
Flare

Release 1.0

Tilman Oestereich, Andreas H. Kelch

May 20, 2022

FLARE

1	About	3
	Python Module Index	167
	Index	169



Web-App development framework for Python

ABOUT

flare is an app development framework for Python-based web-apps running on top of [Pyodide](#) in the browser.

It has integrations to concepts with [ViUR](#), an MVC-framework for the Google App Engine platform, but can also be used stand-alone.

Fire up the tiny [Hello World](#) live demo.

1.1 Getting started

1.1.1 System requirements

soon...

Serving own Pyodide

The script `bin/get-pyodide.py` downloads a minimal Pyodide with only `micropip` and `setuptools` from the Pyodide CDN. Pyodide can also be entirely built and configured on your own, for this check [the documentation](#).

Depending on the location where you want to serve your app, some more configuration might be necessary regarding the WASM mimetype.

Google App Engine

To serve your own Pyodide via Google App Engine, add the following lines to your `app.yaml` file and modify them when needed, as Google App Engine doesn't recognize WASM files correctly.

```
handlers:
- url: /pyodide/(.*\.wasm)$
  static_files: pyodide/\1
  upload: pyodide/.*\.wasm$
  mime_type: application/wasm
- url: /pyodide
  static_dir: pyodide
```

Apache Webserver

For apache web-server, this .htaccess configuration helped to serve the app correctly.

```
RewriteEngine off
Options -ExecCGI +Indexes
IndexOrderDefault Descending Date

#Header always set Access-Control-Allow-Origin "*"
#Header always set Access-Control-Allow-Methods GET

<FilesMatch "\.py$">
    Options +Indexes -ExecCGI -Multiviews
    Order allow,deny
    Allow from all
    RemoveHandler .py
    AddType text/plain .py
</FilesMatch>

<FilesMatch "\.data$">
    Options +Indexes -ExecCGI -Multiviews
    Order allow,deny
    Allow from all
    RemoveHandler .data
    AddType application/octet-stream .data
</FilesMatch>

<FilesMatch "\.wasm$">
    Options +Indexes -ExecCGI -Multiviews
    Order allow,deny
    Allow from all
    RemoveHandler .wasm
    AddType application/wasm .wasm
</FilesMatch>
```

1.1.2 Setup and installation

Linux or WSL

soon...

Mac OS

soon...

1.1.3 Testproject

Setting up a new Python web-app with *flare* is fairly easy. This section describes several things and ways how *flare* can be used and configured.

HTML skeleton

Below is a shortened version of the code from *hello.html* delivered together with the *flare* repo. Such a skeleton must be individually created for an app written with *flare*.

Caution: Depending on where you put the html files, you need to change the source paths:

- `<link rel="stylesheet" href="{path-to-flare-directory}/assets/css/style.css"/>`
- `<script src="{path-to-flare-directory}/assets/js/flare.js"></script>`
- “path”: “{path-to-flare-directory}/flare”

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="assets/css/style.css"/>

  <!-- (1) -->
  <script src="https://pyodide-cdn2.iodide.io/v0.16.1/full/pyodide.js"></script>
  <!-- <script src="pyodide/pyodide.js"></script> -->

  <!-- (2) -->
  <script src="assets/js/flare.js"></script>

  <script>
    window.addEventListener(
      "load",
      (event) => {
        window.init = new flare({
          prelude:                                // (3)
          ,
          print("I'm before any fetch")
          ,
          fetch: {                                // (4)
            "flare": {
              "path": "flare"
            }
          },
          kickoff:                                // (5)
          ,
          from flare import *
          html5.Body().appendChild('<a href="https://www.viur.dev">Hello World</a>')
          flare.popup.Alert("Hello World")
          ,
          });
        }
      );
  </script>
```

(continues on next page)

(continued from previous page)

```

</head>
<body class="is-loading"> <!-- (6) -->
</body>
</html>

```

Notable are the following sections:

1. This is the include for the used Pyodide version. When quickly setting up a project, the default CDN version of Pyodide can be used and is loaded from here. Indeed, it is also possible to serve Pyodide on your own. For this, the utility script `bin/get-pyodide.py` can be used. This script downloads a minimal version of Pyodide delivered from the CDN and stores it into a folder named `pyodide/`. In such a case, the CDN-include here must be removed, and replaced by the local include. `get-pyodide.py` patches some Pyodide-files to directly run from the URL `/pyodide`. You can override this setting by specifying a variable `window.languagePluginLoader` before including the `pyodide.js`.
2. *flare* serves a piece of JavaScript code that is necessary to pre-load flare itself and the Python application. For development, it was useful to directly fetch the py-files from the server and store them into a browser-internal filesystem when the Python interpreter from Pyodide can find it. This is done using the module in `init.js` and the configuration described next.
3. `prelude` is some Python code that is executed before any modules are fetched. It can be omitted, if not wanted.
4. `fetch` describes Python source modules that are being fetched before the application starts. This is very useful for development purposes. For every entry (which is the name of the Python package to be created), a further object describing the fetch path and an optional `optional` attribute is provided. Using the `path`-attribute, the *flare* init script looks for a file `files.json` which provides a listing of the files being fetched. This file is generated using `bin/gen-files-json.py` which is described below. A Pyodide package can also be pre-compiled from source files, but this is not described in detail here, yet.
5. `kickoff` is the Python code that is executed when all fetching is done and nothing failed. It is used as the entry point to start the web-app. In the *hello.html* file, it is just some “Hello World” stuff dumped out using flare.
6. The class `is-loading` is automatically removed when the kickoff code successfully executed. It can be used to show a loading animation or something similar.

Writing huger apps

When writing huger apps with multiple Python files, the above example doesn’t satisfy. For this case, an HTML-file like above still serves as the entry point for the app, but requires a little more configuration.

Let’s thing about the following minimal setup for a huger app:

- `/flare` is the flare repo serving as a library `_ /myapp` contains our app, which exists only of the files
 - `index.html` the app entry HTML
 - `__init__.py` the app source code
 - `files.json` which is the index file for the *flare* init script to find its sources

We only describe the files in `/myapp`:

index.html

```

<!doctype html>
<html>
<head>
  <meta charset="UTF-8">

```

(continues on next page)

(continued from previous page)

```

<script src="https://pyodide-cdn2.iodide.io/v0.16.1/full/pyodide.js"></script>
<script src="/flare/assets/js/flare.js"></script>
<script>
    window.addEventListener(
        "load",
        (event) => {
            window.init = new flare({
                fetch: {
                    "flare": {
                        "path": "/flare/flare"
                    },
                    "myapp": {
                        "path": "."
                    }
                }
            });
        }
    );
</script>
</head>
<body class="is-loading">
</body>
</html>

```

init.py:

```

from flare import *

if __name__ == "myapp":
    html5.Body().appendChild('<a href="https://www.viur.dev">Hello World</a>')
    popup.Alert("Hello World")

```

files.json:

```

[
    "__init__.py"
]

```

The `files.json` was simply generated using the by `../flare/bin/gen-files-json.py`. Whenever a Python file is added, this must be done once. The `files.json` should also be added to version control, to make the app run out-of-the-box.

1.2 Reference Guide

1.2.1 Terminology

Originally, Flare was developed for ViUR. Because of these roots, some terms or objects may cause confusion without further explanation.

Lets start with a quick overview of some main components.

config.py This is a central location where you can store data or information that needs to be shared by the entire application. Some Examples are

- paths
- caches
- configurations
- versions

network.py The default format used for data exchange is json. Each query is made via the NetworkService class in network.py.

views Views are used to divide content within the application. There is always one view that is active. They are saved after their instantiation in a object and are only hooked into the DOM when this view is activated. A view can contain multiple widgets that replace the existing content when activated.

safeeval Executes a string containing Python code. The possible operations are strongly limited for security reasons. With safeeval *flare-if* can show and hide content without the need of coding, and *{{ expressions }}* can directly be interpreted inside of HTML-code.

icons The icon class can use any of the common image types. In most cases, you want to have icons that match the font color. In this case flare requires svg icons.

priorityqueues PriorityQueues are used to provide a plugin capability. For each type there is somewhere centrally an instance of such a PriorityQueue. In this the options are added with the insert function and prioritized with a numerical value and validation function. If now a suitable option is searched, the select function is called with parameters which are passed to the validation function. The first matching option in the prioritized list is then returned.

Only relevant if used with ViUR

bones Bones are in the ViUR ecosystem data field definitions. There are different types and hold besides the type information also display information like a description and tooltips.

moduleInfo Modules are the controllers of a ViUR application, and implement the application logic. In the case of relations, information about the target module may be required, which can be found in the module info.

1.2.2 Configuration

Flare is divided into different components, which have different complexity. In addition to a flare config, the forms and views components have their own config.

Flare config

Here are some default values configured.

- **flare.icon.svg.embedding.path**
 - defines the basepath for the used svg icons.
- **flare.icon.fallback.error**
 - defines the fallback icon name
- **flare.language.current**
 - sets the current active language

The views config will be merged on top.

Bind App

An app that uses flare often has its own config object. Flare provides a `bindApp` function that, in addition to setting the app instance in the configuration, also allows overriding the flare configuration. For example, you can change the default language in your app configuration and all Flare components will use that value instead of the default flare settings.

1.2.3 html5 (core library)

Any **flare** components are entirely established on top of the *html5*-library.

The *html5 library* is flare's core module and key feature, and manages access to the browser's DOM and its items, by implementing a Python object wrapper class for any HTML-element. Such an element is called *widget*. For example, `html5.Div()` is the widget representing a div-element, or `html5.A()` a widget representing an a-element. Widgets can be sub-classed into specialized components, which contain other widgets and components and interact together.

The document's body and head can directly be accessed by the static widgets `html5.Head()` and `html5.Body()`.

All these widgets are inheriting from an abstract widget wrapper called `html5.Widget`. `html5.Widget` is the overall superclass which contains most of the functions used when working with DOM elements. Therefore, all widgets are usually handled the same way, except leaf-type widgets, which may not contain any children.

First steps

When working with native html5-widgets, every widget must be created separately and stacked together in the desired order. This is well known from JavaScript's `createElement`-function.

Here's a little code sample.

```
from flare import html5

# Creating a new a-widget
a = html5.A()
a["href"] = "https://www.viur.dev" # assign value to href-attribute
```

(continues on next page)

(continued from previous page)

```

a["target"] = "_blank"           # assign value to target-attribute
a.addClass("link")               # Add style class "link" to element

# Append text node "Hello World" to the a-element
a.appendChild(html5.TextNode("Hello World"))

# Append the a-widget to the body-widget
html5.Body().appendChild(a)

```

Summarized:

- `html5.Xyz()` creates an instance of the desired widget. The notation is that the first letter is always in uppercase-order, the rest is hold in lowercase-order, therefore e.g. `html5.Textarea()` is used for a textarea.
- Attributes are accessible via the attribute indexing syntax, like `widget["attribute"]`. There are some special attributes like `style` or `data` that are providing a dict-like access, so `widget["style"]["border"] = "1px solid red"` is used.
- Stacking is performed with `widget.appendChild()`. There are also some additional functions for easier element stacking and child modification, these are - `widget.prependChild()` to prepend children, - `widget.insertBefore()` to insert a child before another child, - `widget.removeChild()` to remove a child.
- To access existing child widgets, use `widget.children(n)` to access the *n*-th child, or without *n* to retrieve a list of a children.

Parsing widgets from HTML-code

Above result can also be achieved much faster, by using the build-in *html5-parser and renderer*.

```

from flare import *

html5.Body().appendChild(
    "<a href='https://www.viur.dev' target='_blank' class='viur'>Hello World</a>"
)

```

That's quite simpler, right? This is a very handy feature for prototyping and to quickly integrate new HTML layouts.

`Widget.appendChild()` and other, corresponding functions, allow for an arbitrary number of elements to be added. HTML-code, widgets, text or even lists or tuples of those can be given, like so

```

ul = html5.Ul()
ul.appendChild("<li class='is-active'>lol</li>")
ul.prependChild(html5.Li(1337 * 42))
ul.appendChild("<li>me too</li>", html5.Li("and same as I"))

```

The HTML parser can also do more: When component classes (any class that inherits directly from `html5.Widget`, like `html5.Div` or so) are decorated with the *html5.tag*-decorator, these are automatically made available in the HTML-parser for recognition.

Inheritance is normal

In most cases, both methods shown above are used together where necessary and useful. Especially when creating new components with a custom behavior inside your app, knowledge of both worlds is required.

To create new components, inheriting from existing widgets is usual. If we would like to add our link multiple times within our app, with additional click tracking, we can make it a separate component, like so:

```
import logging
from flare import *

class Link(html5.A): # inherit Link from html5.A widget
    def __init__(self, url, *args, target="_blank", **kwargs):
        super().__init__()
        self.addClass("link")
        self["href"] = url
        self["target"] = "_blank"

        self.appendChild(*args, **kwargs)
        self.sinkEvent("onClick")

    def onClick(self, event):
        logging.info(f"The link to {self['href']} has been clicked")

html5.Body().appendChild(
    # Create a link with text
    Link("https://www.viur.dev", "ViUR Framework"),

    "<br>",

    # Create link with logo
    Link("https://www.python.org", """
        
        """)
)
```

In this example, we just made our first custom component: The Link-class can be arbitrarily used.

Widget basics

Following sections describe the most widely used functions of the :class:`html5.Widget` <flare.html5.Widget>-class which are inherited by any widget or huger component in flare.

Constructor

All widgets share the same `__init__`-function, having the following signature:

```
def __init__(self, *args, appendTo=None, style=None, **kwargs)
```

- `*args` are any positional arguments that are passed to `self.appendChild()`. These can be either other widgets or strings containing HTML-code. Non-container widgets like `html5.Br()` or `html5.Hr()` don't allow anything passed to this parameter, and throw an Exception.
- `appendTo` can be set to another `html5.Widget` where the constructed widget automatically will be appended to. It substitutes an additional `appendChild()`-call to insert the constructed Widget to the parent.
- `style` allows to specify CSS-classes which are added to the constructed widget using
- `**kwargs` specifies any other parameters that are passed to `appendChild()`, like variables.

Insertion and removal

These methods manipulate the DOM and it's nodes

`appendChild()`

Appends another `html5.Widget` as child to the parent element:

```
self.appendChild("""<ul class='navlist'></ul>""")
self.nav.appendChild("""<li>Navigation Point 1</li>""")
```

`prependChild()`

Prepends a new child to the parent element

```
self.appendChild("""<ul class='navlist'></ul>""")
navpoint2 = self.nav.appendChild("""<li>Navigation Point 2</li>""")
navpoint2.prependChild("""<li>Navigation Point 1</li>""")
```


replaceChild()

Same as appendChild(), but removes the current children of the Widget first.

insertBefore()

Inserts a new child element before the target child element

```

self.appendChild("""<ul class='navlist'></ul>""")
navpoint = self.nav.appendChild("""<li>Navigation Point 1</li>""")
navpoint3 = self.nav.appendChild("""<li>Navigation Point 3</li>""")
navpoint2 = self.nav.insertBefore("""<li>Navigation Point 2</li>""", navpoint3)

```

If the child element that the new element is supposed to be inserted before does not exist, the new element is appended to the parent instead.

removeChild(), removeAllChildren()

Either removes one child from the parent element or any available children.

Visibility and usability

Widgets can be switched hidden or disabled. Form elements, for example, might be disabled when a specific condition isn't met. These functions here help to quickly change visibility and usability of widgets, including their child widgets which are switched recursively.

hide(), show()

Hides or shows a widget on demand.

To check whether a widget is hidden or not, evaluate `widget["hidden"]`. In the HTML-parser, this flag can be set using the hidden attribute, e.g. `<div hidden>You can't see me.</div>`.

enable(), disable()

Enable or disable the widget in the DOM. Useful for forms and similar UI applications.

To check whether a widget is disabled or not, evaluate `widget["disabled"]`. In the HTML-parser, this flag can be set using the disabled attribute, e.g. `<div disabled>I'm disabled</div>`.

class-attribute modification

These methods are helpful for adding CSS-classes quickly.

addClass()

Adds a class to the `html5.Widget` and checks to prevent adding the same class multiple times.

```
nav = self.appendChild("""<ul></ul>""")
nav.addClass('navlist')
```

Adding a class multiple times might be wanted and is valid. In this case, modify the widget's `class`-attribute directly by assigning a list to it.

removeClass()

Checks if the widget has that class and removes it

```
nav = self.appendChild("""<ul class='big-red-warning-border-color'></ul>""")
nav.removeClass('big-red-warning-border-color')
```

toggleClass()

Toggles a class *on* or *off*, depending on whether it has the specified class already or not.

hasClass()

Checks if the element has a given class or not. Returns `True` if class name is found and `False` otherwise.

```
nav = self.appendChild("""<ul class='big-red-warning-border-color'></ul>""")
if nav.hasClass('big-red-warning-border-color'):
    print("Help! There is a big red border around this element! Remove the class so we
↪ can feel safe again")
```

HTML parser reference

The `html5`-library built into `flare` brings its own HTML-parser. Using this parser, any HTML-code can directly be turned into a `flare` DOM.

Additionally, some nice extensions regarding `flare` component and widget customization and conditional rendering is supported, as the HTML-renderer automatically creates the DOM from a parsed input and serves as some kind of template processor.

Data-based rendering

Using variables

Any variables provided via kwargs to `html5.fromHTML()` can be inserted in attributes or as `TextNode`-elements with their particular content when surrounded by `{{` and `}}`. Inside this notation, full Python expression syntax is allowed, so that even calculations or concatenations can be done.

```
html5.Body().appendChild("""
    <div class="color-{{ l[1] + 40 }}">{{ d["world"] + "World" * 3 }} and {{ d }}</div>
""", l=[1,2,3], d={"world": "Hello"})
```

renders into

```
<div class="color-42">HelloWorldWorldWorld and {'world': 'Hello'}</div>
```

flare-if, flare-elif, flare-else

The attributes `flare-if`, `flare-elif` and `flare-else` can be used on all tags for conditional rendering.

This allows for any simple Python expression that evaluates to `True` or any computed non-boolean value representing `True`.

```
html5.Body().appendChild("""
    <div>begin</div>
    <div flare-if="i <= 10">i is just low</div>
    <div flare-elif="i <= 50 and j >=100">i and j have normal values</div>
    <div flare-elif="i > 50 and j >= 50">i and j have moderate values</div>
    <div flare-else>i and j are something different</div>
    <div>end</div>
""", i=50, j=151)
```

As variables, any arguments given to `html5.fromHTML()` (or related functions) as kwargs can be used.

html5.parseHTML()

```
def parseHTML(html: str, debug: bool=False) -> HtmlAst
```

Parses the provided HTML-code according to the tags registered by `html5.registerTag()` or components that use the `@tag`-decorator.

The function returns an abstract syntax tree representation (`HtmlAst`) of the HTML-code that can be rendered by `html5.fromHTML()`.

html5.fromHTML()

```
def fromHTML(html: [str, HtmlAst], appendTo: Widget=None, bindTo: Widget=None, debug: ↵
↳ bool=False, **kwargs) -> [Widget]
```

Renders HTML-code or compiled HTML-code (HtmlAst).

- appendTo: Defines the Widget where to append the generated widgets to
- bindTo: Defines the Widget where to bind widgets using the [name]-attribute to
- debug: Debugging output
- **kwargs: Any specified kwargs are available as *variables to any expressions*.

HTML-code can optionally be pre-compiled with `html5.parseHTML()`, and then executed multiple times (but with different variables) by `fromHTML`. This is useful when generating lists of same elements with only replaced variable data.

@html5.tag

Decorator to register a sub-class of `html5.Widget` either under its class-name, or an associated tag-name.

Examples:

```
from flare import html5

# register class Foo as <foo>-Tag
@html5.tag
class Foo(html5.Div):
    pass

# register class Bar as <baz>-Tag
@html5.tag("baz")
class Bar(html5.Div):
    pass
```

1.2.4 Ignite

Ignite is a CSS-framework written in LESS and serving as the base for all components used in flare. <https://ignite.viur.dev/>

In Flare, some simpler and more complex components are already implemented with appropriate CSS-classes.

Button

The Button can be used with `<flare-button>` tag und provides the possibility to add an icon before the Button text.

Input

The Input can be used with `<flare-input>` tag and provides the basis input element with ignite specific css classes.

Label

The Label can be used with `<flare-label>` tag and provides the basis label element with ignite specific css classes.

Switch

The switch is an on/off slide-control and can be used with `<flare-switch>` tag. The component stores the current state internally in a checkbox input field.

Check

The Check Component can be used with `<flare-check>` tag. Like the switch, the internal state is stored in a checkbox input field. Through this component the display of the checkbox can be customized via css.

Radio

The Radio Component can be used with `<flare-radio>` tag. The internal state is stored in a radio input field. Through this component the display of the checkbox can be customized via css.

Select

The Select can be used with `<flare-select>` tag and provides the basis select element with ignite specific css classes. In addition it adds per default a unselectable default option.

Textarea

The Textarea can be used with `<flare-textarea>` tag and provides the basis textarea element with ignite specific css classes.

Progress

The Progress can be used with `<flare-progress>` tag and provides the basis progress element with ignite specific css classes.

Item

The Item component can be used with the tag `<flare-item>` and provides a simple box component. It can contain an image as well as a title and a description

Table

The Table component can be used with the `<flare-table>` tag and provides the basis table element with ignite specific css classes. In addition this component provides the functions `prepareRow` and `prepareCol` to generate the table grid.

Popout

The Popout component can be used with the `<flare-popout>` tag. This component is a floating box and is often used as a tooltip or contextmenu. With the css classes “popout-sw”, “popout-nw”.. you can change the direction.

Popup

There are several types of popup windows. All popups are based on the base Popup Class. Each popup provides a close button, a header, a body and a footer. All Popups are automatically added to the `<body>`-tag.

Prompt

The Prompt is a simple Input box with a cancel and ok button. Use this to get some user Input.

Alert

The Alert is a simple MessageBox with an ok button. Use this for some Feedback.

Confirm

The Confirm is a MessageBox with a yes / no selection. Each button has its own callback so you can bump different actions based on the selection that was made

Textarea Dialog

This Popup basically does the same as the Prompt, but it uses a textarea field instead of an input field.

1.2.5 Network

The *network*-module contains some classes and functions that allow to communicate or work with other services.

Requesting data

The following classes are used to request data from another service.

HTTPRequest

HTTPRequest is a tiny wrapper around the Javascript object XMLHttpRequest. Only the OPENED (1) and DONE (4) statuses are used. In case of OPENED the payload is sent. If it is a post request, a possibly existing content type header is also set. Depending on the status, the success callback or the failure callback specified during instantiation is called.

```
HTTPRequest("GET", url, mySuccessFunction, myFailureFunction)
```

This tiny wrapper is used by the NetworkService, which encapsulates some ViUR-related request types

NetworkService

This function can be passed the following parameters in addition to the callback functions for success, failure and finished:

- module (str): Name of the target ViUR Module or None
- url (str): Path (relative to Module)
- params (dict): Dictionary of key-values paired url parameters
- modifies (bool): previously registered classes can be notified with a onDataChange event
- secure (bool): for this ViUR request is an skey need, so fetch it before the request
- kickoff (bool): by default this value is true, but you can use it to wait before to start a request
- group (requestGroup): use this to bundle multiple requests and get at the end a final callback

This could be a simple request to test on a ViUR System if a user is logged in

```
NetworkService.request( "user", "view/self",
                        successHandler=iamAlreadyLoggedInFunction,
                        failureHandler=loginFunction)
```

Sometimes you need to do a bunch of requests with a callback at the end

```
agroup = requestGroup( allRequestsSuccessFunction )

for aKey in dbKeyListToDelete:
    NetworkService.request( amodule, "delete", { "key": aKey },
                           secure = True, #in case of deletion ViUR needs an skey
                           modifies = False, #avoids the onDataChange event
                           group=agroup,
                           successHandler = singleItemSuccessFunction,
                           failureHandler = singleItemFailureFunction )
```

requestGroup

This class is used to execute several requests of the NetworkService one by one and finally call the callback specified during instantiation. In this case, be sure to set kickoff to False.

Other useful functions

The following functions were often used in connection with data queries and were therefore placed here.

DeferredCall

This is a wrapper around the setTimeout JavascriptObject. After a delay time (default:25ms) the given function is called with the given parameters. This function is called outside the surrounding application flow! Two hidden parameters can be specified during initialization and will not be passed to the function:

- `_delay`: modifies the Timeout delay
- `_callback`: will be called after handling the deferred Funktion

```
DeferredCall(doSomeStuffLaterFunction,  
            anArgumentForMyFunction,  
            _delay=1000,  
            _callback=sayHelloWennFinishedFunction)
```

1.2.6 Utils

soon...

1.2.7 Url handling

Flare Applications are SPA (Single Page Applications) the navigation is done via the #-hash part of the url. This part is treated by Flare like a normal url. The hash should have a form like this.

```
#/path/pathPart2../pathEnd?param1=value&param2=value
```

The following functions split the hash into the corresponding url components or reassemble them.

getUriHashAsString

This function takes the hash of the url and splits it into args and kwargs. The return value is a tuple of the args string and the kwargs string. In most cases you want to use `getUriHashAsObject` instead.

getUrlHashAsObject

Uses the return value of `getUrlHashAsString` and also creates a tuple consisting of args and kwargs. But now the first value is a list and the second is a dictionary.

setUrlHash

This function takes the objects from `getUrlHashAsObject` and reassembles them into a valid hash and finally sets the new url.

```
urlHash, urlParams = getUrlHashAsObject() #read hash
urlParams.update({"key":"newValue"}) #modify
setUrlHash(urlHash,urlParams) #write back
```

example

```
# current URL:
# http://localhost:8080/app/app.html#/user/list?amount=99&status=10
urlHash, urlParams = getUrlHashAsObject() #read hash
print(urlHash,urlParams)
#[ 'user', 'list' ] { "amount": "99", "status": "10" }
urlParams.update({"status": "5"}) # change query
setUrlHash(urlHash,urlParams) #write back to Url
# new URL:
# http://localhost:8080/app/app.html#/user/list?amount=99&status=5
```

1.2.8 i18n

Flare provides the possibility to translate texts depending on the selected language. For each language a Python file with the language abbreviation is created in a folder called ‘translations’. A dictionary with the following name format is then expected in the file:

```
lngDe
lngEn
lngNl
lngFr
...
```

The dictionary itself contains a mapping between a keyword and the translation in the corresponding language.

```
lngDe = {
    "List": "Liste",
    "Username": "Nutzername ist {name}"
    ...
}
```

To use these dictionaries they have to be initialized when starting the application.

```
from flare.i18n import buildTranslations, translate
buildTranslations("app") #the parameter is the name of the root folder
```

(continues on next page)

(continued from previous page)

```
#now you can use the translate function to get the translated text  
  
print(translate('List'))  
# "Liste"
```

translate

The Translate function can additionally have a fallback and any other parameters, which then replace marked positions in a template string.

```
print(translate('Username', {"name":"Alice"}))  
# "Nutzername is Alice"
```

addTranslation

You can also update a translation at runtime. For this you have to specify the language, the keyword and the translation.

```
addTranslation("de", "user", "Nutzer")
```

more functions

the functions `getLanguage` and `setLanguage(lang)` allow to change and request the current language. After changing the language, it must be ensured that templates are rebuilt.

1.2.9 SVG Icons

Icons are dependent on css styling in flare. So icons in a text can have the same color as the surrounding text. This is possible by embedding svg icons. If the tag `flare-svg-icon` is used, the parameter value can be used to specify a path or name to an icon.

```
<flare-svg-icon value="/static/icons/my-icon.svg">
```

To keep the code in flare clear, only the icon name can be specified. If only the name is specified, the config variable `conf["flare.icon.svg.embedding.path"]` is used to compose the path of flare.

```
<flare-svg-icon value="my-icon">
```

It is also possible to define a fallback icon in case the icon cannot be loaded. If the title is set it will be transferred to the svg and in case the fallback icon is not set the first character of the text will be used as placeholder.

```
<flare-svg-icon value="my-icon" fallbackIcon="error" title="My Icon">  
<!-- shows my-icon or on error the error icon -->  
  
<flare-svg-icon value="my-icon" title="My Icon">  
<!-- shows my-icon or the letter M -->
```

Icon

In practice icons can come in different file types. flare-icon can also handle other images and even use filebones directly. In case the icon is not an svg, it is not embedded, but included using img-tag. flare-icon can use the following image types:

- *.svg, *.jpg, *.png, *.gif, *.bmp, *.webp, *.jpeg

```
<flare-icon value="{{skel['image']}}">
<!--loads a filebone-->
```

The Svg-icon parameters fallbackIcon and title are also supported.

1.2.10 Views

Views allow switching between different widgets. A view must inherit from the View class and can update multiple View-Widgets of the conf["app"]. Additionally, the dictOfWidgets must be filled in the constructor before the super call. The key of the dictionary must be present in the conf["app"] widget.

A view widget is the actual content that is then inserted into a widget in the main app. These view widgets must inherit from ViewWidget.

The currently active view is stored in a global state under conf["views_state"].

create a View

```
from flare.views.view import View, ViewWidget

class myView(View):
    def __init__(self):
        dictOfWidgets = {
            "content" : myViewContent
            #each key muss exists as instancevariable in conf["app"]
        }
        super().__init__(dictOfWidgets)

class myViewContent(ViewWidget):
    def initWidget( self ):
        self.appendChild("Hello View")

    def onViewfocusedChanged( self, viewname, *args, **kwargs ):
        pass #here we can execute code, which muss be called wenn e View gets
↪ focus
```

register a View

At this point, we have created a view. We have defined that the widget content from the main app should be replaced by the one from myViewContent. Now we need to register this view.

```
from flare.views.helpers import addView, removeView

addView(myView, "page1")
```

In this example, the view myView is registered under the name page1. A view can also be registered under multiple names. removeView removes the view again.

activate / switch a view

To activate a view the view instance of the state conf["views_state"] must be updated. The status stores the name of the view that is currently displayed and can be updated as follows.

```
conf["views_state"].updateState("activeView", "page1")
```

Views with ViUR

In ViUR, modules have different views depending on the handler. generateView here encapsulates module name, actionname and data away in a params dictionary, which is then available in the view and can be loaded from the view in any ViewWidget.

```
#item is a adminInfo Entry

# generate a unique instancename, because a edit can be opened multiple times with same
↳parameters
instancename = "%s__%s" % (item[ "moduleName" ]+item[ "handler" ], str( time.time() ).
↳replace( ".", "_" ))

#create new viewInstance
viewInst = generateView( myView, item[ "moduleName" ], item[ "handler" ], data = item,
↳name=instancename )

#register this new view
conf[ "views_registered" ].update( { instancename: viewInst } )

# somewhere else in code, i.e in a Navigation
conf["views_state"].updateState("activeView", instancename)
```

1.2.11 ViUR

soon...

1.2.12 Safeeval

soon...

1.3 Tutorials

1.3.1 Hello World

In this tutorial, we will create a basic project that makes use of flare to create a simple web-app.

Project setup

In order to make flare accessible in your project, either download the flare master branch from github and extract it into a flare subdirectory in your project, or - if you are using git - clone it into a git submodule of your project by calling `git submodule add git@github.com:viur-framework/flare.git`.

Once this is done, you can create an `index.html` file that will make use of the now available flare assets.

The HTML

Basically all you need to do is add the flare CSS sheet and javascript file to your HTML file and you are good to go.

```
<link rel="stylesheet" href="flare/assets/css/style.css"/>
<script src="flare/assets/js/flare.js"></script>
```

A simple `index.html` file that uses flare might now look like this:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Hello World</title>
  <link rel="stylesheet" href="flare/assets/css/style.css"/>
  <script src="flare/assets/js/flare.js"></script>
  <script>
    window.addEventListener("load", () => {
      new flare({
        fetch: {
          "flare": {
            "path": "flare/flare"
          }
        },
        kickoff:
      },
    },
  )
  from flare import *
  flare.popup.Alert("Hello World")
```

(continues on next page)

(continued from previous page)

```

        });
    });
</script>
</head>
<body class="is-loading">
</body>
</html>

```

Building from there

The `fetch` block is where the flare python modules are being loaded at application start. It is advisable to add your own python module structure fairly quickly.

1. Add a subdirectory `helloworld` next to your `index.html`.
2. Add a file `__init__.py`:

```
from . import helloworld
```

3. Add a file `helloworld.py`:

```

from flare import *

class HelloWorld(object):
    _message = None

    def __init__(self, message="Hello World"):
        self._message = message

    def show(self):
        popup.Alert(self._message)

```

4. Create a `files.json` file in your module directory and add the following content:

```

[
    "__init__.py",
    "helloworld.py"
]

```

5. Add a second block to the `fetch` in your `index.html`:

```

fetch: {
    "flare": {
        "path": "flare/flare"
    },
    "helloworld": {
        "path": "helloworld"
    }
},

```

6. Change your kickoff script to run the code in your module, instead:

```
from helloworld import *
helloworld.HelloWorld("Hello module world!").show()
```

To execute your hello world sample you can use the test webserver located in the `flare/tools/` folder. Just run `test-server.py` in your project directory and open `http://localhost:8080/index.html` in your browser.

1.3.2 Request JSON data

In this tutorial, we will use flares API to load some JSON data from an API and process it.

Project setup

Please refer to the “Hello World” tutorial on how to set up a basic project with flare.

Using HTTPRequest

Flare comes with a high level API to request data. The `flare.network` module contains a class `HTTPRequest`, whose constructor takes six parameters:

1. `method`: The HTTP method to use for the request (i.e. GET, POST, ...)
2. `url`: The URL to request
3. `callbackSuccess` (optional): A reference to the function which is to be called when the request succeeds (takes a response parameter)
4. `callbackFailure` (optional): A reference to the function which is to be called when the request fails (take the parameters `responseText` and `status`)
5. `payload` (optional): The body of the request, if one is to be sent
6. `content_type` (optional): A value for a Content-Type header (e.g. `application/json`)

Using this constructor immediately sends the request.

Handling the response

In order to parse a JSON response in a success callback, simply use the default `json` functionality:

```
def successCallback(result):
    data = json.loads(result)
```

This will simply turn the response into the appropriate native structure, based on what kind of JSON has been returned:

- Objects will be turned into `dict`
- Arrays will be turned into `list`
- `null` will be turned into `None`
- atomar values will be turned into their respective python counterpart

Example

As an example, we will request the current time of the time zone Europe/Berlin from a public API, then display it in a popup.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Fetching data</title>
  <link rel="stylesheet" href="flare/assets/css/style.css"/>
  <script src="flare/assets/js/flare.js"></script>
  <script>
    window.addEventListener("load", () => {
      new flare({
        fetch: {
          "flare": {
            "path": "flare/flare"
          }
        },
        kickoff:
      },
      import json
      import logging
      from flare import *
      from flare.network import HTTPRequest

      def _successCallback(result):
        data = json.loads(result)
        flare.popup.Alert(data["datetime"])

      def _failureCallback(responseText, status):
        logging.error("Failure: %s %d", responseText, status)

      HTTPRequest(
        "GET",
        "http://worldtimeapi.org/api/timezone/Europe/Berlin",
        _successCallback,
        _failureCallback
      )
    })
  </script>
</head>
<body class="is-loading">
</body>
</html>
```


1.3.3 build a form

soon...

1.3.4 Views

In this tutorial, we will introduce flare's concept of views.

Building blocks

There are two basic concepts in flare in regards to views: The view itself, and the view widgets.

In essence, a view has a name, and it consists of a collection of view widgets, with the information on where in the DOM to actually display them. A view widget is a special kind of widget based on a html5 <div>, that is being hooked into and removed from the appropriate place in the DOM. It also gets a notification whenever view switching occurs.

Views rely on the concept of a central "app" class. Since the view mostly just contains a dictionary on which view widget to place where, it expects to find an app class which has the target elements as fields. The view will then use the field name as key in its view widget dictionary.

Example

As an example, we are going to create simple flip flop views: Two views, each containing a button to show the other view.

For this, we create a new file `views.py` where we put all the following code. We start off with the view widgets of the two views:

```
from flare import html5, bindApp
from flare.button import Button
from flare.config import conf, updateConf
from flare.views.view import View, ViewWidget
from flare.views.helpers import addView, removeView, updateDefaultView

class FlipViewContent(ViewWidget):
    def initWidget(self):
        self.appendChild(Button("Flip!", self.switch))
        self.appendChild(" - Flop!")

    def onViewfocusedChanged(self, viewname, *args, **kwargs):
        pass

    def switch(self):
        conf["views_state"].updateState("activeView", "flop")

class FlopViewContent(ViewWidget):
    def initWidget(self):
        self.appendChild("Flip! - ")
        self.appendChild(Button("Flop!", self.switch))

    def onViewfocusedChanged(self, viewname, *args, **kwargs):
```

(continues on next page)

(continued from previous page)

```

pass

def switch(self):
    conf["views_state"].updateState("activeView", "flip")

```

These two view widgets are virtually identical. They both contain a button that calls their `switch` method, which triggers the switch over to the other view, by changing `activeView` to the name of the other view. Next, we define the view classes themselves.

