epyk.core.js.Js.JsLocation.JsLocation* class method), 616
- `open_new_tab()` (*epyk.core.js.JsLocation.JsLocation* class method), 512
- `orientation` (*epyk.core.html.Aria.Aria* property), 641
- `origin` (*epyk.core.js.Js.JsLocation.JsLocation* property), 616
- `origin` (*epyk.core.js.JsLocation.JsLocation* property), 512
- `OutBrowsers` (class in *epyk.core.py.PyOuts*), 537
- `outs` (*epyk.core.Page.Report* property), 572
- `owns` (*epyk.core.html.Aria.Aria* property), 641
- P**
- `p()` (*epyk.interfaces.components.CompTags.Tags* method), 294
- `Packages` (class in *epyk.core.py.PyNpm*), 547
- `page()` (in module *epyk.core.cli.cli_export*), 68
- `page()` (in module *epyk.core.cli.cli_project*), 71
- `panel()` (*epyk.interfaces.components.CompLayouts.Layouts* method), 190
- `panel()` (*epyk.interfaces.components.CompNavigation.Navigation* method), 239
- `panel()` (*epyk.interfaces.components.CompPanels.Panels* method), 261
- `Panels` (class in *epyk.interfaces.components.CompPanels*), 258
- `panels` (*epyk.interfaces.Interface.Components* property), 487

paragraph() (epyk.interfaces.components.CompTexts.Text method), 311
 parallel_coordinates() (epyk.interfaces.graphs.CompChartsNvd3.Nvd3 method), 429
 parse() (epyk.core.js.Js.Json method), 605
 parseDate() (epyk.core.js.Js.JsonBase static method), 529, 594
 parseFloat() (epyk.core.js.Js.JsonBase static method), 529, 594
 parseInt() (epyk.core.js.Js.JsonBase static method), 529, 594
 password() (epyk.interfaces.components.CompFields.Fields method), 115
 password() (epyk.interfaces.components.CompInputs.Inputs method), 178
 patch() (epyk.core.js.Js.JsonBase method), 529, 594
 path() (epyk.core.py.PyDates.PyDates static method), 544
 path() (epyk.interfaces.components.CompNavigation.Navigation method), 239
 path() (epyk.interfaces.components.CompPictos.Pictogram method), 267
 path() (epyk.interfaces.graphs.CompChartsSvg.SVG method), 455
 pathname (epyk.core.js.Js.JsonLocation.JsonLocation property), 617
 pathname (epyk.core.js.JsonLocation.JsonLocation property), 513
 paths() (epyk.interfaces.components.CompPictos.Pictogram method), 267
 pdf() (epyk.core.data.Data.DataSrc method), 566
 pdf() (epyk.interfaces.components.CompIcons.Icons method), 145
 people() (epyk.interfaces.components.CompPictos.Pictogram method), 267
 percent() (epyk.interfaces.components.CompNumbers.Numbers method), 255
 percentage() (epyk.interfaces.graphs.CompChartsFrappe.CompChartsFrappe method), 411
 perf() (epyk.core.js.Js.JsonConsole method), 493, 603
 period() (epyk.interfaces.components.CompFields.Timeline method), 126
 phone() (epyk.interfaces.components.CompButtons.Buttons method), 84
 PI (epyk.core.js.Js.JsonMaths.JsonMaths property), 609
 Pictogram (class in epyk.interfaces.components.CompPictos), 266
 pictos (epyk.interfaces.Interface.Components property), 487
 pie() (epyk.interfaces.graphs.CompChartsApex.ApexChart method), 366
 pie() (epyk.interfaces.graphs.CompChartsBillboard.Billboard method), 374
 pie() (epyk.interfaces.graphs.CompChartsC3.C3 method), 382
 pie() (epyk.interfaces.graphs.CompChartsChartJs.ChartJs method), 394
 pie() (epyk.interfaces.graphs.CompChartsD3.D3 method), 401
 pie() (epyk.interfaces.graphs.CompChartsDc.DC method), 405
 pie() (epyk.interfaces.graphs.CompChartsFrappe.CompChartFrappe method), 411
 pie() (epyk.interfaces.graphs.CompChartsGoogle.ChartGoogle method), 418
 pie() (epyk.interfaces.graphs.CompChartsNvd3.Nvd3 method), 429
 pie() (epyk.interfaces.graphs.CompChartsPlotly.Plotly2D method), 439
 pie() (epyk.interfaces.graphs.CompChartsRoughViz.CompRoughViz method), 447
 pie() (epyk.interfaces.graphs.CompChartsSparkline.Sparkline method), 450
 pilcrow() (epyk.interfaces.components.CompNavigation.Navigation method), 240
 pill() (epyk.interfaces.components.CompButtons.Buttons method), 85
 pill() (epyk.interfaces.components.CompCalendars.Calendar method), 96
 pills() (epyk.interfaces.components.CompLists.Lists method), 207
 pills() (epyk.interfaces.components.CompMenus.Menus method), 218
 pills() (epyk.interfaces.components.CompPanels.Panels method), 262
 pin() (epyk.interfaces.components.CompNavigation.Navigation method), 240
 pingback() (epyk.core.html.Header.Links method), 632
 pivot() (epyk.interfaces.tables.CompPivot.Pivottable method), 469
 pivots (epyk.interfaces.tables.CompTables.Tables property), 474
 Pivottable (class in epyk.interfaces.tables.CompPivot), 467
 pixelDepth (epyk.core.js.Js.JsonScreen property), 607
 pkgs (epyk.core.js.Imports.ImportManager property), 60
 placeholder (epyk.core.html.Aria.Aria property), 641
 play() (epyk.interfaces.components.CompIcons.Icons method), 145
 plot() (epyk.interfaces.graphs.CompCharts.Graphs method), 358
 plot() (epyk.interfaces.graphs.CompChartsApex.ApexChart method), 366
 plot() (epyk.interfaces.graphs.CompChartsBillboard.Billboard method), 375
 plot() (epyk.interfaces.graphs.CompChartsC3.C3 method), 383

plot() (epyk.interfaces.graphs.CompChartsChartCss.CompChartsChartCss method), 388