```

class FlipView(View):
    def __init__(self):
        super().__init__({
            "content": FlipViewContent
        })

class FlopView(View):
    def __init__(self):
        super().__init__({
            "content": FlopViewContent
        })

```

Again, these two views are virtually identical. All they do is contain the information on where to put their respective content. In this case, they both only have one view widget, and they both bind it to the same place: An element named `content`. In order for this resolution to work, there needs to be an app class, which has a field named `content` which points to the element where the view shall be rendered. Let's build one.

```

class App(html5.Div):

    def __init__(self):
        super(App, self).__init__()
        html5.Body().appendChild(self)
        bindApp(self, conf)

```

As you can see, we derive our app class from a div, hook it into the DOM, and call `bindApp` to register it in the configuration, so that the view system can access it. Now we add the `content` field to it, and make sure that it is properly connected to the DOM:

```

class App(html5.Div):
    content = html5.Div()

    def __init__(self):
        super(App, self).__init__()
        html5.Body().appendChild(self)
        bindApp(self, conf)
        self.appendChild(self.content)

```

Only one final step remains: Registering the two views, setting the flip view to active, and actually running the app.

```

class App(html5.Div):
    content = html5.Div()

    def __init__(self):

```

(continues on next page)

(continued from previous page)

```
super(App, self).__init__()
html5.Body().appendChild(self)
bindApp(self, conf)
self.appendChild(self.content)
addView(FlipView, "flip")
addView(FlopView, "flop")
conf["views_state"].updateState("activeView", "flip")
```

```
app = App()
```

As seen with the `addView` calls, we register the two view classes, giving them the names “flip” and “flop”. These names are then used to switch the `activeView`, as we already did earlier in the `switch` methods of the view widgets.

That’s it. Make flare load your `views.py` by adding the following lines of code to your `__init__.py` file:

```
from . import views

views.App()
```

Now and you can have fun with flipping and flopping the two views.

1.3.5 Translation

soon...

1.3.6 Update url

soon...

1.4 API Reference

This page contains auto-generated API reference documentation¹.

1.4.1 flare

Flare is an application development framework for writing software frontends in pure Python.

¹ Created with `sphinx-autoapi`

Subpackages

`flare.html5`

Submodules

`flare.html5.core`

HTML5 Widget abstraction library.

- Provides a Widget-abstraction for each HTML-element
- Routing of attribute getter/setter and JQuery-style helpers
- Fully-integrated HTML-parser for quick Widget prototyping

Module Contents

Classes

<i>TextNode</i>	Represents a piece of text inside the DOM.
<i>_WidgetClassWrapper</i>	Built-in mutable sequence.
<i>_WidgetDataWrapper</i>	dict() -> new empty dictionary
<i>_WidgetStyleWrapper</i>	dict() -> new empty dictionary
<i>Widget</i>	
<i>_attrLabel</i>	
<i>_attrCharset</i>	
<i>_attrCite</i>	
<i>_attrDatetime</i>	
<i>_attrForm</i>	
<i>_attrAlt</i>	
<i>_attrAutofocus</i>	
<i>_attrDisabled</i>	
<i>_attrChecked</i>	
<i>_attrIndeterminate</i>	
<i>_attrName</i>	
<i>_attrValue</i>	

continues on next page

Table 1 – continued from previous page

<i>_attrAutocomplete</i>
<i>_attrRequired</i>
<i>_attrMultiple</i>
<i>_attrSize</i>
<i>_attrFor</i>
<i>_attrInputs</i>
<i>_attrFormhead</i>
<i>_attrHref</i>
<i>_attrTarget</i>
<i>_attrType</i>
<i>_attrMedia</i>
<i>_attrDimensions</i>
<i>_attrUsemap</i>
<i>_attrMultimedia</i>
<i>_attrRel</i>
<i>_attrSrc</i>
<i>A</i>
<i>Area</i>
<i>Audio</i>
<i>Bdo</i>
<i>Blockquote</i>
<i>BodyCls</i>
<i>Canvas</i>
<i>Command</i>
<i>_Del</i>

continues on next page

Table 1 – continued from previous page

<i>Dialog</i>
<i>Abbr</i>
<i>Address</i>
<i>Article</i>
<i>Aside</i>
<i>B</i>
<i>Bdi</i>
<i>Br</i>
<i>Caption</i>
<i>Cite</i>
<i>Code</i>
<i>Datalist</i>
<i>Dfn</i>
<i>Div</i>
<i>Em</i>
<i>Embed</i>
<i>Figcaption</i>
<i>Figure</i>
<i>Footer</i>
<i>Header</i>
<i>H1</i>
<i>H2</i>
<i>H3</i>
<i>H4</i>
<i>H5</i>

continues on next page

Table 1 – continued from previous page

<i>H6</i>
<i>Hr</i>
<i>I</i>
<i>Kdb</i>
<i>Legend</i>
<i>Mark</i>
<i>Noscript</i>
<i>P</i>
<i>Rq</i>
<i>Rt</i>
<i>Ruby</i>
<i>S</i>
<i>Samp</i>
<i>Section</i>
<i>Small</i>
<i>Strong</i>
<i>Sub</i>
<i>Summery</i>
<i>Sup</i>
<i>U</i>
<i>Var</i>
<i>Wbr</i>
<i>Button</i>
<i>Fieldset</i>
<i>Form</i>

continues on next page

Table 1 – continued from previous page

<i>Input</i>
<i>Label</i>
<i>Optgroup</i>
<i>Option</i>
<i>Output</i>
<i>Select</i>
<i>Textarea</i>
<i>HeadCls</i>
<i>Iframe</i>
<i>Img</i>
<i>Ins</i>
<i>Keygen</i>
<i>Link</i>
<i>Ul</i>
<i>Ol</i>
<i>Li</i>
<i>Dl</i>
<i>Dt</i>
<i>Dd</i>
<i>Map</i>
<i>Menu</i>
<i>Meta</i>
<i>Meter</i>
<i>Nav</i>
<i>Object</i>

continues on next page

Table 1 – continued from previous page

<i>Param</i>	
<i>Progress</i>	
<i>Q</i>	
<i>Script</i>	
<i>Source</i>	
<i>Span</i>	
<i>Details</i>	
<i>Summary</i>	
<i>Style</i>	
<i>Tr</i>	
<i>Td</i>	
<i>Th</i>	
<i>Thead</i>	
<i>Tbody</i>	
<i>ColWrapper</i>	
<i>RowWrapper</i>	
<i>Table</i>	
<i>Time</i>	
<i>Track</i>	
<i>Video</i>	
<i>Template</i>	
<i>HtmlAst</i>	Abstract syntax tree element used by <code>parseHTML()</code> .

Functions

<i>domCreateAttribute</i> (tag, ns=None)	Creates a new HTML/SVG/... attribute.
<i>domCreateElement</i> (tag, ns=None)	Creates a new HTML/SVG/... tag.
<i>domCreateTextNode</i> (txt="")	
<i>domGetElementById</i> (idTag)	
<i>domElementFromPoint</i> (x, y)	
<i>domGetElementsByTagName</i> (tag)	
<i>domConvertEncodedText</i> (txt)	Convert HTML-encoded text (containing HTML entities) into its decoded string representation.
<i>Body</i> ()	
<i>Head</i> ()	
<i>unescape</i> (val, maxLength=0)	Unquotes several HTML-quoted characters in a string.
<i>doesEventHitWidgetOrParents</i> (event, widget)	Test if event 'event' hits widget 'widget' (or <i>any</i> of its parents).
<i>doesEventHitWidgetOrChildren</i> (event, widget)	Test if event 'event' hits widget 'widget' (or <i>any</i> of its children).
<i>textToHtml</i> (node, text)	Generates html nodes from text by splitting text into content and into line breaks <code>html5.Br</code> .
<i>parseInt</i> (s, ret=0)	Parses a value as int.
<i>parseFloat</i> (s, ret=0.0)	Parses a value as float.
<i>getKey</i> (event)	Returns the Key Identifier of the given event.
<i>isArrowLeft</i> (event)	
<i>isArrowUp</i> (event)	
<i>isArrowRight</i> (event)	
<i>isArrowDown</i> (event)	
<i>isEscape</i> (event)	
<i>isReturn</i> (event)	
<i>isControl</i> (event)	
<i>isShift</i> (event)	
<i>isMeta</i> (event)	
<i>registerTag</i> (tagName, widgetClass, override=True)	
<i>tag</i> (arg)	Decorator to register a sub-class of <code>html5.Widget</code> either under its class-name or an associated tag-name.
<i>_buildTags</i> (debug=False)	Generates a dictionary of all to the <code>html5</code> -library known tags and their associated objects and attributes.
<i>parseHTML</i> (html: str, debug: bool = False) → <code>HtmlAst</code>	Parses the provided HTML-code according to the tags registered by <code>html5.registerTag()</code> or components that used the <code>html5.tag-decorator</code> .
1.4. API Reference	39
<i>fromHTML</i> (html: [str, <code>HtmlAst</code>], appendTo: <code>Widget</code> = None, bindTo: <code>Widget</code> = None, debug: bool = False, **kwargs) → [<code>Widget</code>]	Parses the provided HTML code according to the objects defined in the <code>html5</code> -library.

Attributes

htmlExpressionEvaluator

document

__domParser

_body

_head

__tags

__reVarReplacer

flare.html5.core.**htmlExpressionEvaluator**

flare.html5.core.**document**

flare.html5.core.**domCreateAttribute**(tag, ns=None)

Creates a new HTML/SVG/... attribute.

Parameters **ns** – the namespace. Default: HTML. Possible values: HTML, SVG, XBL, XUL

flare.html5.core.**domCreateElement**(tag, ns=None)

Creates a new HTML/SVG/... tag.

Parameters **ns** – the namespace. Default: HTML. Possible values: HTML, SVG, XBL, XUL

flare.html5.core.**domCreateTextNode**(txt="")

flare.html5.core.**domGetElementById**(idTag)

flare.html5.core.**domElementFromPoint**(x, y)

flare.html5.core.**domGetElementsByTagName**(tag)

flare.html5.core.**__domParser**

flare.html5.core.**domConvertEncodedText**(txt)

Convert HTML-encoded text (containing HTML entities) into its decoded string representation.

The reason for this function is the handling of HTML entities, which is not properly supported by native JavaScript.

We use the browser's DOM parser to do this, according to <https://stackoverflow.com/questions/3700326/decode-amp-back-to-in-javascript>

Parameters **txt** – The encoded text.

Returns The decoded text.

class flare.html5.core.**TextNode**(txt=None, *args, **kwargs)

Bases: object

Represents a piece of text inside the DOM.

This is the *only* object not deriving from “Widget”, as it does not support any of its properties.

_setText(*self*, *txt*)

_getText(*self*)

__str__(*self*)

Return str(*self*).

onAttach(*self*)

onDetach(*self*)

_setDisabled(*self*, *disabled*)

_getDisabled(*self*)

children(*self*)

class flare.html5.core._WidgetClassWrapper(*targetWidget*)

Bases: list

Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

set(*self*, *value*)

_updateElem(*self*)

append(*self*, *p_object*)

Append object to the end of the list.

clear(*self*)

Remove all items from list.

remove(*self*, *value*)

Remove first occurrence of value.

Raises ValueError if the value is not present.

extend(*self*, *iterable*)

Extend list by appending elements from the iterable.

insert(*self*, *index*, *p_object*)

Insert object before index.

pop(*self*, *index=None*)

Remove and return item at index (default last).

Raises IndexError if list is empty or index is out of range.

class flare.html5.core._WidgetDataWrapper(*targetWidget*)

Bases: dict

dict() -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's

(key, value) pairs

dict(iterable) -> new dictionary initialized as if via: d = { } for k, v in iterable:

d[k] = v

dict(kwargs)** -> new dictionary initialized with the name=value pairs in the keyword argument list. For example: dict(one=1, two=2)

__setitem__(self, key, value)

Set self[key] to value.

update(self, E=None, **F)

D.update([E,]**F) -> None. Update D from dict/iterable E and F. If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

class flare.html5.core._WidgetStyleWrapper(targetWidget)

Bases: dict

dict() -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's

(key, value) pairs

dict(iterable) -> new dictionary initialized as if via: d = { } for k, v in iterable:

d[k] = v

dict(kwargs)** -> new dictionary initialized with the name=value pairs in the keyword argument list. For example: dict(one=1, two=2)

__setitem__(self, key, value)

Set self[key] to value.

update(self, E=None, **F)

D.update([E,]**F) -> None. Update D from dict/iterable E and F. If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

class flare.html5.core.Widget(*args, appendTo=None, style=None, **kwargs)

Bases: object

_namespace

_tagName

_leafTag = False

style = []

sinkEvent(self, *args)

unsinkEvent(self, *args)

addEventListener(self, event, callback)

Adds an event listener callback to an event on a Widget.

Parameters

- **event** – The event string, e.g. “click” or “mouseover”
- **callback** – The callback function to be called on the given event. This callback function can either accept no parameters, receive the pure Event-object from JavaScript as one parameter, or receive both the pure Event-object from JavaScript and the Widget-instance where the event was triggered on.

removeEventListener(*self, event, callback*)

Removes an event listener callback from a Widget.

The event listener must be previously added by `Widget.addEventListener()`.

Parameters

- **event** – The event string, e.g. “click” or “mouseover”
- **callback** – The callback function to be removed

disable(*self*)

Disables an element, in case it is not already disabled.

On disabled elements, events are not triggered anymore.

enable(*self*)

Enables an element, in case it is not already enabled.

_getTargetfuncName(*self, key, type*)

__getitem__(*self, key*)

__setitem__(*self, key, value*)

__str__(*self*)

Return `str(self)`.

__iter__(*self*)

_getData(*self*)

Custom data attributes are intended to store custom data private to the page or application, for which there are no more appropriate attributes or elements.

Parameters **name** –

Returns

_getTranslate(*self*)

Specifies whether an elements attribute values and contents of its children are to be translated when the page is localized, or whether to leave them unchanged.

Returns True | False

_setTranslate(*self, val*)

Specifies whether an elements attribute values and contents of its children are to be translated when the page is localized, or whether to leave them unchanged.

Parameters **val** – True | False

_getTitle(*self*)

Advisory information associated with the element.

Returns str

_setTitle(*self, val*)

Advisory information associated with the element.

Parameters **val** – str

_getTabIndex(*self*)

Specifies whether the element represents an element that is focusable (that is, an element which is part of the sequence of focusable elements in the document), and the relative order of the element in the sequence of focusable elements in the document.

Returns number

_setTabIndex(*self*, *val*)

Specifies whether the element represents an element that is focusable (that is, an element which is part of the sequence of focusable elements in the document), and the relative order of the element in the sequence of focusable elements in the document.

Parameters **val** – number

_getSpellcheck(*self*)

Specifies whether the element represents an element whose contents are subject to spell checking and grammar checking.

Returns True | False

_setSpellcheck(*self*, *val*)

Specifies whether the element represents an element whose contents are subject to spell checking and grammar checking.

Parameters **val** – True | False

_getLang(*self*)

Specifies the primary language for the contents of the element and for any of the elements attributes that contain text.

Returns language tag e.g. de|en|fr|es|it|ru|

_setLang(*self*, *val*)

Specifies the primary language for the contents of the element and for any of the elements attributes that contain text.

Parameters **val** – language tag

_getHidden(*self*)

Specifies that the element represents an element that is not yet, or is no longer, relevant.

Returns True | False

_setHidden(*self*, *val*)

Specifies that the element represents an element that is not yet, or is no longer, relevant.

Parameters **val** – True | False

_getDisabled(*self*)**_setDisabled(*self*, *disable*)****_getDropzone(*self*)**

Specifies what types of content can be dropped on the element, and instructs the UA about which actions to take with content when it is dropped on the element.

Returns “copy” | “move” | “link”

_setDropzone(*self*, *val*)

Specifies what types of content can be dropped on the element, and instructs the UA about which actions to take with content when it is dropped on the element.

Parameters **val** – “copy” | “move” | “link”

_getDraggable(*self*)

Specifies whether the element is draggable.

Returns True | False | “auto”

_setDraggable(*self*, *val*)

Specifies whether the element is draggable.

Parameters **val** – True | False | “auto”

_getDir(*self*)

Specifies the elements text directionality.

Returns ltr | rtl | auto

_setDir(*self*, *val*)

Specifies the elements text directionality.

Parameters **val** – ltr | rtl | auto

_getContextmenu(*self*)

The value of the id attribute on the menu with which to associate the element as a context menu.

Returns

_setContextmenu(*self*, *val*)

The value of the id attribute on the menu with which to associate the element as a context menu.

Parameters **val** –

_getContenteditable(*self*)

Specifies whether the contents of the element are editable.

Returns True | False

_setContenteditable(*self*, *val*)

Specifies whether the contents of the element are editable.

Parameters **val** – True | False

_getAccesskey(*self*)

A key label or list of key labels with which to associate the element; each key label represents a keyboard shortcut which UAs can use to activate the element or give focus to the element.

Parameters **self** –

Returns

_setAccesskey(*self*, *val*)

A key label or list of key labels with which to associate the element; each key label represents a keyboard shortcut which UAs can use to activate the element or give focus to the element.

Parameters

- **self** –
- **val** –

_getId(*self*)

Specifies a unique id for an element.

Parameters **self** –

Returns

_setId(*self*, *val*)

Specifies a unique id for an element.

Parameters

- **self** –
- **val** –

_getClass(*self*)

The class attribute specifies one or more classnames for an element.

Returns

_setClass(*self*, *value*)

The class attribute specifies one or more classnames for an element.

Parameters

- **self** –
- **value** –

@raise ValueError:

_getStyle(*self*)

The style attribute specifies an inline style for an element.

Parameters **self** –

Returns

_getRole(*self*)

Specifies a role for an element.

@param self: @return:

_setRole(*self*, *val*)

Specifies a role for an element.

@param self: @param val:

hide(*self*)

Hide element, if shown.

Returns

show(*self*)

Show element, if hidden.

Returns

isHidden(*self*)

Checks if a widget is hidden.

Returns True if hidden, False otherwise.

isVisible(*self*)

Checks if a widget is visible.

Returns True if visible, False otherwise.

onBind(*self*, *widget*, *name*)

Event function that is called on the widget when it is bound to another widget with a name.

This is only done by the HTML parser, a manual binding by the user is not triggered.

onAttach(*self*)

onDetach(*self*)

__collectChildren(*self*, **args*, ***kwargs*)

Internal function for collecting children from args.

This is used by `appendChild()`, `prependChild()`, `insertChild()` etc.

insertBefore(*self*, *insert*, *child*, ***kwargs*)

insertAfter(*self*, *insert*, *child*, ***kwargs*)

prependChild(*self*, **args*, ***kwargs*)

appendChild(*self*, **args*, ***kwargs*)

replaceChild(*self*, **args*, ***kwargs*)

removeChild(*self*, *child*)

removeAllChildren(*self*)

Removes all child widgets of the current widget.

isParentOf(*self*, *widget*)

Checks if an object is the parent of widget.

Parameters **widget** ([Widget](#)) – The widget to check for.

Returns True, if widget is a child of the object, else False.

isChildOf(*self*, *widget*)

Checks if an object is the child of widget.

Parameters **widget** ([Widget](#)) – The widget to check for.

Returns True, if object is a child of widget, else False.

hasClass(*self*, *className*)

Determine whether the current widget is assigned the given class.

Parameters **className** (*str*) – The class name to search for.

addClass(*self*, **args*)

Adds a class or a list of classes to the current widget.

If the widget already has the class, it is ignored.

Parameters **args** (*list of str* | *list of list of str*) – A list of class names. This can also be a list.

removeClass(*self*, **args*)

Removes a class or a list of classes from the current widget.

Parameters **args** (*list of str* | *list of list of str*) – A list of class names. This can also be a list.

toggleClass(*self*, *on*, *off*=None)

Toggles the class *on*.

If the widget contains a class *on*, it is toggled by *off*. *off* can either be a class name that is substituted, or nothing.

Parameters

- **on** (*str*) – Classname to test for. If *on* does not exist, but *off*, *off* is replaced by *on*.
- **off** (*str*) – Classname to replace if *on* existed.

Returns Returns True, if *on* was switched, else False.

Return type bool

onBlur(*self*, *event*)

onChange(*self*, *event*)

onContextMenu(*self*, *event*)

onFocus(*self*, *event*)

onFocusIn(*self*, *event*)

onFocusOut(*self*, *event*)

onFormChange(*self*, *event*)

onFormInput(*self*, *event*)

onInput(*self*, *event*)

onInvalid(*self*, *event*)

onReset(*self*, *event*)

onSelect(*self*, *event*)

onSubmit(*self*, *event*)

onKeyDown(*self*, *event*)

onKeyPress(*self*, *event*)

onKeyUp(*self*, *event*)

onClick(*self*, *event*, *wdg*=None)

onDbClick(*self*, *event*)

onDrag(*self*, *event*)

onDragEnd(*self*, *event*)

onDragEnter(*self*, *event*)

onDragLeave(*self*, *event*)

onDragOver(*self*, *event*)

onDragStart(*self, event*)

onDrop(*self, event*)

onMouseDown(*self, event*)

onMouseMove(*self, event*)

onMouseOut(*self, event*)

onMouseOver(*self, event*)

onMouseUp(*self, event*)

onMouseWheel(*self, event*)

onScroll(*self, event*)

onTouchStart(*self, event*)

onTouchEnd(*self, event*)

onTouchMove(*self, event*)

onTouchCancel(*self, event*)

focus(*self*)

blur(*self*)

parent(*self*)

children(*self, n=None*)

Access children of widget.

If *n* is omitted, it returns a list of all child-widgets; Else, it returns the *N*'th child, or None if its out of bounds.

Parameters *n* (*int*) – Optional offset of child widget to return.

Returns Returns all children or only the requested one.

Return type list | *Widget* | None

sortChildren(*self, key, reversed=False*)

Sorts our direct children. They are rearranged on DOM level.

Key must be a function accepting one widget as parameter and must return the key used to sort these widgets.

fromHTML(*self, html, appendTo=None, bindTo=None, replace=False, vars=None, **kwargs*)

Parses html and constructs its elements as part of self.

Parameters

- **html** – HTML code.
- **appendTo** – The entity where the HTML code is constructed below. This defaults to self in usual case.
- **bindTo** – The entity where the named objects are bound to. This defaults to self in usual case.
- **replace** – Clear entire content of appendTo before appending.

- **vars** – Deprecated; Same as kwargs.
- ****kwargs** – Additional variables provided as a dict for { {placeholders} } inside the HTML

Returns

```
class flare.html5.core._attrLabel
```

Bases: object

```
    _getLabel(self)
```

```
    _setLabel(self, val)
```

```
class flare.html5.core._attrCharset
```

Bases: object

```
    _getCharset(self)
```

```
    _setCharset(self, val)
```

```
class flare.html5.core._attrCite
```

Bases: object

```
    _getCite(self)
```

```
    _setCite(self, val)
```

```
class flare.html5.core._attrDatetime
```

Bases: object

```
    _getDatetime(self)
```

```
    _setDatetime(self, val)
```

```
class flare.html5.core._attrForm
```

Bases: object

```
    _getForm(self)
```

```
    _setForm(self, val)
```

```
class flare.html5.core._attrAlt
```

Bases: object

```
    _getAlt(self)
```

```
    _setAlt(self, val)
```

```
class flare.html5.core._attrAutofocus
```

Bases: object

```
    _getAutofocus(self)
```

```
    _setAutofocus(self, val)
```

```
class flare.html5.core._attrDisabled
```

Bases: object

```
class flare.html5.core._attrChecked
```

Bases: object

```
    _getChecked(self)
    _setChecked(self, val)
class flare.html5.core._attrIndeterminate
    Bases: object
    _getIndeterminate(self)
    _setIndeterminate(self, val)
class flare.html5.core._attrName
    Bases: object
    _getName(self)
    _setName(self, val)
class flare.html5.core._attrValue
    Bases: object
    _getValue(self)
    _setValue(self, val)
class flare.html5.core._attrAutocomplete
    Bases: object
    _getAutocomplete(self)
    _setAutocomplete(self, val)
class flare.html5.core._attrRequired
    Bases: object
    _getRequired(self)
    _setRequired(self, val)
class flare.html5.core._attrMultiple
    Bases: object
    _getMultiple(self)
    _setMultiple(self, val)
class flare.html5.core._attrSize
    Bases: object
    _getSize(self)
    _setSize(self, val)
class flare.html5.core._attrFor
    Bases: object
    _getFor(self)
    _setFor(self, val)
```

```
class flare.html5.core._attrInputs
```

```
    Bases: _attrRequired
```

```
    _getMaxlength(self)
```

```
    _setMaxlength(self, val)
```

```
    _getPlaceholder(self)
```

```
    _setPlaceholder(self, val)
```

```
    _getReadonly(self)
```

```
    _setReadonly(self, val)
```

```
class flare.html5.core._attrFormhead
```

```
    Bases: object
```

```
    _getFormaction(self)
```

```
    _setFormaction(self, val)
```

```
    _getFormenctype(self)
```

```
    _setFormenctype(self, val)
```

```
    _getFormmethod(self)
```

```
    _setFormmethod(self, val)
```

```
    _getFormtarget(self)
```

```
    _setFormtarget(self, val)
```

```
    _getFormnovalidate(self)
```

```
    _setFormnovalidate(self, val)
```

```
class flare.html5.core._attrHref
```

```
    Bases: object
```

```
    _getHref(self)
```

```
        Url of a Page.
```

```
        Parameters self –
```

```
    _setHref(self, val)
```

```
        Url of a Page.
```

```
        Parameters val – URL
```

```
    _getHreflang(self)
```

```
    _setHreflang(self, val)
```

```
class flare.html5.core._attrTarget
```

```
    Bases: object
```

```
    _getTarget(self)
```

```
    _setTarget(self, val)
```



```
class flare.html5.core._attrType
    Bases: object
    _getType(self)
    _setType(self, val)

class flare.html5.core._attrMedia
    Bases: _attrType
    _getMedia(self)
    _setMedia(self, val)

class flare.html5.core._attrDimensions
    Bases: object
    _getWidth(self)
    _setWidth(self, val)
    _getHeight(self)
    _setHeight(self, val)

class flare.html5.core._attrUsemap
    Bases: object
    _getUsemap(self)
    _setUsemap(self, val)

class flare.html5.core._attrMultimedia
    Bases: object
    _getAutoplay(self)
    _setAutoplay(self, val)
    _getPlaysinline(self)
    _setPlaysinline(self, val)
    _getControls(self)
    _setControls(self, val)
    _getLoop(self)
    _setLoop(self, val)
    _getMuted(self)
    _setMuted(self, val)
    _getPreload(self)
    _setPreload(self, val)
```

```
class flare.html5.core._attrRel
    Bases: object
    _getRel(self)
    _setRel(self, val)

class flare.html5.core._attrSrc
    Bases: object
    _getSrc(self)
    _setSrc(self, val)

class flare.html5.core.A(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrHref, _attrTarget, _attrMedia, _attrRel, _attrName
    _tagName = a
    _getDownload(self)
        The download attribute specifies the path to a download.
        Returns filename
    _setDownload(self, val)
        The download attribute specifies the path to a download.
        Parameters val – filename

class flare.html5.core.Area(*args, appendTo=None, style=None, **kwargs)
    Bases: A, _attrAlt
    _tagName = area
    _leafTag = True
    _getCoords(self)
    _setCoords(self, val)
    _getShape(self)
    _setShape(self, val)

class flare.html5.core.Audio(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrSrc, _attrMultimedia
    _tagName = audio

class flare.html5.core.Bdo(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = bdo

class flare.html5.core.Blockquote(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = blockquote
    _getBlockquote(self)
```

```

    _setBlockquote(self, val)

class flare.html5.core.BodyCls(*args, **kwargs)
    Bases: Widget
flare.html5.core._body
flare.html5.core.Body()

class flare.html5.core.Canvas(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDimensions
    _tagName = canvas

class flare.html5.core.Command(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrLabel, _attrType, _attrDisabled, _attrChecked
    _tagName = command
    _getIcon(self)
    _setIcon(self, val)
    _getRadiogroup(self)
    _setRadiogroup(self, val)

class flare.html5.core._Del(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrCite, _attrDatetime
    _tagName = _del

class flare.html5.core.Dialog(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = dialog
    _getOpen(self)
    _setOpen(self, val)

class flare.html5.core.Abbbr(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = abbr

class flare.html5.core.Address(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = address

class flare.html5.core.Article(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = article

class flare.html5.core.Aside(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = aside

```

```
class flare.html5.core.B(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = b

class flare.html5.core.Bdi(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = bdi

class flare.html5.core.Br(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = br
    _leafTag = True

class flare.html5.core.Caption(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = caption

class flare.html5.core.Cite(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = cite

class flare.html5.core.Code(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = code

class flare.html5.core.Datalist(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = datalist

class flare.html5.core.Dfn(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = dfn

class flare.html5.core.Div(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = div

class flare.html5.core.Em(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = em

class flare.html5.core.Embed(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrSrc, _attrType, _attrDimensions
    _tagName = embed
    _leafTag = True
```

```

class flare.html5.core.Figcaption(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = figcaption

class flare.html5.core.Figure(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = figure

class flare.html5.core.Footer(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = footer

class flare.html5.core.Header(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = header

class flare.html5.core.H1(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = h1

class flare.html5.core.H2(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = h2

class flare.html5.core.H3(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = h3

class flare.html5.core.H4(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = h4

class flare.html5.core.H5(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = h5

class flare.html5.core.H6(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = h6

class flare.html5.core.Hr(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = hr
    _leafTag = True

class flare.html5.core.I(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget

```

```
    _tagName = i

class flare.html5.core.Kdb(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = kdb

class flare.html5.core.Legend(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = legend

class flare.html5.core.Mark(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = mark

class flare.html5.core.Noscript(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = noscript

class flare.html5.core.P(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = p

class flare.html5.core.Rq(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = rq

class flare.html5.core.Rt(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = rt

class flare.html5.core.Ruby(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = ruby

class flare.html5.core.S(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = s

class flare.html5.core.Samp(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = samp

class flare.html5.core.Section(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = section

class flare.html5.core.Small(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
```

```

    _tagName = small

class flare.html5.core.Strong(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = strong

class flare.html5.core.Sub(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = sub

class flare.html5.core.Summary(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = summary

class flare.html5.core.Sup(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = sup

class flare.html5.core.U(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = u

class flare.html5.core.Var(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = var

class flare.html5.core.Wbr(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = wbr

class flare.html5.core.Button(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrType, _attrForm, _attrAutofocus, _attrName, _attrValue,
    _attrFormhead
    _tagName = button

class flare.html5.core.Fieldset(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrForm, _attrName
    _tagName = fieldset

class flare.html5.core.Form(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrName, _attrTarget, _attrAutocomplete
    _tagName = form

    _getNovalidate(self)

    _setNovalidate(self, val)

    _getAction(self)

```

```
_setAction(self, val)
_getMethod(self)
_setMethod(self, val)
_getEnctype(self)
_setEnctype(self, val)
_getAccept_attrCharset(self)
_setAccept_attrCharset(self, val)
```

```
class flare.html5.core.Input(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrType, _attrForm, _attrAlt, _attrAutofocus,
    _attrChecked, _attrIndeterminate, _attrName, _attrDimensions, _attrValue, _attrFormhead,
    _attrAutocomplete, _attrInputs, _attrMultiple, _attrSize, _attrSrc
    _tagName = input
    _leafTag = True
    _getAccept(self)
    _setAccept(self, val)
    _getList(self)
    _setList(self, val)
    _getMax(self)
    _setMax(self, val)
    _getMin(self)
    _setMin(self, val)
    _getPattern(self)
    _setPattern(self, val)
    _getStep(self)
    _setStep(self, val)
```

```
class flare.html5.core.Label(*args, forElem=None, **kwargs)
    Bases: Widget, _attrForm, _attrFor
    _tagName = label
    autoIdCounter = 0
```

```
class flare.html5.core.Optgroup(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrLabel
    _tagName = optgroup
```



```

class flare.html5.core.Option(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrLabel, _attrValue
    _tagName = option
    _getSelected(self)
    _setSelected(self, val)

class flare.html5.core.Output(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrForm, _attrName, _attrFor
    _tagName = output

class flare.html5.core.Select(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrForm, _attrAutofocus, _attrName, _attrRequired,
    _attrMultiple, _attrSize
    _tagName = select
    _getSelectedIndex(self)
    _getOptions(self)

class flare.html5.core.Textarea(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrForm, _attrAutofocus, _attrName, _attrInputs, _attrValue
    _tagName = textarea
    _getCols(self)
    _setCols(self, val)
    _getRows(self)
    _setRows(self, val)
    _getWrap(self)
    _setWrap(self, val)

class flare.html5.core.HeadCls(*args, **kwargs)
    Bases: Widget
flare.html5.core._head
flare.html5.core.Head()

class flare.html5.core.Iframe(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrSrc, _attrName, _attrDimensions
    _tagName = iframe
    _getSandbox(self)
    _setSandbox(self, val)
    _getSrcdoc(self)
    _setSrcdoc(self, val)

```

```
    _getSeamless(self)
    _setSeamless(self, val)
class flare.html5.core.Img(src=None, *args, **kwargs)
    Bases: Widget, _attrSrc, _attrDimensions, _attrUsemap, _attrAlt
    _tagName = img
    _leafTag = True
    _getCrossorigin(self)
    _setCrossorigin(self, val)
    _getIsmap(self)
    _setIsmap(self, val)
class flare.html5.core.Ins(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrCite, _attrDatetime
    _tagName = ins
class flare.html5.core.Keygen(*args, appendTo=None, style=None, **kwargs)
    Bases: Form, _attrAutofocus, _attrDisabled
    _tagName = keygen
    _getChallenge(self)
    _setChallenge(self, val)
    _getKeytype(self)
    _setKeytype(self, val)
class flare.html5.core.Link(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrHref, _attrMedia, _attrRel
    _tagName = link
    _leafTag = True
    _getSizes(self)
    _setSizes(self, val)
class flare.html5.core.Ul(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = ul
class flare.html5.core.Ol(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = ol
class flare.html5.core.Li(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
```

```

    _tagName = li
class flare.html5.core.Dl(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = dl
class flare.html5.core.Dt(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = dt
class flare.html5.core.Dd(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = dd
class flare.html5.core.Map(*args, forElem=None, **kwargs)
    Bases: Label, _attrType
    _tagName = map
class flare.html5.core.Menu(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = menu
class flare.html5.core.Meta(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrName, _attrCharset
    _tagName = meta
    _leafTag = True
    _getContent(self)
    _setContent(self, val)
class flare.html5.core.Meter(*args, appendTo=None, style=None, **kwargs)
    Bases: Form, _attrValue
    _tagName = meter
    _getHigh(self)
    _setHigh(self, val)
    _getLow(self)
    _setLow(self, val)
    _getMax(self)
    _setMax(self, val)
    _getMin(self)
    _setMin(self, val)
    _getOptimum(self)

```

```
    _setOptimum(self, val)

class flare.html5.core.Nav(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = nav

class flare.html5.core.Object(*args, appendTo=None, style=None, **kwargs)
    Bases: Form, _attrType, _attrName, _attrDimensions, _attrUsemap
    _tagName = object

class flare.html5.core.Param(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrName, _attrValue
    _tagName = param
    _leafTag = True

class flare.html5.core.Progress(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrValue
    _tagName = progress
    _getMax(self)
    _setMax(self, val)

class flare.html5.core.Q(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrCite
    _tagName = q

class flare.html5.core.Script(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrSrc, _attrCharset
    _tagName = script
    _getAsync(self)
    _setAsync(self, val)
    _getDefer(self)
    _setDefer(self, val)

class flare.html5.core.Source(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrMedia, _attrSrc
    _tagName = source
    _leafTag = True

class flare.html5.core.Span(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = span

class flare.html5.core.Details(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
```

```

    _tagName = details
    _getOpen(self)
    _setOpen(self, val)
class flare.html5.core.Summary(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = summary
class flare.html5.core.Style(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrMedia
    _tagName = style
    _getScoped(self)
    _setScoped(self, val)
class flare.html5.core.Tr(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = tr
    _getRowspan(self)
    _setRowspan(self, span)
class flare.html5.core.Td(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = td
    _getColspan(self)
    _setColspan(self, span)
    _getRowspan(self)
    _setRowspan(self, span)
class flare.html5.core.Th(*args, appendTo=None, style=None, **kwargs)
    Bases: Td
    _tagName = th
class flare.html5.core.Thead(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = thead
class flare.html5.core.Tbody(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = tbody
class flare.html5.core.ColWrapper(parentElem, *args, **kwargs)
    Bases: object

```

```
    __getitem__(self, item)

    __setitem__(self, key, value)

class flare.html5.core.RowWrapper(parentElem, *args, **kwargs)
    Bases: object
    __getitem__(self, item)

class flare.html5.core.Table(*args, **kwargs)
    Bases: Widget
    _tagName = table
    prepareRow(self, row)
    prepareCol(self, row, col)
    prepareGrid(self, rows, cols)
    clear(self)
    _getCell(self)
    getRowCount(self)

class flare.html5.core.Time(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDatetime
    _tagName = time

class flare.html5.core.Track(*args, forElem=None, **kwargs)
    Bases: Label, _attrSrc
    _tagName = track
    _leafTag = True
    _getKind(self)
    _setKind(self, val)
    _getSrclang(self)
    _setSrclang(self, val)
    _getDefault(self)
    _setDefault(self, val)

class flare.html5.core.Video(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrSrc, _attrDimensions, _attrMultimedia
    _tagName = video
    _getPoster(self)
    _setPoster(self, val)
```

class flare.html5.core.Template(*args, appendTo=None, style=None, **kwargs)

Bases: *Widget*

_tagName = **template**

flare.html5.core.unescape(val, maxLength=0)

Unquotes several HTML-quoted characters in a string.

Parameters

- **val** (*str*) – The value to be unescaped.
- **maxLength** (*int*) – Cut-off after maxLength characters. A value of 0 means “unlimited”. (default)

Returns The unquoted string.

Return type *str*

flare.html5.core.doesEventHitWidgetOrParents(event, widget)

Test if event ‘event’ hits widget ‘widget’ (or *any* of its parents).

flare.html5.core.doesEventHitWidgetOrChildren(event, widget)

Test if event ‘event’ hits widget ‘widget’ (or *any* of its children).

flare.html5.core.textToHtml(node, text)

Generates html nodes from text by splitting text into content and into line breaks html5.Br.

Parameters

- **node** – The node where the nodes are appended to.
- **text** – The text to be inserted.

flare.html5.core.parseInt(s, ret=0)

Parses a value as int.

flare.html5.core.parseFloat(s, ret=0.0)

Parses a value as float.

flare.html5.core.getKey(event)

Returns the Key Identifier of the given event.

Available Codes: <https://www.w3.org/TR/2006/WD-DOM-Level-3-Events-20060413/keyset.html#KeySet-Set>

flare.html5.core.isArrowLeft(event)

flare.html5.core.isArrowUp(event)

flare.html5.core.isArrowRight(event)

flare.html5.core.isArrowDown(event)

flare.html5.core.isEscape(event)

flare.html5.core.isReturn(event)

flare.html5.core.isControl(event)

flare.html5.core.isShift(event)

`flare.html5.core.isMeta(event)`

`flare.html5.core.__tags`

`flare.html5.core.__reVarReplacer`

`flare.html5.core.registerTag(tagName, widgetClass, override=True)`

`flare.html5.core.tag(arg)`

Decorator to register a sub-class of `html5.Widget` either under its class-name or an associated tag-name.

```
"""python # register class Foo as <foo>-Tag @html5.tag class Foo(html5.Div):
    pass
# register class Bar as <baz>-Tag @html5.tag("baz") class Bar(html5.Div):
    pass
"""
```

`flare.html5.core._buildTags(debug=False)`

Generates a dictionary of all to the `html5-library` known tags and their associated objects and attributes.

class `flare.html5.core.HtmlAst`

Bases: `list`

Abstract syntax tree element used by `parseHTML()`.

`flare.html5.core.parseHTML(html: str, debug: bool = False) → HtmlAst`

Parses the provided HTML-code according to the tags registered by `html5.registerTag()` or components that used the `html5.tag`-decorator.

`flare.html5.core.fromHTML(html: [str, HtmlAst], appendTo: Widget = None, bindTo: Widget = None, debug: bool = False, **kwargs) → [Widget]`

Parses the provided HTML code according to the objects defined in the `html5-library`.

`html` can also be pre-compiled by `parseHTML()` so that it executes faster.

Constructs all objects as DOM nodes. The first level is chained into `appendTo`. If no `appendTo` is provided, `appendTo` will be set to `html5.Body()`.

If `bindTo` is provided, objects are bound to this widget.

```
"""python from vi import html5
div = html5.Div() html5.parse.fromHTML("""
    <div>Yeah! <a href="hello world" [name]="myLink" class="trullman bernd" disabled> hah
        ala malla" bababtschga" st <em>ah</em>ralla <i>malla tralla</i> da </a>lala
    </div>""", div)
div.myLink.appendChild("appended!") """
```


flare.html5.svg

SVG abstraction layer integrations for HTML5.

Module Contents

Classes

_attrSvgViewBox

_attrSvgDimensions

_attrSvgPoints

_attrSvgTransform

_attrSvgXlink

_attrSvgStyles

SvgWidget

Svg

SvgCircle

SvgEllipse

SvgG

SvgImage

SvgLine

SvgPath

SvgPolygon

SvgPolyline

SvgRect

SvgText

```
class flare.html5.svg._attrSvgViewBox
```

```
    Bases: object
```

```
    _getViewbox(self)
```

```
_setViewbox(self, val)
_getPreserveaspectratio(self)
_setPreserveaspectratio(self, val)
```

```
class flare.html5.svg._attrSvgDimensions
```

```
    Bases: object
```

```
    _getWidth(self)
    _setWidth(self, val)
    _getHeight(self)
    _setHeight(self, val)
    _getX(self)
    _setX(self, val)
    _getY(self)
    _setY(self, val)
    _getR(self)
    _setR(self, val)
    _getRx(self)
    _setRx(self, val)
    _getRy(self)
    _setRy(self, val)
    _getCx(self)
    _setCx(self, val)
    _getCy(self)
    _setCy(self, val)
```

```
class flare.html5.svg._attrSvgPoints
```

```
    Bases: object
```

```
    _getPoints(self)
    _setPoints(self, val)
    _getX1(self)
    _setX1(self, val)
    _getY1(self)
    _setY1(self, val)
    _getX2(self)
```

```

    _setX2(self, val)
    _getY2(self)
    _setY2(self, val)
class flare.html5.svg._attrSvgTransform
    Bases: object
    _getTransform(self)
    _setTransform(self, val)
class flare.html5.svg._attrSvgXlink
    Bases: object
    _getXlinkhref(self)
    _setXlinkhref(self, val)
class flare.html5.svg._attrSvgStyles
    Bases: object
    _getFill(self)
    _setFill(self, val)
    _getStroke(self)
    _setStroke(self, val)
class flare.html5.svg.SvgWidget(*args, appendTo=None, style=None, **kwargs)
    Bases: flare.html5.core.Widget
    _namespace = SVG
class flare.html5.svg.Svg(*args, appendTo=None, style=None, **kwargs)
    Bases: SvgWidget, \_attrSvgViewBox, \_attrSvgDimensions, \_attrSvgTransform
    _tagName = svg
    _getVersion(self)
    _setVersion(self, val)
    _getXmlns(self)
    _setXmlns(self, val)
class flare.html5.svg.SvgCircle(*args, appendTo=None, style=None, **kwargs)
    Bases: SvgWidget, \_attrSvgTransform, \_attrSvgDimensions
    _tagName = circle
class flare.html5.svg.SvgEllipse(*args, appendTo=None, style=None, **kwargs)
    Bases: SvgWidget, \_attrSvgTransform, \_attrSvgDimensions
    _tagName = ellipse

```

```
class flare.html5.svg.SvgG(*args, appendTo=None, style=None, **kwargs)
    Bases: SvgWidget, _attrSvgTransform, _attrSvgStyles
    _tagName = g
    _getSvgTransform(self)
    _setSvgTransform(self, val)

class flare.html5.svg.SvgImage(*args, appendTo=None, style=None, **kwargs)
    Bases: SvgWidget, _attrSvgViewBox, _attrSvgDimensions, _attrSvgTransform, _attrSvgXlink
    _tagName = image

class flare.html5.svg.SvgLine(*args, appendTo=None, style=None, **kwargs)
    Bases: SvgWidget, _attrSvgTransform, _attrSvgPoints
    _tagName = line

class flare.html5.svg.SvgPath(*args, appendTo=None, style=None, **kwargs)
    Bases: SvgWidget, _attrSvgTransform
    _tagName = path
    _getD(self)
    _setD(self, val)
    _getPathLength(self)
    _setPathLength(self, val)

class flare.html5.svg.SvgPolygon(*args, appendTo=None, style=None, **kwargs)
    Bases: SvgWidget, _attrSvgTransform, _attrSvgPoints
    _tagName = polygon

class flare.html5.svg.SvgPolyline(*args, appendTo=None, style=None, **kwargs)
    Bases: SvgWidget, _attrSvgTransform, _attrSvgPoints
    _tagName = polyline

class flare.html5.svg.SvgRect(*args, appendTo=None, style=None, **kwargs)
    Bases: SvgWidget, _attrSvgDimensions, _attrSvgTransform, _attrSvgStyles
    _tagName = rect

class flare.html5.svg.SvgText(*args, appendTo=None, style=None, **kwargs)
    Bases: SvgWidget, _attrSvgDimensions, _attrSvgTransform, _attrSvgStyles
    _tagName = text
```

Package Contents

Classes

<i>TextNode</i>	Represents a piece of text inside the DOM.
<i>_WidgetClassWrapper</i>	Built-in mutable sequence.
<i>_WidgetDataWrapper</i>	dict() -> new empty dictionary
<i>_WidgetStyleWrapper</i>	dict() -> new empty dictionary
<i>Widget</i>	
<i>_attrLabel</i>	
<i>_attrCharset</i>	
<i>_attrCite</i>	
<i>_attrDatetime</i>	
<i>_attrForm</i>	
<i>_attrAlt</i>	
<i>_attrAutofocus</i>	
<i>_attrDisabled</i>	
<i>_attrChecked</i>	
<i>_attrIndeterminate</i>	
<i>_attrName</i>	
<i>_attrValue</i>	
<i>_attrAutocomplete</i>	
<i>_attrRequired</i>	
<i>_attrMultiple</i>	
<i>_attrSize</i>	
<i>_attrFor</i>	
<i>_attrInputs</i>	
<i>_attrFormhead</i>	
<i>_attrHref</i>	

continues on next page

Table 2 – continued from previous page

<i>_attrTarget</i>
<i>_attrType</i>
<i>_attrMedia</i>
<i>_attrDimensions</i>
<i>_attrUsemap</i>
<i>_attrMultimedia</i>
<i>_attrRel</i>
<i>_attrSrc</i>
<i>A</i>
<i>Area</i>
<i>Audio</i>
<i>Bdo</i>
<i>Blockquote</i>
<i>BodyCls</i>
<i>Canvas</i>
<i>Command</i>
<i>_Del</i>
<i>Dialog</i>
<i>Abbr</i>
<i>Address</i>
<i>Article</i>
<i>Aside</i>
<i>B</i>
<i>Bdi</i>
<i>Br</i>

continues on next page

Table 2 – continued from previous page

<i>Caption</i>
<i>Cite</i>
<i>Code</i>
<i>Datalist</i>
<i>Dfn</i>
<i>Div</i>
<i>Em</i>
<i>Embed</i>
<i>Figcaption</i>
<i>Figure</i>
<i>Footer</i>
<i>Header</i>
<i>H1</i>
<i>H2</i>
<i>H3</i>
<i>H4</i>
<i>H5</i>
<i>H6</i>
<i>Hr</i>
<i>I</i>
<i>Kdb</i>
<i>Legend</i>
<i>Mark</i>
<i>Noscript</i>
<i>P</i>

continues on next page

Table 2 – continued from previous page

<i>Rq</i>
<i>Rt</i>
<i>Ruby</i>
<i>S</i>
<i>Samp</i>
<i>Section</i>
<i>Small</i>
<i>Strong</i>
<i>Sub</i>
<i>Summery</i>
<i>Sup</i>
<i>U</i>
<i>Var</i>
<i>Wbr</i>
<i>Button</i>
<i>Fieldset</i>
<i>Form</i>
<i>Input</i>
<i>Label</i>
<i>Optgroup</i>
<i>Option</i>
<i>Output</i>
<i>Select</i>
<i>Textarea</i>
<i>HeadCls</i>

continues on next page

Table 2 – continued from previous page

<i>Iframe</i>
<i>Img</i>
<i>Ins</i>
<i>Keygen</i>
<i>Link</i>
<i>Ul</i>
<i>Ol</i>
<i>Li</i>
<i>Dl</i>
<i>Dt</i>
<i>Dd</i>
<i>Map</i>
<i>Menu</i>
<i>Meta</i>
<i>Meter</i>
<i>Nav</i>
<i>Object</i>
<i>Param</i>
<i>Progress</i>
<i>Q</i>
<i>Script</i>
<i>Source</i>
<i>Span</i>
<i>Details</i>
<i>Summary</i>

continues on next page

Table 2 – continued from previous page

<i>Style</i>	
<i>Tr</i>	
<i>Td</i>	
<i>Th</i>	
<i>Thead</i>	
<i>Tbody</i>	
<i>ColWrapper</i>	
<i>RowWrapper</i>	
<i>Table</i>	
<i>Time</i>	
<i>Track</i>	
<i>Video</i>	
<i>Template</i>	
<i>HtmlAst</i>	Abstract syntax tree element used by parseHTML().

Functions

<i>domCreateAttribute</i> (tag, ns=None)	Creates a new HTML/SVG/... attribute.
<i>domCreateElement</i> (tag, ns=None)	Creates a new HTML/SVG/... tag.
<i>domCreateTextNode</i> (txt="")	
<i>domGetElementById</i> (idTag)	
<i>domElementFromPoint</i> (x, y)	
<i>domGetElementsByTagName</i> (tag)	
<i>domConvertEncodedText</i> (txt)	Convert HTML-encoded text (containing HTML entities) into its decoded string representation.
<i>Body</i> ()	
<i>Head</i> ()	
<i>unescape</i> (val, maxLength=0)	Unquotes several HTML-quoted characters in a string.
<i>doesEventHitWidgetOrParents</i> (event, widget)	Test if event 'event' hits widget 'widget' (or <i>any</i> of its parents).
<i>doesEventHitWidgetOrChildren</i> (event, widget)	Test if event 'event' hits widget 'widget' (or <i>any</i> of its children).
<i>textToHtml</i> (node, text)	Generates html nodes from text by splitting text into content and into line breaks html5.Br.
<i>parseInt</i> (s, ret=0)	Parses a value as int.
<i>parseFloat</i> (s, ret=0.0)	Parses a value as float.
<i>getKey</i> (event)	Returns the Key Identifier of the given event.
<i>isArrowLeft</i> (event)	
<i>isArrowUp</i> (event)	
<i>isArrowRight</i> (event)	
<i>isArrowDown</i> (event)	
<i>isEscape</i> (event)	
<i>isReturn</i> (event)	
<i>isControl</i> (event)	
<i>isShift</i> (event)	
<i>isMeta</i> (event)	
<i>registerTag</i> (tagName, widgetClass, override=True)	
<i>tag</i> (arg)	Decorator to register a sub-class of html5.Widget either under its class-name or an associated tag-name.
<i>_buildTags</i> (debug=False)	Generates a dictionary of all to the html5-library known tags and their associated objects and attributes.
<i>parseHTML</i> (html: str, debug: bool = False) → HtmlAst	Parses the provided HTML-code according to the tags registered by <code>html5.registerTag()</code> or components that used the <code>html5.tag-decorator</code> .
1.4. API Reference	79
<i>fromHTML</i> (html: [str, HtmlAst], appendTo: Widget = None, bindTo: Widget = None, debug: bool = False, **kwargs) → [Widget]	Parses the provided HTML code according to the objects defined in the html5-library.

Attributes

htmlExpressionEvaluator

document

__domParser

_body

_head

__tags

__reVarReplacer

flare.html5.htmlExpressionEvaluator**flare.html5.document****flare.html5.domCreateAttribute**(*tag*, *ns=None*)

Creates a new HTML/SVG/... attribute.

Parameters **ns** – the namespace. Default: HTML. Possible values: HTML, SVG, XBL, XUL**flare.html5.domCreateElement**(*tag*, *ns=None*)

Creates a new HTML/SVG/... tag.

Parameters **ns** – the namespace. Default: HTML. Possible values: HTML, SVG, XBL, XUL**flare.html5.domCreateTextNode**(*txt=""*)**flare.html5.domGetElementById**(*idTag*)**flare.html5.domElementFromPoint**(*x*, *y*)**flare.html5.domGetElementsByTagName**(*tag*)**flare.html5.__domParser****flare.html5.domConvertEncodedText**(*txt*)

Convert HTML-encoded text (containing HTML entities) into its decoded string representation.

The reason for this function is the handling of HTML entities, which is not properly supported by native JavaScript.

We use the browser's DOM parser to do this, according to <https://stackoverflow.com/questions/3700326/decode-amp-back-to-in-javascript>**Parameters** **txt** – The encoded text.**Returns** The decoded text.**class** flare.html5.**TextNode**(*txt=None*, **args*, ***kwargs*)

Bases: object

Represents a piece of text inside the DOM.

This is the *only* object not deriving from “Widget”, as it does not support any of its properties.

_setText(*self*, *txt*)

_getText(*self*)

__str__(*self*)

Return str(*self*).

onAttach(*self*)

onDetach(*self*)

_setDisabled(*self*, *disabled*)

_getDisabled(*self*)

children(*self*)

class flare.html5._WidgetClassWrapper(*targetWidget*)

Bases: list

Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

set(*self*, *value*)

_updateElem(*self*)

append(*self*, *p_object*)

Append object to the end of the list.

clear(*self*)

Remove all items from list.

remove(*self*, *value*)

Remove first occurrence of value.

Raises ValueError if the value is not present.

extend(*self*, *iterable*)

Extend list by appending elements from the iterable.

insert(*self*, *index*, *p_object*)

Insert object before index.

pop(*self*, *index=None*)

Remove and return item at index (default last).

Raises IndexError if list is empty or index is out of range.

class flare.html5._WidgetDataWrapper(*targetWidget*)

Bases: dict

dict() -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's

(key, value) pairs

dict(iterable) -> new dictionary initialized as if via: d = { } for k, v in iterable:

d[k] = v

dict(kwargs)** -> new dictionary initialized with the name=value pairs in the keyword argument list. For example: dict(one=1, two=2)

__setitem__(self, key, value)

Set self[key] to value.

update(self, E=None, **F)

D.update([E,]**F) -> None. Update D from dict/iterable E and F. If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

class flare.html5._WidgetStyleWrapper(targetWidget)

Bases: dict

dict() -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's

(key, value) pairs

dict(iterable) -> new dictionary initialized as if via: d = { } for k, v in iterable:

d[k] = v

dict(kwargs)** -> new dictionary initialized with the name=value pairs in the keyword argument list. For example: dict(one=1, two=2)

__setitem__(self, key, value)

Set self[key] to value.

update(self, E=None, **F)

D.update([E,]**F) -> None. Update D from dict/iterable E and F. If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

class flare.html5.Widget(*args, appendTo=None, style=None, **kwargs)

Bases: object

_namespace

_tagName

_leafTag = False

style = []

sinkEvent(self, *args)

unsinkEvent(self, *args)

addEventListener(self, event, callback)

Adds an event listener callback to an event on a Widget.

Parameters

- **event** – The event string, e.g. “click” or “mouseover”
- **callback** – The callback function to be called on the given event. This callback function can either accept no parameters, receive the pure Event-object from JavaScript as one parameter, or receive both the pure Event-object from JavaScript and the Widget-instance where the event was triggered on.

removeEventListener(*self, event, callback*)

Removes an event listener callback from a Widget.

The event listener must be previously added by `Widget.addEventListener()`.

Parameters

- **event** – The event string, e.g. “click” or “mouseover”
- **callback** – The callback function to be removed

disable(*self*)

Disables an element, in case it is not already disabled.

On disabled elements, events are not triggered anymore.

enable(*self*)

Enables an element, in case it is not already enabled.

_getTargetfuncName(*self, key, type*)

__getitem__(*self, key*)

__setitem__(*self, key, value*)

__str__(*self*)

Return `str(self)`.

__iter__(*self*)

_getData(*self*)

Custom data attributes are intended to store custom data private to the page or application, for which there are no more appropriate attributes or elements.

Parameters **name** –

Returns

_getTranslate(*self*)

Specifies whether an elements attribute values and contents of its children are to be translated when the page is localized, or whether to leave them unchanged.

Returns True | False

_setTranslate(*self, val*)

Specifies whether an elements attribute values and contents of its children are to be translated when the page is localized, or whether to leave them unchanged.

Parameters **val** – True | False

_getTitle(*self*)

Advisory information associated with the element.

Returns str

_setTitle(*self, val*)

Advisory information associated with the element.

Parameters **val** – str

_getTabIndex(*self*)

Specifies whether the element represents an element that is focusable (that is, an element which is part of the sequence of focusable elements in the document), and the relative order of the element in the sequence of focusable elements in the document.

Returns number

_setTabIndex(*self*, *val*)

Specifies whether the element represents an element that is focusable (that is, an element which is part of the sequence of focusable elements in the document), and the relative order of the element in the sequence of focusable elements in the document.

Parameters **val** – number

_getSpellcheck(*self*)

Specifies whether the element represents an element whose contents are subject to spell checking and grammar checking.

Returns True | False

_setSpellcheck(*self*, *val*)

Specifies whether the element represents an element whose contents are subject to spell checking and grammar checking.

Parameters **val** – True | False

_getLang(*self*)

Specifies the primary language for the contents of the element and for any of the elements attributes that contain text.

Returns language tag e.g. de|en|fr|es|it|ru|

_setLang(*self*, *val*)

Specifies the primary language for the contents of the element and for any of the elements attributes that contain text.

Parameters **val** – language tag

_getHidden(*self*)

Specifies that the element represents an element that is not yet, or is no longer, relevant.

Returns True | False

_setHidden(*self*, *val*)

Specifies that the element represents an element that is not yet, or is no longer, relevant.

Parameters **val** – True | False

_getDisabled(*self*)**_setDisabled(*self*, *disable*)****_getDropzone(*self*)**

Specifies what types of content can be dropped on the element, and instructs the UA about which actions to take with content when it is dropped on the element.

Returns “copy” | “move” | “link”

_setDropzone(*self*, *val*)

Specifies what types of content can be dropped on the element, and instructs the UA about which actions to take with content when it is dropped on the element.

Parameters **val** – “copy” | “move” | “link”

_getDraggable(*self*)

Specifies whether the element is draggable.

Returns True | False | “auto”

_setDraggable(*self*, *val*)

Specifies whether the element is draggable.

Parameters **val** – True | False | “auto”

_getDir(*self*)

Specifies the elements text directionality.

Returns ltr | rtl | auto

_setDir(*self*, *val*)

Specifies the elements text directionality.

Parameters **val** – ltr | rtl | auto

_getContextmenu(*self*)

The value of the id attribute on the menu with which to associate the element as a context menu.

Returns

_setContextmenu(*self*, *val*)

The value of the id attribute on the menu with which to associate the element as a context menu.

Parameters **val** –

_getContenteditable(*self*)

Specifies whether the contents of the element are editable.

Returns True | False

_setContenteditable(*self*, *val*)

Specifies whether the contents of the element are editable.

Parameters **val** – True | False

_getAccesskey(*self*)

A key label or list of key labels with which to associate the element; each key label represents a keyboard shortcut which UAs can use to activate the element or give focus to the element.

Parameters **self** –

Returns

_setAccesskey(*self*, *val*)

A key label or list of key labels with which to associate the element; each key label represents a keyboard shortcut which UAs can use to activate the element or give focus to the element.

Parameters

- **self** –
- **val** –

_getId(*self*)

Specifies a unique id for an element.

Parameters **self** –

Returns

_setId(*self*, *val*)

Specifies a unique id for an element.

Parameters

- **self** –
- **val** –

_getClass(*self*)

The class attribute specifies one or more classnames for an element.

Returns

_setClass(*self*, *value*)

The class attribute specifies one or more classnames for an element.

Parameters

- **self** –
- **value** –

@raise ValueError:

_getStyle(*self*)

The style attribute specifies an inline style for an element.

Parameters **self** –

Returns

_getRole(*self*)

Specifies a role for an element.

@param self: @return:

_setRole(*self*, *val*)

Specifies a role for an element.

@param self: @param val:

hide(*self*)

Hide element, if shown.

Returns

show(*self*)

Show element, if hidden.

Returns

isHidden(*self*)

Checks if a widget is hidden.

Returns True if hidden, False otherwise.

isVisible(*self*)

Checks if a widget is visible.

Returns True if visible, False otherwise.

onBind(*self*, *widget*, *name*)

Event function that is called on the widget when it is bound to another widget with a name.

This is only done by the HTML parser, a manual binding by the user is not triggered.

onAttach(*self*)

onDetach(*self*)

__collectChildren(*self*, **args*, ***kwargs*)

Internal function for collecting children from args.

This is used by `appendChild()`, `prependChild()`, `insertChild()` etc.

insertBefore(*self*, *insert*, *child*, ***kwargs*)

insertAfter(*self*, *insert*, *child*, ***kwargs*)

prependChild(*self*, **args*, ***kwargs*)

appendChild(*self*, **args*, ***kwargs*)

replaceChild(*self*, **args*, ***kwargs*)

removeChild(*self*, *child*)

removeAllChildren(*self*)

Removes all child widgets of the current widget.

isParentOf(*self*, *widget*)

Checks if an object is the parent of widget.

Parameters **widget** ([Widget](#)) – The widget to check for.

Returns True, if widget is a child of the object, else False.

isChildOf(*self*, *widget*)

Checks if an object is the child of widget.

Parameters **widget** ([Widget](#)) – The widget to check for.

Returns True, if object is a child of widget, else False.

hasClass(*self*, *className*)

Determine whether the current widget is assigned the given class.

Parameters **className** (*str*) – The class name to search for.

addClass(*self*, **args*)

Adds a class or a list of classes to the current widget.

If the widget already has the class, it is ignored.

Parameters **args** (*list of str* | *list of list of str*) – A list of class names. This can also be a list.

removeClass(*self*, **args*)

Removes a class or a list of classes from the current widget.

Parameters **args** (*list of str* | *list of list of str*) – A list of class names. This can also be a list.

toggleClass(*self*, *on*, *off*=None)

Toggles the class *on*.

If the widget contains a class *on*, it is toggled by *off*. *off* can either be a class name that is substituted, or nothing.

Parameters

- **on** (*str*) – Classname to test for. If *on* does not exist, but *off*, *off* is replaced by *on*.
- **off** (*str*) – Classname to replace if *on* existed.

Returns Returns True, if *on* was switched, else False.

Return type bool

onBlur(*self*, *event*)

onChange(*self*, *event*)

onContextMenu(*self*, *event*)

onFocus(*self*, *event*)

onFocusIn(*self*, *event*)

onFocusOut(*self*, *event*)

onFormChange(*self*, *event*)

onFormInput(*self*, *event*)

onInput(*self*, *event*)

onInvalid(*self*, *event*)

onReset(*self*, *event*)

onSelect(*self*, *event*)

onSubmit(*self*, *event*)

onKeyDown(*self*, *event*)

onKeyPress(*self*, *event*)

onKeyUp(*self*, *event*)

onClick(*self*, *event*, *wdg*=None)

onDbClick(*self*, *event*)

onDrag(*self*, *event*)

onDragEnd(*self*, *event*)

onDragEnter(*self*, *event*)

onDragLeave(*self*, *event*)

onDragOver(*self*, *event*)

onDragStart(*self, event*)

onDrop(*self, event*)

onMouseDown(*self, event*)

onMouseMove(*self, event*)

onMouseOut(*self, event*)

onMouseOver(*self, event*)

onMouseUp(*self, event*)

onMouseWheel(*self, event*)

onScroll(*self, event*)

onTouchStart(*self, event*)

onTouchEnd(*self, event*)

onTouchMove(*self, event*)

onTouchCancel(*self, event*)

focus(*self*)

blur(*self*)

parent(*self*)

children(*self, n=None*)

Access children of widget.

If *n* is omitted, it returns a list of all child-widgets; Else, it returns the *N*'th child, or None if its out of bounds.

Parameters *n* (*int*) – Optional offset of child widget to return.

Returns Returns all children or only the requested one.

Return type list | *Widget* | None

sortChildren(*self, key, reversed=False*)

Sorts our direct children. They are rearranged on DOM level.

Key must be a function accepting one widget as parameter and must return the key used to sort these widgets.

fromHTML(*self, html, appendTo=None, bindTo=None, replace=False, vars=None, **kwargs*)

Parses html and constructs its elements as part of self.

Parameters

- **html** – HTML code.
- **appendTo** – The entity where the HTML code is constructed below. This defaults to self in usual case.
- **bindTo** – The entity where the named objects are bound to. This defaults to self in usual case.
- **replace** – Clear entire content of appendTo before appending.

- **vars** – Deprecated; Same as kwargs.
- ****kwargs** – Additional variables provided as a dict for { {placeholders} } inside the HTML

Returns

```
class flare.html5._attrLabel
```

Bases: object

```
    _getLabel(self)
```

```
    _setLabel(self, val)
```

```
class flare.html5._attrCharset
```

Bases: object

```
    _getCharset(self)
```

```
    _setCharset(self, val)
```

```
class flare.html5._attrCite
```

Bases: object

```
    _getCite(self)
```

```
    _setCite(self, val)
```

```
class flare.html5._attrDatetime
```

Bases: object

```
    _getDatetime(self)
```

```
    _setDatetime(self, val)
```

```
class flare.html5._attrForm
```

Bases: object

```
    _getForm(self)
```

```
    _setForm(self, val)
```

```
class flare.html5._attrAlt
```

Bases: object

```
    _getAlt(self)
```

```
    _setAlt(self, val)
```

```
class flare.html5._attrAutofocus
```

Bases: object

```
    _getAutofocus(self)
```

```
    _setAutofocus(self, val)
```

```
class flare.html5._attrDisabled
```

Bases: object

```
class flare.html5._attrChecked
```

Bases: object

```
    _getChecked(self)
    _setChecked(self, val)
class flare.html5._attrIndeterminate
    Bases: object
    _getIndeterminate(self)
    _setIndeterminate(self, val)
class flare.html5._attrName
    Bases: object
    _getName(self)
    _setName(self, val)
class flare.html5._attrValue
    Bases: object
    _getValue(self)
    _setValue(self, val)
class flare.html5._attrAutocomplete
    Bases: object
    _getAutocomplete(self)
    _setAutocomplete(self, val)
class flare.html5._attrRequired
    Bases: object
    _getRequired(self)
    _setRequired(self, val)
class flare.html5._attrMultiple
    Bases: object
    _getMultiple(self)
    _setMultiple(self, val)
class flare.html5._attrSize
    Bases: object
    _getSize(self)
    _setSize(self, val)
class flare.html5._attrFor
    Bases: object
    _getFor(self)
    _setFor(self, val)
```

```
class flare.html5._attrInputs
```

```
    Bases: _attrRequired
```

```
    _getMaxlength(self)
```

```
    _setMaxlength(self, val)
```

```
    _getPlaceholder(self)
```

```
    _setPlaceholder(self, val)
```

```
    _getReadonly(self)
```

```
    _setReadonly(self, val)
```

```
class flare.html5._attrFormhead
```

```
    Bases: object
```

```
    _getFormaction(self)
```

```
    _setFormaction(self, val)
```

```
    _getFormenctype(self)
```

```
    _setFormenctype(self, val)
```

```
    _getFormmethod(self)
```

```
    _setFormmethod(self, val)
```

```
    _getFormtarget(self)
```

```
    _setFormtarget(self, val)
```

```
    _getFormnovalidate(self)
```

```
    _setFormnovalidate(self, val)
```

```
class flare.html5._attrHref
```

```
    Bases: object
```

```
    _getHref(self)
```

```
        Url of a Page.
```

```
        Parameters self –
```

```
    _setHref(self, val)
```

```
        Url of a Page.
```

```
        Parameters val – URL
```

```
    _getHreflang(self)
```

```
    _setHreflang(self, val)
```

```
class flare.html5._attrTarget
```

```
    Bases: object
```

```
    _getTarget(self)
```

```
    _setTarget(self, val)
```



```
class flare.html5._attrType
    Bases: object
    _getType(self)
    _setType(self, val)

class flare.html5._attrMedia
    Bases: _attrType
    _getMedia(self)
    _setMedia(self, val)

class flare.html5._attrDimensions
    Bases: object
    _getWidth(self)
    _setWidth(self, val)
    _getHeight(self)
    _setHeight(self, val)

class flare.html5._attrUsemap
    Bases: object
    _getUsemap(self)
    _setUsemap(self, val)

class flare.html5._attrMultimedia
    Bases: object
    _getAutoplay(self)
    _setAutoplay(self, val)
    _getPlaysinline(self)
    _setPlaysinline(self, val)
    _getControls(self)
    _setControls(self, val)
    _getLoop(self)
    _setLoop(self, val)
    _getMuted(self)
    _setMuted(self, val)
    _getPreload(self)
    _setPreload(self, val)
```

```
class flare.html5._attrRel
    Bases: object
    _getRel(self)
    _setRel(self, val)

class flare.html5._attrSrc
    Bases: object
    _getSrc(self)
    _setSrc(self, val)

class flare.html5.A(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrHref, _attrTarget, _attrMedia, _attrRel, _attrName
    _tagName = a
    _getDownload(self)
        The download attribute specifies the path to a download.
        Returns filename
    _setDownload(self, val)
        The download attribute specifies the path to a download.
        Parameters val – filename

class flare.html5.Area(*args, appendTo=None, style=None, **kwargs)
    Bases: A, _attrAlt
    _tagName = area
    _leafTag = True
    _getCoords(self)
    _setCoords(self, val)
    _getShape(self)
    _setShape(self, val)

class flare.html5.Audio(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrSrc, _attrMultimedia
    _tagName = audio

class flare.html5.Bdo(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = bdo

class flare.html5.Blockquote(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = blockquote
    _getBlockquote(self)
```

```

    _setBlockquote(self, val)

class flare.html5.BodyCls(*args, **kwargs)
    Bases: Widget
    flare.html5._body
    flare.html5.Body()

class flare.html5.Canvas(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDimensions
    _tagName = canvas

class flare.html5.Command(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrLabel, _attrType, _attrDisabled, _attrChecked
    _tagName = command
    _getIcon(self)
    _setIcon(self, val)
    _getRadiogroup(self)
    _setRadiogroup(self, val)

class flare.html5._Del(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrCite, _attrDatetime
    _tagName = _del

class flare.html5.Dialog(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = dialog
    _getOpen(self)
    _setOpen(self, val)

class flare.html5.Abbbr(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = abbr

class flare.html5.Address(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = address

class flare.html5.Article(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = article

class flare.html5.Aside(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = aside

```

```
class flare.html5.B(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = b

class flare.html5.Bdi(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = bdi

class flare.html5.Br(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = br
    _leafTag = True

class flare.html5.Caption(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = caption

class flare.html5.Cite(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = cite

class flare.html5.Code(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = code

class flare.html5.Datalist(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = datalist

class flare.html5.Dfn(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = dfn

class flare.html5.Div(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = div

class flare.html5.Em(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = em

class flare.html5.Embed(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrSrc, _attrType, _attrDimensions
    _tagName = embed
    _leafTag = True
```

```

class flare.html5.Figcaption(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = figcaption

class flare.html5.Figure(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = figure

class flare.html5.Footer(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = footer

class flare.html5.Header(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = header

class flare.html5.H1(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = h1

class flare.html5.H2(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = h2

class flare.html5.H3(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = h3

class flare.html5.H4(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = h4

class flare.html5.H5(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = h5

class flare.html5.H6(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = h6

class flare.html5.Hr(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = hr
    _leafTag = True

class flare.html5.I(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget

```

```
    _tagName = i
class flare.html5.Kdb(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = kdb
class flare.html5.Legend(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = legend
class flare.html5.Mark(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = mark
class flare.html5.Noscript(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = noscript
class flare.html5.P(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = p
class flare.html5.Rq(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = rq
class flare.html5.Rt(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = rt
class flare.html5.Ruby(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = ruby
class flare.html5.S(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = s
class flare.html5.Samp(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = samp
class flare.html5.Section(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = section
class flare.html5.Small(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
```

```

    _tagName = small
class flare.html5.Strong(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = strong
class flare.html5.Sub(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = sub
class flare.html5.Summary(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = summary
class flare.html5.Sup(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = sup
class flare.html5.U(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = u
class flare.html5.Var(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = var
class flare.html5.Wbr(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = wbr
class flare.html5.Button(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrType, _attrForm, _attrAutofocus, _attrName, _attrValue,
    _attrFormhead
    _tagName = button
class flare.html5.Fieldset(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrForm, _attrName
    _tagName = fieldset
class flare.html5.Form(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrName, _attrTarget, _attrAutocomplete
    _tagName = form
    _getNovalidate(self)
    _setNovalidate(self, val)
    _getAction(self)

```

```
_setAction(self, val)
_getMethod(self)
_setMethod(self, val)
_getEnctype(self)
_setEnctype(self, val)
_getAccept_attrCharset(self)
_setAccept_attrCharset(self, val)
```

```
class flare.html5.Input(*args, appendTo=None, style=None, **kwargs)
```

```
    Bases: Widget, _attrDisabled, _attrType, _attrForm, _attrAlt, _attrAutofocus,
           _attrChecked, _attrIndeterminate, _attrName, _attrDimensions, _attrValue, _attrFormhead,
           _attrAutocomplete, _attrInputs, _attrMultiple, _attrSize, _attrSrc
```