 plot() (epyk.interfaces.graphs.CompChartsChartJs.ChartJs method), 395
 plot() (epyk.interfaces.graphs.CompChartsDc.Dc method), 406
 plot() (epyk.interfaces.graphs.CompChartsFrappe.CompChartsFrappe method), 412
 plot() (epyk.interfaces.graphs.CompChartsGoogle.ChartGoogle method), 419
 plot() (epyk.interfaces.graphs.CompChartsNvd3.Nvd3 method), 430
 plot() (epyk.interfaces.graphs.CompChartsRoughViz.CompChartsRoughViz method), 447
 plot() (epyk.interfaces.graphs.CompChartsVega.VegaEmbed method), 458
 plot() (epyk.interfaces.graphs.CompChartsVis.Vis2D method), 462
 Plotly (class in epyk.interfaces.geo.CompGeoPlotly), 343
 Plotly (class in epyk.interfaces.graphs.CompChartsPlotly), 432
 Plotly (class in epyk.interfaces.tables.CompTablesPlotly), 475
 plotly (epyk.core.data.Data.DataSrc property), 566
 plotly (epyk.interfaces.geo.CompGeo.Geo property), 332
 plotly (epyk.interfaces.graphs.CompCharts.Chart2d property), 354
 plotly (epyk.interfaces.graphs.CompCharts.Chart3d property), 355
 plotly (epyk.interfaces.graphs.CompCharts.Graphs property), 359
 plotly() (epyk.interfaces.components.CompNumbers.Numbers method), 256
 plotly() (epyk.interfaces.tables.CompPivot.Pivottable method), 470
 Plotly2D (class in epyk.interfaces.graphs.CompChartsPlotly), 432
 Plotly3D (class in epyk.interfaces.graphs.CompChartsPlotly), 441
 plotly_map (epyk.interfaces.geo.CompGeo.Geo property), 332
 plotly_with_delta() (epyk.interfaces.components.CompNumbers.Numbers method), 256
 PlotlyBubble (class in epyk.interfaces.geo.CompGeoPlotly), 345
 PlotlyChoropleth (class in epyk.interfaces.geo.CompGeoPlotly), 349
 plotlys (epyk.interfaces.tables.CompTables.Tables property), 474
 plus() (epyk.interfaces.components.CompIcons.Icons method), 146
 plus() (epyk.interfaces.components.CompPanels.Slidings method), 265
 points() (epyk.interfaces.components.CompLists.Lists method), 207
 points() (epyk.interfaces.components.CompNavigation.Navigation method), 241
 plot() (epyk.interfaces.graphs.CompChartsApex.ApexChart method), 366
 plot() (epyk.interfaces.graphs.CompChartsChartJs.ChartJs method), 396
 Poller (class in epyk.interfaces.components.CompPollers), 269
 poller() (epyk.interfaces.Interface.Components property), 487
 polyline() (epyk.interfaces.graphs.CompChartsSvg.SVG method), 455
 polyline() (epyk.interfaces.graphs.CompChartsSvg.SVG method), 456
 popup() (epyk.interfaces.components.CompLayouts.Layouts method), 191
 popup() (epyk.interfaces.components.CompModals.Modals method), 225
 port (epyk.core.js.Js.JsLocation.JsLocation property), 617
 port (epyk.core.js.JsLocation.JsLocation property), 513
 posinset (epyk.core.html.Aria.Aria property), 641
 post() (epyk.core.js.Js.JsBase method), 530, 595
 post() (epyk.core.py.PyRest.PyRest method), 548
 postData() (epyk.core.js.Js.Window.JsWindow method), 506
 postit() (epyk.interfaces.Interface.Components method), 488
 postTo() (epyk.core.js.Js.JsLocation.JsLocation class method), 617
 postTo() (epyk.core.js.JsLocation.JsLocation class method), 513
 pound() (epyk.interfaces.components.CompNumbers.Numbers method), 257
 pow() (epyk.core.js.Js.JsMaths.JsMaths static method), 611
 powered() (epyk.interfaces.components.CompRich.Rich method), 275
 preconnect() (epyk.core.html.Header.Links method), 632
 prefetch() (epyk.core.html.Header.Links method), 632
 preformat() (epyk.interfaces.components.CompTexts.Texts method), 312
 preload() (epyk.core.html.Header.Links method), 633
 prerender() (epyk.core.html.Header.Links method), 633
 pressed (epyk.core.html.Aria.Aria property), 642
 previous() (epyk.interfaces.components.CompIcons.Icons method), 146
 price() (epyk.interfaces.components.CompVignets.Vignets

- method), 328
- print() (epyk.core.js.Js.JsBase method), 530, 595
- print() (epyk.interfaces.Interface.Components method), 488
- print_() (epyk.core.js.Js.Window.JsWindow method), 506
- profile() (epyk.core.js.Js.JsBase method), 531, 596
- progress() (epyk.interfaces.components.CompSliders.Sliders method), 280
- progressbar() (epyk.interfaces.components.CompSliders.Sliders method), 280
- properties (epyk.core.Page.Report property), 573
- proxy() (epyk.core.py.PyRest.PyRest method), 549
- publish() (epyk.core.py.PyOuts.PyOuts method), 540
- pushState() (epyk.core.js.Js.Window.JsHistory method), 496
- put() (epyk.core.js.Js.JsBase method), 531, 596
- py (epyk.core.Page.Report property), 573
- PyCrypto (class in epyk.core.py.PyCrypto), 558
- PyDates (class in epyk.core.py.PyDates), 541
- PyExt (class in epyk.core.py.PyExt), 559
- PyGeo (class in epyk.core.py.PyGeo), 562
- pyk (epyk.interfaces.Interface.Components property), 488
- PyOuts (class in epyk.core.py.PyOuts), 538
- PyRest (class in epyk.core.py.PyRest), 547
- python() (epyk.interfaces.components.CompCodes.Code method), 100
- python() (epyk.interfaces.components.CompIcons.Icons method), 147
- ## Q
- q() (epyk.interfaces.components.CompTags.Tags method), 294
- qrcode() (epyk.interfaces.Interface.Components method), 489
- quarters (epyk.core.py.PyDates.PyDates property), 544
- querySelector() (epyk.core.js.Js.JsBase method), 531, 596
- querySelectorAll() (epyk.core.js.Js.JsBase method), 531, 596
- queueMicrotask() (epyk.core.js.Js.JsBase method), 532, 597
- quote() (epyk.interfaces.components.CompNavigation.Banner method), 231
- quote() (epyk.interfaces.components.CompPictos.Pictogram method), 267
- ## R
- r() (epyk.interfaces.components.CompCodes.Code method), 100
- radar() (epyk.interfaces.graphs.CompChartsApex.ApexChart method), 367
- radar() (epyk.interfaces.graphs.CompChartsBillboard.Billboard method), 375
- radar() (epyk.interfaces.graphs.CompChartsChartJs.ChartJs method), 396
- radial() (epyk.interfaces.graphs.CompChartsApex.ApexChart method), 367
- radio() (epyk.interfaces.components.CompButtons.Buttons method), 85
- radio() (epyk.interfaces.components.CompFields.Fields method), 115
- radio() (epyk.interfaces.components.CompInputs.Inputs method), 178
- radios() (epyk.interfaces.components.CompLists.Lists method), 208
- random() (epyk.core.js.Js.JsMaths.JsMaths method), 612
- range() (epyk.interfaces.components.CompFields.Fields method), 116
- range() (epyk.interfaces.components.CompSliders.Sliders method), 281
- range_dates() (epyk.core.py.PyDates.PyDates method), 544
- react() (in module epyk.core.cli.cli_npm), 69
- react_parser() (in module epyk.core.cli.cli_npm), 69
- readonly (epyk.core.html.Aria.Aria property), 642
- readyState (epyk.core.js.Js.JsWebSocket.WebSocket property), 622
- receive() (epyk.core.js.Js.JsWebSocket.WebSocket method), 622
- reconnect() (epyk.core.js.Js.JsWebSocket.WebSocket method), 622
- record() (epyk.core.data.Data.DataJs method), 563
- records (epyk.core.py.PySql.SqlConn property), 554
- rectangle() (epyk.interfaces.graphs.CompChartsSvg.SVG method), 456
- references (epyk.interfaces.components.CompTexts.Texts property), 312
- refresh() (epyk.core.html.Header.Meta method), 635
- refresh() (epyk.interfaces.components.CompButtons.Buttons method), 86
- refresh() (epyk.interfaces.components.CompIcons.Icons method), 147
- register() (epyk.core.Page.Report method), 573
- registerFunction() (epyk.core.js.Js.JsBase method), 532, 597
- relevant (epyk.core.html.Aria.Aria property), 642
- reload() (epyk.core.js.Js.JsLocation.JsLocation class method), 617
- reload() (epyk.core.js.Js.JsLocation.JsLocation class method), 513
- remove() (epyk.interfaces.components.CompButtons.Buttons method), 87
- remove() (epyk.interfaces.components.CompIcons.Icons method), 148

- [removeItem\(\)](#) ([epyk.core.js.JsWindow.JsLocalStorage](#) method), 499
[removeItem\(\)](#) ([epyk.core.js.JsWindow.JsSessionStorage](#) method), 500
[replace\(\)](#) ([epyk.core.js.Js.JsLocation.JsLocation](#) class method), 617
[replace\(\)](#) ([epyk.core.js.JsLocation.JsLocation](#) class method), 514
[replaceState\(\)](#) ([epyk.core.js.JsWindow.JsHistory](#) method), 497
[Report](#) (class in [epyk.core.Page](#)), 569
[repositories\(\)](#) ([epyk.core.py.PyNpm.Packages](#) class method), 547
[request\(\)](#) ([epyk.core.py.PyRest.PyRest](#) static method), 549
[request_http\(\)](#) ([epyk.core.js.Js.JsBase](#) method), 532, 597
[request_rpc\(\)](#) ([epyk.core.js.Js.JsBase](#) method), 533, 598
[requests](#) ([epyk.core.py.PyExt.PyExt](#) property), 561
[requirements](#) ([epyk.core.js.Imports.ImportManager](#) property), 61
[requirements\(\)](#) (in module [epyk.core.cli.cli_npm](#)), 69
[resolve\(\)](#) ([epyk.core.py.PyMarkdown.MarkDown](#) method), 546
[rest\(\)](#) ([epyk.core.data.Data.DataSrc](#) method), 566
[rest\(\)](#) ([epyk.core.js.Js.JsBase](#) method), 533, 598
[return_\(\)](#) ([epyk.core.js.Js.JsBase](#) method), 534, 599
[ribbon\(\)](#) ([epyk.interfaces.graphs.CompChartsPlotly.Plotly3D](#) method), 443
[Rich](#) (class in [epyk.interfaces.components.CompRich](#)), 270
[rich](#) ([epyk.interfaces.Interface.Components](#) property), 489
[right\(\)](#) ([epyk.core.html.KeyCodes.KeyCode](#) method), 646
[right\(\)](#) ([epyk.interfaces.components.CompDrawers.Drawer](#) method), 105
[right\(\)](#) ([epyk.interfaces.components.CompMenus.Menus](#) method), 218
[right\(\)](#) ([epyk.interfaces.components.CompPanels.Slidings](#) method), 265
[role](#) ([epyk.core.html.Aria.Aria](#) property), 642
[roledescription](#) ([epyk.core.html.Aria.Aria](#) property), 642
[roman\(\)](#) ([epyk.interfaces.components.CompLists.Lists](#) method), 208
[room\(\)](#) ([epyk.interfaces.components.CompNetwork.Network](#) method), 249
[root__script](#) ([epyk.core.Page.Report](#) property), 573
[roughviz](#) ([epyk.interfaces.graphs.CompCharts.Graphs](#) property), 359
[round\(\)](#) ([epyk.core.js.Js.JsMaths.JsMaths](#) method), 612
[row\(\)](#) ([epyk.interfaces.components.CompLayouts.Layouts](#) method), 191
[row\(\)](#) ([epyk.interfaces.components.CompNavigation.Banners](#) method), 231
[rowindex](#) ([epyk.core.html.Aria.Aria](#) property), 643