```
    _tagName = input
    _leafTag = True
    _getAccept(self)
    _setAccept(self, val)
    _getList(self)
    _setList(self, val)
    _getMax(self)
    _setMax(self, val)
    _getMin(self)
    _setMin(self, val)
    _getPattern(self)
    _setPattern(self, val)
    _getStep(self)
    _setStep(self, val)
```

```
class flare.html5.Label(*args, forElem=None, **kwargs)
```

```
    Bases: Widget, _attrForm, _attrFor
```

```
    _tagName = label
    autoIdCounter = 0
```

```
class flare.html5.Optgroup(*args, appendTo=None, style=None, **kwargs)
```

```
    Bases: Widget, _attrDisabled, _attrLabel
```

```
    _tagName = optgroup
```



```

class flare.html5.Option(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrLabel, _attrValue
    _tagName = option
    _getSelected(self)
    _setSelected(self, val)

class flare.html5.Output(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrForm, _attrName, _attrFor
    _tagName = output

class flare.html5.Select(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrForm, _attrAutofocus, _attrName, _attrRequired,
    _attrMultiple, _attrSize
    _tagName = select
    _getSelectedIndex(self)
    _getOptions(self)

class flare.html5.Textarea(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrForm, _attrAutofocus, _attrName, _attrInputs, _attrValue
    _tagName = textarea
    _getCols(self)
    _setCols(self, val)
    _getRows(self)
    _setRows(self, val)
    _getWrap(self)
    _setWrap(self, val)

class flare.html5.HeadCls(*args, **kwargs)
    Bases: Widget
flare.html5._head
flare.html5.Head()

class flare.html5.Iframe(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrSrc, _attrName, _attrDimensions
    _tagName = iframe
    _getSandbox(self)
    _setSandbox(self, val)
    _getSrcdoc(self)
    _setSrcdoc(self, val)

```

```
    _getSeamless(self)
    _setSeamless(self, val)
class flare.html5.Img(src=None, *args, **kwargs)
    Bases: Widget, _attrSrc, _attrDimensions, _attrUsemap, _attrAlt
    _tagName = img
    _leafTag = True
    _getCrossorigin(self)
    _setCrossorigin(self, val)
    _getIsmap(self)
    _setIsmap(self, val)
class flare.html5.Ins(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrCite, _attrDatetime
    _tagName = ins
class flare.html5.Keygen(*args, appendTo=None, style=None, **kwargs)
    Bases: Form, _attrAutofocus, _attrDisabled
    _tagName = keygen
    _getChallenge(self)
    _setChallenge(self, val)
    _getKeytype(self)
    _setKeytype(self, val)
class flare.html5.Link(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrHref, _attrMedia, _attrRel
    _tagName = link
    _leafTag = True
    _getSizes(self)
    _setSizes(self, val)
class flare.html5.Ul(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = ul
class flare.html5.Ol(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = ol
class flare.html5.Li(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
```

```

    _tagName = li
class flare.html5.Dl(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = dl
class flare.html5.Dt(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = dt
class flare.html5.Dd(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = dd
class flare.html5.Map(*args, forElem=None, **kwargs)
    Bases: Label, _attrType
    _tagName = map
class flare.html5.Menu(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = menu
class flare.html5.Meta(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrName, _attrCharset
    _tagName = meta
    _leafTag = True
    _getContent(self)
    _setContent(self, val)
class flare.html5.Meter(*args, appendTo=None, style=None, **kwargs)
    Bases: Form, _attrValue
    _tagName = meter
    _getHigh(self)
    _setHigh(self, val)
    _getLow(self)
    _setLow(self, val)
    _getMax(self)
    _setMax(self, val)
    _getMin(self)
    _setMin(self, val)
    _getOptimum(self)

```

```
    _setOptimum(self, val)

class flare.html5.Nav(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = nav

class flare.html5.Object(*args, appendTo=None, style=None, **kwargs)
    Bases: Form, _attrType, _attrName, _attrDimensions, _attrUsemap
    _tagName = object

class flare.html5.Param(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrName, _attrValue
    _tagName = param
    _leafTag = True

class flare.html5.Progress(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrValue
    _tagName = progress
    _getMax(self)
    _setMax(self, val)

class flare.html5.Q(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrCite
    _tagName = q

class flare.html5.Script(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrSrc, _attrCharset
    _tagName = script
    _getAsync(self)
    _setAsync(self, val)
    _getDefer(self)
    _setDefer(self, val)

class flare.html5.Source(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrMedia, _attrSrc
    _tagName = source
    _leafTag = True

class flare.html5.Span(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = span

class flare.html5.Details(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
```

```

    _tagName = details
    _getOpen(self)
    _setOpen(self, val)
class flare.html5.Summary(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = summary
class flare.html5.Style(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrMedia
    _tagName = style
    _getScoped(self)
    _setScoped(self, val)
class flare.html5.Tr(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = tr
    _getRowspan(self)
    _setRowspan(self, span)
class flare.html5.Td(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = td
    _getColspan(self)
    _setColspan(self, span)
    _getRowspan(self)
    _setRowspan(self, span)
class flare.html5.Th(*args, appendTo=None, style=None, **kwargs)
    Bases: Td
    _tagName = th
class flare.html5.Thead(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = thead
class flare.html5.Tbody(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = tbody
class flare.html5.ColWrapper(parentElem, *args, **kwargs)
    Bases: object

```

```
    __getitem__(self, item)

    __setitem__(self, key, value)

class flare.html5.RowWrapper(parentElem, *args, **kwargs)
    Bases: object
    __getitem__(self, item)

class flare.html5.Table(*args, **kwargs)
    Bases: Widget
    _tagName = table
    prepareRow(self, row)
    prepareCol(self, row, col)
    prepareGrid(self, rows, cols)
    clear(self)
    _getCell(self)
    getRowCount(self)

class flare.html5.Time(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDatetime
    _tagName = time

class flare.html5.Track(*args, forElem=None, **kwargs)
    Bases: Label, _attrSrc
    _tagName = track
    _leafTag = True
    _getKind(self)
    _setKind(self, val)
    _getSrclang(self)
    _setSrclang(self, val)
    _getDefault(self)
    _setDefault(self, val)

class flare.html5.Video(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrSrc, _attrDimensions, _attrMultimedia
    _tagName = video
    _getPoster(self)
    _setPoster(self, val)
```

class flare.html5.Template(*args, appendTo=None, style=None, **kwargs)

Bases: *Widget*

_tagName = **template**

flare.html5.unescape(val, maxLength=0)

Unquotes several HTML-quoted characters in a string.

Parameters

- **val** (*str*) – The value to be unescaped.
- **maxLength** (*int*) – Cut-off after maxLength characters. A value of 0 means “unlimited”. (default)

Returns The unquoted string.

Return type *str*

flare.html5.doesEventHitWidgetOrParents(event, widget)

Test if event ‘event’ hits widget ‘widget’ (or *any* of its parents).

flare.html5.doesEventHitWidgetOrChildren(event, widget)

Test if event ‘event’ hits widget ‘widget’ (or *any* of its children).

flare.html5.textToHtml(node, text)

Generates html nodes from text by splitting text into content and into line breaks html5.Br.

Parameters

- **node** – The node where the nodes are appended to.
- **text** – The text to be inserted.

flare.html5.parseInt(s, ret=0)

Parses a value as int.

flare.html5.parseFloat(s, ret=0.0)

Parses a value as float.

flare.html5.getKey(event)

Returns the Key Identifier of the given event.

Available Codes: <https://www.w3.org/TR/2006/WD-DOM-Level-3-Events-20060413/keyset.html#KeySet-Set>

flare.html5.isArrowLeft(event)

flare.html5.isArrowUp(event)

flare.html5.isArrowRight(event)

flare.html5.isArrowDown(event)

flare.html5.isEscape(event)

flare.html5.isReturn(event)

flare.html5.isControl(event)

flare.html5.isShift(event)

`flare.html5.isMeta(event)`

`flare.html5.__tags`

`flare.html5.__reVarReplacer`

`flare.html5.registerTag(tagName, widgetClass, override=True)`

`flare.html5.tag(arg)`

Decorator to register a sub-class of `html5.Widget` either under its class-name or an associated tag-name.

```
"""python # register class Foo as <foo>-Tag @html5.tag class Foo(html5.Div):
    pass
# register class Bar as <baz>-Tag @html5.tag("baz") class Bar(html5.Div):
    pass
"""
```

`flare.html5._buildTags(debug=False)`

Generates a dictionary of all to the `html5-library` known tags and their associated objects and attributes.

class `flare.html5.HtmlAst`

Bases: `list`

Abstract syntax tree element used by `parseHTML()`.

`flare.html5.parseHTML(html: str, debug: bool = False) → HtmlAst`

Parses the provided HTML-code according to the tags registered by `html5.registerTag()` or components that used the `html5.tag`-decorator.

`flare.html5.fromHTML(html: [str, HtmlAst], appendTo: Widget = None, bindTo: Widget = None, debug: bool = False, **kwargs) → [Widget]`

Parses the provided HTML code according to the objects defined in the `html5-library`.

`html` can also be pre-compiled by `parseHTML()` so that it executes faster.

Constructs all objects as DOM nodes. The first level is chained into `appendTo`. If no `appendTo` is provided, `appendTo` will be set to `html5.Body()`.

If `bindTo` is provided, objects are bound to this widget.

```
"""python from vi import html5
div = html5.Div() html5.parse.fromHTML("""
    <div>Yeah! <a href="hello world" [name]="myLink" class="trullman bernd" disabled> hah
        ala malla" bababtschga" st <em>ah</em>ralla <i>malla tralla</i> da </a>lala
    </div>""", div)
div.myLink.appendChild("appended!") """
```


flare.translations

Submodules

flare.translations.de

Module Contents

flare.translations.de.lngDe

flare.translations.en

Module Contents

flare.translations.en.lngEn

Package Contents

flare.translations.lngDe

flare.translations.lngEn

flare.views

Submodules

flare.views.helpers

Module Contents

Functions

<i>generateView</i> (view: flare.views.view.View, module-Name, actionName, name=None, data=())	
<i>addView</i> (view: flare.views.view.View, name=None)	Add a View and make it available.
<i>updateDefaultView</i> (name)	
<i>removeView</i> (name, targetView=None)	
<i>registerViews</i> (root, path)	Add all Views in a folder.
<i>zip_listdir</i> (zip_file, target_dir)	

Attributes

sitepackagespath

`flare.views.helpers.sitepackagespath`

`flare.views.helpers.generateView(view: flare.views.view.View, moduleName, actionName, name=None, data=())`

`flare.views.helpers.addView(view: flare.views.view.View, name=None)`

Add a View and make it available.

`flare.views.helpers.updateDefaultView(name)`

`flare.views.helpers.removeView(name, targetView=None)`

`flare.views.helpers.registerViews(root, path)`

Add all Views in a folder.

`flare.views.helpers.zip_listdir(zip_file, target_dir)`

`flare.views.view`

Module Contents

Classes

View

ViewWidget

Attributes

params

`class flare.views.view.View(dictOfWidgets=None, name=None)`

`onActiveViewChanged(self, viewName, *args, **kwargs)`

`loadView(self)`

`flare.views.view.params`

`class flare.views.view.ViewWidget(view)`

Bases: [flare.html5.Div](#)

`onViewfocusedChanged(self, viewname, *args, **kwargs)`

`initWidget(self)``onDetach(self)`

Package Contents

Classes

StateHandler

Attributes

conf

```
class flare.views.StateHandler(initialize=(), widget=None)
```

```
    updateState(self, key, value)
```

```
    getState(self, key, empty=None)
```

```
    register(self, key, widget)
```

```
    unregister(self, key, widget)
```

```
flare.views.conf
```

```
flare.viur
```

Subpackages

```
flare.viur.bones
```

Expose all bones.

Submodules

```
flare.viur.bones.base
```

Collection of Basebone related classes.

Module Contents

Classes

<i>ReadFromClientErrorSeverity</i>	Enum for Errors.
<i>BaseEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>BaseViewWidget</i>	Base class for a bone-compliant view widget implementation using a div.
<i>BaseMultiEditWidgetEntry</i>	Base class for an entry in a MultiBone container.
<i>BaseMultiEditWidget</i>	Class for encapsulating multiple bones inside a container.
<i>BaseMultiViewWidget</i>	
<i>BaseLanguageEditWidget</i>	Class for encapsulating a bone for each language inside a container.
<i>BaseBone</i>	

```
class flare.viur.bones.base.ReadFromClientErrorSeverity
```

```
    Bases: enum.IntEnum
```

```
    Enum for Errors.
```

```
    NotSet = 0
```

```
    InvalidatesOther = 1
```

```
    Empty = 2
```

```
    Invalid = 3
```

```
class flare.viur.bones.base.BaseEditWidget(bone, **kwargs)
```

```
    Bases: flare.ignite.html5.Div
```

```
    Base class for a bone-compliant edit widget implementation using an input field.
```

```
    This widget defines the general interface of a bone edit control.
```

```
    style = ['flr-value']
```

```
    createWidget(self)
```

```
        Function for creating the Widget or multiple Widgets that represent the bone.
```

```
    updateWidget(self)
```

```
        Function for updating the Widget or multiple Widgets that represent the bone.
```

```
    unserialize(self, value=None)
```

```
        Unserialize the widget value.
```

```
    serialize(self)
```

```
        Serialize the widget value.
```

```
class flare.viur.bones.base.BaseViewWidget(bone, **kwargs)
```

```
    Bases: flare.ignite.html5.Div
```

```
    Base class for a bone-compliant view widget implementation using a div.
```

```

style = ['flr-value']

unserialize(self, value=None)
    Unserialize the widget value.

serialize(self)
    Serialize the widget value.

class flare.viur.bones.base.BaseMultiEditWidgetEntry(widget: flare.ignite.html5.Widget,
    errorInformation=None)

    Bases: flare.ignite.html5.Div
    Base class for an entry in a MultiBone container.

    style = ['flr-bone-widgets-item']

    onRemoveBtnClick(self)

    onDragStart(self, event)

    onDragOver(self, event)

    onDragLeave(self, event)

    onDragEnd(self, event)

    onDrop(self, event)

class flare.viur.bones.base.BaseMultiEditWidget(bone, widgetFactory: callable, **kwargs)
    Bases: flare.ignite.html5.Div
    Class for encapsulating multiple bones inside a container.

    entryFactory

    style = ['flr-value-container']

    onAddBtnClick(self)

    onRemoveBtnClick(self)

    addEntry(self, value=None)

    unserialize(self, value)

    serialize(self)

class flare.viur.bones.base.BaseMultiViewWidget(bone, widgetFactory: callable, **kwargs)
    Bases: flare.ignite.html5.Ul

    unserialize(self, value)

    serialize(self)

class flare.viur.bones.base.BaseLanguageEditWidget(bone, widgetFactory: callable, **kwargs)
    Bases: flare.ignite.html5.Div
    Class for encapsulating a bone for each language inside a container.

    onLangBtnClick(self, sender)

```

unserialize(*self*, *value*)

serialize(*self*)

class flare.viur.bones.base.**BaseBone**(*moduleName*, *boneName*, *skelStructure*, *errors=None*,
errorQueue=None, **args*, ***kwargs*)

Bases: object

editWidgetFactory

viewWidgetFactory

multiEditWidgetFactory

multiViewWidgetFactory

languageEditWidgetFactory

languageViewWidgetFactory

Base “Catch-All” delegate for everything not handled separately.

editWidget(*self*, *value=None*, *errorInformation=None*) → flare.ignite.html5.Widget

viewWidget(*self*, *value=None*)

labelWidget(*self*)

tooltipWidget(*self*)

errorWidget(*self*)

boneWidget(*self*, **args*, ***kwargs*)

flare.viur.bones.boolean

Module Contents

Classes

<i>BooleanEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>BooleanViewWidget</i>	Base class for a bone-compliant view widget implementation using a div.
<i>BooleanBone</i>	

class flare.viur.bones.boolean.**BooleanEditWidget**(*bone*, ***kwargs*)

Bases: *flare.viur.bones.base.BaseEditWidget*

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

style = ['flr-value', 'flr-value--boolean']

createWidget(self)

Function for creating the Widget or multiple Widgets that represent the bone.

updateWidget(self)

Function for updating the Widget or multiple Widgets that represent the bone.

unserialize(self, value=None)

Unserialize the widget value.

serialize(self)

Serialize the widget value.

class flare.viur.bones.boolean.**BooleanViewWidget**(bone, **kwargs)

Bases: [flare.viur.bones.base.BaseViewWidget](#)

Base class for a bone-compliant view widget implementation using a div.

unserialize(self, value=None)

Unserialize the widget value.

class flare.viur.bones.boolean.**BooleanBone**(moduleName, boneName, skelStructure, errors=None, errorQueue=None, *args, **kwargs)

Bases: [flare.viur.bones.base.BaseBone](#)

editWidgetFactory**viewWidgetFactory**

static **checkFor**(moduleName, boneName, skelStructure, *args, **kwargs)

[flare.viur.bones.color](#)

Module Contents**Classes**

ColorEditWidget	Base class for a bone-compliant edit widget implementation using an input field.
ColorViewWidget	Base class for a bone-compliant view widget implementation using a div.
ColorBone	

class flare.viur.bones.color.**ColorEditWidget**(bone, **kwargs)

Bases: [flare.viur.bones.base.BaseEditWidget](#)

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

style = ['flr-value', 'flr-value--color']

createWidget(self)

Function for creating the Widget or multiple Widgets that represent the bone.

updateWidget(*self*)

Function for updating the Widget or multiple Widgets that represent the bone.

onUnsetBtnClick(*self*)

serialize(*self*)

Serialize the widget value.

class flare.viur.bones.color.**ColorViewWidget**(*bone*, ****kwargs**)

Bases: [flare.viur.bones.base.BaseViewWidget](#)

Base class for a bone-compliant view widget implementation using a div.

unserialize(*self*, *value=None*)

Unserialize the widget value.

class flare.viur.bones.color.**ColorBone**(*moduleName*, *boneName*, *skelStructure*, *errors=None*,
errorQueue=None, **args*, ****kwargs**)

Bases: [flare.viur.bones.base.BaseBone](#)

editWidgetFactory

viewWidgetFactory

static checkFor(*moduleName*, *boneName*, *skelStructure*, **args*, ****kwargs**)

[flare.viur.bones.date](#)

Module Contents

Classes

DateEditWidget	Base class for a bone-compliant edit widget implementation using an input field.
DateViewWidget	Base class for a bone-compliant view widget implementation using a div.
DateBone	

class flare.viur.bones.date.**DateEditWidget**(*bone*, ****kwargs**)

Bases: [flare.viur.bones.base.BaseEditWidget](#)

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

style = ['flr-value', 'flr-value--date']

createWidget(*self*)

Function for creating the Widget or multiple Widgets that represent the bone.

updateWidget(*self*)

Function for updating the Widget or multiple Widgets that represent the bone.

unserialize(*self*, *value=None*)

Unserialize the widget value.

serialize(*self*)

Serialize the widget value.

class flare.viur.bones.date.**DateViewWidget**(*bone*, ****kwargs**)

Bases: [flare.viur.bones.base.BaseViewWidget](#)

Base class for a bone-compliant view widget implementation using a div.

unserialize(*self*, *value=None*)

Unserialize the widget value.

class flare.viur.bones.date.**DateBone**(*moduleName*, *boneName*, *skelStructure*, *errors=None*,
errorQueue=None, **args*, ****kwargs**)

Bases: [flare.viur.bones.base.BaseBone](#)

editWidgetFactory

viewWidgetFactory

static checkFor(*moduleName*, *boneName*, *skelStructure*, **args*, ****kwargs**)

[flare.viur.bones.email](#)

Module Contents

Classes

EmailEditWidget	Base class for a bone-compliant edit widget implementation using an input field.
EmailViewWidget	Base class for a bone-compliant view widget implementation using a div.
EmailBone	

class flare.viur.bones.email.**EmailEditWidget**(*bone*, ****kwargs**)

Bases: [flare.viur.bones.base.BaseEditWidget](#)

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

updateWidget(*self*)

Function for updating the Widget or multiple Widgets that represent the bone.

class flare.viur.bones.email.**EmailViewWidget**(*bone*, ****kwargs**)

Bases: [flare.viur.bones.base.BaseViewWidget](#)

Base class for a bone-compliant view widget implementation using a div.

unserialize(*self*, *value=None*)

Unserialize the widget value.

```
class flare.viur.bones.email.EmailBone(moduleName, boneName, skelStructure, errors=None,
                                         errorQueue=None, *args, **kwargs)

    Bases: flare.viur.bones.base.BaseBone

    editWidgetFactory

    viewWidgetFactory

    static checkFor(moduleName, boneName, skelStructure, *args, **kwargs)
```

`flare.viur.bones.numeric`

Module Contents

Classes

<i>NumericEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>NumericViewWidget</i>	Base class for a bone-compliant view widget implementation using a div.
<i>NumericBone</i>	

Functions

<i>_formatCurrencyValue</i> (value, bone)	Internal helper function that formats a numeric value which is a string according to the bone's formatting
---	--

`flare.viur.bones.numeric._formatCurrencyValue(value, bone)`

Internal helper function that formats a numeric value which is a string according to the bone's formatting

```
class flare.viur.bones.numeric.NumericEditWidget(bone, **kwargs)
```

Bases: *flare.viur.bones.base.BaseEditWidget*

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

```
style = ['flr-value', 'flr-value--numeric']
```

```
createWidget(self)
```

Function for creating the Widget or multiple Widgets that represent the bone.

```
updateWidget(self)
```

Function for updating the Widget or multiple Widgets that represent the bone.

```
setValue(self, value)
```

```
onChange(self, event)
```

```
unserialize(self, value=None)
```

Unserialize the widget value.

serialize(*self*)

Serialize the widget value.

class flare.viur.bones.numeric.**NumericViewWidget**(*bone*, ****kwargs**)

Bases: [flare.viur.bones.base.BaseViewWidget](#)

Base class for a bone-compliant view widget implementation using a div.

unserialize(*self*, *value=None*)

Unserialize the widget value.

class flare.viur.bones.numeric.**NumericBone**(***args**, ****kwargs**)

Bases: [flare.viur.bones.base.BaseBone](#)

editWidgetFactory

viewWidgetFactory

static checkFor(*moduleName*, *boneName*, *skelStructure*, ***args**, ****kwargs**)

[flare.viur.bones.password](#)

Module Contents

Classes

[PasswordEditWidget](#)

Base class for a bone-compliant edit widget implementation using an input field.

[PasswordBone](#)

class flare.viur.bones.password.**PasswordEditWidget**(*bone*, ****kwargs**)

Bases: [flare.viur.bones.base.BaseEditWidget](#)

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

style = ['flr-value', 'flr-value--password', 'flr-value-container', 'input-group']

createWidget(*self*)

Function for creating the Widget or multiple Widgets that represent the bone.

updateWidget(*self*)

Function for updating the Widget or multiple Widgets that represent the bone.

serialize(*self*)

Serialize the widget value.

class flare.viur.bones.password.**PasswordBone**(*moduleName*, *boneName*, *skelStructure*, *errors=None*, *errorQueue=None*, ***args**, ****kwargs**)

Bases: [flare.viur.bones.base.BaseBone](#)

editWidgetFactory

static checkFor(*moduleName*, *boneName*, *skelStructure*, ***args**, ****kwargs**)

`flare.viur.bones.raw`

Module Contents

Classes

<i>RawEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>RawViewWidget</i>	Base class for a bone-compliant view widget implementation using a div.
<i>RawBone</i>	

class `flare.viur.bones.raw.RawEditWidget`(*bone*, ***kwargs*)

Bases: *[flare.viur.bones.base.BaseEditWidget](#)*

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

style = ['flr-value', 'flr-value--raw']

createWidget(*self*)

Function for creating the Widget or multiple Widgets that represent the bone.

updateWidget(*self*)

Function for updating the Widget or multiple Widgets that represent the bone.

class `flare.viur.bones.raw.RawViewWidget`(*bone*, ***kwargs*)

Bases: *[flare.viur.bones.base.BaseViewWidget](#)*

Base class for a bone-compliant view widget implementation using a div.

unserialize(*self*, *value=None*)

Unserialize the widget value.

class `flare.viur.bones.raw.RawBone`(*moduleName*, *boneName*, *skelStructure*, *errors=None*,
errorQueue=None, **args*, ***kwargs*)

Bases: *[flare.viur.bones.base.BaseBone](#)*

editWidgetFactory

viewWidgetFactory

static checkFor(*moduleName*, *boneName*, *skelStructure*, **args*, ***kwargs*)

`flare.viur.bones.record`

Module Contents

Classes

<i>RecordEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>RecordViewWidget</i>	Base class for a bone-compliant view widget implementation using a div.
<i>RecordBone</i>	

class `flare.viur.bones.record.RecordEditWidget`(*bone*, ***kwargs*)

Bases: `flare.viur.bones.base.BaseEditWidget`

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

style = ['flr-value', 'flr-value--record']

createWidget(*self*)

Function for creating the Widget or multiple Widgets that represent the bone.

updateWidget(*self*)

Function for updating the Widget or multiple Widgets that represent the bone.

unserialize(*self*, *value=None*)

Unserialize the widget value.

serialize(*self*)

Serialize the widget value.

class `flare.viur.bones.record.RecordViewWidget`(*bone*, *language=None*, ***kwargs*)

Bases: `flare.viur.bones.base.BaseViewWidget`

Base class for a bone-compliant view widget implementation using a div.

style = ['flr-value', 'flr-value--record']

unserialize(*self*, *value=None*)

Unserialize the widget value.

class `flare.viur.bones.record.RecordBone`(*moduleName*, *boneName*, *skelStructure*, *errors=None*, *errorQueue=None*, **args*, ***kwargs*)

Bases: `flare.viur.bones.base.BaseBone`

editWidgetFactory

viewWidgetFactory

static checkFor(*moduleName*, *boneName*, *skelStructure*, **args*, ***kwargs*)

flare.viur.bones.relational

Module Contents

Classes

<i>RelationalEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>RelationalViewWidget</i>	
<i>RelationalMultiEditWidget</i>	Class for encapsulating multiple bones inside a container.
<i>RelationalBone</i>	
<i>HierarchyBone</i>	
<i>TreeItemBone</i>	
<i>TreeDirBone</i>	
<i>FileEditDirectWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>FileViewWidget</i>	
<i>FileMultiEditDirectWidget</i>	Class for encapsulating multiple bones inside a container.
<i>FileDirectBone</i>	
<i>FileEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>FileBone</i>	

Functions

<i>_getDefaultValues</i> (structure)	Gets defaultValues from a structure.
--------------------------------------	--------------------------------------

flare.viur.bones.relational._getDefaultValues(*structure*)

Gets defaultValues from a structure.

class flare.viur.bones.relational.**RelationalEditWidget**(*bone*, *language=None*, ***kwargs*)

Bases: *flare.viur.bones.base.BaseEditWidget*

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

style = ['flr-value', 'flr-value--relational']

createWidget(*self*)

Function for creating the Widget or multiple Widgets that represent the bone.

```

updateWidget(self)
    Function for updating the Widget or multiple Widgets that represent the bone.
updateString(self)
onChange(self, event)
unserialize(self, value=None)
    Unserialize the widget value.
serialize(self)
    Serialize the widget value.
onSelectBtnClick(self)
onDeleteBtnClick(self)
class flare.viur.bones.relational.RelationalViewWidget(bone, language=None, **kwargs)
    Bases: flare.html5.Div
    style = ['flr-value', 'flr-value--relational']
unserialize(self, value=None)
serialize(self)
class flare.viur.bones.relational.RelationalMultiEditWidget(*args, **kwargs)
    Bases: flare.viur.bones.base.BaseMultiEditWidget
    Class for encapsulating multiple bones inside a container.
onAddBtnClick(self)
    _addEntriesFromSelection(self, selector, selection)
class flare.viur.bones.relational.RelationalBone(*args, **kwargs)
    Bases: flare.viur.bones.base.BaseBone
    editWidgetFactory
    viewWidgetFactory
    multiEditWidgetFactory
    selectorAllow
    static checkFor(moduleName, boneName, skelStructure, *args, **kwargs)
class flare.viur.bones.relational.HierarchyBone(*args, **kwargs)
    Bases: RelationalBone
    static checkFor(moduleName, boneName, skelStructure, *args, **kwargs)
class flare.viur.bones.relational.TreeItemBone(*args, **kwargs)
    Bases: RelationalBone
    selectorAllow
    static checkFor(moduleName, boneName, skelStructure, *args, **kwargs)

```

```
class flare.viur.bones.relational.TreeDirBone(*args, **kwargs)
    Bases: RelationalBone
    selectorAllow

    static checkFor(moduleName, boneName, skelStructure, *args, **kwargs)

class flare.viur.bones.relational.FileEditDirectWidget(bone, language=None, **kwargs)
    Bases: RelationalEditWidget
    Base class for a bone-compliant edit widget implementation using an input field.
    This widget defines the general interface of a bone edit control.
    style = ['flr-value', 'flr-value--file']

    createWidget(self)
        Function for creating the Widget or multiple Widgets that represent the bone.
    updateWidget(self)
        Function for updating the Widget or multiple Widgets that represent the bone.
    onChange(self, event)
    startUpload(self, file)
    onDragEnter(self, event)
    onDragOver(self, event)
    onDragLeave(self, event)
    onDrop(self, event)
    onUploadSuccess(self, uploader, entry)
    onUploadFailed(self, uploader, errorCode)
    unserialize(self, value=None)
        Unserialize the widget value.
    onDeleteBtnClick(self)

class flare.viur.bones.relational.FileViewWidget(bone, language=None, **kwargs)
    Bases: RelationalViewWidget
    unserialize(self, value=None)

class flare.viur.bones.relational.FileMultiEditDirectWidget(bone, widgetFactory: callable,
                                                             **kwargs)
    Bases: flare.html5.Div
    Class for encapsulating multiple bones inside a container.
    entryFactory
    style = ['flr-value-container']
    onChange(self, event)
```



```

startUpload(self, file)

onDragEnter(self, event)

onDragOver(self, event)

onDragLeave(self, event)

onDrop(self, event)

onUploadSuccess(self, uploader, entry)

onUploadFailed(self, uploader, errorCode)

addEntry(self, value=None)

unserialize(self, value)

serialize(self)

class flare.viur.bones.relational.FileDirectBone(*args, **kwargs)
    Bases: TreeItemBone

    editWidgetFactory

    viewWidgetFactory

    multiEditWidgetFactory

    static checkFor(moduleName, boneName, skelStructure, *args, **kwargs)

class flare.viur.bones.relational.FileEditWidget(bone, language=None, **kwargs)
    Bases: RelationalEditWidget

    Base class for a bone-compliant edit widget implementation using an input field.

    This widget defines the general interface of a bone edit control.

    style = ['flr-value', 'flr-value--relational', 'flr-value--file']

    createWidget(self)
        Function for creating the Widget or multiple Widgets that represent the bone.

    unserialize(self, value=None)
        Unserialize the widget value.

class flare.viur.bones.relational.FileBone(*args, **kwargs)
    Bases: TreeItemBone

    editWidgetFactory

    viewWidgetFactory

    static checkFor(moduleName, boneName, skelStructure, *args, **kwargs)

```

`flare.viur.bones.select`

Module Contents

Classes

<i>SelectMultipleEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>SelectSingleEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>SelectViewWidget</i>	Base class for a bone-compliant view widget implementation using a div.
<i>SelectMultipleBone</i>	
<i>SelectSingleBone</i>	

class `flare.viur.bones.select.SelectMultipleEditWidget`(*bone*, ***kwargs*)

Bases: `flare.viur.bones.base.BaseEditWidget`

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

style = ['flr-value-container', 'option-group']

entryTemplate

createWidget(*self*)

Function for creating the Widget or multiple Widgets that represent the bone.

updateWidget(*self*)

Function for updating the Widget or multiple Widgets that represent the bone.

unserialize(*self*, *value=None*)

Unserialize the widget value.

serialize(*self*)

Serialize the widget value.

class `flare.viur.bones.select.SelectSingleEditWidget`(*bone*, ***kwargs*)

Bases: `flare.viur.bones.base.BaseEditWidget`

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

entryTemplate

createWidget(*self*)

Function for creating the Widget or multiple Widgets that represent the bone.

updateWidget(*self*)

Function for updating the Widget or multiple Widgets that represent the bone.

unserialize(*self*, *value=None*)

Unserialize the widget value.

serialize(*self*)

Serialize the widget value.

class flare.viur.bones.select.**SelectViewWidget**(*bone*, ***kwargs*)

Bases: [flare.viur.bones.base.BaseViewWidget](#)

Base class for a bone-compliant view widget implementation using a div.

unserialize(*self*, *value=None*)

Unserialize the widget value.

class flare.viur.bones.select.**SelectMultipleBone**(**args*, ***kwargs*)

Bases: [flare.viur.bones.base.BaseBone](#)

editWidgetFactory

multiEditWidgetFactory

viewWidgetFactory

Base “Catch-All” delegate for everything not handled separately.

static checkFor(*moduleName*, *boneName*, *skelStructure*, **args*, ***kwargs*)

class flare.viur.bones.select.**SelectSingleBone**(**args*, ***kwargs*)

Bases: [SelectMultipleBone](#)

editWidgetFactory

static checkFor(*moduleName*, *boneName*, *skelStructure*, **args*, ***kwargs*)

flare.viur.bones.spatial

Module Contents

Classes

[SpatialEditWidget](#)

Base class for a bone-compliant edit widget implementation using an input field.

[SpatialBone](#)

class flare.viur.bones.spatial.**SpatialEditWidget**(*bone*, ***kwargs*)

Bases: [flare.viur.bones.base.BaseEditWidget](#)

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

createWidget(*self*)

Function for creating the Widget or multiple Widgets that represent the bone.

updateWidget(*self*)

Function for updating the Widget or multiple Widgets that represent the bone.

unserialize(*self*, *value=None*)

Unserialize the widget value.

serialize(*self*)

Serialize the widget value.

class flare.viur.bones.spatial.**SpatialBone**(*moduleName*, *boneName*, *skelStructure*, *errors=None*,
errorQueue=None, **args*, ***kwargs*)Bases: [flare.viur.bones.base.BaseBone](#)**editWidgetFactory****static checkFor**(*moduleName*, *boneName*, *skelStructure*, **args*, ***kwargs*)**flare.viur.bones.string**

Module Contents

Classes

StringEditWidget	Base class for a bone-compliant edit widget implementation using an input field.
StringViewWidget	Base class for a bone-compliant view widget implementation using a div.
StringBone	

class flare.viur.bones.string.**StringEditWidget**(*bone*, ***kwargs*)Bases: [flare.viur.bones.base.BaseEditWidget](#)

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

style = ['flr-value', 'flr-value--string']**createWidget**(*self*)

Function for creating the Widget or multiple Widgets that represent the bone.

updateWidget(*self*)

Function for updating the Widget or multiple Widgets that represent the bone.

onChange(*self*, *event*)**onKeyUp**(*self*, *event*)**renderTimeout**(*self*)**updateLength**(*self*)**unserialize**(*self*, *value=None*)

Unserialize the widget value.

serialize(*self*)

Serialize the widget value.

class flare.viur.bones.string.**StringViewWidget**(*bone*, ***kwargs*)

Bases: [flare.viur.bones.base.BaseViewWidget](#)

Base class for a bone-compliant view widget implementation using a div.

unserialize(*self*, *value=None*)

Unserialize the widget value.

class flare.viur.bones.string.**StringBone**(*moduleName*, *boneName*, *skelStructure*, *errors=None*, *errorQueue=None*, **args*, ***kwargs*)

Bases: [flare.viur.bones.base.BaseBone](#)

editWidgetFactory

viewWidgetFactory

static checkFor(*moduleName*, *boneName*, *skelStructure*, **args*, ***kwargs*)

[flare.viur.bones.text](#)

Module Contents

Classes

TextEditWidget	Base class for a bone-compliant edit widget implementation using an input field.
TextViewWidget	Base class for a bone-compliant view widget implementation using a div.
TextBone	

class flare.viur.bones.text.**TextEditWidget**(*bone*, ***kwargs*)

Bases: [flare.viur.bones.base.BaseEditWidget](#)

Base class for a bone-compliant edit widget implementation using an input field.

This widget defines the general interface of a bone edit control.

style = ['flr-value', 'flr-value--text']

createWidget(*self*)

Function for creating the Widget or multiple Widgets that represent the bone.

updateWidget(*self*)

Function for updating the Widget or multiple Widgets that represent the bone.

_setDisabled(*self*, *disable*)

class flare.viur.bones.text.**TextViewWidget**(*bone*, ***kwargs*)

Bases: [flare.viur.bones.base.BaseViewWidget](#)

Base class for a bone-compliant view widget implementation using a div.

unserialize(*self*, *value=None*)

Unserialize the widget value.

class flare.viur.bones.text.**TextBone**(*moduleName*, *boneName*, *skelStructure*, *errors=None*,
errorQueue=None, **args*, ***kwargs*)

Bases: *flare.viur.bones.base.BaseBone*

editWidgetFactory

viewWidgetFactory

static checkFor(*moduleName*, *boneName*, *skelStructure*, **args*, ***kwargs*)

flare.viur.widgets

Submodules

flare.viur.widgets.file

Module Contents

Classes

Search

FileImagePopup

FilePreviewImage

<i>Uploader</i>	Uploads a file to the server while providing visual feedback of the progress.
-----------------	---

FileLeafWidget

FileNodeWidget

<i>FileWidget</i>	Base Widget that renders a tree.
-------------------	----------------------------------

Functions

getImagePreview(*data*, *cropped=False*, *size=150*)

flare.viur.widgets.file.**getImagePreview**(*data*, *cropped=False*, *size=150*)

class flare.viur.widgets.file.**Search**(**args*, ***kwargs*)

Bases: *flare.ignite.html5.Div*

doSearch(*self*, **args*, ***kwargs*)

```

    resetSearch(self)

    onKeyDown(self, event)

    resetLoadingState(self)

    reevaluate(self)

    focus(self)

class flare.viur.widgets.file.FileImagePopup(preview, *args, **kwargs)
    Bases: flare.popup.Popup
    onClick(self, event)

    onDownloadBtnClick(self, sender=None)

class flare.viur.widgets.file.FilePreviewImage(file=None, size=150, *args, **kwargs)
    Bases: flare.ignite.html5.Div
    setFile(self, file)

    download(self)

    onClick(self, sender=None)

class flare.viur.widgets.file.Uploader(file, node, context=None, showResultMessage=True,
                                         module='file', *args, **kwargs)
    Bases: flare.ignite.Progress
    Uploads a file to the server while providing visual feedback of the progress.

    onUploadUrlAvailable(self, req)
        Internal callback - the actual upload url (retrieved by calling /file/getUploadURL) is known.

    onSkeyAvailable(self, req)
        Internal callback - the Security-Key is known.
        # Only for core 2.x needed

    onLoad(self, *args, **kwargs)
        Internal callback - The state of our upload changed.

    onUploadAdded(self, req)

    onProgress(self, event)
        Internal callback - further bytes have been transmitted.

    onSuccess(self, *args, **kwargs)
        Internal callback - The upload succeeded.

    onFailed(self, errorCode, *args, **kwargs)

    replaceWithMessage(self, message, isSuccess)

class flare.viur.widgets.file.FileLeafWidget(module, data, structure, widget, *args, **kwargs)
    Bases: flare.viur.widgets.tree.TreeLeafWidget
    EntryIcon(self)
        Leafs have a different Icon.

```

setStyle(*self*)

Leaf have a different color.

class flare.viur.widgets.file.**FileNodeWidget**(*module, data, structure, widget, *args, **kwargs*)

Bases: *flare.viur.widgets.tree.TreeNodeWidget*

setStyle(*self*)

Is used to define the appearance of the element.

class flare.viur.widgets.file.**FileWidget**(*module, rootNode=None, selectMode=None, node=None, context=None, *args, **kwargs*)

Bases: *flare.viur.widgets.tree.TreeBrowserWidget*

Base Widget that renders a tree.

leafWidget

nodeWidget

searchWidget(*self*)

onStartSearch(*self, searchStr, *args, **kwargs*)

getChildKey(*self, widget*)

Derives a string used to sort the entries on each level.

static canHandle(*module, moduleInfo*)

flare.viur.widgets.htmleditor

Module Contents

Classes

TextInsertImageAction

Extended version for a button with a text and icon, which binds itself to an event function.

HtmlEditor

Attributes

summernoteEditor

flare.viur.widgets.htmleditor.**summernoteEditor**

class flare.viur.widgets.htmleditor.**TextInsertImageAction**(*summernote=None, boneName="", *args, **kwargs*)

Bases: *flare.button.Button*

Extended version for a button with a text and icon, which binds itself to an event function.