[rowspan](#) ([epyk.core.html.Aria.Aria](#) property), 643
[rpc\(\)](#) ([epyk.core.data.Data.DataSrc](#) method), 567
[rss\(\)](#) ([epyk.core.data.Data.DataSrc](#) method), 567
[rss\(\)](#) ([epyk.interfaces.components.CompIcons.Icons](#) method), 148
[rst\(\)](#) ([epyk.interfaces.components.CompCodes.Code](#) method), 101
[rubric\(\)](#) ([epyk.interfaces.components.CompTitles.Titles](#) method), 319
[run\(\)](#) ([epyk.interfaces.components.CompButtons.Buttons](#) method), 87
- ## S
- [s\(\)](#) ([epyk.interfaces.components.CompTags.Tags](#) method), 295
[samp\(\)](#) ([epyk.interfaces.components.CompTags.Tags](#) method), 295
[samples](#) ([epyk.core.js.Js.JsBase](#) property), 534, 599
[satellite\(\)](#) ([epyk.interfaces.geo.CompGeoGoogle.GeoGoogle](#) method), 337
[save\(\)](#) ([epyk.core.html.KeyCodes.KeyCode](#) method), 646
[save\(\)](#) ([epyk.interfaces.components.CompIcons.Icons](#) method), 149
[save_cache\(\)](#) ([epyk.core.data.Data.DataSrc](#) method), 567
[scatter\(\)](#) ([epyk.interfaces.graphs.CompChartsApex.ApexChart](#) method), 368
[scatter\(\)](#) ([epyk.interfaces.graphs.CompChartsBillboard.Billboard](#) method), 375
[scatter\(\)](#) ([epyk.interfaces.graphs.CompChartsC3.C3](#) method), 383
[scatter\(\)](#) ([epyk.interfaces.graphs.CompChartsChartJs.ChartJs](#) method), 397
[scatter\(\)](#) ([epyk.interfaces.graphs.CompChartsD3.D3](#) method), 401
[scatter\(\)](#) ([epyk.interfaces.graphs.CompChartsDc.DC](#) method), 406
[scatter\(\)](#) ([epyk.interfaces.graphs.CompChartsGoogle.ChartGoogle](#) method), 419
[scatter\(\)](#) ([epyk.interfaces.graphs.CompChartsNvd3.Nvd3](#) method), 430
[scatter\(\)](#) ([epyk.interfaces.graphs.CompChartsPlotly.Plotly2D](#) method), 439
[scatter\(\)](#) ([epyk.interfaces.graphs.CompChartsPlotly.Plotly3D](#) method), 444
[scatter\(\)](#) ([epyk.interfaces.graphs.CompChartsRoughViz.CompRoughViz](#) method), 448
[scatter\(\)](#) ([epyk.interfaces.graphs.CompChartsVis.Vis2D](#) method), 462

`scatter()` (*epyk.interfaces.graphs.CompChartsVis.Vis3D method*), 465
`scatter3d()` (*epyk.interfaces.graphs.CompChartsVis.Vis method*), 459
`scattergl()` (*epyk.interfaces.graphs.CompChartsPlotly.Plotly method*), 440
`scattermapbox()` (*epyk.interfaces.geo.CompGeoPlotly.Plotly method*), 344
`scatterpolar()` (*epyk.interfaces.graphs.CompChartsPlotly.Plotly2D method*), 440
`screen` (*epyk.core.js.Js.JsBase property*), 534, 599
`script()` (*epyk.interfaces.graphs.CompChartsD3.D3 method*), 402
`scroll()` (*epyk.core.js.JsWindow.JsWindow method*), 506
`scroll()` (*epyk.interfaces.components.CompNavigation.Navigation method*), 241
`scrollbar_webkit` (*epyk.core.css.styles.GrpCls.ClassPage property*), 576
`scrollbar_webkit_thumb` (*epyk.core.css.styles.GrpCls.ClassPage property*), 576
`scrollbar_webkit_track` (*epyk.core.css.styles.GrpCls.ClassPage property*), 576
`scrollEndPage` (*epyk.core.js.JsWindow.JsWindow property*), 507
`scrollMaxY` (*epyk.core.js.JsWindow.JsWindow property*), 507
`scrollPercentage` (*epyk.core.js.JsWindow.JsWindow property*), 507
`scrollTo()` (*epyk.core.js.JsWindow.JsWindow method*), 507
`scrollUp()` (*epyk.core.js.JsWindow.JsWindow method*), 507
`scrollY` (*epyk.core.js.JsWindow.JsWindow property*), 507
`search` (*epyk.core.js.Js.JsLocation.JsLocation property*), 618
`search` (*epyk.core.js.JsLocation.JsLocation property*), 514
`search()` (*epyk.interfaces.components.CompInputs.Inputs method*), 179
`search_input()` (*epyk.interfaces.components.CompRich.Rich method*), 275
`search_results()` (*epyk.interfaces.components.CompRich.Rich method*), 276
`section()` (*epyk.interfaces.components.CompImages.Images method*), 167
`section()` (*epyk.interfaces.components.CompLayouts.Layouts method*), 192
`section()` (*epyk.interfaces.components.CompTitles.Titles method*), 319
`select()` (*epyk.core.py.PySql.SqlConn method*), 554
`select()` (*epyk.interfaces.components.CompFields.Fields method*), 117
`select()` (*epyk.interfaces.components.CompLists.Lists method*), 209
`selected` (*epyk.core.html.Aria.Aria property*), 643
`selection` (*epyk.core.css.styles.GrpCls.ClassPage property*), 576
`selections()` (*epyk.interfaces.components.CompMenus.Menus method*), 219
`send()` (*epyk.core.js.Js.JsWebSocket.WebSocket method*), 622
`sendText()` (*epyk.core.js.Js.JsWebSocket.WebSocket method*), 623
`series()` (*epyk.interfaces.graphs.CompChartsDc.DC method*), 407
`sequence()` (*epyk.core.data.Data.DataJs method*), 563
`serverSentEvent()` (*epyk.core.js.Js.JsBase method*), 534, 599
`service()` (*epyk.core.js.Js.JsConsole method*), 493, 604
`set()` (*epyk.core.html.Aria.Aria method*), 643
`set()` (*epyk.core.js.Js.JsLocation.URLSearchParams method*), 619
`set()` (*epyk.core.js.JsLocation.URLSearchParams method*), 515
`set_crossfilter()` (*epyk.interfaces.graphs.CompChartsDc.DC method*), 407
`set_version()` (*epyk.interfaces.graphs.CompChartsChartJs.ChartJs method*), 397
`setInterval()` (*epyk.core.js.JsWindow.JsWindow method*), 507
`setItem()` (*epyk.core.js.JsWindow.JsLocalStorage method*), 499
`setItem()` (*epyk.core.js.JsWindow.JsSessionStorage method*), 501
`setResourceTimingBufferSize()` (*epyk.core.js.Js.JsPerformance.JsPerformance method*), 626
`setSize` (*epyk.core.html.Aria.Aria property*), 643
`setTimeout()` (*epyk.core.js.JsWindow.JsWindow method*), 508
`setVersion()` (*epyk.core.js.Imports.ImportManager method*), 61
`shift()` (*epyk.core.html.KeyCodes.KeyCode method*), 647
`shift_with()` (*epyk.core.html.KeyCodes.KeyCode method*), 647
`shortcut()` (*epyk.interfaces.components.CompNavigation.Navigation method*), 241
`shortlink()` (*epyk.core.html.Header.Links method*), 633
`show()` (*epyk.core.js.Imports.ImportManager method*), 61
`side()` (*epyk.interfaces.components.CompNavigation.Navigation method*), 242

signin() (epyk.interfaces.components.CompIcons.Icons method), 149
 sin() (epyk.core.js.Js.JsMaths.JsMaths method), 613
 SipHash (class in epyk.core.py.PyHash), 562
 skillbars() (epyk.interfaces.graphs.CompCharts.Graphs method), 359
 skins (epyk.core.Page.Report property), 573
 slider() (epyk.interfaces.components.CompFields.Fields method), 118
 slider() (epyk.interfaces.components.CompSliders.Sliders method), 281
 Sliders (class in epyk.interfaces.components.CompSliders), 278
 sliders (epyk.interfaces.Interface.Components property), 489
 slides() (epyk.interfaces.components.CompVignets.Vignets method), 328
 slideshow() (epyk.interfaces.Interface.Components method), 489
 sliding() (epyk.interfaces.components.CompPanels.Panels method), 262
 Slidings (class in epyk.interfaces.components.CompPanels), 265
 slidings (epyk.interfaces.components.CompPanels.Panels property), 263
 small() (epyk.interfaces.components.CompButtons.Buttons method), 88
 small() (epyk.interfaces.components.CompTags.Tags method), 296
 soap() (epyk.core.data.Data.DataSrc method), 567
 socket() (epyk.core.data.Data.DataSrc method), 568
 socketio() (epyk.core.js.Js.JsBase method), 534, 599
 sort (epyk.core.html.Aria.Aria property), 643
 south_america() (epyk.interfaces.geo.CompGeoJqV.JqueryMap method), 341
 south_america() (epyk.interfaces.geo.CompGeoPlotly.PlotlyBubble method), 347
 south_america() (epyk.interfaces.geo.CompGeoPlotly.PlotlyChoropleth method), 351
 space() (epyk.core.html.KeyCodes.KeyCode method), 647
 span() (epyk.interfaces.components.CompTags.Tags method), 296
 span() (epyk.interfaces.components.CompTexts.Texts method), 312
 Sparkline (class in epyk.interfaces.graphs.CompCharts.Sparkline), 448
 sparkline() (epyk.interfaces.graphs.CompCharts.Graphs method), 360
 sparklines (epyk.interfaces.graphs.CompCharts.Graphs property), 361
 speechRecognition() (epyk.core.js.Js.JsBase method), 534, 599
 spline() (epyk.interfaces.graphs.CompChartsBillboard.Billboard method), 376
 spline() (epyk.interfaces.graphs.CompChartsC3.C3 method), 384
 split() (epyk.interfaces.components.CompPanels.Panels method), 263
 sponsor() (epyk.interfaces.components.CompNavigation.Banners method), 232
 sql() (epyk.interfaces.components.CompCodes.Code method), 102
 SqlConnection (class in epyk.core.py.PySql), 550
 SqlConnectionNeo4j (class in epyk.core.py.PySql), 557
 SqlConnectionOdbc (class in epyk.core.py.PySql), 557
 sqrt() (epyk.core.js.Js.JsMaths.JsMaths method), 613
 Sqrt1_2 (epyk.core.js.Js.JsMaths.JsMaths property), 609
 Sqrt2 (epyk.core.js.Js.JsMaths.JsMaths property), 609
 squares() (epyk.interfaces.components.CompLists.Lists method), 210
 stack() (epyk.interfaces.components.CompPictos.Pictogram method), 268
 stackblitz() (epyk.core.py.PyOuts.OutBrowsers method), 538
 stacked() (epyk.interfaces.graphs.CompChartsBillboard.Billboard method), 377
 stacked() (epyk.interfaces.graphs.CompChartsChartCss.CompChartCss method), 388
 stackoverflow() (epyk.interfaces.components.CompIcons.Icons method), 149
 stanford() (epyk.interfaces.graphs.CompChartsC3.C3 method), 384
 star() (epyk.interfaces.graphs.CompChartsSvg.SVG method), 457
 stars() (epyk.interfaces.components.CompRich.RichMap method), 277
 states (epyk.core.js.Js.JsWebSocket.WebSocket property), 623
 static() (epyk.interfaces.components.CompFields.Fields method), 118
 status() (epyk.interfaces.components.CompRich.Rich method), 277
 std (epyk.interfaces.Interface.WebComponents property), 64
 step() (epyk.interfaces.graphs.CompChartsBillboard.Billboard method), 377
 step() (epyk.interfaces.graphs.CompChartsC3.C3 method), 385
 step() (epyk.interfaces.graphs.CompChartsChartJs.ChartJs method), 397
 step() (epyk.interfaces.graphs.CompChartsDc.DC method), 408
 stepper() (epyk.interfaces.components.CompModals.Modals method), 226
 Steppers (class in epyk.interfaces.components.CompSteps), 283

- steppers (*epyk.interfaces.Interface.Components* property), 490
 steps (*epyk.interfaces.Interface.Components* property), 490
 stop() (*epyk.interfaces.components.CompIcons.Icons* method), 150
 store() (*epyk.interfaces.components.CompButtons.Buttons* method), 89
 string() (*epyk.core.js.Js.JsBase* method), 535, 600