```
onClick(self, sender=None)

onSelectionActivated(self, selectWdg, selection)

static isSuitableFor(modul, handler, actionName)

resetLoadingState(self)

class flare.viur.widgets.htmleditor.HtmlEditor(*args, **kwargs)
    Bases: flare.html5.Textarea
    initSources = False

    _attachSummernote(self, retry=0)

    onAttach(self)

    onDetach(self)

    onEditorChange(self, e, *args, **kwargs)

    _getValue(self)

    _setValue(self, val)

    enable(self)
        Enables an element, in case it is not already enabled.

    disable(self)
        Disables an element, in case it is not already disabled.
        On disabled elements, events are not triggered anymore.
```

flare.viur.widgets.list

Module Contents

Classes

ListWidget	Provides the interface to list-applications.
SkellistItem	Extended version for a button with a text and icon, which binds itself to an event function.
ListSelection	

```
class flare.viur.widgets.list.ListWidget(module, filter=None, columns=None, filterID=None,
                                          filterDescr=None, batchSize=None, context=None,
                                          autoload=True, *args, **kwargs)

    Bases: flare.html5.Div

    Provides the interface to list-applications.

    It acts as a data-provider for a DataTable and binds an action-bar to this table.
```

setSelector(*self, callback, multi=True, allow=None*)

Configures the widget as selector for a relationalBone and shows it.

onAcceptSelectionChanged(*self, event, *args, **kwargs*)

static canHandle(*moduleName, moduleInfo*)

class flare.viur.widgets.list.**SkellistItem**(*skel*)

Bases: [flare.button.Button](#)

Extended version for a button with a text and icon, which binds itself to an event function.

buildWidget(*self*)

onActiveSelectionChanged(*self, event, *args, **kwargs*)

class flare.viur.widgets.list.**ListSelection**(*modulname, filter=None, title=None, id=None, className=None, icon=None, enableShortcuts=True, closeable=True, footer=True, *args, **kwargs*)

Bases: [flare.popup.Popup](#)

requestClients(*self*)

onRequestList(*self, skellist*)

onActiveSelectionChanged(*self, event, *args, **kwargs*)

activateSelection(*self, widget*)

reloadList(*self*)

buildListSelection(*self*)

onApplyfilterChanged(*self, value, *args, **kwargs*)

onAcceptSelectionChanged(*self, event, *args, **kwargs*)

onActiveButtonChanged(*self, event, *args, **kwargs*)

acceptSelection(*self*)

setContent(*self, widget*)

[flare.viur.widgets.tree](#)

Module Contents

Classes

TreeItemWidget

TreeLeafWidget

TreeNodeWidget

TreeWidget Base Widget that renders a tree.

BrowserLeafWidget

BrowserNodeWidget

BreadcrumbNodeWidget

TreeBrowserWidget Base Widget that renders a tree.

class flare.viur.widgets.tree.**TreeItemWidget**(*module, data, structure, widget, *args, **kwargs*)

Bases: *flare.html5.Li*

setStyle(*self*)

Is used to define the appearance of the element.

additionalDropAreas(*self*)

Drag and Drop areas.

markDraggedElement(*self*)

Mark the current dragged Element.

unmarkDraggedElement(*self*)

onDragStart(*self, event*)

onDragEnd(*self, event*)

onDragOver(*self, event*)

Test wherever the current drag would mean.

“make it a child of us”, “insert before us” or “insert after us” and apply the correct classes.

onDragLeave(*self, event*)

Remove all drop indicating classes.

disableDragMarkers(*self*)

onDrop(*self, event*)

We received a drop.

Test wherever its means “make it a child of us”, “insert before us” or “insert after us” and initiate the corresponding NetworkService requests.

EntryIcon(*self*)

toggleArrow(*self*)

buildDescription(*self*)

Creates the visual representation of our entry.

onClick(*self*, *event*)

onDbClick(*self*, *event*)

toggleExpand(*self*)

Toggle a Node and request if needed child elements.

class flare.viur.widgets.tree.**TreeLeafWidget**(*module*, *data*, *structure*, *widget*, **args*, ***kwargs*)

Bases: [TreeItemWidget](#)

skelType = leaf

setStyle(*self*)

Leaf have a different color.

toggleArrow(*self*)

Leafes cant be toggled.

EntryIcon(*self*)

Leafs have a different Icon.

class flare.viur.widgets.tree.**TreeNodeWidget**(*module*, *data*, *structure*, *widget*, **args*, ***kwargs*)

Bases: [TreeItemWidget](#)

skelType = node

class flare.viur.widgets.tree.**TreeWidget**(*module*, *rootNode=None*, *node=None*, *context=None*, **args*, ***kwargs*)

Bases: [flare.html5.Div](#)

Base Widget that renders a tree.

nodeWidget

leafWidget

setSelector(*self*, *callback*, *multi=True*, *allow=None*)

Configures the widget as selector for a relationalBone and shows it.

static canHandle(*moduleName*, *moduleInfo*)

class flare.viur.widgets.tree.**BrowserLeafWidget**(*module*, *data*, *structure*, *widget*, **args*, ***kwargs*)

Bases: [TreeLeafWidget](#)

setStyle(*self*)

Leaf have a different color.

class flare.viur.widgets.tree.**BrowserNodeWidget**(*module*, *data*, *structure*, *widget*, **args*, ***kwargs*)

Bases: [TreeNodeWidget](#)

setStyle(*self*)

Is used to define the appearance of the element.

class flare.viur.widgets.tree.**BreadcrumbNodeWidget**(*module*, *data*, *structure*, *widget*, **args*, ***kwargs*)

Bases: [TreeNodeWidget](#)

setStyle(*self*)

Is used to define the appearance of the element.

class flare.viur.widgets.tree.**TreeBrowserWidget**(*module*, *rootNode=None*, *node=None*, *context=None*,
args*, *kwargs*)

Bases: [TreeWidget](#)

Base Widget that renders a tree.

leafWidget**nodeWidget**

static **canHandle**(*module*, *moduleInfo*)

Submodules

flare.viur.formatString

Module Contents**Functions**

[formatString](#)(*format*: str, *data*: Dict, *structure=None*, *language=None*) Central entryPoint

[formatStringHandler](#)(*format*: str, *value*: Dict, *structure*: Dict, *language*: str = 'de') → str

[displayStringHandler](#)(*display*: str, *value*: Dict, *structure*: Dict, *language*: str = 'de') →
[flare.html5.Widget]

[evalStringHandler](#)(*format*, *data*, *structure*, *language*)

flare.viur.formatString.**formatString**(*format*: str, *data*: Dict, *structure=None*, *language=None*)
Central entryPoint

if string contains \$(we use old formatstrings else we use evalStrings (core 3.0 draft)

displayStrings actually only used in relations and records. This handler can be used with display param

flare.viur.formatString.**formatStringHandler**(*format*: str, *value*: Dict, *structure*: Dict, *language*: str =
'de') → str

flare.viur.formatString.**displayStringHandler**(*display*: str, *value*: Dict, *structure*: Dict, *language*: str =
'de') → [flare.html5.Widget]

flare.viur.formatString.**evalStringHandler**(*format*, *data*, *structure*, *language*)

flare.viur.formconf

Module Contents

flare.viur.formconf.conf

```
# A value displayed as "empty value" "emptyValue": translate("-",)
# Language settings "flare.language.current": "de",
# Global holder to main admin window "mainWindow": None,
# Modules list "modules": {"_tasks": {"handler": "singleton", "name": "Tasks"}},
# Language settings "defaultLanguage": "de",
# Cached selector widgets on relationalBones for re-use "selectors": {},
```

flare.viur.formerrors

Module Contents

Functions

<i>collectBoneErrors</i> (errorList, currentKey, boneStructure)	Collect Errors from given errorList.
---	--------------------------------------

flare.viur.formerrors.**collectBoneErrors**(errorList, currentKey, boneStructure)

Collect Errors from given errorList.

severity: NotSet = 0 InvalidatesOther = 1 Empty = 2 Invalid = 3

flare.viur.forms

Module Contents

Classes

<i>ViurForm</i>	Handles an input form for a VIUR skeleton.
<i>ViurFormBone</i>	
<i>ViurFormSubmit</i>	Extended version for a button with a text and icon, which binds itself to an event function.

```
class flare.viur.forms.ViurForm(formName: str = None, moduleName: str = None, actionName: str = 'add',
                                skel=None, structure=None, visible=(), ignore=(), hide=(), errors=None,
                                context=None, *args, **kwargs)
```

Bases: *flare.html5.Form*

Handles an input form for a VIUR skeleton.

```

onChange(self, event)

onBoneChange(self, bone)

_setModuleName(self, val)

_setActionname(self, val)

_setFormname(self, val)

buildForm(self)
    Builds a form with save button.

buildInternalForm(self)
    Builds only the form.

registerField(self, key, widget)

update(self)
    Updates current form view state regarding conditional input fields.

submitForm(self)

unserialize(self, skel: Dict = None)
    Unserializes a dict of values into this form. :param skel: Either a dict of values to be unserialized into this
    form, or None for emptying all values.

serialize(self, all=False) → Dict
    Serializes all bone's values into a dict to be sent to ViUR or the be evaluated.

actionSuccess(self, req)

handleErrors(self)

createFormSuccessMessage(self)

createFormErrorMessage(self)

actionFailed(self, req, *args, **kwargs)

onFormSuccess(self, event)

onSubmitStatusChanged(self, value, *args, **kwargs)

class flare.viur.forms.ViurFormBone(boneName=None, form=None, defaultvalue=None, hidden=False,
                                     filter=None)
    Bases: flare.html5.Div

    onAttach(self)

    onChange(self, event, *args, **kwargs)

    unserialize(self, data=None)

    serialize(self)

    _setBonename(self, val)

    _setLabel(self, val)

```

```
_setPlaceholder(self, val)
```

```
_setHide(self, val)
```

```
_setValue(self, val)
```

```
setInvalid(self, errors=None)
```

```
setValid(self)
```

```
class flare.viur.forms.ViurFormSubmit(text=None, callback=None, className='btn--submit btn--primary',  
                                     icon=None, badge=None, form=None)
```

Bases: *flare.button.Button*

Extended version for a button with a text and icon, which binds itself to an event function.

```
onAttach(self)
```

```
sendViurForm(self, sender=None)
```

```
onSubmitStatusChanged(self, value, *args, **kwargs)
```

flare.viur.formtooltip

Module Contents

Classes

ToolTip

Small utility class for providing tooltips.

```
class flare.viur.formtooltip.ToolTip(shortText="", longText="", *args, **kwargs)
```

Bases: *flare.html5.Div*

Small utility class for providing tooltips.

```
onClick(self, event)
```

```
_setDisabled(self, disabled)
```

Package Contents

Classes

PriorityQueue

Functions

formatString(format: str, data: Dict, structure=None, Central entryPoint language=None)

displayStringHandler(display: str, value: Dict, structure: Dict, language: str = 'de') → [flare.html5.Widget]

Attributes

conf # A value displayed as "empty value"

BoneSelector

ModuleWidgetSelector

DisplayDelegateSelector

class flare.viur.**PriorityQueue**

Bases: object

insert(self, priority, validateFunc, generator)

select(self, *args, **kwargs)

flare.viur.**conf**

A value displayed as “empty value” “emptyValue”: translate(“-“),

Language settings “flare.language.current”: “de”,

Global holder to main admin window “mainWindow”: None,

Modules list “modules”: {“_tasks”: {“handler”: “singleton”, “name”: “Tasks”}},

Language settings “defaultLanguage”: “de”,

Cached selector widgets on relationalBones for re-use “selectors”: {},

flare.viur.**formatString**(format: str, data: Dict, structure=None, language=None)

Central entryPoint

if string contains \$(we use old formatstrings else we use evalStrings (core 3.0 draft)

displayStrings actually only used in relations and records. This handler can be used with display param

flare.viur.**displayStringHandler**(display: str, value: Dict, structure: Dict, language: str = 'de') → [flare.html5.Widget]

flare.viur.**BoneSelector**

flare.viur.**ModuleWidgetSelector**

flare.viur.**DisplayDelegateSelector**

exception `flare.viur.InvalidBoneValueException`

Bases: `ValueError`

Inappropriate argument value (of correct type).

`flare.widgets`

Submodules

`flare.widgets.buttonbar`

Module Contents

Classes

ButtonBar

ButtonBarButton

Extended version for a button with a text and icon, which binds itself to an event function.

ButtonBarSearch

class `flare.widgets.buttonbar.ButtonBar`

Bases: *flare.html5.Div*

onActiveButtonChanged(*self*, *event*, *args, **kwargs)

addButton(*self*, *name*, *btnStr*)

buttonClicked(*self*, *widget*)

class `flare.widgets.buttonbar.ButtonBarButton`

Bases: *flare.button.Button*

Extended version for a button with a text and icon, which binds itself to an event function.

onActiveButtonChanged(*self*, *event*, *args, **kwargs)

class `flare.widgets.buttonbar.ButtonBarSearch`

Bases: *flare.html5.Div*

applyFilter(*self*, *widget*)

onApplyfilterChanged(*self*, *event*, *args, **kwargs)

onActiveButtonChanged(*self*, *event*, *args, **kwargs)

Submodules

flare.button

Flare-styled button Widgets.

Module Contents

Classes

<i>Button</i>	Extended version for a button with a text and icon, which binds itself to an event function.
---------------	--

class flare.button.**Button**(*text=None, callback=None, className="", icon=None*)
Bases: *flare.html5.Button*
Extended version for a button with a text and icon, which binds itself to an event function.
onBind(*self, widget, name*)
Event function that is called on the widget when it is bound to another widget with a name.
This is only done by the HTML parser, a manual binding by the user is not triggered.
onClick(*self, event, widget=None*)
resetIcon(*self*)
update(*self*)
_setIcon(*self, icon*)
_getIcon(*self*)
_setText(*self, text*)
_getText(*self*)

flare.cache

The cache module is set on top of the network module and caches any entries read.
When the same entry (identified by module and key) is requested, it first is returned from the cache, when already there.

Module Contents

Classes

<i>Cache</i>
<i>Plan</i>

class flare.cache.Cache

Bases: object

updateStructure(self, module, structure)

update(self, module, key, data, structure=None)

lookup(self, module, key='current')

struct(self, module)

start(self, plan, finishHandler=None, failureHandler=None)

finish(self, plan)

require(self, *args)

invalidate(self, *args)

onDataChanged(self, module, key=None, **kwargs)

request(self, *args, finishHandler=None, failureHandler=None)

class flare.cache.Plan(module, action, params=None, follow=None, alias='current', local=True)

Bases: object

run(self, cache)

finish(self, cache)

_onRequestSuccess(self, req)

_onRequestFailure(self, req, code)

flare.config

Flare configuration.

Module Contents

Functions

<i>updateConf</i> (other: Dict)	Merges other into conf.
---	-------------------------

Attributes

<i>conf</i>

<i>htmlExpressionEvaluator</i>
--

`flare.config.updateConf`(*other*: Dict)
Merges other into conf.
`flare.config.conf`
`flare.config.htmlExpressionEvaluator`

`flare.debug`

still WIP

Module Contents

Functions

<code>debug</code> (<i>element</i> =None)	Debug popup
<code>debugElement</code> (<i>element</i>)	recursive debug tree

`flare.debug.debug`(*element*=None)
Debug popup
`flare.debug.debugElement`(*element*)
recursive debug tree

`flare.event`

Event dispatcher for non-browser Events which occur on Widget state changes.

Module Contents

Classes

<code>EventDispatcher</code>	Base class for event notifier.
------------------------------	--------------------------------

`class flare.event.EventDispatcher`(*name*)
Bases: object
Base class for event notifier.
`_genTargetFuncName`(*self*)
Return the name of the function called on the receiving object.
`register`(*self*, *cb*, *reset*=False)
Append “cb” to the list of objects to inform of the given Event.
Does nothing if cb has already subscribed. :param cb: the object to register :type cb: object

unregister(*self*, *cb*)

Remove “cb” from the list of objects to inform of the given Event.

Does nothing if cb is not in that list. :param cb: the object to remove :type cb: object

fire(*self*, **args*, ***kwargs*)

Fire the event.

Informs all subscribed listeners. All parameters passed to the receiving function.

flare.handler

Flare base handlers for ViUR prototypes.

Module Contents

Classes

requestHandler

ListHandler

SyncHandler

class flare.handler.**requestHandler**(*module*, *action*, *params*=(), *eventName*='listUpdated', *secure*=False)

requestData(*self*, **args*, ***kwargs*)

requestSuccess(*self*, *req*)

_requestFailed(*self*, *req*, **args*, ***kwargs*)

onListStatusChanged(*self*, *event*, **args*, ***kwargs*)

getDescrFromValue(*self*, *definition*, *val*)

buildSelectDescr(*self*, *skel*, *structure*)

class flare.handler.**ListHandler**(*module*, *action*, *params*=(), *eventName*='listUpdated', *secure*=False)

Bases: *requestHandler*

reload(*self*)

filter(*self*, *filterparams*)

getCurrentAmount(*self*)

requestNext(*self*)

requestSuccess(*self*, *req*)

class flare.handler.**SyncHandler**

Bases: object

```
static request(url, params=None, jsonResult=None)

genReqStr(self, params)

_request(self, url, params)

onCompletion(self, text)

onError(self, text, code)
```

flare.i18n

Internationalization tools to easily implement multi-language applications.

Module Contents

Functions

<hr/>	
<i>buildTranslations</i> (pathToFolder)	
<hr/>	
<i>translate</i> (key, fallback=None, **kwargs)	Tries to translate the given string in the currently selected language.
<hr/>	
<i>addTranslation</i> (lang, a, b=None)	Adds or updates new translations.
<hr/>	
<i>setLanguage</i> (lang)	Sets the current language to lang.
<hr/>	
<i>getLanguage</i> ()	Returns the current language.
<hr/>	

Attributes

<i>_currentLanguage</i>	
<hr/>	
<i>_currentLanguage</i>	
<hr/>	
<i>_currentLanguage</i>	
<hr/>	
<i>_currentLanguage</i>	
<hr/>	
<i>_runtimeTranslations</i>	
<hr/>	
<i>_lngMap</i>	
<hr/>	

```
flare.i18n._currentLanguage
flare.i18n._currentLanguage
flare.i18n._currentLanguage = en
flare.i18n._currentLanguage
```

`flare.i18n._runtimeTranslations`

`flare.i18n._lngMap`

`flare.i18n.buildTranslations(pathToFolder)`

`flare.i18n.translate(key, fallback=None, **kwargs)`

Tries to translate the given string in the currently selected language.

Supports replacing markers (using {markerName} syntax).

Parameters

- **key** – The string to translate
- **fallback** – Return string when no translation is found.

Returns The translated string

`flare.i18n.addTranslation(lang, a, b=None)`

Adds or updates new translations.

`flare.i18n.setLanguage(lang)`

Sets the current language to lang.

`flare.i18n.getLanguage()`

Returns the current language.

flare.icons

Components for displaying icons.

Module Contents

Classes

<i>SvgIcon</i>	A raw, embedded SVG icon-component.
<i>Icon</i>	Icon component with first-letter fallback, normally shown as embedded SVG.
<i>BadgeIcon</i>	A badge icon is an icon-component with a little badge, e.g. a number of new messages or items in the cart or so.

class `flare.icons.SvgIcon(value=None, fallbackIcon=None, title="")`

Bases: [*flare.html5.svg.Svg*](#)

A raw, embedded SVG icon-component.

_leafTag = True

_setValue(self, value)

_setTitle(self, val)

Advisory information associated with the element.

Parameters val – str

getIcon(*self*)

replaceSVG(*self*, *icondata*)

requestFallBack(*self*, *data*, *status*)

class flare.icons.**Icon**(*value=None, fallbackIcon=None, title="", classes=[]*)

Bases: [flare.html5.I](#)

Icon component with first-letter fallback, normally shown as embedded SVG.

_leafTag = **True**

_setValue(*self*, *value*)

_setTitle(*self*, *val*)

Advisory information associated with the element.

Parameters **val** – str

_setFallback(*self*, *val*)

onError(*self*)

class flare.icons.**BadgeIcon**(*title="", value=None, fallbackIcon=None, badge=None*)

Bases: [Icon](#)

A badge icon is an icon-component with a little badge, e.g. a number of new messages or items in the cart or so.

_setBadge(*self*, *badge*)

_getBadge(*self*)

flare.ignite

Flare-specific form Widgets with specialized classes and behavior.

Module Contents

Classes

Label

Input

Switch

Check

Radio

Select

Textarea

Progress

Item

Table

```
class flare.ignite.Label(*args, **kwargs)
```

```
    Bases: flare.html5.Label
```

```
class flare.ignite.Input(*args, **kwargs)
```

```
    Bases: flare.html5.Input
```

```
class flare.ignite.Switch(*args, **kwargs)
```

```
    Bases: flare.html5.Div
```

```
    _setChecked(self, value)
```

```
    _getChecked(self)
```

```
class flare.ignite.Check(*args, **kwargs)
```

```
    Bases: flare.html5.Input
```

```
class flare.ignite.Radio(*args, **kwargs)
```

```
    Bases: flare.html5.Div
```

```
class flare.ignite.Select(*args, **kwargs)
```

```
    Bases: flare.html5.Select
```

```
class flare.ignite.Textarea(*args, **kwargs)
```

```
    Bases: flare.html5.Textarea
```

```
class flare.ignite.Progress(*args, **kwargs)
```

```
    Bases: flare.html5.Progress
```

```
class flare.ignite.Item(title=None, descr=None, className=None, *args, **kwargs)
```

```
    Bases: flare.html5.Div
```

```

class flare.ignite.Table(*args, **kwargs)
    Bases: flare.html5.Table
    prepareRow(self, row)
    prepareCol(self, row, col)
    fastGrid(self, rows, cols, createHidden=False)

```

flare.input

Input widget with additional event handling.

Module Contents

Classes

Input

```

class flare.input.Input(type='text', placeholder=None, callback=None, id=None, focusCallback=None,
                        *args, **kwargs)
    Bases: flare.html5.Input
    onChange(self, event)
    onFocus(self, event)
    onDetach(self)

```

flare.intersectionObserver

Module Contents

Classes

IntersectionObserver

Python wrapper for IntersectionObserver.

```

class flare.intersectionObserver.IntersectionObserver(callback, rootWidget=None,
                                                       rootMargin='0px', threshold=0.2)

```

Python wrapper for IntersectionObserver.

Usage: myObserver = IntersectionObserver(myChangeFunction) myObserver.observe(aWidget)

jsObserver

observableWidgets = []

observe(self, widget)

unobserve(self, widget)

flare.log

Generalized Python logging for Pyodide.

Module Contents

Classes

<i>FlareLogRecord</i>	A LogRecord instance represents an event being logged.
<i>JSConsoleHandler</i>	Brings our awesome log messages onto the js console.

Functions

<i>prepareLogger</i> (level: str, mergeArgs: bool = False) → None	Call this before first usage of logging or getLogger().
<i>getLogger</i> (name: str) → Any	Creates a child logger of our 'root' logger with a name.

Attributes

<i>loggers</i>

```
flare.log.loggers = []
```

```
class flare.log.FlareLogRecord(name, level, pathname, lineno, msg, args, exc_info, func=None, sinfo=None, mergeArgs=False, **kwargs)
```

Bases: logging.LogRecord

A LogRecord instance represents an event being logged.

LogRecord instances are created every time something is logged. They contain all the information pertinent to the event being logged. The main information passed in is in msg and args, which are combined using str(msg) % args to create the message field of the record. The record also includes information such as when the record was created, the source line where the logging call was made, and any exception information to be logged.

NOTE: This is mostly the same as the original LogRecord. Differences:

- Do not use a single dict as keyword args because pyodites' Proxy objects cannot be used

with isinstance(proxy, collections.abc.Mapping). This will be discussed upstream. * User-supplied arguments to logging messages will not be replaced in message, but will be forwarded to js console via separate arguments.

getMessage(self) → str

Optionally merge args into message driven by mergeArgs flag in ctor, otherwise this will happen later in js console as objects.

Returns

class flare.log.JSConsoleHandler(*stream=None*)

Bases: logging.StreamHandler

Brings our awesome log messages onto the js console.

emit(*self, record: logging.LogRecord*) → None

Emit a record.

If a formatter is specified, it is used to format the record. The record is then written to the stream with a trailing newline. If exception information is present, it is formatted using `traceback.print_exception` and appended to the stream. If the stream has an ‘encoding’ attribute, it is used to determine how to do the output to the stream.

flare.log.**prepareLogger**(*level: str, mergeArgs: bool = False*) → None

Call this before first usage of logging or getLogger().

:param level Log level as str as of all, info, debug, warning, error or critical :param mergeArgs: If True we’re merging args into resulting message resulting in possible duplicated output or get the ‘raw’ message output if False.

flare.log.**getLogger**(*name: str*) → Any

Creates a child logger of our ‘root’ logger with a name.

Usually it’s the `__name__` attribute of the module you want to use a logger for.

Parameters *name* –

Returns

flare.network

Wrapper to handle ViUR-related Ajax requests.

Module Contents

Classes

<i>DeferredCall</i>	Calls the given function with a fixed delay.
<i>HTTPRequest</i>	Wrapper around XMLHttpRequest.
<i>NetworkService</i>	Generic wrapper around ajax requests.
<i>requestGroup</i>	

Functions

<i>NiceError</i> (req, code, params="", then=None)	Displays a descriptive error message using an Alert dialog to the user.
<i>NiceErrorAndThen</i> (function)	Returns a callback which first displays a descriptive error message to the user and then calls another function.
<i>processSkelQueue</i> ()	
<i>getUrlHashAsString</i> (urlHash=None)	
<i>getUrlHashAsObject</i> (urlHash=None)	
<i>setUrlHash</i> (hash, param=None)	

Attributes

<i>skeyRequestQueue</i>

class flare.network.DeferredCall(*func*, **args*, ***kwargs*)

Bases: object

Calls the given function with a fixed delay.

This allows assuming that calls to NetworkService are always asynchronous, so its guaranteed that any initialization code can run before the Network-Call yields results.

run(*self*)

Internal callback that executes the callback function.

class flare.network.HTTPRequest(*method*, *url*, *callbackSuccess*=None, *callbackFailure*=None, *payload*=None, *content_type*=None, *response_type*=None, *asynchronous*=True)

Bases: object

Wrapper around XMLHttpRequest.

onReadyStateChange(*self*, **args*, ***kwargs*)

Internal callback.

flare.network.NiceError(*req*, *code*, *params*="", *then*=None)

Displays a descriptive error message using an Alert dialog to the user.

flare.network.NiceErrorAndThen(*function*)

Returns a callback which first displays a descriptive error message to the user and then calls another function.

flare.network.skeyRequestQueue = []

flare.network.processSkelQueue()

```
class flare.network.NetworkService(module, url, params, successHandler, failureHandler, finishedHandler,  
                                   modifies, secure, kickoff, group=None)
```

Bases: object

Generic wrapper around ajax requests.

Handles caching and multiplexing multiple concurrent requests to the same resource. It also acts as the central proxy to notify currently active widgets of changes made to data on the server.

```
changeListeners = []
```

```
host =
```

```
prefix = /json
```

```
defaultFailureHandler
```

```
retryCodes
```

```
retryMax = 3
```

```
retryDelay = 5000
```

```
static notifyChange(module, **kwargs)
```

Broadcasts a change made to data of module 'module' to all currently registered changeListeners.

Parameters **module** (*str*) – Name of the module where the change occurred

```
static registerChangeListener(listener)
```

Registers object 'listener' for change notifications.

'listener' must provide an 'onDataChanged' function accepting one parameter: the name of the module. Does nothing if that object has already registered. :param listener: The object to register :type listener: object

```
static removeChangeListener(listener)
```

Unregisters the object 'listener' from change notifications.

Parameters **listener** (*object*) – The object to unregister. It must be currently registered.

```
static genReqStr(params)
```

```
static decode(req)
```

Decodes a response received from the server (ie parsing the json).

Returns object

```
static isOkay(req)
```

```
static urlForArgs(module, path)
```

Constructs the final url for that request.

If module is given, it prepends "/prefix" If module is None, path is returned unchanged. :param module: Name of the target module or None :type module: str or None :param path: Path (either relative to 'module' or absolute if 'module' is None :type path: str :returns: str

```
kickoff(self)
```

```
static request(module, url, params=None, successHandler=None, failureHandler=None,  
               finishedHandler=None, modifies=False, secure=False, kickoff=True, group=None)
```

Performs an AJAX request. Handles caching and security-keys.

Calls made to this function are guaranteed to be async.

Parameters

- **module** (*str* or *None*) – Target module on the server. Set to *None* if you want to call anything else
- **url** (*str* or *None*) – The path (relative to module) or a full url if module is *None*
- **successHandler** (*callable*) – function beeing called if the request succeeds. Must take one argument (the request).
- **failureHandler** (*callable*) – function beeing called if the request failes. Must take two arguments (the request and an error-code).
- **finishedHandler** (*callable*) – function beeing called if the request finished (regardless wherever it succeeded or not). Must take one argument (the request).
- **modifies** (*bool*) – If set to *True*, it will automatically broadcast an *onDataChanged* event for that module.
- **secure** (*bool*) – If true, include a fresh securitykey in this request. Defaults to *False*.

```
doFetch(self, url, params, skey)
```

Internal function performing the actual AJAX request.

```
onCompletion(self, text)
```

Internal hook for the AJAX call.

```
onError(self, text, code)
```

Internal hook for the AJAX call.

```
onTimeout(self, text)
```

Internal hook for the AJAX call.