 stringify() (*epyk.core.js.Js.JsJson* method), 606
 strong() (*epyk.interfaces.components.CompTags.Tags* method), 297
 Style (class in *epyk.core.css.styles.classes.CssStyle*), 577
 stylesheet() (*epyk.core.html.Header.Links* method), 633
 sub() (*epyk.interfaces.components.CompTags.Tags* method), 297
 sub_total() (*epyk.interfaces.tables.CompPivot.Pivottable* method), 470
 subscribe() (*epyk.interfaces.components.CompForms.Forms* method), 130
 subtitle() (*epyk.interfaces.components.CompTitles.Titles* method), 320
 success() (*epyk.interfaces.components.CompIcons.Icons* method), 150
 success() (*epyk.interfaces.components.CompModals.Modals* method), 226
 success() (*epyk.interfaces.components.CompNetwork.Network* method), 249
 sunburst() (*epyk.interfaces.graphs.CompChartsDc.DC* method), 408
 sunburst() (*epyk.interfaces.graphs.CompChartsNvd3.Nvd3* method), 431
 sup() (*epyk.interfaces.components.CompTags.Tags* method), 298
 surface() (*epyk.interfaces.graphs.CompChartsPlotly.Plotly3D* method), 444
 surface() (*epyk.interfaces.graphs.CompChartsVis.Vis* method), 460
 surface() (*epyk.interfaces.graphs.CompChartsVis.Vis3D* method), 465
 SVG (class in *epyk.interfaces.graphs.CompChartsSvg*), 451
 svg (*epyk.interfaces.graphs.CompCharts.Chart2d* property), 354
 svg (*epyk.interfaces.graphs.CompCharts.Graphs* property), 361
 svg() (*epyk.core.html.Header.Icons* method), 630
 svg() (*epyk.interfaces.graphs.CompChartsD3.D3* method), 403
 switch() (*epyk.core.js.Js.JsBase* method), 535, 600
 symbols (*epyk.core.Page.Report* property), 573
- ## T
- tab() (*epyk.core.html.KeyCodes.KeyCode* method), 647
 table() (*epyk.core.js.Js.JsConsole* method), 494, 604
 table() (*epyk.core.py.PySql.SqlConn* method), 555
 table() (*epyk.interfaces.components.CompIcons.Icons* method), 151
 table() (*epyk.interfaces.components.CompLayouts.Layouts* method), 192
 table() (*epyk.interfaces.tables.CompAgGrid.AgGrid* method), 466
 table() (*epyk.interfaces.tables.CompDatatable.Datatables* method), 467
 table() (*epyk.interfaces.tables.CompTableD3.D3* method), 471
 table() (*epyk.interfaces.tables.CompTableGoogle.Google* method), 472
 table() (*epyk.interfaces.tables.CompTablesPlotly.Plotly* method), 475
 table() (*epyk.interfaces.tables.CompTabulator.Tabulators* method), 478
 table_clone() (*epyk.core.py.PySql.SqlConn* method), 555
 table_create() (*epyk.core.py.PySql.SqlConn* method), 555
 table_create_from_file() (*epyk.core.py.PySql.SqlConn* method), 555
 table_empty() (*epyk.core.py.PySql.SqlConn* method), 556
 table_migrate() (*epyk.core.py.PySql.SqlConn* method), 556
 Tables (class in *epyk.interfaces.tables.CompTables*), 472
 tables (*epyk.core.py.PySql.SqlConn* property), 556
 tables (*epyk.core.py.PySql.SqlConnOdbc* property), 557
 tables (*epyk.interfaces.Interface.Components* property), 490
 tabs() (*epyk.interfaces.components.CompPanels.Panels* method), 264
 Tabulators (class in *epyk.interfaces.tables.CompTabulator*), 476
 tabulators (*epyk.interfaces.tables.CompTables.Tables* property), 475
 Tags (class in *epyk.interfaces.components.CompTags*), 283
 tags (*epyk.interfaces.Interface.Components* property), 491
 team() (*epyk.interfaces.components.CompPictos.Pictogram* method), 268
 terrain() (*epyk.interfaces.geo.CompGeoGoogle.GeoGoogle* method), 337
 text() (*epyk.interfaces.components.CompButtons.Buttons* method), 89
 text() (*epyk.interfaces.components.CompFields.Fields* method), 119

`text()` (`epyk.interfaces.components.CompNavigation.Banners` method), 232
`text()` (`epyk.interfaces.components.CompTexts.Texts` method), 313
`text()` (`epyk.interfaces.components.CompVignets.Vignets` method), 329
`textarea()` (`epyk.interfaces.components.CompFields.Fields` method), 120
`textarea()` (`epyk.interfaces.components.CompInputs.Inputs` method), 180
`TextReferences` (class in `epyk.interfaces.components.CompTexts`), 300
`Texts` (class in `epyk.interfaces.components.CompTexts`), 302
`texts` (`epyk.interfaces.Interface.Components` property), 491
`theme` (`epyk.core.Page.Report` property), 574
`thumbs_down()` (`epyk.interfaces.components.CompButtons.Buttons` method), 90
`thumbs_up()` (`epyk.interfaces.components.CompButtons.Buttons` method), 90
`tick()` (`epyk.interfaces.components.CompIcons.Icons` method), 151
`tick()` (`epyk.interfaces.components.CompPictos.Pictogram` method), 268
`time()` (`epyk.core.js.Js.JsConsole` method), 494, 604
`time()` (`epyk.interfaces.components.CompFields.Fields` method), 121
`time()` (`epyk.interfaces.components.CompTags.Tags` method), 298
`timeEnd()` (`epyk.core.js.Js.JsConsole` method), 494, 604
`timeline()` (`epyk.interfaces.graphs.CompChartsVis.Vis2D` method), 463
`Timelines` (class in `epyk.interfaces.components.CompFields.Fields`), 124
`timelines` (`epyk.interfaces.Interface.Components` property), 491
`timer()` (`epyk.interfaces.components.CompCalendars.Calendars` method), 97
`timer()` (`epyk.interfaces.components.CompIcons.Icons` method), 152
`timeseries()` (`epyk.interfaces.graphs.CompChartsBillboard.Billboard` method), 378
`timeseries()` (`epyk.interfaces.graphs.CompChartsC3.C3` method), 385
`timeseries()` (`epyk.interfaces.graphs.CompChartsChart.JsChart` method), 398
`timeseries()` (`epyk.interfaces.graphs.CompChartsNvd3.Nvd3` method), 431
`timeseries()` (`epyk.interfaces.graphs.CompChartsPlotly.Plotly` method), 441
`title()` (`epyk.core.html.Header.Header` method), 629
`title()` (`epyk.core.js.Js.JsBase` static method), 535, 600
`title()` (`epyk.interfaces.components.CompNavigation.Banners` method), 232
`title()` (`epyk.interfaces.components.CompTexts.Texts` method), 314
`title()` (`epyk.interfaces.components.CompTitles.Titles` method), 320
`Titles` (class in `epyk.interfaces.components.CompTitles`), 316
`titles` (`epyk.interfaces.Interface.Components` property), 491
`to()` (`epyk.interfaces.components.CompNavigation.Navigation` method), 242
`to_mime()` (`epyk.core.py.PyMails.Email` method), 546
`to_requireJs()` (`epyk.core.js.Imports.ImportManager` method), 61
`to_server_time()` (`epyk.core.py.PyDates.PyDates` static method), 545
`to_user_time()` (`epyk.core.py.PyDates.PyDates` static method), 545
`today` (`epyk.core.py.PyDates.PyDates` property), 546
`today()` (`epyk.interfaces.components.CompFields.Fields` method), 121
`toggle()` (`epyk.interfaces.components.CompButtons.Buttons` method), 91
`toggle()` (`epyk.interfaces.components.CompFields.Fields` method), 122
`toggle()` (`epyk.interfaces.components.CompPollers.Poller` method), 269
`toggleInterval()` (`epyk.core.js.JsWindow.JsWindow` method), 508
`Toggles` (class in `epyk.interfaces.components.CompIcons`), 156
`toggles` (`epyk.interfaces.components.CompIcons.Icons` property), 152
`toJSON()` (`epyk.core.js.Js.JsPerformance.JsPerformance` method), 627
`toolbar()` (`epyk.interfaces.components.CompMenus.Menus` method), 219
`toolbar()` (`epyk.interfaces.components.CompMenus.Menus` method), 220
`top()` (`epyk.interfaces.components.CompNavigation.Banners` method), 233
`top()` (`epyk.interfaces.components.CompNavigation.NavBars` method), 234
`trafficlights()` (`epyk.interfaces.tables.CompTabulator.Tabulators` method), 478
`transition()` (`epyk.core.css.styles.classes.CssStyle.Style` method), 580
`translate()` (`epyk.core.py.PyMarkdown.MarkDown` class method), 546
`TruthyDate()` (in module `epyk.core.cli.cli_project`), 71
`transparent()` (`epyk.interfaces.components.CompNavigation.NavBars` method), 235
`transpile()` (in module `epyk.core.cli.cli_export`), 68

- transpile_all() (in module epyk.core.cli.cli_project), 71
 tree() (epyk.interfaces.components.CompLists.Lists method), 210
 tree() (epyk.interfaces.components.CompTrees.Trees method), 324
 treemap() (epyk.interfaces.graphs.CompChartsApex.ApexChart method), 368
 treemap() (epyk.interfaces.graphs.CompChartsChartJs.ChartJs method), 398
 treemap() (epyk.interfaces.graphs.CompChartsGoogle.ChartGoogle method), 282
 Trees (class in epyk.interfaces.components.CompTrees), 322
 trees (epyk.interfaces.Interface.Components property), 491
 triangle() (epyk.interfaces.graphs.CompChartsSvg.SVG method), 457
 tristate() (epyk.interfaces.graphs.CompChartsSparkline.Sparkline method), 451
 trunc() (epyk.core.js.Js.JsMaths.JsMaths method), 613
 tryCatch() (epyk.core.js.Js.JsConsole method), 494, 604
 tui (epyk.interfaces.Interface.WebComponents property), 64
 twitch() (epyk.interfaces.components.CompIcons.Icons method), 152
 twitter() (epyk.interfaces.components.CompIcons.Icons method), 153
 typeof() (epyk.core.js.Js.JsBase static method), 536, 601
- ## U
- u() (epyk.interfaces.components.CompTags.Tags method), 299
 ui (epyk.core.Page.Report property), 574
 ui() (epyk.interfaces.tables.CompPivot.Pivottable method), 471
 uk() (epyk.interfaces.geo.CompGeoChartJs.Choropleth method), 334
 underline() (epyk.interfaces.components.CompLayouts.Layouts method), 193
 underline() (epyk.interfaces.components.CompTitles.Titles method), 321
 up() (epyk.core.html.KeyCodes.KeyCode method), 647
 up() (epyk.interfaces.components.CompNavigation.Navigation method), 243
 up_down() (epyk.interfaces.components.CompTexts.Texts method), 315
 update() (epyk.core.py.PySql.SqlConn method), 556
 update() (epyk.interfaces.components.CompRich.Rich method), 278
 update() (in module epyk.core.cli.cli_npm), 70
 updateState() (epyk.core.js.JsWindow.JsHistory method), 497
 updateStateFromComponent() (epyk.core.js.JsWindow.JsHistory method), 498
 upload() (epyk.interfaces.components.CompLinks.Links method), 197
 upload() (epyk.interfaces.components.CompNetwork.Network method), 250
 upper() (epyk.interfaces.components.CompSliders.Sliders method), 321
 upper() (epyk.interfaces.components.CompTitles.Titles method), 321
 url (epyk.core.js.Js.JsBreadCrumb property), 607
 URL (epyk.core.js.JsWindow.JsWindow property), 501
 URLSearchParams (class in epyk.core.js.Js.JsLocation), 618
 URLSearchParams (class in epyk.core.js.JsLocation), 514