```
clear(self)
```

```
onFinished(self, success)
```

```
class flare.network.requestGroup(callback=None)
```

```
    addRequest(self, request)
```

```
    call(self)
```

```
    onFinished(self, success)
```

```
flare.network.getUrlHashAsString(urlHash=None)
```

```
flare.network.getUrlHashAsObject(urlHash=None)
```

```
flare.network.setUrlHash(hash, param=None)
```


flare.observable

Observed values firing events when changed.

Module Contents

Classes

ObservableValue

StateHandler

```
class flare.observable.ObservableValue(key, value=None)
```

```
    Bases: object
```

```
    value
```

```
    setValue(self, value)
```

```
class flare.observable.StateHandler(initialize=(), widget=None)
```

```
    updateState(self, key, value)
```

```
    getState(self, key, empty=None)
```

```
    register(self, key, widget)
```

```
    unregister(self, key, widget)
```

flare.popout

Popout menu that is expanded when hovering.

Example:

```
"""html <popout icon="icon-arrowhead-down">
    <popout-item @click="onEdit">edit</popout-item> <popout-item @click="onLeave">leave</popout-
    item> <popout-item @click="onDelete">delete</popout-item>
</popout> """
```

Module Contents

Classes

<i>PopoutItem</i>	It's an item in a popout menu.
<i>Popout</i>	Popout menu.

class flare.popout.**PopoutItem**(*args, appendTo=None, style=None, **kwargs)

Bases: *flare.html5.Div*

It's an item in a popout menu.

style = ['item', 'has-hover']

class flare.popout.**Popout**(*args, **kwargs)

Bases: *flare.html5.Div*

Popout menu.

style = ['popout-opener', 'popout-anchor']

_setIcon(self, icon)

_getIcon(self)

_setText(self, text)

_getText(self)

flare.popup

Pre-defined dialog widgets for user interaction.

Module Contents

Classes

<i>Popup</i>	
<i>Prompt</i>	
<i>Alert</i>	Just displaying an alerting message box with OK-button.
<i>Confirm</i>	
<i>TextareaDialog</i>	
<i>radioButtonDialog</i>	

```

class flare.popup.Popup(title="", id=None, className=None, icon=None, enableShortcuts=True,
                        closeable=True, *args, **kwargs)

    Bases: flare.html5.Div

    onAttach(self)

    onDetach(self)

    onDocumentKeyDown(self, event)

    close(self)

    onClose(self)

class flare.popup.Prompt(text, value="", successHandler=None, abortHandler=None, successLbl=None,
                        abortLbl=None, placeholder="", *args, **kwargs)

    Bases: Popup

    onKeyDown(self, event)

    onKeyUp(self, event)

    onDocumentKeyDown(self, event)

    onOkay(self, *args, **kwargs)

    onCancel(self, *args, **kwargs)

class flare.popup.Alert(msg, title=None, className=None, okCallback=None, okLabel=None,
                        icon='icon-info', closeable=True, *args, **kwargs)

    Bases: Popup

    Just displaying an alerting message box with OK-button.

    drop(self)

    onOkBtnClick(self)

    onKeyDown(self, event)

class flare.popup.Confirm(question, title=None, yesCallback=None, noCallback=None, yesLabel=None,
                        noLabel=None, icon='icon-question', closeable=True, *args, **kwargs)

    Bases: Popup

    onKeyDown(self, event)

    onDocumentKeyDown(self, event)

    drop(self)

    onYesClicked(self, *args, **kwargs)

    onNoClicked(self, *args, **kwargs)

class flare.popup.TextareaDialog(text, value="", successHandler=None, abortHandler=None,
                                successLbl=None, abortLbl=None, *args, **kwargs)

    Bases: Popup

    onDocumentKeyDown(self, event)

```

```
onOkay(self, *args, **kwargs)
```

```
onCancel(self, *args, **kwargs)
```

```
class flare.popup.radioButtonDialog(title, radioValues: list, radioButtonGroupName='radioButtonGroup',  
                                     checkedValue=None, icon='icon-question', closeable=True,  
                                     successHandler=None, abortHandler=None, successLbl=None,  
                                     abortLbl=None, *args, **kwargs)
```

Bases: [Popup](#)

```
onOkay(self, *args, **kwargs)
```

```
onCancel(self, *args, **kwargs)
```

flare.priorityqueue

Select object generators by priority.

This is used when implementing pluggable features, which can optionally be registered for specific use-cases.

Module Contents

Classes

[*PriorityQueue*](#)

```
class flare.priorityqueue.PriorityQueue
```

Bases: object

```
insert(self, priority, validateFunc, generator)
```

```
select(self, *args, **kwargs)
```

flare.safeeval

Here we are trying to provide an secure and safe space for evaluate simple python expressions on some 'data'.

If you only need a oneshot evaluation, you call safeEval and enjoy the result. Otherwise call first compile to get the ast representation and execute that compiled expression multiple times with different data. A plain instance of SafeEval without allowedCallables argument will not accept any method/function like call on execution

Module Contents

Classes

SafeEval

Safely evaluate an expression from an untrusted party.

class flare.safeeval.**SafeEval**(*allowedCallables: Union[None, Dict[str, Any]] = None*)

Safely evaluate an expression from an untrusted party.

_BoolOp(*self, node, names*)

Handling ast.BoolOp in a Pythonic style.

callNode(*self, node: ast.Call, names: Dict[str, Any]*) → Any

Evaluates the call if present in allowed callables.

Parameters

- **node** – The call node to evaluate
- **names** – a mapping of local objects which is used as ‘locals’ namespace

Returns If allowed to evaluate the node, its result will be returned

compareNode(*self, node: ast.Compare, names: Dict[str, Any]*) → bool

Evaluates an ‘if’ expression.

These are a bit tricky as they can have more than two operands (eg. “if 1 < 2 < 3”)

Parameters

- **node** – The compare node to evaluate
- **names** – a mapping of local objects which is used as ‘locals’ namespace

listNode(*self, node, names*)

execute(*self, node: [str, ast.AST], names: Dict[str, Any]*) → Any

Evaluates the current node with optional data.

Parameters

- **node** – The compare node to evaluate
- **names** – a mapping of local objects which is used as ‘locals’ namespace

Returns whatever the expression wants to return

compile(*self, expr: str*) → ast.AST

Compiles a python expression string to an ast.

Afterwards you can use execute to run the compiled ast with optional data. If you only want to run a ‘oneshot’ expression feel free to use our safeEval method.

Parameters **expr** – the expression to compile

Returns the ready to use ast node

safeEval(*self, expr: str, names: Dict[str, Any]*) → Any

Safely evaluate an expression.

If you want to evaluate the expression multiple times with different variables use compile to generate the AST once and call execute for each set of variables.

Parameters

- **expr** – the string to compile and evaluate
- **names** – a mapping of local objects which is used as ‘locals’ namespace

Returns the result of evaluation of the expression with env provided by names

flare.utils

Utility functions.

Module Contents**Functions**

<code>unescape(val, maxLength=0)</code>	Unquotes several HTML-quoted characters in a string.
<code>doesEventHitWidgetOrParents(event, widget)</code>	Test if event 'event' hits widget 'widget' (or <i>any</i> of its parents).
<code>doesEventHitWidgetOrChildren(event, widget)</code>	Test if event 'event' hits widget 'widget' (or <i>any</i> of its children).
<code>textToHtml(node, text)</code>	Generates html nodes from text by splitting text into content and into line breaks html5.Br.
<code>parseInt(s, ret=0)</code>	Parses a value as int.
<code>parseFloat(s, ret=0.0)</code>	Parses a value as float.
<code>createWorker(pythonCode, callback=None, errorCallback=None, context={})</code>	Generates and starts a Pyodide Webworker.

`flare.utils.unescape(val, maxLength=0)`

Unquotes several HTML-quoted characters in a string.

Parameters

- **val** (*str*) – The value to be unescaped.
- **maxLength** (*int*) – Cut-off after maxLength characters. A value of 0 means “unlimited”. (default)

Returns The unquoted string.

Return type `str`

`flare.utils.doesEventHitWidgetOrParents(event, widget)`

Test if event ‘event’ hits widget ‘widget’ (or *any* of its parents).

`flare.utils.doesEventHitWidgetOrChildren(event, widget)`

Test if event ‘event’ hits widget ‘widget’ (or *any* of its children).

`flare.utils.textToHtml(node, text)`

Generates html nodes from text by splitting text into content and into line breaks html5.Br.

Parameters

- **node** – The node where the nodes are appended to.
- **text** – The text to be inserted.

`flare.utils.parseInt(s, ret=0)`

Parses a value as int.

`flare.utils.parseFloat(s, ret=0.0)`

Parses a value as float.

`flare.utils.createWorker(pythonCode, callback=None, errorCallback=None, context={})`

Generates and starts a Pyodide Webworker.

def callog(txt=None): result = txt.data.to_py() if “error” in result:

print(result[“error”])

if “msg” in result: print(result[“msg”])

code=

import statistics,time from js import self as js_self for i in range(0,100):

js_self.postMessage(“POST %s”%i)

“”

createWorker(code,callog,callog)

context can take variables, these are like global startparameters

Package Contents

Classes

Cache

Functions

updateConf(other: Dict)

Merges other into conf.

_html5WidgetSetHidden(widget, hidden)

_html5WidgetSetDisabled(widget, disabled)

loadProjectConf(aconf)

Update the default flare config with project configuration.

bindApp(app, injectdata)

Add the app instance as "app" to global config.

Attributes

conf

_super_setHidden

_setHidden

_super_setDisabled

_setDisabled

class flare.Cache

Bases: object

updateStructure(*self, module, structure*)**update**(*self, module, key, data, structure=None*)**lookup**(*self, module, key='current'*)**struct**(*self, module*)**start**(*self, plan, finishHandler=None, failureHandler=None*)**finish**(*self, plan*)**require**(*self, *args*)**invalidate**(*self, *args*)**onDataChanged**(*self, module, key=None, **kwargs*)**request**(*self, *args, finishHandler=None, failureHandler=None*)**flare.conf****flare.updateConf**(*other: Dict*)

Merges other into conf.

flare._html5WidgetSetHidden(*widget, hidden*)**flare._super_setHidden****flare._setHidden****flare._html5WidgetSetDisabled**(*widget, disabled*)**flare._super_setDisabled****flare._setDisabled****flare.loadProjectConf**(*aconf*)

Update the default flare config with project configuration.

flare.bindApp(*app, injectdata*)

Add the app instance as “app” to global config.

1.4.2 webworker_scripts

WARNING! THIS SCRIPTS ARE USED IN A SANDBOX SO ALL DEPENDENCIES SHOULD BE HANDELED HERE!

THIS USES PYODIDE V0.17!

Module Contents

Classes

weblog

Attributes

log

```
class webworker_scripts.weblog
    static info(text)
    static warn(text)
    static error(text)
webworker_scripts.log
```


PYTHON MODULE INDEX

f

- flare, 31
- flare.button, 143
- flare.cache, 143
- flare.config, 144
- flare.debug, 145
- flare.event, 145
- flare.handler, 146
- flare.html5, 32
- flare.html5.core, 32
- flare.html5.svg, 69
- flare.i18n, 147
- flare.icons, 148
- flare.ignite, 149
- flare.input, 151
- flare.intersectionObserver, 151
- flare.log, 152
- flare.network, 153
- flare.observable, 157
- flare.popout, 157
- flare.popup, 158
- flare.priorityqueue, 160
- flare.safeeval, 160
- flare.translations, 109
- flare.translations.de, 109
- flare.translations.en, 109
- flare.utils, 162
- flare.views, 109
- flare.views.helpers, 109
- flare.views.view, 110
- flare.viur, 111
- flare.viur.bones, 111
- flare.viur.bones.base, 111
- flare.viur.bones.boolean, 114
- flare.viur.bones.color, 115
- flare.viur.bones.date, 116
- flare.viur.bones.email, 117
- flare.viur.bones.numeric, 118
- flare.viur.bones.password, 119
- flare.viur.bones.raw, 120
- flare.viur.bones.record, 121
- flare.viur.bones.relational, 122

- flare.viur.bones.select, 126
- flare.viur.bones.spatial, 127
- flare.viur.bones.string, 128
- flare.viur.bones.text, 129
- flare.viur.formatString, 137
- flare.viur.formconf, 138
- flare.viur.formerrors, 138
- flare.viur.forms, 138
- flare.viur.formtooltip, 140
- flare.viur.widgets, 130
- flare.viur.widgets.file, 130
- flare.viur.widgets.htmleditor, 132
- flare.viur.widgets.list, 133
- flare.viur.widgets.tree, 134
- flare.widgets, 142
- flare.widgets.buttonbar, 142

W

- webworker_scripts, 165

Symbols

- `_BoolOp()` (*flare.safeeval.SafeEval* method), 161
- `_Del` (class in *flare.html5*), 95
- `_Del` (class in *flare.html5.core*), 55
- `_WidgetClassWrapper` (class in *flare.html5*), 81
- `_WidgetClassWrapper` (class in *flare.html5.core*), 41
- `_WidgetDataWrapper` (class in *flare.html5*), 81
- `_WidgetDataWrapper` (class in *flare.html5.core*), 41
- `_WidgetStyleWrapper` (class in *flare.html5*), 82
- `_WidgetStyleWrapper` (class in *flare.html5.core*), 42
- `__collectChildren()` (*flare.html5.Widget* method), 87
- `__collectChildren()` (*flare.html5.core.Widget* method), 47
- `__domParser` (in module *flare.html5*), 80
- `__domParser` (in module *flare.html5.core*), 40
- `__getitem__()` (*flare.html5.ColWrapper* method), 105
- `__getitem__()` (*flare.html5.RowWrapper* method), 106
- `__getitem__()` (*flare.html5.Widget* method), 83
- `__getitem__()` (*flare.html5.core.ColWrapper* method), 65
- `__getitem__()` (*flare.html5.core.RowWrapper* method), 66
- `__getitem__()` (*flare.html5.core.Widget* method), 43
- `__iter__()` (*flare.html5.Widget* method), 83
- `__iter__()` (*flare.html5.core.Widget* method), 43
- `__reVarReplacer` (in module *flare.html5*), 108
- `__reVarReplacer` (in module *flare.html5.core*), 68
- `__setitem__()` (*flare.html5.ColWrapper* method), 106
- `__setitem__()` (*flare.html5.Widget* method), 83
- `__setitem__()` (*flare.html5._WidgetDataWrapper* method), 82
- `__setitem__()` (*flare.html5._WidgetStyleWrapper* method), 82
- `__setitem__()` (*flare.html5.core.ColWrapper* method), 66
- `__setitem__()` (*flare.html5.core.Widget* method), 43
- `__setitem__()` (*flare.html5.core._WidgetDataWrapper* method), 42
- `__setitem__()` (*flare.html5.core._WidgetStyleWrapper* method), 42
- `__str__()` (*flare.html5.TextNode* method), 81
- `__str__()` (*flare.html5.Widget* method), 83
- `__str__()` (*flare.html5.core.TextNode* method), 41
- `__str__()` (*flare.html5.core.Widget* method), 43
- `__tags` (in module *flare.html5*), 108
- `__tags` (in module *flare.html5.core*), 68
- `_addEntriesFromSelection()` (*flare.viur.bones.relational.RelationalMultiEditWidget* method), 123
- `_attachSummernote()` (*flare.viur.widgets.htmleditor.HtmlEditor* method), 133
- `_attrAlt` (class in *flare.html5*), 90
- `_attrAlt` (class in *flare.html5.core*), 50
- `_attrAutocomplete` (class in *flare.html5*), 91
- `_attrAutocomplete` (class in *flare.html5.core*), 51
- `_attrAutofocus` (class in *flare.html5*), 90
- `_attrAutofocus` (class in *flare.html5.core*), 50
- `_attrCharset` (class in *flare.html5*), 90
- `_attrCharset` (class in *flare.html5.core*), 50
- `_attrChecked` (class in *flare.html5*), 90
- `_attrChecked` (class in *flare.html5.core*), 50
- `_attrCite` (class in *flare.html5*), 90
- `_attrCite` (class in *flare.html5.core*), 50
- `_attrDatetime` (class in *flare.html5*), 90
- `_attrDatetime` (class in *flare.html5.core*), 50
- `_attrDimensions` (class in *flare.html5*), 93
- `_attrDimensions` (class in *flare.html5.core*), 53
- `_attrDisabled` (class in *flare.html5*), 90
- `_attrDisabled` (class in *flare.html5.core*), 50
- `_attrFor` (class in *flare.html5*), 91
- `_attrFor` (class in *flare.html5.core*), 51
- `_attrForm` (class in *flare.html5*), 90
- `_attrForm` (class in *flare.html5.core*), 50
- `_attrFormhead` (class in *flare.html5*), 92
- `_attrFormhead` (class in *flare.html5.core*), 52
- `_attrHref` (class in *flare.html5*), 92
- `_attrHref` (class in *flare.html5.core*), 52
- `_attrIndeterminate` (class in *flare.html5*), 91
- `_attrIndeterminate` (class in *flare.html5.core*), 51
- `_attrInputs` (class in *flare.html5*), 91
- `_attrInputs` (class in *flare.html5.core*), 51
- `_attrLabel` (class in *flare.html5*), 90
- `_attrLabel` (class in *flare.html5.core*), 50

_attrMedia (class in flare.html5), 93
 _attrMedia (class in flare.html5.core), 53
 _attrMultimedia (class in flare.html5), 93
 _attrMultimedia (class in flare.html5.core), 53
 _attrMultiple (class in flare.html5), 91
 _attrMultiple (class in flare.html5.core), 51
 _attrName (class in flare.html5), 91
 _attrName (class in flare.html5.core), 51
 _attrRel (class in flare.html5), 93
 _attrRel (class in flare.html5.core), 53
 _attrRequired (class in flare.html5), 91
 _attrRequired (class in flare.html5.core), 51
 _attrSize (class in flare.html5), 91
 _attrSize (class in flare.html5.core), 51
 _attrSrc (class in flare.html5), 94
 _attrSrc (class in flare.html5.core), 54
 _attrSvgDimensions (class in flare.html5.svg), 70
 _attrSvgPoints (class in flare.html5.svg), 70
 _attrSvgStyles (class in flare.html5.svg), 71
 _attrSvgTransform (class in flare.html5.svg), 71
 _attrSvgViewBox (class in flare.html5.svg), 69
 _attrSvgXlink (class in flare.html5.svg), 71
 _attrTarget (class in flare.html5), 92
 _attrTarget (class in flare.html5.core), 52
 _attrType (class in flare.html5), 92
 _attrType (class in flare.html5.core), 52
 _attrUsemap (class in flare.html5), 93
 _attrUsemap (class in flare.html5.core), 53
 _attrValue (class in flare.html5), 91
 _attrValue (class in flare.html5.core), 51
 _body (in module flare.html5), 95
 _body (in module flare.html5.core), 55
 _buildTags() (in module flare.html5), 108
 _buildTags() (in module flare.html5.core), 68
 _currentLanguage (in module flare.i18n), 147
 _formatCurrencyValue() (in module flare.viur.bones.numeric), 118
 _genTargetFuncName() (flare.event.EventDispatcher method), 145
 _getAccept() (flare.html5.Input method), 100
 _getAccept() (flare.html5.core.Input method), 60
 _getAccept_attrCharset() (flare.html5.Form method), 100
 _getAccept_attrCharset() (flare.html5.core.Form method), 60
 _getAccesskey() (flare.html5.Widget method), 85
 _getAccesskey() (flare.html5.core.Widget method), 45
 _getAction() (flare.html5.Form method), 99
 _getAction() (flare.html5.core.Form method), 59
 _getAlt() (flare.html5._attrAlt method), 90
 _getAlt() (flare.html5.core._attrAlt method), 50
 _getAsync() (flare.html5.Script method), 104
 _getAsync() (flare.html5.core.Script method), 64
 _getAutocomplete() (flare.html5._attrAutocomplete method), 91
 _getAutocomplete() (flare.html5.core._attrAutocomplete method), 51
 _getAutofocus() (flare.html5._attrAutofocus method), 90
 _getAutofocus() (flare.html5.core._attrAutofocus method), 50
 _getAutoplay() (flare.html5._attrMultimedia method), 93
 _getAutoplay() (flare.html5.core._attrMultimedia method), 53
 _getBadge() (flare.icons.BadgeIcon method), 149
 _getBlockquote() (flare.html5.Blockquote method), 94
 _getBlockquote() (flare.html5.core.Blockquote method), 54
 _getCell() (flare.html5.Table method), 106
 _getCell() (flare.html5.core.Table method), 66
 _getChallenge() (flare.html5.Keygen method), 102
 _getChallenge() (flare.html5.core.Keygen method), 62
 _getCharset() (flare.html5._attrCharset method), 90
 _getCharset() (flare.html5.core._attrCharset method), 50
 _getChecked() (flare.html5._attrChecked method), 90
 _getChecked() (flare.html5.core._attrChecked method), 50
 _getChecked() (flare.ignite.Switch method), 150
 _getCite() (flare.html5._attrCite method), 90
 _getCite() (flare.html5.core._attrCite method), 50
 _getClass() (flare.html5.Widget method), 86
 _getClass() (flare.html5.core.Widget method), 46
 _getCols() (flare.html5.Textarea method), 101
 _getCols() (flare.html5.core.Textarea method), 61
 _getColspan() (flare.html5.Td method), 105
 _getColspan() (flare.html5.core.Td method), 65
 _getContent() (flare.html5.Meta method), 103
 _getContent() (flare.html5.core.Meta method), 63
 _getContenteditable() (flare.html5.Widget method), 85
 _getContenteditable() (flare.html5.core.Widget method), 45
 _getContextmenu() (flare.html5.Widget method), 85
 _getContextmenu() (flare.html5.core.Widget method), 45
 _getControls() (flare.html5._attrMultimedia method), 93
 _getControls() (flare.html5.core._attrMultimedia method), 53
 _getCoords() (flare.html5.Area method), 94
 _getCoords() (flare.html5.core.Area method), 54
 _getCrossorigin() (flare.html5.Img method), 102
 _getCrossorigin() (flare.html5.core.Img method), 62
 _getCx() (flare.html5.svg._attrSvgDimensions method), 70

_getCy() (flare.html5.svg._attrSvgDimensions method), 70
 _getD() (flare.html5.svg.SvgPath method), 72
 _getData() (flare.html5.Widget method), 83
 _getData() (flare.html5.core.Widget method), 43
 _getDatetime() (flare.html5._attrDatetime method), 90
 _getDatetime() (flare.html5.core._attrDatetime method), 50
 _getDefault() (flare.html5.Track method), 106
 _getDefault() (flare.html5.core.Track method), 66
 _getDefaultValues() (in module flare.viur.bones.relational), 122
 _getDefer() (flare.html5.Script method), 104
 _getDefer() (flare.html5.core.Script method), 64
 _getDir() (flare.html5.Widget method), 85
 _getDir() (flare.html5.core.Widget method), 45
 _getDisabled() (flare.html5.TextNode method), 81
 _getDisabled() (flare.html5.Widget method), 84
 _getDisabled() (flare.html5.core.TextNode method), 41
 _getDisabled() (flare.html5.core.Widget method), 44
 _getDownload() (flare.html5.A method), 94
 _getDownload() (flare.html5.core.A method), 54
 _getDraggable() (flare.html5.Widget method), 85
 _getDraggable() (flare.html5.core.Widget method), 45
 _getDropzone() (flare.html5.Widget method), 84
 _getDropzone() (flare.html5.core.Widget method), 44
 _getEnctype() (flare.html5.Form method), 100
 _getEnctype() (flare.html5.core.Form method), 60
 _getFill() (flare.html5.svg._attrSvgStyles method), 71
 _getFor() (flare.html5._attrFor method), 91
 _getFor() (flare.html5.core._attrFor method), 51
 _getForm() (flare.html5._attrForm method), 90
 _getForm() (flare.html5.core._attrForm method), 50
 _getFormaction() (flare.html5._attrFormhead method), 92
 _getFormaction() (flare.html5.core._attrFormhead method), 52
 _getFormenctype() (flare.html5._attrFormhead method), 92
 _getFormenctype() (flare.html5.core._attrFormhead method), 52
 _getFormmethod() (flare.html5._attrFormhead method), 92
 _getFormmethod() (flare.html5.core._attrFormhead method), 52
 _getFormnovalidate() (flare.html5._attrFormhead method), 92
 _getFormnovalidate() (flare.html5.core._attrFormhead method), 52
 _getFormtarget() (flare.html5._attrFormhead method), 92
 _getFormtarget() (flare.html5.core._attrFormhead method), 52
 _getHeight() (flare.html5._attrDimensions method), 93
 _getHeight() (flare.html5.core._attrDimensions method), 53
 _getHeight() (flare.html5.svg._attrSvgDimensions method), 70
 _getHidden() (flare.html5.Widget method), 84
 _getHidden() (flare.html5.core.Widget method), 44
 _getHigh() (flare.html5.Meter method), 103
 _getHigh() (flare.html5.core.Meter method), 63
 _getHref() (flare.html5._attrHref method), 92
 _getHref() (flare.html5.core._attrHref method), 52
 _getHreflang() (flare.html5._attrHref method), 92
 _getHreflang() (flare.html5.core._attrHref method), 52
 _getIcon() (flare.button.Button method), 143
 _getIcon() (flare.html5.Command method), 95
 _getIcon() (flare.html5.core.Command method), 55
 _getIcon() (flare.popout.Popout method), 158
 _getId() (flare.html5.Widget method), 85
 _getId() (flare.html5.core.Widget method), 45
 _getIndeterminate() (flare.html5._attrIndeterminate method), 91
 _getIndeterminate() (flare.html5.core._attrIndeterminate method), 51
 _getIsmap() (flare.html5.Img method), 102
 _getIsmap() (flare.html5.core.Img method), 62
 _getKeytype() (flare.html5.Keygen method), 102
 _getKeytype() (flare.html5.core.Keygen method), 62
 _getKind() (flare.html5.Track method), 106
 _getKind() (flare.html5.core.Track method), 66
 _getLabel() (flare.html5._attrLabel method), 90
 _getLabel() (flare.html5.core._attrLabel method), 50
 _getLang() (flare.html5.Widget method), 84
 _getLang() (flare.html5.core.Widget method), 44
 _getList() (flare.html5.Input method), 100
 _getList() (flare.html5.core.Input method), 60
 _getLoop() (flare.html5._attrMultimedia method), 93
 _getLoop() (flare.html5.core._attrMultimedia method), 53
 _getLow() (flare.html5.Meter method), 103
 _getLow() (flare.html5.core.Meter method), 63
 _getMax() (flare.html5.Input method), 100
 _getMax() (flare.html5.Meter method), 103
 _getMax() (flare.html5.Progress method), 104
 _getMax() (flare.html5.core.Input method), 60
 _getMax() (flare.html5.core.Meter method), 63
 _getMax() (flare.html5.core.Progress method), 64
 _getMaxlength() (flare.html5._attrInputs method), 92
 _getMaxlength() (flare.html5.core._attrInputs method), 52
 _getMedia() (flare.html5._attrMedia method), 93

`_getMedia()` (*flare.html5.core._attrMedia method*), 53
`_getMethod()` (*flare.html5.Form method*), 100
`_getMethod()` (*flare.html5.core.Form method*), 60
`_getMin()` (*flare.html5.Input method*), 100
`_getMin()` (*flare.html5.Meter method*), 103
`_getMin()` (*flare.html5.core.Input method*), 60
`_getMin()` (*flare.html5.core.Meter method*), 63
`_getMultiple()` (*flare.html5._attrMultiple method*), 91
`_getMultiple()` (*flare.html5.core._attrMultiple method*), 51
`_getMuted()` (*flare.html5._attrMultimedia method*), 93
`_getMuted()` (*flare.html5.core._attrMultimedia method*), 53
`_getName()` (*flare.html5._attrName method*), 91
`_getName()` (*flare.html5.core._attrName method*), 51
`_getNovalidate()` (*flare.html5.Form method*), 99
`_getNovalidate()` (*flare.html5.core.Form method*), 59
`_getOpen()` (*flare.html5.Details method*), 105
`_getOpen()` (*flare.html5.Dialog method*), 95
`_getOpen()` (*flare.html5.core.Details method*), 65
`_getOpen()` (*flare.html5.core.Dialog method*), 55
`_getOptimum()` (*flare.html5.Meter method*), 103
`_getOptimum()` (*flare.html5.core.Meter method*), 63
`_getOptions()` (*flare.html5.Select method*), 101
`_getOptions()` (*flare.html5.core.Select method*), 61
`_getPathLength()` (*flare.html5.svg.SvgPath method*), 72
`_getPattern()` (*flare.html5.Input method*), 100
`_getPattern()` (*flare.html5.core.Input method*), 60
`_getPlaceholder()` (*flare.html5._attrInputs method*), 92
`_getPlaceholder()` (*flare.html5.core._attrInputs method*), 52
`_getPlaysinline()` (*flare.html5._attrMultimedia method*), 93
`_getPlaysinline()` (*flare.html5.core._attrMultimedia method*), 53
`_getPoints()` (*flare.html5.svg._attrSvgPoints method*), 70
`_getPoster()` (*flare.html5.Video method*), 106
`_getPoster()` (*flare.html5.core.Video method*), 66
`_getPreload()` (*flare.html5._attrMultimedia method*), 93
`_getPreload()` (*flare.html5.core._attrMultimedia method*), 53
`_getPreserveaspectratio()` (*flare.html5.svg._attrSvgViewBox method*), 70
`_getR()` (*flare.html5.svg._attrSvgDimensions method*), 70
`_getRadiogroup()` (*flare.html5.Command method*), 95
`_getRadiogroup()` (*flare.html5.core.Command method*), 55
`_getReadonly()` (*flare.html5._attrInputs method*), 92
`_getReadonly()` (*flare.html5.core._attrInputs method*), 52
`_getRel()` (*flare.html5._attrRel method*), 94
`_getRel()` (*flare.html5.core._attrRel method*), 54
`_getRequired()` (*flare.html5._attrRequired method*), 91
`_getRequired()` (*flare.html5.core._attrRequired method*), 51
`_getRole()` (*flare.html5.Widget method*), 86
`_getRole()` (*flare.html5.core.Widget method*), 46
`_getRows()` (*flare.html5.Textarea method*), 101
`_getRows()` (*flare.html5.core.Textarea method*), 61
`_getRowspan()` (*flare.html5.Td method*), 105
`_getRowspan()` (*flare.html5.Tr method*), 105
`_getRowspan()` (*flare.html5.core.Td method*), 65
`_getRowspan()` (*flare.html5.core.Tr method*), 65
`_getRx()` (*flare.html5.svg._attrSvgDimensions method*), 70
`_getRy()` (*flare.html5.svg._attrSvgDimensions method*), 70
`_getSandbox()` (*flare.html5.Iframe method*), 101
`_getSandbox()` (*flare.html5.core.Iframe method*), 61
`_getScoped()` (*flare.html5.Style method*), 105
`_getScoped()` (*flare.html5.core.Style method*), 65
`_getSeamless()` (*flare.html5.Iframe method*), 101
`_getSeamless()` (*flare.html5.core.Iframe method*), 61
`_getSelected()` (*flare.html5.Option method*), 101
`_getSelected()` (*flare.html5.core.Option method*), 61
`_getSelectedIndex()` (*flare.html5.Select method*), 101
`_getSelectedIndex()` (*flare.html5.core.Select method*), 61
`_getShape()` (*flare.html5.Area method*), 94
`_getShape()` (*flare.html5.core.Area method*), 54
`_getSize()` (*flare.html5._attrSize method*), 91
`_getSize()` (*flare.html5.core._attrSize method*), 51
`_getSizes()` (*flare.html5.Link method*), 102
`_getSizes()` (*flare.html5.core.Link method*), 62
`_getSpellcheck()` (*flare.html5.Widget method*), 84
`_getSpellcheck()` (*flare.html5.core.Widget method*), 44
`_getSrc()` (*flare.html5._attrSrc method*), 94
`_getSrc()` (*flare.html5.core._attrSrc method*), 54
`_getSrcdoc()` (*flare.html5.Iframe method*), 101
`_getSrcdoc()` (*flare.html5.core.Iframe method*), 61
`_getSrclang()` (*flare.html5.Track method*), 106
`_getSrclang()` (*flare.html5.core.Track method*), 66
`_getStep()` (*flare.html5.Input method*), 100
`_getStep()` (*flare.html5.core.Input method*), 60
`_getStroke()` (*flare.html5.svg._attrSvgStyles method*), 71
`_getStyle()` (*flare.html5.Widget method*), 86
`_getStyle()` (*flare.html5.core.Widget method*), 46
`_getSvgTransform()` (*flare.html5.svg.SvgG method*), 72
`_getTabIndex()` (*flare.html5.Widget method*), 83

_getTabIndex() (flare.html5.core.Widget method), 43
 _getTarget() (flare.html5._attrTarget method), 92
 _getTarget() (flare.html5.core._attrTarget method), 52
 _getTargetfuncName() (flare.html5.Widget method), 83
 _getTargetfuncName() (flare.html5.core.Widget method), 43
 _getText() (flare.button.Button method), 143
 _getText() (flare.html5.TextNode method), 81
 _getText() (flare.html5.core.TextNode method), 41
 _getText() (flare.popout.Popout method), 158
 _getTitle() (flare.html5.Widget method), 83
 _getTitle() (flare.html5.core.Widget method), 43
 _getTransform() (flare.html5.svg._attrSvgTransform method), 71
 _getTranslate() (flare.html5.Widget method), 83
 _getTranslate() (flare.html5.core.Widget method), 43
 _getType() (flare.html5._attrType method), 93
 _getType() (flare.html5.core._attrType method), 53
 _getUsemap() (flare.html5._attrUsemap method), 93
 _getUsemap() (flare.html5.core._attrUsemap method), 53
 _getValue() (flare.html5._attrValue method), 91
 _getValue() (flare.html5.core._attrValue method), 51
 _getValue() (flare.viur.widgets.htmleditor.HtmlEditor method), 133
 _getVersion() (flare.html5.svg.Svg method), 71
 _getViewbox() (flare.html5.svg._attrSvgViewBox method), 69
 _getWidth() (flare.html5._attrDimensions method), 93
 _getWidth() (flare.html5.core._attrDimensions method), 53
 _getWidth() (flare.html5.svg._attrSvgDimensions method), 70
 _getWrap() (flare.html5.Textarea method), 101
 _getWrap() (flare.html5.core.Textarea method), 61
 _getX() (flare.html5.svg._attrSvgDimensions method), 70
 _getX1() (flare.html5.svg._attrSvgPoints method), 70
 _getX2() (flare.html5.svg._attrSvgPoints method), 70
 _getXlinkhref() (flare.html5.svg._attrSvgXlink method), 71
 _getXmlns() (flare.html5.svg.Svg method), 71
 _getY() (flare.html5.svg._attrSvgDimensions method), 70
 _getY1() (flare.html5.svg._attrSvgPoints method), 70
 _getY2() (flare.html5.svg._attrSvgPoints method), 71
 _head (in module flare.html5), 101
 _head (in module flare.html5.core), 61
 _html5WidgetSetDisabled() (in module flare), 164
 _html5WidgetSetHidden() (in module flare), 164
 _leafTag (flare.html5.Area attribute), 94
 _leafTag (flare.html5.Br attribute), 96
 _leafTag (flare.html5.Embed attribute), 96
 _leafTag (flare.html5.Hr attribute), 97
 _leafTag (flare.html5.Img attribute), 102
 _leafTag (flare.html5.Input attribute), 100
 _leafTag (flare.html5.Link attribute), 102
 _leafTag (flare.html5.Meta attribute), 103
 _leafTag (flare.html5.Param attribute), 104
 _leafTag (flare.html5.Source attribute), 104
 _leafTag (flare.html5.Track attribute), 106
 _leafTag (flare.html5.Widget attribute), 82
 _leafTag (flare.html5.core.Area attribute), 54
 _leafTag (flare.html5.core.Br attribute), 56
 _leafTag (flare.html5.core.Embed attribute), 56
 _leafTag (flare.html5.core.Hr attribute), 57
 _leafTag (flare.html5.core.Img attribute), 62
 _leafTag (flare.html5.core.Input attribute), 60
 _leafTag (flare.html5.core.Link attribute), 62
 _leafTag (flare.html5.core.Meta attribute), 63
 _leafTag (flare.html5.core.Param attribute), 64
 _leafTag (flare.html5.core.Source attribute), 64
 _leafTag (flare.html5.core.Track attribute), 66
 _leafTag (flare.html5.core.Widget attribute), 42
 _leafTag (flare.icons.Icon attribute), 149
 _leafTag (flare.icons.SvgIcon attribute), 148
 _lngMap (in module flare.i18n), 148
 _namespace (flare.html5.Widget attribute), 82
 _namespace (flare.html5.core.Widget attribute), 42
 _namespace (flare.html5.svg.SvgWidget attribute), 71
 _onRequestFailure() (flare.cache.Plan method), 144
 _onRequestSuccess() (flare.cache.Plan method), 144
 _request() (flare.handler.SyncHandler method), 147
 _requestFailed() (flare.handler.requestHandler method), 146
 _runtimeTranslations (in module flare.i18n), 147
 _setAccept() (flare.html5.Input method), 100
 _setAccept() (flare.html5.core.Input method), 60
 _setAccept_attrCharset() (flare.html5.Form method), 100
 _setAccept_attrCharset() (flare.html5.core.Form method), 60
 _setAccesskey() (flare.html5.Widget method), 85
 _setAccesskey() (flare.html5.core.Widget method), 45
 _setAction() (flare.html5.Form method), 99
 _setAction() (flare.html5.core.Form method), 59
 _setActionname() (flare.viur.forms.ViurForm method), 139
 _setAlt() (flare.html5._attrAlt method), 90
 _setAlt() (flare.html5.core._attrAlt method), 50
 _setAsync() (flare.html5.Script method), 104
 _setAsync() (flare.html5.core.Script method), 64
 _setAutocomplete() (flare.html5._attrAutocomplete method), 91
 _setAutocomplete() (flare.html5.core._attrAutocomplete method), 51

_setAutofocus() (flare.html5._attrAutofocus method), 90
 _setAutofocus() (flare.html5.core._attrAutofocus method), 50
 _setAutoplay() (flare.html5._attrMultimedia method), 93
 _setAutoplay() (flare.html5.core._attrMultimedia method), 53
 _setBadge() (flare.icons.BadgeIcon method), 149
 _setBlockquote() (flare.html5.Blockquote method), 94
 _setBlockquote() (flare.html5.core.Blockquote method), 54
 _setBonename() (flare.viur.forms.ViurFormBone method), 139
 _setChallenge() (flare.html5.Keygen method), 102
 _setChallenge() (flare.html5.core.Keygen method), 62
 _setCharset() (flare.html5._attrCharset method), 90
 _setCharset() (flare.html5.core._attrCharset method), 50
 _setChecked() (flare.html5._attrChecked method), 91
 _setChecked() (flare.html5.core._attrChecked method), 51
 _setChecked() (flare.ignite.Switch method), 150
 _setCite() (flare.html5._attrCite method), 90
 _setCite() (flare.html5.core._attrCite method), 50
 _setClass() (flare.html5.Widget method), 86
 _setClass() (flare.html5.core.Widget method), 46
 _setCols() (flare.html5.Textarea method), 101
 _setCols() (flare.html5.core.Textarea method), 61
 _setColspan() (flare.html5.Td method), 105
 _setColspan() (flare.html5.core.Td method), 65
 _setContent() (flare.html5.Meta method), 103
 _setContent() (flare.html5.core.Meta method), 63
 _setContenteditable() (flare.html5.Widget method), 85
 _setContenteditable() (flare.html5.core.Widget method), 45
 _setContextmenu() (flare.html5.Widget method), 85
 _setContextmenu() (flare.html5.core.Widget method), 45
 _setControls() (flare.html5._attrMultimedia method), 93
 _setControls() (flare.html5.core._attrMultimedia method), 53
 _setCoords() (flare.html5.Area method), 94
 _setCoords() (flare.html5.core.Area method), 54
 _setCrossorigin() (flare.html5.Img method), 102
 _setCrossorigin() (flare.html5.core.Img method), 62
 _setCx() (flare.html5.svg._attrSvgDimensions method), 70
 _setCy() (flare.html5.svg._attrSvgDimensions method), 70
 _setD() (flare.html5.svg.SvgPath method), 72
 _setDatetime() (flare.html5._attrDatetime method), 90
 _setDatetime() (flare.html5.core._attrDatetime method), 50
 _setDefault() (flare.html5.Track method), 106
 _setDefault() (flare.html5.core.Track method), 66
 _setDefer() (flare.html5.Script method), 104
 _setDefer() (flare.html5.core.Script method), 64
 _setDir() (flare.html5.Widget method), 85
 _setDir() (flare.html5.core.Widget method), 45
 _setDisabled (in module flare), 164
 _setDisabled() (flare.html5.TextNode method), 81
 _setDisabled() (flare.html5.Widget method), 84
 _setDisabled() (flare.html5.core.TextNode method), 41
 _setDisabled() (flare.html5.core.Widget method), 44
 _setDisabled() (flare.viur.bones.text.TextEditWidget method), 129
 _setDisabled() (flare.viur.formtooltip.ToolTip method), 140
 _setDownload() (flare.html5.A method), 94
 _setDownload() (flare.html5.core.A method), 54
 _setDraggable() (flare.html5.Widget method), 85
 _setDraggable() (flare.html5.core.Widget method), 45
 _setDropzone() (flare.html5.Widget method), 84
 _setDropzone() (flare.html5.core.Widget method), 44
 _setEnctype() (flare.html5.Form method), 100
 _setEnctype() (flare.html5.core.Form method), 60
 _setFallback() (flare.icons.Icon method), 149
 _setFill() (flare.html5.svg._attrSvgStyles method), 71
 _setFor() (flare.html5._attrFor method), 91
 _setFor() (flare.html5.core._attrFor method), 51
 _setForm() (flare.html5._attrForm method), 90
 _setForm() (flare.html5.core._attrForm method), 50
 _setFormaction() (flare.html5._attrFormhead method), 92
 _setFormaction() (flare.html5.core._attrFormhead method), 52
 _setFormenctype() (flare.html5._attrFormhead method), 92
 _setFormenctype() (flare.html5.core._attrFormhead method), 52
 _setFormmethod() (flare.html5._attrFormhead method), 92
 _setFormmethod() (flare.html5.core._attrFormhead method), 52
 _setFormname() (flare.viur.forms.ViurForm method), 139
 _setFormnovalidate() (flare.html5._attrFormhead method), 92
 _setFormnovalidate() (flare.html5.core._attrFormhead method), 52
 _setFormtarget() (flare.html5._attrFormhead method), 92
 _setFormtarget() (flare.html5.core._attrFormhead method), 92

method), 52
 _setHeight() (flare.html5._attrDimensions method), 93
 _setHeight() (flare.html5.core._attrDimensions method), 53
 _setHeight() (flare.html5.svg._attrSvgDimensions method), 70
 _setHidden (in module flare), 164
 _setHidden() (flare.html5.Widget method), 84
 _setHidden() (flare.html5.core.Widget method), 44
 _setHide() (flare.viur.forms.ViurFormBone method), 140
 _setHigh() (flare.html5.Meter method), 103
 _setHigh() (flare.html5.core.Meter method), 63
 _setHref() (flare.html5._attrHref method), 92
 _setHref() (flare.html5.core._attrHref method), 52
 _setHreflang() (flare.html5._attrHref method), 92
 _setHreflang() (flare.html5.core._attrHref method), 52
 _setIcon() (flare.button.Button method), 143
 _setIcon() (flare.html5.Command method), 95
 _setIcon() (flare.html5.core.Command method), 55
 _setIcon() (flare.popout.Popout method), 158
 _setId() (flare.html5.Widget method), 86
 _setId() (flare.html5.core.Widget method), 46
 _setIndeterminate() (flare.html5._attrIndeterminate method), 91
 _setIndeterminate() (flare.html5.core._attrIndeterminate method), 51
 _setIsmap() (flare.html5.Img method), 102
 _setIsmap() (flare.html5.core.Img method), 62
 _setKeytype() (flare.html5.Keygen method), 102
 _setKeytype() (flare.html5.core.Keygen method), 62
 _setKind() (flare.html5.Track method), 106
 _setKind() (flare.html5.core.Track method), 66
 _setLabel() (flare.html5._attrLabel method), 90
 _setLabel() (flare.html5.core._attrLabel method), 50
 _setLabel() (flare.viur.forms.ViurFormBone method), 139
 _setLang() (flare.html5.Widget method), 84
 _setLang() (flare.html5.core.Widget method), 44
 _setList() (flare.html5.Input method), 100
 _setList() (flare.html5.core.Input method), 60
 _setLoop() (flare.html5._attrMultimedia method), 93
 _setLoop() (flare.html5.core._attrMultimedia method), 53
 _setLow() (flare.html5.Meter method), 103
 _setLow() (flare.html5.core.Meter method), 63
 _setMax() (flare.html5.Input method), 100
 _setMax() (flare.html5.Meter method), 103
 _setMax() (flare.html5.Progress method), 104
 _setMax() (flare.html5.core.Input method), 60
 _setMax() (flare.html5.core.Meter method), 63
 _setMax() (flare.html5.core.Progress method), 64
 _setMaxlength() (flare.html5._attrInputs method), 92
 _setMaxlength() (flare.html5.core._attrInputs method), 52
 _setMedia() (flare.html5._attrMedia method), 93
 _setMedia() (flare.html5.core._attrMedia method), 53
 _setMethod() (flare.html5.Form method), 100
 _setMethod() (flare.html5.core.Form method), 60
 _setMin() (flare.html5.Input method), 100
 _setMin() (flare.html5.Meter method), 103
 _setMin() (flare.html5.core.Input method), 60
 _setMin() (flare.html5.core.Meter method), 63
 _setModulename() (flare.viur.forms.ViurForm method), 139
 _setMultiple() (flare.html5._attrMultiple method), 91
 _setMultiple() (flare.html5.core._attrMultiple method), 51
 _setMuted() (flare.html5._attrMultimedia method), 93
 _setMuted() (flare.html5.core._attrMultimedia method), 53
 _setName() (flare.html5._attrName method), 91
 _setName() (flare.html5.core._attrName method), 51
 _setNovalidate() (flare.html5.Form method), 99
 _setNovalidate() (flare.html5.core.Form method), 59
 _setOpen() (flare.html5.Details method), 105
 _setOpen() (flare.html5.Dialog method), 95
 _setOpen() (flare.html5.core.Details method), 65
 _setOpen() (flare.html5.core.Dialog method), 55
 _setOptimum() (flare.html5.Meter method), 103
 _setOptimum() (flare.html5.core.Meter method), 63
 _setPathLength() (flare.html5.svg.SvgPath method), 72
 _setPattern() (flare.html5.Input method), 100
 _setPattern() (flare.html5.core.Input method), 60
 _setPlaceholder() (flare.html5._attrInputs method), 92
 _setPlaceholder() (flare.html5.core._attrInputs method), 52
 _setPlaceholder() (flare.viur.forms.ViurFormBone method), 139
 _setPlaysinline() (flare.html5._attrMultimedia method), 93
 _setPlaysinline() (flare.html5.core._attrMultimedia method), 53
 _setPoints() (flare.html5.svg._attrSvgPoints method), 70
 _setPoster() (flare.html5.Video method), 106
 _setPoster() (flare.html5.core.Video method), 66
 _setPreload() (flare.html5._attrMultimedia method), 93
 _setPreload() (flare.html5.core._attrMultimedia method), 53
 _setPreserveaspectratio() (flare.html5.svg._attrSvgViewBox method), 70

`_setR()` (*flare.html5.svg._attrSvgDimensions* method), 70
`_setRadiogroup()` (*flare.html5.Command* method), 95
`_setRadiogroup()` (*flare.html5.core.Command* method), 55
`_setReadOnly()` (*flare.html5._attrInputs* method), 92
`_setReadOnly()` (*flare.html5.core._attrInputs* method), 52
`_setRel()` (*flare.html5._attrRel* method), 94
`_setRel()` (*flare.html5.core._attrRel* method), 54
`_setRequired()` (*flare.html5._attrRequired* method), 91
`_setRequired()` (*flare.html5.core._attrRequired* method), 51
`_setRole()` (*flare.html5.Widget* method), 86
`_setRole()` (*flare.html5.core.Widget* method), 46
`_setRows()` (*flare.html5.Textarea* method), 101
`_setRows()` (*flare.html5.core.Textarea* method), 61
`_setRowspan()` (*flare.html5.Td* method), 105
`_setRowspan()` (*flare.html5.Tr* method), 105
`_setRowspan()` (*flare.html5.core.Td* method), 65
`_setRowspan()` (*flare.html5.core.Tr* method), 65
`_setRx()` (*flare.html5.svg._attrSvgDimensions* method), 70
`_setRy()` (*flare.html5.svg._attrSvgDimensions* method), 70
`_setSandbox()` (*flare.html5.Iframe* method), 101
`_setSandbox()` (*flare.html5.core.Iframe* method), 61
`_setScoped()` (*flare.html5.Style* method), 105
`_setScoped()` (*flare.html5.core.Style* method), 65
`_setSeamless()` (*flare.html5.Iframe* method), 102
`_setSeamless()` (*flare.html5.core.Iframe* method), 62
`_setSelected()` (*flare.html5.Option* method), 101
`_setSelected()` (*flare.html5.core.Option* method), 61
`_setShape()` (*flare.html5.Area* method), 94
`_setShape()` (*flare.html5.core.Area* method), 54
`_setSize()` (*flare.html5._attrSize* method), 91
`_setSize()` (*flare.html5.core._attrSize* method), 51
`_setSizes()` (*flare.html5.Link* method), 102
`_setSizes()` (*flare.html5.core.Link* method), 62
`_setSpellcheck()` (*flare.html5.Widget* method), 84
`_setSpellcheck()` (*flare.html5.core.Widget* method), 44
`_setSrc()` (*flare.html5._attrSrc* method), 94
`_setSrc()` (*flare.html5.core._attrSrc* method), 54
`_setSrcdoc()` (*flare.html5.Iframe* method), 101
`_setSrcdoc()` (*flare.html5.core.Iframe* method), 61
`_setSrcLang()` (*flare.html5.Track* method), 106
`_setSrcLang()` (*flare.html5.core.Track* method), 66
`_setStep()` (*flare.html5.Input* method), 100
`_setStep()` (*flare.html5.core.Input* method), 60
`_setStroke()` (*flare.html5.svg._attrSvgStyles* method), 71
`_setSvgTransform()` (*flare.html5.svg.SvgG* method), 72
`_setTabIndex()` (*flare.html5.Widget* method), 84
`_setTabIndex()` (*flare.html5.core.Widget* method), 44
`_setTarget()` (*flare.html5._attrTarget* method), 92
`_setTarget()` (*flare.html5.core._attrTarget* method), 52
`_setText()` (*flare.button.Button* method), 143
`_setText()` (*flare.html5.TextNode* method), 81
`_setText()` (*flare.html5.core.TextNode* method), 41
`_setText()` (*flare.popout.Popout* method), 158
`_setTitle()` (*flare.html5.Widget* method), 83
`_setTitle()` (*flare.html5.core.Widget* method), 43
`_setTitle()` (*flare.icons.Icon* method), 149
`_setTitle()` (*flare.icons.SvgIcon* method), 148
`_setTransform()` (*flare.html5.svg._attrSvgTransform* method), 71
`_setTranslate()` (*flare.html5.Widget* method), 83
`_setTranslate()` (*flare.html5.core.Widget* method), 43
`_setType()` (*flare.html5._attrType* method), 93
`_setType()` (*flare.html5.core._attrType* method), 53
`_setUsemaph()` (*flare.html5._attrUsemaph* method), 93
`_setUsemaph()` (*flare.html5.core._attrUsemaph* method), 53
`_setValue()` (*flare.html5._attrValue* method), 91
`_setValue()` (*flare.html5.core._attrValue* method), 51
`_setValue()` (*flare.icons.Icon* method), 149
`_setValue()` (*flare.icons.SvgIcon* method), 148
`_setValue()` (*flare.viur.forms.ViurFormBone* method), 140
`_setValue()` (*flare.viur.widgets.htmleditor.HtmlEditor* method), 133
`_setVersion()` (*flare.html5.svg.Svg* method), 71
`_setViewbox()` (*flare.html5.svg._attrSvgViewBox* method), 69
`_setWidth()` (*flare.html5._attrDimensions* method), 93
`_setWidth()` (*flare.html5.core._attrDimensions* method), 53
`_setWidth()` (*flare.html5.svg._attrSvgDimensions* method), 70
`_setWrap()` (*flare.html5.Textarea* method), 101
`_setWrap()` (*flare.html5.core.Textarea* method), 61
`_setX()` (*flare.html5.svg._attrSvgDimensions* method), 70
`_setX1()` (*flare.html5.svg._attrSvgPoints* method), 70
`_setX2()` (*flare.html5.svg._attrSvgPoints* method), 70
`_setXlinkhref()` (*flare.html5.svg._attrSvgXlink* method), 71
`_setXmlns()` (*flare.html5.svg.Svg* method), 71
`_setY()` (*flare.html5.svg._attrSvgDimensions* method), 70
`_setY1()` (*flare.html5.svg._attrSvgPoints* method), 70
`_setY2()` (*flare.html5.svg._attrSvgPoints* method), 71
`super_setDisabled` (in module *flare*), 164
`super_setHidden` (in module *flare*), 164
`_tagName` (*flare.html5.A* attribute), 94
`_tagName` (*flare.html5.Abbbr* attribute), 95

_tagName (flare.html5.Address attribute), 95
 _tagName (flare.html5.Area attribute), 94
 _tagName (flare.html5.Article attribute), 95
 _tagName (flare.html5.Aside attribute), 95
 _tagName (flare.html5.Audio attribute), 94
 _tagName (flare.html5.B attribute), 96
 _tagName (flare.html5.Bdi attribute), 96
 _tagName (flare.html5.Bdo attribute), 94
 _tagName (flare.html5.Blockquote attribute), 94
 _tagName (flare.html5.Br attribute), 96
 _tagName (flare.html5.Button attribute), 99
 _tagName (flare.html5.Canvas attribute), 95
 _tagName (flare.html5.Caption attribute), 96
 _tagName (flare.html5.Cite attribute), 96
 _tagName (flare.html5.Code attribute), 96
 _tagName (flare.html5.Command attribute), 95
 _tagName (flare.html5.Datalist attribute), 96
 _tagName (flare.html5.Dd attribute), 103
 _tagName (flare.html5.Details attribute), 104
 _tagName (flare.html5.Dfn attribute), 96
 _tagName (flare.html5.Dialog attribute), 95
 _tagName (flare.html5.Div attribute), 96
 _tagName (flare.html5.Dl attribute), 103
 _tagName (flare.html5.Dt attribute), 103
 _tagName (flare.html5.Em attribute), 96
 _tagName (flare.html5.Embed attribute), 96
 _tagName (flare.html5.Fieldset attribute), 99
 _tagName (flare.html5.Figcaption attribute), 97
 _tagName (flare.html5.Figure attribute), 97
 _tagName (flare.html5.Footer attribute), 97
 _tagName (flare.html5.Form attribute), 99
 _tagName (flare.html5.H1 attribute), 97
 _tagName (flare.html5.H2 attribute), 97
 _tagName (flare.html5.H3 attribute), 97
 _tagName (flare.html5.H4 attribute), 97
 _tagName (flare.html5.H5 attribute), 97
 _tagName (flare.html5.H6 attribute), 97
 _tagName (flare.html5.Header attribute), 97
 _tagName (flare.html5.Hr attribute), 97
 _tagName (flare.html5.I attribute), 97
 _tagName (flare.html5.Iframe attribute), 101
 _tagName (flare.html5.Img attribute), 102
 _tagName (flare.html5.Input attribute), 100
 _tagName (flare.html5.Ins attribute), 102
 _tagName (flare.html5.Kdb attribute), 98
 _tagName (flare.html5.Keygen attribute), 102
 _tagName (flare.html5.Label attribute), 100
 _tagName (flare.html5.Legend attribute), 98
 _tagName (flare.html5.Li attribute), 102
 _tagName (flare.html5.Link attribute), 102
 _tagName (flare.html5.Map attribute), 103
 _tagName (flare.html5.Mark attribute), 98
 _tagName (flare.html5.Menu attribute), 103
 _tagName (flare.html5.Meta attribute), 103
 _tagName (flare.html5.Meter attribute), 103
 _tagName (flare.html5.Nav attribute), 104
 _tagName (flare.html5.Noscript attribute), 98
 _tagName (flare.html5.Object attribute), 104
 _tagName (flare.html5.Ol attribute), 102
 _tagName (flare.html5.Optgroup attribute), 100
 _tagName (flare.html5.Option attribute), 101
 _tagName (flare.html5.Output attribute), 101
 _tagName (flare.html5.P attribute), 98
 _tagName (flare.html5.Param attribute), 104
 _tagName (flare.html5.Progress attribute), 104
 _tagName (flare.html5.Q attribute), 104
 _tagName (flare.html5.Rq attribute), 98
 _tagName (flare.html5.Rt attribute), 98
 _tagName (flare.html5.Ruby attribute), 98
 _tagName (flare.html5.S attribute), 98
 _tagName (flare.html5.Samp attribute), 98
 _tagName (flare.html5.Script attribute), 104
 _tagName (flare.html5.Section attribute), 98
 _tagName (flare.html5.Select attribute), 101
 _tagName (flare.html5.Small attribute), 98
 _tagName (flare.html5.Source attribute), 104
 _tagName (flare.html5.Span attribute), 104
 _tagName (flare.html5.Strong attribute), 99
 _tagName (flare.html5.Style attribute), 105
 _tagName (flare.html5.Sub attribute), 99
 _tagName (flare.html5.Summary attribute), 105
 _tagName (flare.html5.Summary attribute), 99
 _tagName (flare.html5.Sup attribute), 99
 _tagName (flare.html5.Table attribute), 106
 _tagName (flare.html5.Tbody attribute), 105
 _tagName (flare.html5.Td attribute), 105
 _tagName (flare.html5.Template attribute), 107
 _tagName (flare.html5.Textarea attribute), 101
 _tagName (flare.html5.Th attribute), 105
 _tagName (flare.html5.Thead attribute), 105
 _tagName (flare.html5.Time attribute), 106
 _tagName (flare.html5.Tr attribute), 105
 _tagName (flare.html5.Track attribute), 106
 _tagName (flare.html5.U attribute), 99
 _tagName (flare.html5.Ul attribute), 102
 _tagName (flare.html5.Var attribute), 99
 _tagName (flare.html5.Video attribute), 106
 _tagName (flare.html5.Wbr attribute), 99
 _tagName (flare.html5.Widget attribute), 82
 _tagName (flare.html5._Del attribute), 95
 _tagName (flare.html5.core.A attribute), 54
 _tagName (flare.html5.core.Abbbr attribute), 55
 _tagName (flare.html5.core.Address attribute), 55
 _tagName (flare.html5.core.Area attribute), 54
 _tagName (flare.html5.core.Article attribute), 55
 _tagName (flare.html5.core.Aside attribute), 55
 _tagName (flare.html5.core.Audio attribute), 54
 _tagName (flare.html5.core.B attribute), 56

`_tagName` (*flare.html5.core.Bdi* attribute), 56
`_tagName` (*flare.html5.core.Bdo* attribute), 54
`_tagName` (*flare.html5.core.Blockquote* attribute), 54
`_tagName` (*flare.html5.core.Br* attribute), 56
`_tagName` (*flare.html5.core.Button* attribute), 59
`_tagName` (*flare.html5.core.Canvas* attribute), 55
`_tagName` (*flare.html5.core.Caption* attribute), 56
`_tagName` (*flare.html5.core.Cite* attribute), 56
`_tagName` (*flare.html5.core.Code* attribute), 56
`_tagName` (*flare.html5.core.Command* attribute), 55
`_tagName` (*flare.html5.core.Datalist* attribute), 56
`_tagName` (*flare.html5.core.Dd* attribute), 63
`_tagName` (*flare.html5.core.Details* attribute), 64
`_tagName` (*flare.html5.core.Dfn* attribute), 56
`_tagName` (*flare.html5.core.Dialog* attribute), 55
`_tagName` (*flare.html5.core.Div* attribute), 56
`_tagName` (*flare.html5.core.Dl* attribute), 63
`_tagName` (*flare.html5.core.Dt* attribute), 63
`_tagName` (*flare.html5.core.Em* attribute), 56
`_tagName` (*flare.html5.core.Embed* attribute), 56
`_tagName` (*flare.html5.core.Fieldset* attribute), 59
`_tagName` (*flare.html5.core.Figcaption* attribute), 57
`_tagName` (*flare.html5.core.Figure* attribute), 57
`_tagName` (*flare.html5.core.Footer* attribute), 57
`_tagName` (*flare.html5.core.Form* attribute), 59
`_tagName` (*flare.html5.core.H1* attribute), 57
`_tagName` (*flare.html5.core.H2* attribute), 57
`_tagName` (*flare.html5.core.H3* attribute), 57
`_tagName` (*flare.html5.core.H4* attribute), 57
`_tagName` (*flare.html5.core.H5* attribute), 57
`_tagName` (*flare.html5.core.H6* attribute), 57
`_tagName` (*flare.html5.core.Header* attribute), 57
`_tagName` (*flare.html5.core.Hr* attribute), 57
`_tagName` (*flare.html5.core.I* attribute), 57
`_tagName` (*flare.html5.core.Iframe* attribute), 61
`_tagName` (*flare.html5.core.Img* attribute), 62
`_tagName` (*flare.html5.core.Input* attribute), 60
`_tagName` (*flare.html5.core.Ins* attribute), 62
`_tagName` (*flare.html5.core.Kdb* attribute), 58
`_tagName` (*flare.html5.core.Keygen* attribute), 62
`_tagName` (*flare.html5.core.Label* attribute), 60
`_tagName` (*flare.html5.core.Legend* attribute), 58
`_tagName` (*flare.html5.core.Li* attribute), 62
`_tagName` (*flare.html5.core.Link* attribute), 62
`_tagName` (*flare.html5.core.Map* attribute), 63
`_tagName` (*flare.html5.core.Mark* attribute), 58
`_tagName` (*flare.html5.core.Menu* attribute), 63
`_tagName` (*flare.html5.core.Meta* attribute), 63
`_tagName` (*flare.html5.core.Meter* attribute), 63
`_tagName` (*flare.html5.core.Nav* attribute), 64
`_tagName` (*flare.html5.core.Noscript* attribute), 58
`_tagName` (*flare.html5.core.Object* attribute), 64
`_tagName` (*flare.html5.core.Ol* attribute), 62
`_tagName` (*flare.html5.core.Optgroup* attribute), 60
`_tagName` (*flare.html5.core.Option* attribute), 61
`_tagName` (*flare.html5.core.Output* attribute), 61
`_tagName` (*flare.html5.core.P* attribute), 58