 urlSearchParams (epyk.core.js.Js.JsLocation.JsLocation property), 618
 urlSearchParams (epyk.core.js.JsLocation.JsLocation property), 514
 us() (epyk.interfaces.geo.CompGeoChartJs.BubbleMaps method), 333
 us() (epyk.interfaces.geo.CompGeoChartJs.Choropleth method), 335
 usa() (epyk.interfaces.geo.CompGeoDc.Dc method), 336
 usa() (epyk.interfaces.geo.CompGeoJqV.JqueryVectorMap method), 342
 usa() (epyk.interfaces.geo.CompGeoPlotly.PlotlyBubble method), 348
 usa() (epyk.interfaces.geo.CompGeoPlotly.PlotlyChoropleth method), 351
- ## V
- valid (epyk.core.css.styles.classes.CssStyle.Style property), 580
 validate() (epyk.interfaces.components.CompButtons.Buttons method), 226
 validation() (epyk.interfaces.components.CompModals.Modals method), 226
 valuemax (epyk.core.html.Aria.Aria property), 643
 valuemax (epyk.core.html.Aria.Aria property), 644
 valuenow (epyk.core.html.Aria.Aria property), 644
 valuetext (epyk.core.html.Aria.Aria property), 644
 var() (epyk.interfaces.components.CompTags.Tags method), 299
 vega (epyk.interfaces.graphs.CompCharts.Graphs property), 361
 VegaEmbedded (class in epyk.interfaces.graphs.CompChartsVega), 458

- versions()** (*epyk.core.py.PyNpm.Packages class method*), 547
video() (*epyk.interfaces.components.CompMedia.Media method*), 212
video() (*epyk.interfaces.components.CompVignets.Vignets method*), 329
view() (*epyk.interfaces.components.CompFields.Timelines method*), 126
viewHeight (*epyk.core.js.Js.JsBase property*), 536, 601
viewport() (*epyk.core.html.Header.Meta method*), 636
vignet() (*epyk.interfaces.components.CompVignets.Vignets method*), 330
Vignets (*class in epyk.interfaces.components.CompVignets*), 325
vignets (*epyk.interfaces.Interface.Components property*), 491
Vis (*class in epyk.interfaces.graphs.CompChartsVis*), 459
vis (*epyk.core.data.Data.DataSrc property*), 568
vis (*epyk.interfaces.graphs.CompCharts.Chart2d property*), 355
vis (*epyk.interfaces.graphs.CompCharts.Chart3d property*), 355
vis (*epyk.interfaces.graphs.CompCharts.Graphs property*), 361
Vis2D (*class in epyk.interfaces.graphs.CompChartsVis*), 461
Vis3D (*class in epyk.interfaces.graphs.CompChartsVis*), 464
visited (*epyk.core.css.styles.classes.CssStyle.Style property*), 580
votes() (*epyk.interfaces.components.CompNetwork.Network method*), 250
vue() (*in module epyk.core.cli.cli_npm*), 70
vue_parser() (*in module epyk.core.cli.cli_npm*), 70
- ## W
- w3cTryIt()** (*epyk.core.py.PyOuts.PyOuts method*), 541
wallpaper() (*epyk.interfaces.components.CompImages.Images method*), 167
warn() (*epyk.core.js.Js.JsConsole method*), 495, 605
warning() (*epyk.interfaces.components.CompIcons.Icons method*), 153
warning() (*epyk.interfaces.components.CompNetwork.Network method*), 250
wbr() (*epyk.interfaces.components.CompTags.Tags method*), 300
web (*epyk.core.Page.Report property*), 574
web() (*epyk.core.py.PyOuts.PyOuts method*), 541
WebComponents (*class in epyk.interfaces.Interface*), 63
webkit_slider_thumb (*epyk.core.css.styles.classes.CssStyle.Style property*), 580
- webscrapping()** (*epyk.core.data.Data.DataSrc method*), 568
webscrapping() (*epyk.core.py.PyRest.PyRest static method*), 550
website() (*epyk.core.js.Imports.ImportManager method*), 61
website() (*epyk.interfaces.components.CompTexts.TextReferences method*), 301
WebSocket (*class in epyk.core.js.Js.JsWebSocket*), 620
websocket() (*epyk.core.js.Js.JsBase method*), 536, 601
week() (*epyk.interfaces.components.CompFields.Timelines method*), 126
weeks() (*epyk.interfaces.components.CompFields.Fields method*), 123
where() (*epyk.core.py.PySql.SqlConn method*), 557
while_() (*epyk.core.js.Js.JsBase method*), 536, 601
width (*epyk.core.js.Js.JsScreen property*), 607
wordcloud() (*epyk.interfaces.graphs.CompChartsChartJs.ChartJs method*), 399
worker() (*epyk.core.js.Js.JsBase method*), 536, 601
workload() (*epyk.interfaces.components.CompFields.Timelines method*), 127
world() (*epyk.interfaces.geo.CompGeoChartJs.BubbleMaps method*), 333
world() (*epyk.interfaces.geo.CompGeoChartJs.Choropleth method*), 335
world() (*epyk.interfaces.geo.CompGeoJqV.JqueryVektorMap method*), 342
world() (*epyk.interfaces.geo.CompGeoPlotly.PlotlyBubble method*), 348
world() (*epyk.interfaces.geo.CompGeoPlotly.PlotlyChoropleth method*), 352
wrench() (*epyk.interfaces.components.CompIcons.Icons method*), 154
writeln() (*epyk.core.js.Js.JsBase method*), 537, 602
- ## X
- xml()** (*epyk.interfaces.components.CompCodes.Code method*), 102
- ## Y
- years()** (*epyk.interfaces.components.CompFields.Fields method*), 124
youtube() (*epyk.interfaces.components.CompIcons.Icons method*), 154
youtube() (*epyk.interfaces.components.CompImages.Images method*), 168
youtube() (*epyk.interfaces.components.CompMedia.Media method*), 212
- ## Z
- zoom()** (*epyk.interfaces.components.CompIcons.Icons method*), 155

`zoom_in()` (*epyk.interfaces.components.CompIcons.Icons*
method), [155](#)
`zoom_out()` (*epyk.interfaces.components.CompIcons.Icons*
method), [156](#)