`_tagName` (*flare.html5.core.Param* attribute), 64
`_tagName` (*flare.html5.core.Progress* attribute), 64
`_tagName` (*flare.html5.core.Q* attribute), 64
`_tagName` (*flare.html5.core.Rq* attribute), 58
`_tagName` (*flare.html5.core.Rt* attribute), 58
`_tagName` (*flare.html5.core.Ruby* attribute), 58
`_tagName` (*flare.html5.core.S* attribute), 58
`_tagName` (*flare.html5.core.Samp* attribute), 58
`_tagName` (*flare.html5.core.Script* attribute), 64
`_tagName` (*flare.html5.core.Section* attribute), 58
`_tagName` (*flare.html5.core.Select* attribute), 61
`_tagName` (*flare.html5.core.Small* attribute), 58
`_tagName` (*flare.html5.core.Source* attribute), 64
`_tagName` (*flare.html5.core.Span* attribute), 64
`_tagName` (*flare.html5.core.Strong* attribute), 59
`_tagName` (*flare.html5.core.Style* attribute), 65
`_tagName` (*flare.html5.core.Sub* attribute), 59
`_tagName` (*flare.html5.core.Summary* attribute), 65
`_tagName` (*flare.html5.core.Summary* attribute), 59
`_tagName` (*flare.html5.core.Sup* attribute), 59
`_tagName` (*flare.html5.core.Table* attribute), 66
`_tagName` (*flare.html5.core.Tbody* attribute), 65
`_tagName` (*flare.html5.core.Td* attribute), 65
`_tagName` (*flare.html5.core.Template* attribute), 67
`_tagName` (*flare.html5.core.Textarea* attribute), 61
`_tagName` (*flare.html5.core.Th* attribute), 65
`_tagName` (*flare.html5.core.Thead* attribute), 65
`_tagName` (*flare.html5.core.Time* attribute), 66
`_tagName` (*flare.html5.core.Tr* attribute), 65
`_tagName` (*flare.html5.core.Track* attribute), 66
`_tagName` (*flare.html5.core.U* attribute), 59
`_tagName` (*flare.html5.core.Ul* attribute), 62
`_tagName` (*flare.html5.core.Var* attribute), 59
`_tagName` (*flare.html5.core.Video* attribute), 66
`_tagName` (*flare.html5.core.Wbr* attribute), 59
`_tagName` (*flare.html5.core.Widget* attribute), 42
`_tagName` (*flare.html5.core._Del* attribute), 55
`_tagName` (*flare.html5.svg.Svg* attribute), 71
`_tagName` (*flare.html5.svg.SvgCircle* attribute), 71
`_tagName` (*flare.html5.svg.SvgEllipse* attribute), 71
`_tagName` (*flare.html5.svg.SvgG* attribute), 72
`_tagName` (*flare.html5.svg.SvgImage* attribute), 72
`_tagName` (*flare.html5.svg.SvgLine* attribute), 72
`_tagName` (*flare.html5.svg.SvgPath* attribute), 72
`_tagName` (*flare.html5.svg.SvgPolygon* attribute), 72
`_tagName` (*flare.html5.svg.SvgPolyline* attribute), 72
`_tagName` (*flare.html5.svg.SvgRect* attribute), 72
`_tagName` (*flare.html5.svg.SvgText* attribute), 72
`_updateElem()` (*flare.html5._WidgetClassWrapper* method), 81

`_updateElem()` (*flare.html5.core._WidgetClassWrapper* method), 41

A

`A` (class in *flare.html5*), 94

`A` (class in *flare.html5.core*), 54

`Abbr` (class in *flare.html5*), 95

`Abbr` (class in *flare.html5.core*), 55

`acceptSelection()` (*flare.viur.widgets.list.ListSelection* method), 134

`actionFailed()` (*flare.viur.forms.ViurForm* method), 139

`actionSuccess()` (*flare.viur.forms.ViurForm* method), 139

`activateSelection()` (*flare.viur.widgets.list.ListSelection* method), 134

`addButton()` (*flare.widgets.buttonbar.ButtonBar* method), 142

`addClass()` (*flare.html5.core.Widget* method), 47

`addClass()` (*flare.html5.Widget* method), 87

`addEntry()` (*flare.viur.bones.base.BaseMultiEditWidget* method), 113

`addEntry()` (*flare.viur.bones.relational.FileMultiEditDirect* method), 125

`addEventListener()` (*flare.html5.core.Widget* method), 42

`addEventListener()` (*flare.html5.Widget* method), 82

`additionalDropAreas()` (*flare.viur.widgets.tree.TreeItemWidget* method), 135

`addRequest()` (*flare.network.requestGroup* method), 156

`Address` (class in *flare.html5*), 95

`Address` (class in *flare.html5.core*), 55

`addTranslation()` (in module *flare.i18n*), 148

`addView()` (in module *flare.views.helpers*), 110

`Alert` (class in *flare.popup*), 159

`append()` (*flare.html5._WidgetClassWrapper* method), 81

`append()` (*flare.html5.core._WidgetClassWrapper* method), 41

`appendChild()` (*flare.html5.core.Widget* method), 47

`appendChild()` (*flare.html5.Widget* method), 87

`applyFilter()` (*flare.widgets.buttonbar.ButtonBarSearch* method), 142

`Area` (class in *flare.html5*), 94

`Area` (class in *flare.html5.core*), 54

`Article` (class in *flare.html5*), 95

`Article` (class in *flare.html5.core*), 55

`Aside` (class in *flare.html5*), 95

`Aside` (class in *flare.html5.core*), 55

`Audio` (class in *flare.html5*), 94

`Audio` (class in *flare.html5.core*), 54

`autoIdCounter` (*flare.html5.core.Label* attribute), 60

`autoIdCounter` (*flare.html5.Label* attribute), 100

B

`B` (class in *flare.html5*), 95

`B` (class in *flare.html5.core*), 55

`BadgeIcon` (class in *flare.icons*), 149

`BaseBone` (class in *flare.viur.bones.base*), 114

`BaseEditWidget` (class in *flare.viur.bones.base*), 112

`BaseLanguageEditWidget` (class in *flare.viur.bones.base*), 113

`BaseMultiEditWidget` (class in *flare.viur.bones.base*), 113

`BaseMultiEditWidgetEntry` (class in *flare.viur.bones.base*), 113

`BaseMultiViewWidget` (class in *flare.viur.bones.base*), 113

`BaseViewWidget` (class in *flare.viur.bones.base*), 112

`Bdi` (class in *flare.html5*), 96

`Bdi` (class in *flare.html5.core*), 56

`Bdo` (class in *flare.html5*), 94

`Bdo` (class in *flare.html5.core*), 54

`bindApp()` (in module *flare*), 164

`Blockquote` (class in *flare.html5*), 94

`Blockquote` (class in *flare.html5.core*), 54

`blur()` (*flare.html5.core.Widget* method), 49

`blur()` (*flare.html5.Widget* method), 89

`Body()` (in module *flare.html5*), 95

`Body()` (in module *flare.html5.core*), 55

`BodyCls` (class in *flare.html5*), 95

`BodyCls` (class in *flare.html5.core*), 55

`BoneSelector` (in module *flare.viur*), 141

`boneWidget()` (*flare.viur.bones.base.BaseBone* method), 114

`BooleanBone` (class in *flare.viur.bones.boolean*), 115

`BooleanEditWidget` (class in *flare.viur.bones.boolean*), 114

`BooleanViewWidget` (class in *flare.viur.bones.boolean*), 115

`Br` (class in *flare.html5*), 96

`Br` (class in *flare.html5.core*), 56

`BreadcrumbNodeWidget` (class in *flare.viur.widgets.tree*), 136

`BrowserLeafWidget` (class in *flare.viur.widgets.tree*), 136

`BrowserNodeWidget` (class in *flare.viur.widgets.tree*), 136

`buildDescription()` (*flare.viur.widgets.tree.TreeItemWidget* method), 135

`buildForm()` (*flare.viur.forms.ViurForm* method), 139

`buildInternalForm()` (*flare.viur.forms.ViurForm* method), 139

`buildListSelection()` (*flare.viur.widgets.list.ListSelection* method),

134
 buildSelectDescr() (*flare.handler.requestHandler* method), 146
 buildTranslations() (*in module flare.i18n*), 148
 buildWidget() (*flare.viur.widgets.list.SkellistItem* method), 134
 Button (*class in flare.button*), 143
 Button (*class in flare.html5*), 99
 Button (*class in flare.html5.core*), 59
 ButtonBar (*class in flare.widgets.buttonbar*), 142
 ButtonBarButton (*class in flare.widgets.buttonbar*), 142
 ButtonBarSearch (*class in flare.widgets.buttonbar*), 142
 buttonClicked() (*flare.widgets.buttonbar.ButtonBar* method), 142

C

Cache (*class in flare*), 164
 Cache (*class in flare.cache*), 143
 call() (*flare.network.requestGroup* method), 156
 callNode() (*flare.safeeval.SafeEval* method), 161
 canHandle() (*flare.viur.widgets.file.FileWidget* static method), 132
 canHandle() (*flare.viur.widgets.list.ListWidget* static method), 134
 canHandle() (*flare.viur.widgets.tree.TreeBrowserWidget* static method), 137
 canHandle() (*flare.viur.widgets.tree.TreeWidget* static method), 136
 Canvas (*class in flare.html5*), 95
 Canvas (*class in flare.html5.core*), 55
 Caption (*class in flare.html5*), 96
 Caption (*class in flare.html5.core*), 56
 changeListeners (*flare.network.NetworkService* attribute), 155
 Check (*class in flare.ignite*), 150
 checkFor() (*flare.viur.bones.boolean.BooleanBone* static method), 115
 checkFor() (*flare.viur.bones.color.ColorBone* static method), 116
 checkFor() (*flare.viur.bones.date.DateBone* static method), 117
 checkFor() (*flare.viur.bones.email.EmailBone* static method), 118
 checkFor() (*flare.viur.bones.numeric.NumericBone* static method), 119
 checkFor() (*flare.viur.bones.password.PasswordBone* static method), 119
 checkFor() (*flare.viur.bones.raw.RawBone* static method), 120
 checkFor() (*flare.viur.bones.record.RecordBone* static method), 121

checkFor() (*flare.viur.bones.relational.FileBone* static method), 125
 checkFor() (*flare.viur.bones.relational.FileDirectBone* static method), 125
 checkFor() (*flare.viur.bones.relational.HierarchyBone* static method), 123
 checkFor() (*flare.viur.bones.relational.RelationalBone* static method), 123
 checkFor() (*flare.viur.bones.relational.TreeDirBone* static method), 124
 checkFor() (*flare.viur.bones.relational.TreeItemBone* static method), 123
 checkFor() (*flare.viur.bones.select.SelectMultipleBone* static method), 127
 checkFor() (*flare.viur.bones.select.SelectSingleBone* static method), 127
 checkFor() (*flare.viur.bones.spatial.SpatialBone* static method), 128
 checkFor() (*flare.viur.bones.string.StringBone* static method), 129
 checkFor() (*flare.viur.bones.text.TextBone* static method), 130
 children() (*flare.html5.core.TextNode* method), 41
 children() (*flare.html5.core.Widget* method), 49
 children() (*flare.html5.TextNode* method), 81
 children() (*flare.html5.Widget* method), 89
 Cite (*class in flare.html5*), 96
 Cite (*class in flare.html5.core*), 56
 clear() (*flare.html5._WidgetClassWrapper* method), 81
 clear() (*flare.html5.core._WidgetClassWrapper* method), 41
 clear() (*flare.html5.core.Table* method), 66
 clear() (*flare.html5.Table* method), 106
 clear() (*flare.network.NetworkService* method), 156
 close() (*flare.popup.Popup* method), 159
 Code (*class in flare.html5*), 96
 Code (*class in flare.html5.core*), 56
 collectBoneErrors() (*in module flare.viur.formerrors*), 138
 ColorBone (*class in flare.viur.bones.color*), 116
 ColorEditWidget (*class in flare.viur.bones.color*), 115
 ColorViewWidget (*class in flare.viur.bones.color*), 116
 ColWrapper (*class in flare.html5*), 105
 ColWrapper (*class in flare.html5.core*), 65
 Command (*class in flare.html5*), 95
 Command (*class in flare.html5.core*), 55
 compareNode() (*flare.safeeval.SafeEval* method), 161
 compile() (*flare.safeeval.SafeEval* method), 161
 conf (*in module flare*), 164
 conf (*in module flare.config*), 145
 conf (*in module flare.views*), 111
 conf (*in module flare.viur*), 141
 conf (*in module flare.viur.formconf*), 138
 Confirm (*class in flare.popup*), 159

createFormErrorMessage()
 (flare.viur.forms.ViurForm method), 139
 createFormSuccessMessage()
 (flare.viur.forms.ViurForm method), 139
 createWidget() (flare.viur.bones.base.BaseEditWidget
 method), 112
 createWidget() (flare.viur.bones.boolean.BooleanEditWidget
 method), 114
 createWidget() (flare.viur.bones.color.ColorEditWidget
 method), 115
 createWidget() (flare.viur.bones.date.DateEditWidget
 method), 116
 createWidget() (flare.viur.bones.numeric.NumericEditWidget
 method), 118
 createWidget() (flare.viur.bones.password.PasswordEditWidget
 method), 119
 createWidget() (flare.viur.bones.raw.RawEditWidget
 method), 120
 createWidget() (flare.viur.bones.record.RecordEditWidget
 method), 121
 createWidget() (flare.viur.bones.relational.FileEditWidget
 method), 124
 createWidget() (flare.viur.bones.relational.FileEditWidget
 method), 125
 createWidget() (flare.viur.bones.relational.RelationalEditWidget
 method), 122
 createWidget() (flare.viur.bones.select.SelectMultipleEditWidget
 method), 126
 createWidget() (flare.viur.bones.select.SelectSingleEditWidget
 method), 126
 createWidget() (flare.viur.bones.spatial.SpatialEditWidget
 method), 127
 createWidget() (flare.viur.bones.string.StringEditWidget
 method), 128
 createWidget() (flare.viur.bones.text.TextEditWidget
 method), 129
 createWorker() (in module flare.utils), 163

D
 Datalist (class in flare.html5), 96
 Datalist (class in flare.html5.core), 56
 DateBone (class in flare.viur.bones.date), 117
 DateEditWidget (class in flare.viur.bones.date), 116
 DateViewWidget (class in flare.viur.bones.date), 117
 Dd (class in flare.html5), 103
 Dd (class in flare.html5.core), 63
 debug() (in module flare.debug), 145
 debugElement() (in module flare.debug), 145
 decode() (flare.network.NetworkService static method),
 155
 defaultFailureHandler
 (flare.network.NetworkService attribute),
 155
 DeferredCall (class in flare.network), 154

 Details (class in flare.html5), 104
 Details (class in flare.html5.core), 64
 Dfn (class in flare.html5), 96
 Dfn (class in flare.html5.core), 56
 Dialog (class in flare.html5), 95
 Dialog (class in flare.html5.core), 55
 disable() (flare.html5.core.Widget method), 43
 disable() (flare.html5.Widget method), 83
 disable() (flare.viur.widgets.htmleditor.HtmlEditor
 method), 133
 disableDragMarkers()
 (flare.viur.widgets.tree.TreeItemWidget
 method), 135
 DisplayDelegateSelector (in module flare.viur), 141
 DisplayStringHandler() (in module flare.viur), 141
 displayStringHandler() (in module
 flare.viur.formatString), 137
 Div (class in flare.html5), 96
 Div (class in flare.html5.core), 56
 Dl (class in flare.html5), 103
 Dl (class in flare.html5.core), 63
 document (in module flare.html5), 80
 document (in module flare.html5.core), 40
 doesEventHitWidgetOrChildren() (in module
 flare.html5), 107
 doesEventHitWidgetOrChildren() (in module
 flare.html5.core), 67
 doesEventHitWidgetOrChildren() (in module
 flare.utils), 162
 doesEventHitWidgetOrParents() (in module
 flare.html5), 107
 doesEventHitWidgetOrParents() (in module
 flare.html5.core), 67
 doesEventHitWidgetOrParents() (in module
 flare.utils), 162
 doFetch() (flare.network.NetworkService method), 156
 domConvertEncodedText() (in module flare.html5), 80
 domConvertEncodedText() (in module
 flare.html5.core), 40
 domCreateAttribute() (in module flare.html5), 80
 domCreateAttribute() (in module flare.html5.core),
 40
 domCreateElement() (in module flare.html5), 80
 domCreateElement() (in module flare.html5.core), 40
 domCreateTextNode() (in module flare.html5), 80
 domCreateTextNode() (in module flare.html5.core), 40
 domElementFromPoint() (in module flare.html5), 80
 domElementFromPoint() (in module flare.html5.core),
 40
 domGetElementById() (in module flare.html5), 80
 domGetElementById() (in module flare.html5.core), 40
 domGetElementsByTagName() (in module flare.html5),
 80

`domGetElementsByTagName()` (in module `flare.html5.core`), 40
`doSearch()` (`flare.viur.widgets.file.Search` method), 130
`download()` (`flare.viur.widgets.file.FilePreviewImage` method), 131
`drop()` (`flare.popup.Alert` method), 159
`drop()` (`flare.popup.Confirm` method), 159
`Dt` (class in `flare.html5`), 103
`Dt` (class in `flare.html5.core`), 63

E

`editWidget()` (`flare.viur.bones.base.BaseBone` method), 114
`editWidgetFactory` (`flare.viur.bones.base.BaseBone` attribute), 114
`editWidgetFactory` (`flare.viur.bones.boolean.BooleanBone` attribute), 115
`editWidgetFactory` (`flare.viur.bones.color.ColorBone` attribute), 116
`editWidgetFactory` (`flare.viur.bones.date.DateBone` attribute), 117
`editWidgetFactory` (`flare.viur.bones.email.EmailBone` attribute), 118
`editWidgetFactory` (`flare.viur.bones.numeric.NumericBone` attribute), 119
`editWidgetFactory` (`flare.viur.bones.password.PasswordBone` attribute), 119
`editWidgetFactory` (`flare.viur.bones.raw.RawBone` attribute), 120
`editWidgetFactory` (`flare.viur.bones.record.RecordBone` attribute), 121
`editWidgetFactory` (`flare.viur.bones.relational.FileBone` attribute), 125
`editWidgetFactory` (`flare.viur.bones.relational.FileDirectBone` attribute), 125
`editWidgetFactory` (`flare.viur.bones.relational.RelationalBone` attribute), 123
`editWidgetFactory` (`flare.viur.bones.select.SelectMultipleBone` attribute), 127
`editWidgetFactory` (`flare.viur.bones.select.SelectSingleBone` attribute), 127
`editWidgetFactory` (`flare.viur.bones.spatial.SpatialBone` attribute), 128
`editWidgetFactory` (`flare.viur.bones.string.StringBone` attribute), 129
`editWidgetFactory` (`flare.viur.bones.text.TextBone` attribute), 130
`Em` (class in `flare.html5`), 96
`Em` (class in `flare.html5.core`), 56
`EmailBone` (class in `flare.viur.bones.email`), 117
`EmailEditWidget` (class in `flare.viur.bones.email`), 117
`EmailViewWidget` (class in `flare.viur.bones.email`), 117
`Embed` (class in `flare.html5`), 96
`Embed` (class in `flare.html5.core`), 56

`emit()` (`flare.log.JSConsoleHandler` method), 153
`Empty` (`flare.viur.bones.base.ReadFromClientErrorSeverity` attribute), 112
`enable()` (`flare.html5.core.Widget` method), 43
`enable()` (`flare.html5.Widget` method), 83
`enable()` (`flare.viur.widgets.htmleditor.HtmlEditor` method), 133
`entryFactory` (`flare.viur.bones.base.BaseMultiEditWidget` attribute), 113
`entryFactory` (`flare.viur.bones.relational.FileMultiEditDirectWidget` attribute), 124
`EntryIcon()` (`flare.viur.widgets.file.FileLeafWidget` method), 131
`EntryIcon()` (`flare.viur.widgets.tree.TreeItemWidget` method), 135
`EntryIcon()` (`flare.viur.widgets.tree.TreeLeafWidget` method), 136
`entryTemplate` (`flare.viur.bones.select.SelectMultipleEditWidget` attribute), 126
`entryTemplate` (`flare.viur.bones.select.SelectSingleEditWidget` attribute), 126
`error()` (`webworker_scripts.weblog` static method), 165
`errorWidget()` (`flare.viur.bones.base.BaseBone` method), 114
`evalStringHandler()` (in module `flare.viur.formatString`), 137
`EventDispatcher` (class in `flare.event`), 145
`execute()` (`flare.safeeval.SafeEval` method), 161
`extend()` (`flare.html5._WidgetClassWrapper` method), 81
`extend()` (`flare.html5.core._WidgetClassWrapper` method), 41

F

`fastGrid()` (`flare.ignite.Table` method), 151
`Fieldset` (class in `flare.html5`), 99
`Fieldset` (class in `flare.html5.core`), 59
`Figcaption` (class in `flare.html5`), 96
`Figcaption` (class in `flare.html5.core`), 56
`Figure` (class in `flare.html5`), 97
`Figure` (class in `flare.html5.core`), 57
`FileBone` (class in `flare.viur.bones.relational`), 125
`FileDirectBone` (class in `flare.viur.bones.relational`), 125
`FileEditDirectWidget` (class in `flare.viur.bones.relational`), 124
`FileEditWidget` (class in `flare.viur.bones.relational`), 125
`FileImagePopup` (class in `flare.viur.widgets.file`), 131
`FileLeafWidget` (class in `flare.viur.widgets.file`), 131
`FileMultiEditDirectWidget` (class in `flare.viur.bones.relational`), 124
`FileNodeWidget` (class in `flare.viur.widgets.file`), 132
`FilePreviewImage` (class in `flare.viur.widgets.file`), 131

FileViewWidget (class in *flare.viur.bones.relational*), 124
 FileWidget (class in *flare.viur.widgets.file*), 132
 filter() (*flare.handler.ListHandler* method), 146
 finish() (*flare.Cache* method), 164
 finish() (*flare.cache.Cache* method), 144
 finish() (*flare.cache.Plan* method), 144
 fire() (*flare.event.EventDispatcher* method), 146
 flare
 module, 31
 flare.button
 module, 143
 flare.cache
 module, 143
 flare.config
 module, 144
 flare.debug
 module, 145
 flare.event
 module, 145
 flare.handler
 module, 146
 flare.html5
 module, 32
 flare.html5.core
 module, 32
 flare.html5.svg
 module, 69
 flare.i18n
 module, 147
 flare.icons
 module, 148
 flare.ignite
 module, 149
 flare.input
 module, 151
 flare.intersectionObserver
 module, 151
 flare.log
 module, 152
 flare.network
 module, 153
 flare.observable
 module, 157
 flare.popout
 module, 157
 flare.popup
 module, 158
 flare.priorityqueue
 module, 160
 flare.safeeval
 module, 160
 flare.translations
 module, 109
 flare.translations.de
 module, 109
 flare.translations.en
 module, 109
 flare.utils
 module, 162
 flare.views
 module, 109
 flare.views.helpers
 module, 109
 flare.views.view
 module, 110
 flare.viur
 module, 111
 flare.viur.bones
 module, 111
 flare.viur.bones.base
 module, 111
 flare.viur.bones.boolean
 module, 114
 flare.viur.bones.color
 module, 115
 flare.viur.bones.date
 module, 116
 flare.viur.bones.email
 module, 117
 flare.viur.bones.numeric
 module, 118
 flare.viur.bones.password
 module, 119
 flare.viur.bones.raw
 module, 120
 flare.viur.bones.record
 module, 121
 flare.viur.bones.relational
 module, 122
 flare.viur.bones.select
 module, 126
 flare.viur.bones.spatial
 module, 127
 flare.viur.bones.string
 module, 128
 flare.viur.bones.text
 module, 129
 flare.viur.formatString
 module, 137
 flare.viur.formconf
 module, 138
 flare.viur.formerrors
 module, 138
 flare.viur.forms
 module, 138
 flare.viur.formtooltip
 module, 140

flare.viur.widgets
 module, 130
 flare.viur.widgets.file
 module, 130
 flare.viur.widgets.htmleditor
 module, 132
 flare.viur.widgets.list
 module, 133
 flare.viur.widgets.tree
 module, 134
 flare.widgets
 module, 142
 flare.widgets.buttonbar
 module, 142
 FlareLogRecord (class in flare.log), 152
 focus() (flare.html5.core.Widget method), 49
 focus() (flare.html5.Widget method), 89
 focus() (flare.viur.widgets.file.Search method), 131
 Footer (class in flare.html5), 97
 Footer (class in flare.html5.core), 57
 Form (class in flare.html5), 99
 Form (class in flare.html5.core), 59
 formatString() (in module flare.viur), 141
 formatString() (in module flare.viur.formatString), 137
 formatStringHandler() (in module flare.viur.formatString), 137
 fromHTML() (flare.html5.core.Widget method), 49
 fromHTML() (flare.html5.Widget method), 89
 fromHTML() (in module flare.html5), 108
 fromHTML() (in module flare.html5.core), 68

G

generateView() (in module flare.views.helpers), 110
 genReqStr() (flare.handler.SyncHandler method), 147
 genReqStr() (flare.network.NetworkService static method), 155
 getChildKey() (flare.viur.widgets.file.FileWidget method), 132
 getCurrentAmount() (flare.handler.ListHandler method), 146
 getDescrFromValue() (flare.handler.requestHandler method), 146
 getIcon() (flare.icons.SvgIcon method), 148
 getImagePreview() (in module flare.viur.widgets.file), 130
 getKey() (in module flare.html5), 107
 getKey() (in module flare.html5.core), 67
 getLanguage() (in module flare.i18n), 148
 getLogger() (in module flare.log), 153
 getMessage() (flare.log.FlareLogRecord method), 152
 getRowCount() (flare.html5.core.Table method), 66
 getRowCount() (flare.html5.Table method), 106

getState() (flare.observable.StateHandler method), 157
 getState() (flare.views.StateHandler method), 111
 getUriHashAsObject() (in module flare.network), 156
 getUriHashAsString() (in module flare.network), 156

H

H1 (class in flare.html5), 97
 H1 (class in flare.html5.core), 57
 H2 (class in flare.html5), 97
 H2 (class in flare.html5.core), 57
 H3 (class in flare.html5), 97
 H3 (class in flare.html5.core), 57
 H4 (class in flare.html5), 97
 H4 (class in flare.html5.core), 57
 H5 (class in flare.html5), 97
 H5 (class in flare.html5.core), 57
 H6 (class in flare.html5), 97
 H6 (class in flare.html5.core), 57
 handleErrors() (flare.viur.forms.ViurForm method), 139
 hasClass() (flare.html5.core.Widget method), 47
 hasClass() (flare.html5.Widget method), 87
 Head() (in module flare.html5), 101
 Head() (in module flare.html5.core), 61
 HeadCls (class in flare.html5), 101
 HeadCls (class in flare.html5.core), 61
 Header (class in flare.html5), 97
 Header (class in flare.html5.core), 57
 hide() (flare.html5.core.Widget method), 46
 hide() (flare.html5.Widget method), 86
 HierarchyBone (class in flare.viur.bones.relational), 123
 host (flare.network.NetworkService attribute), 155
 Hr (class in flare.html5), 97
 Hr (class in flare.html5.core), 57
 HtmlAst (class in flare.html5), 108
 HtmlAst (class in flare.html5.core), 68
 HtmlEditor (class in flare.viur.widgets.htmleditor), 133
 htmlExpressionEvaluator (in module flare.config), 145
 htmlExpressionEvaluator (in module flare.html5), 80
 htmlExpressionEvaluator (in module flare.html5.core), 40
 HTTPRequest (class in flare.network), 154

I

I (class in flare.html5), 97
 I (class in flare.html5.core), 57
 Icon (class in flare.icons), 149
 Iframe (class in flare.html5), 101
 Iframe (class in flare.html5.core), 61
 Img (class in flare.html5), 102
 Img (class in flare.html5.core), 62

- info() (*webworker_scripts.weblog static method*), 165
 initSources (*flare.viur.widgets.htmleditor.HtmlEditor attribute*), 133
 initWidget() (*flare.views.view.ViewWidget method*), 110
 Input (*class in flare.html5*), 100
 Input (*class in flare.html5.core*), 60
 Input (*class in flare.ignite*), 150
 Input (*class in flare.input*), 151
 Ins (*class in flare.html5*), 102
 Ins (*class in flare.html5.core*), 62
 insert() (*flare.html5._WidgetClassWrapper method*), 81
 insert() (*flare.html5.core._WidgetClassWrapper method*), 41
 insert() (*flare.priorityqueue.PriorityQueue method*), 160
 insert() (*flare.viur.PriorityQueue method*), 141
 insertAfter() (*flare.html5.core.Widget method*), 47
 insertAfter() (*flare.html5.Widget method*), 87
 insertBefore() (*flare.html5.core.Widget method*), 47
 insertBefore() (*flare.html5.Widget method*), 87
 IntersectionObserver (*class in flare.intersectionObserver*), 151
 Invalid (*flare.viur.bones.base.ReadFromClientErrorSeverity attribute*), 112
 invalidate() (*flare.Cache method*), 164
 invalidate() (*flare.cache.Cache method*), 144
 InvalidatesOther (*flare.viur.bones.base.ReadFromClientErrorSeverity attribute*), 112
 InvalidBoneValueException, 141
 isArrowDown() (*in module flare.html5*), 107
 isArrowDown() (*in module flare.html5.core*), 67
 isArrowLeft() (*in module flare.html5*), 107
 isArrowLeft() (*in module flare.html5.core*), 67
 isArrowRight() (*in module flare.html5*), 107
 isArrowRight() (*in module flare.html5.core*), 67
 isArrowUp() (*in module flare.html5*), 107
 isArrowUp() (*in module flare.html5.core*), 67
 isChildOf() (*flare.html5.core.Widget method*), 47
 isChildOf() (*flare.html5.Widget method*), 87
 isControl() (*in module flare.html5*), 107
 isControl() (*in module flare.html5.core*), 67
 isEscape() (*in module flare.html5*), 107
 isEscape() (*in module flare.html5.core*), 67
 isHidden() (*flare.html5.core.Widget method*), 46
 isHidden() (*flare.html5.Widget method*), 86
 isMeta() (*in module flare.html5*), 107
 isMeta() (*in module flare.html5.core*), 67
 isOkay() (*flare.network.NetworkService static method*), 155
 isParentOf() (*flare.html5.core.Widget method*), 47
 isParentOf() (*flare.html5.Widget method*), 87
 isReturn() (*in module flare.html5*), 107
 isReturn() (*in module flare.html5.core*), 67
 isShift() (*in module flare.html5*), 107
 isShift() (*in module flare.html5.core*), 67
 isSuitableFor() (*flare.viur.widgets.htmleditor.TextInsertImageAction static method*), 133
 isVisible() (*flare.html5.core.Widget method*), 46
 isVisible() (*flare.html5.Widget method*), 86
 Item (*class in flare.ignite*), 150
- ## J
- JSConsoleHandler (*class in flare.log*), 152
 jsObserver (*flare.intersectionObserver.IntersectionObserver attribute*), 151
- ## K
- Kdb (*class in flare.html5*), 98
 Kdb (*class in flare.html5.core*), 58
 Keygen (*class in flare.html5*), 102
 Keygen (*class in flare.html5.core*), 62
 kickoff() (*flare.network.NetworkService method*), 155
- ## L
- Label (*class in flare.html5*), 100
 Label (*class in flare.html5.core*), 60
 Label (*class in flare.ignite*), 150
 labelWidget() (*flare.viur.bones.base.BaseBone method*), 114
 languageEditWidgetFactory (*flare.viur.bones.base.BaseBone attribute*), 114
 languageViewWidgetFactory (*flare.viur.bones.base.BaseBone attribute*), 114
 leafWidget (*flare.viur.widgets.file.FileWidget attribute*), 132
 leafWidget (*flare.viur.widgets.tree.TreeBrowserWidget attribute*), 137
 leafWidget (*flare.viur.widgets.tree.TreeWidget attribute*), 136
 Legend (*class in flare.html5*), 98
 Legend (*class in flare.html5.core*), 58
 Li (*class in flare.html5*), 102
 Li (*class in flare.html5.core*), 62
 Link (*class in flare.html5*), 102
 Link (*class in flare.html5.core*), 62
 ListHandler (*class in flare.handler*), 146
 listNode() (*flare.safeeval.SafeEval method*), 161
 ListSelection (*class in flare.viur.widgets.list*), 134
 ListWidget (*class in flare.viur.widgets.list*), 133
 lngDe (*in module flare.translations*), 109
 lngDe (*in module flare.translations.de*), 109
 lngEn (*in module flare.translations*), 109
 lngEn (*in module flare.translations.en*), 109
 loadProjectConf() (*in module flare*), 164

`loadView()` (*flare.views.view.View method*), 110
`log` (*in module webworker_scripts*), 165
`loggers` (*in module flare.log*), 152
`lookup()` (*flare.Cache method*), 164
`lookup()` (*flare.cache.Cache method*), 144

M

`Map` (*class in flare.html5*), 103
`Map` (*class in flare.html5.core*), 63
`Mark` (*class in flare.html5*), 98
`Mark` (*class in flare.html5.core*), 58
`markDraggedElement()`
 (*flare.viur.widgets.tree.TreeItemWidget method*), 135
`Menu` (*class in flare.html5*), 103
`Menu` (*class in flare.html5.core*), 63
`Meta` (*class in flare.html5*), 103
`Meta` (*class in flare.html5.core*), 63
`Meter` (*class in flare.html5*), 103
`Meter` (*class in flare.html5.core*), 63
`module`
 `flare`, 31
 `flare.button`, 143
 `flare.cache`, 143
 `flare.config`, 144
 `flare.debug`, 145
 `flare.event`, 145
 `flare.handler`, 146
 `flare.html5`, 32
 `flare.html5.core`, 32
 `flare.html5.svg`, 69
 `flare.i18n`, 147
 `flare.icons`, 148
 `flare.ignite`, 149
 `flare.input`, 151
 `flare.intersectionObserver`, 151
 `flare.log`, 152
 `flare.network`, 153
 `flare.observable`, 157
 `flare.popout`, 157
 `flare.popup`, 158
 `flare.priorityqueue`, 160
 `flare.safeeval`, 160
 `flare.translations`, 109
 `flare.translations.de`, 109
 `flare.translations.en`, 109
 `flare.utils`, 162
 `flare.views`, 109
 `flare.views.helpers`, 109
 `flare.views.view`, 110
 `flare.viur`, 111
 `flare.viur.bones`, 111
 `flare.viur.bones.base`, 111
 `flare.viur.bones.boolean`, 114

`flare.viur.bones.color`, 115
 `flare.viur.bones.date`, 116
 `flare.viur.bones.email`, 117
 `flare.viur.bones.numeric`, 118
 `flare.viur.bones.password`, 119
 `flare.viur.bones.raw`, 120
 `flare.viur.bones.record`, 121
 `flare.viur.bones.relational`, 122
 `flare.viur.bones.select`, 126
 `flare.viur.bones.spatial`, 127
 `flare.viur.bones.string`, 128
 `flare.viur.bones.text`, 129
 `flare.viur.formatString`, 137
 `flare.viur.formconf`, 138
 `flare.viur.formerrors`, 138
 `flare.viur.forms`, 138
 `flare.viur.formtooltip`, 140
 `flare.viur.widgets`, 130
 `flare.viur.widgets.file`, 130
 `flare.viur.widgets.htmleditor`, 132
 `flare.viur.widgets.list`, 133
 `flare.viur.widgets.tree`, 134
 `flare.widgets`, 142
 `flare.widgets.buttonbar`, 142
 `webworker_scripts`, 165

`ModuleWidgetSelector` (*in module flare.viur*), 141

`multiEditWidgetFactory`
 (*flare.viur.bones.base.BaseBone attribute*), 114

`multiEditWidgetFactory`
 (*flare.viur.bones.relational.FileDirectBone attribute*), 125

`multiEditWidgetFactory`
 (*flare.viur.bones.relational.RelationalBone attribute*), 123

`multiEditWidgetFactory`
 (*flare.viur.bones.select.SelectMultipleBone attribute*), 127

`multiViewWidgetFactory`
 (*flare.viur.bones.base.BaseBone attribute*), 114

N

`Nav` (*class in flare.html5*), 104

`Nav` (*class in flare.html5.core*), 64

`NetworkService` (*class in flare.network*), 154

`NiceError()` (*in module flare.network*), 154

`NiceErrorAndThen()` (*in module flare.network*), 154

`nodeWidget` (*flare.viur.widgets.file.FileWidget attribute*), 132

`nodeWidget` (*flare.viur.widgets.tree.TreeBrowserWidget attribute*), 137

`nodeWidget` (*flare.viur.widgets.tree.TreeWidget attribute*), 136

- Noscript (class in flare.html5), 98
 Noscript (class in flare.html5.core), 58
 notifyChange() (flare.network.NetworkService static method), 155
 NotSet (flare.viur.bones.base.ReadFromClientErrorSeverity attribute), 112
 NumericBone (class in flare.viur.bones.numeric), 119
 NumericEditWidget (class in flare.viur.bones.numeric), 118
 NumericViewWidget (class in flare.viur.bones.numeric), 119
- ## O
- Object (class in flare.html5), 104
 Object (class in flare.html5.core), 64
 ObservableValue (class in flare.observable), 157
 observableWidgets (flare.intersectionObserver.IntersectionObserver attribute), 151
 observe() (flare.intersectionObserver.IntersectionObserver method), 151
 Ol (class in flare.html5), 102
 Ol (class in flare.html5.core), 62
 onAcceptSelectionChanged() (flare.viur.widgets.list.ListSelection method), 134
 onAcceptSelectionChanged() (flare.viur.widgets.list.ListWidget method), 134
 onActiveButtonChanged() (flare.viur.widgets.list.ListSelection method), 134
 onActiveButtonChanged() (flare.widgets.buttonbar.ButtonBar method), 142
 onActiveButtonChanged() (flare.widgets.buttonbar.ButtonBarButton method), 142
 onActiveButtonChanged() (flare.widgets.buttonbar.ButtonBarSearch method), 142
 onActiveSelectionChanged() (flare.viur.widgets.list.ListSelection method), 134
 onActiveSelectionChanged() (flare.viur.widgets.list.SkellistItem method), 134
 onActiveViewChanged() (flare.views.view.View method), 110
 onAddBtnClick() (flare.viur.bones.base.BaseMultiEditWidget method), 113
 onAddBtnClick() (flare.viur.bones.relational.RelationalMultiEditWidget method), 123
 onApplyfilterChanged() (flare.viur.widgets.list.ListSelection method), 134
 onApplyfilterChanged() (flare.widgets.buttonbar.ButtonBarSearch method), 142
 onAttach() (flare.html5.core.TextNode method), 41
 onAttach() (flare.html5.core.Widget method), 47
 onAttach() (flare.html5.TextNode method), 81
 onAttach() (flare.html5.Widget method), 87
 onAttach() (flare.popup.Popup method), 159
 onAttach() (flare.viur.forms.ViurFormBone method), 139
 onAttach() (flare.viur.forms.ViurFormSubmit method), 140
 onAttach() (flare.viur.widgets.htmleditor.HtmlEditor method), 133
 onBind() (flare.button.Button method), 143
 onBind() (flare.html5.core.Widget method), 46
 onBind() (flare.html5.Widget method), 86
 onBlur() (flare.html5.core.Widget method), 48
 onBlur() (flare.html5.Widget method), 88
 onBoneChange() (flare.viur.forms.ViurForm method), 139
 onCancel() (flare.popup.Prompt method), 159
 onCancel() (flare.popup.radioButtonDialog method), 160
 onCancel() (flare.popup.TextareaDialog method), 160
 onChange() (flare.html5.core.Widget method), 48
 onChange() (flare.html5.Widget method), 88
 onChange() (flare.input.Input method), 151
 onChange() (flare.viur.bones.numeric.NumericEditWidget method), 118
 onChange() (flare.viur.bones.relational.FileEditDirectWidget method), 124
 onChange() (flare.viur.bones.relational.FileMultiEditDirectWidget method), 124
 onChange() (flare.viur.bones.relational.RelationalEditWidget method), 123
 onChange() (flare.viur.bones.string.StringEditWidget method), 128
 onChange() (flare.viur.forms.ViurForm method), 138
 onChange() (flare.viur.forms.ViurFormBone method), 139
 onClick() (flare.button.Button method), 143
 onClick() (flare.html5.core.Widget method), 48
 onClick() (flare.html5.Widget method), 88
 onClick() (flare.viur.formtooltip.ToolTip method), 140
 onClick() (flare.viur.widgets.file.FileImagePopup method), 131
 onClick() (flare.viur.widgets.file.FilePreviewImage method), 131
 onClick() (flare.viur.widgets.htmleditor.TextInsertImageAction method), 132
 onClick() (flare.viur.widgets.tree.TreeItemWidget method), 136

- onClose() (*flare.popup.Popup* method), 159
- onCompletion() (*flare.handler.SyncHandler* method), 147
- onCompletion() (*flare.network.NetworkService* method), 156
- onContextMenu() (*flare.html5.core.Widget* method), 48
- onContextMenu() (*flare.html5.Widget* method), 88
- onDataChanged() (*flare.Cache* method), 164
- onDataChanged() (*flare.cache.Cache* method), 144
- onDbClick() (*flare.html5.core.Widget* method), 48
- onDbClick() (*flare.html5.Widget* method), 88
- onDbClick() (*flare.viur.widgets.tree.TreeItemWidget* method), 136
- onDeleteBtnClick() (*flare.viur.bones.relational.FileEditDirectWidget* method), 124
- onDeleteBtnClick() (*flare.viur.bones.relational.RelationalFileEditDirectWidget* method), 123
- onDetach() (*flare.html5.core.TextNode* method), 41
- onDetach() (*flare.html5.core.Widget* method), 47
- onDetach() (*flare.html5.TextNode* method), 81
- onDetach() (*flare.html5.Widget* method), 87
- onDetach() (*flare.input.Input* method), 151
- onDetach() (*flare.popup.Popup* method), 159
- onDetach() (*flare.views.view.ViewWidget* method), 111
- onDetach() (*flare.viur.widgets.htmleditor.HtmlEditor* method), 133
- onDocumentKeyDown() (*flare.popup.Confirm* method), 159
- onDocumentKeyDown() (*flare.popup.Popup* method), 159
- onDocumentKeyDown() (*flare.popup.Prompt* method), 159
- onDocumentKeyDown() (*flare.popup.TextareaDialog* method), 159
- onDownloadBtnClick() (*flare.viur.widgets.file.FileImagePopup* method), 131
- onDrag() (*flare.html5.core.Widget* method), 48
- onDrag() (*flare.html5.Widget* method), 88
- onDragEnd() (*flare.html5.core.Widget* method), 48
- onDragEnd() (*flare.html5.Widget* method), 88
- onDragEnd() (*flare.viur.bones.base.BaseMultiEditWidgetEntry* method), 113
- onDragEnd() (*flare.viur.widgets.tree.TreeItemWidget* method), 135
- onDragEnter() (*flare.html5.core.Widget* method), 48
- onDragEnter() (*flare.html5.Widget* method), 88
- onDragEnter() (*flare.viur.bones.relational.FileEditDirectWidget* method), 124
- onDragEnter() (*flare.viur.bones.relational.FileMultiEditDirectWidget* method), 125
- onDragLeave() (*flare.html5.core.Widget* method), 48
- onDragLeave() (*flare.html5.Widget* method), 88
- onDragLeave() (*flare.viur.bones.base.BaseMultiEditWidgetEntry* method), 113
- onDragLeave() (*flare.viur.bones.relational.FileEditDirectWidget* method), 124
- onDragLeave() (*flare.viur.bones.relational.FileMultiEditDirectWidget* method), 125
- onDragLeave() (*flare.viur.widgets.tree.TreeItemWidget* method), 135
- onDragOver() (*flare.html5.core.Widget* method), 48
- onDragOver() (*flare.html5.Widget* method), 88
- onDragOver() (*flare.viur.bones.base.BaseMultiEditWidgetEntry* method), 113
- onDragOver() (*flare.viur.bones.relational.FileEditDirectWidget* method), 124
- onDragOver() (*flare.viur.bones.relational.FileMultiEditDirectWidget* method), 125
- onDragOver() (*flare.viur.widgets.tree.TreeItemWidget* method), 135
- onDragStart() (*flare.html5.core.Widget* method), 48
- onDragStart() (*flare.html5.Widget* method), 88
- onDragStart() (*flare.viur.bones.base.BaseMultiEditWidgetEntry* method), 113
- onDragStart() (*flare.viur.widgets.tree.TreeItemWidget* method), 135
- onDrop() (*flare.html5.core.Widget* method), 49
- onDrop() (*flare.html5.Widget* method), 89
- onDrop() (*flare.viur.bones.base.BaseMultiEditWidgetEntry* method), 113
- onDrop() (*flare.viur.bones.relational.FileEditDirectWidget* method), 124
- onDrop() (*flare.viur.bones.relational.FileMultiEditDirectWidget* method), 125
- onDrop() (*flare.viur.widgets.tree.TreeItemWidget* method), 135
- onEditorChange() (*flare.viur.widgets.htmleditor.HtmlEditor* method), 133
- onError() (*flare.handler.SyncHandler* method), 147
- onError() (*flare.icons.Icon* method), 149
- onError() (*flare.network.NetworkService* method), 156
- onFailed() (*flare.viur.widgets.file.Uploader* method), 131
- onFinished() (*flare.network.NetworkService* method), 156
- onFinished() (*flare.network.requestGroup* method), 156
- onFocus() (*flare.html5.core.Widget* method), 48
- onFocus() (*flare.html5.Widget* method), 88
- onFocus() (*flare.input.Input* method), 151
- onFocusIn() (*flare.html5.core.Widget* method), 48
- onFocusIn() (*flare.html5.Widget* method), 88
- onFocusIn() (*flare.html5.core.Widget* method), 48
- onFocusIn() (*flare.html5.Widget* method), 88
- onFocusOut() (*flare.html5.Widget* method), 88
- onFormChange() (*flare.html5.core.Widget* method), 48
- onFormChange() (*flare.html5.Widget* method), 88
- onFormInput() (*flare.html5.core.Widget* method), 48

- onFormInput() (*flare.html5.Widget method*), 88
- onFormSuccess() (*flare.viur.forms.ViurForm method*), 139
- onInput() (*flare.html5.core.Widget method*), 48
- onInput() (*flare.html5.Widget method*), 88
- onInvalid() (*flare.html5.core.Widget method*), 48
- onInvalid() (*flare.html5.Widget method*), 88
- onKeyDown() (*flare.html5.core.Widget method*), 48
- onKeyDown() (*flare.html5.Widget method*), 88
- onKeyDown() (*flare.popup.Alert method*), 159
- onKeyDown() (*flare.popup.Confirm method*), 159
- onKeyDown() (*flare.popup.Prompt method*), 159
- onKeyDown() (*flare.viur.widgets.file.Search method*), 131
- onKeyPress() (*flare.html5.core.Widget method*), 48
- onKeyPress() (*flare.html5.Widget method*), 88
- onKeyUp() (*flare.html5.core.Widget method*), 48
- onKeyUp() (*flare.html5.Widget method*), 88
- onKeyUp() (*flare.popup.Prompt method*), 159
- onKeyUp() (*flare.viur.bones.string.StringEditWidget method*), 128
- onLangBtnClick() (*flare.viur.bones.base.BaseLanguageEditWidget method*), 113
- onListStatusChanged() (*flare.handler.requestHandler method*), 146
- onLoad() (*flare.viur.widgets.file.Uploader method*), 131
- onMouseDown() (*flare.html5.core.Widget method*), 49
- onMouseDown() (*flare.html5.Widget method*), 89
- onMouseMove() (*flare.html5.core.Widget method*), 49
- onMouseMove() (*flare.html5.Widget method*), 89
- onMouseOut() (*flare.html5.core.Widget method*), 49
- onMouseOut() (*flare.html5.Widget method*), 89
- onMouseOver() (*flare.html5.core.Widget method*), 49
- onMouseOver() (*flare.html5.Widget method*), 89
- onMouseUp() (*flare.html5.core.Widget method*), 49
- onMouseUp() (*flare.html5.Widget method*), 89
- onMouseWheel() (*flare.html5.core.Widget method*), 49
- onMouseWheel() (*flare.html5.Widget method*), 89
- onNoClicked() (*flare.popup.Confirm method*), 159
- onOkay() (*flare.popup.Prompt method*), 159
- onOkay() (*flare.popup.radioButtonDialog method*), 160
- onOkay() (*flare.popup.TextareaDialog method*), 159
- onOkBtnClick() (*flare.popup.Alert method*), 159
- onProgress() (*flare.viur.widgets.file.Uploader method*), 131
- onReadyStateChange() (*flare.network.HTTPRequest method*), 154
- onRemoveBtnClick() (*flare.viur.bones.base.BaseMultiEditWidget method*), 113
- onRemoveBtnClick() (*flare.viur.bones.base.BaseMultiEditWidget method*), 113
- onRequestList() (*flare.viur.widgets.list.ListSelection method*), 134
- onReset() (*flare.html5.core.Widget method*), 48
- onReset() (*flare.html5.Widget method*), 88
- onScroll() (*flare.html5.core.Widget method*), 49
- onScroll() (*flare.html5.Widget method*), 89
- onSelect() (*flare.html5.core.Widget method*), 48
- onSelect() (*flare.html5.Widget method*), 88
- onSelectBtnClick() (*flare.viur.bones.relational.RelationalEditWidget method*), 123
- onSelectionActivated() (*flare.viur.widgets.htmleditor.TextInsertImageAction method*), 133
- onSkeyAvailable() (*flare.viur.widgets.file.Uploader method*), 131
- onStartSearch() (*flare.viur.widgets.file.FileWidget method*), 132
- onSubmit() (*flare.html5.core.Widget method*), 48
- onSubmit() (*flare.html5.Widget method*), 88
- onSubmitStatusChanged() (*flare.viur.forms.ViurForm method*), 139
- onSubmitStatusChanged() (*flare.viur.forms.ViurFormSubmit method*), 140
- onSubmitStatusChanged() (*flare.viur.widgets.file.Uploader method*), 131
- onTimeout() (*flare.network.NetworkService method*), 156
- onTouchCancel() (*flare.html5.core.Widget method*), 49
- onTouchCancel() (*flare.html5.Widget method*), 89
- onTouchEnd() (*flare.html5.core.Widget method*), 49
- onTouchEnd() (*flare.html5.Widget method*), 89
- onTouchMove() (*flare.html5.core.Widget method*), 49
- onTouchMove() (*flare.html5.Widget method*), 89
- onTouchStart() (*flare.html5.core.Widget method*), 49
- onTouchStart() (*flare.html5.Widget method*), 89
- onUnsetBtnClick() (*flare.viur.bones.color.ColorEditWidget method*), 116
- onUploadAdded() (*flare.viur.widgets.file.Uploader method*), 131
- onUploadFailed() (*flare.viur.bones.relational.FileEditDirectWidget method*), 124
- onUploadFailed() (*flare.viur.bones.relational.FileMultiEditDirectWidget method*), 125
- onUploadSuccess() (*flare.viur.bones.relational.FileEditDirectWidget method*), 124
- onUploadSuccess() (*flare.viur.bones.relational.FileMultiEditDirectWidget method*), 125
- onUploadUrlAvailable() (*flare.viur.widgets.file.Uploader method*), 131
- onViewfocusedChanged() (*flare.views.view.ViewWidget method*), 110
- onYesClicked() (*flare.popup.Confirm method*), 159
- Optgroup (class in *flare.html5*), 100
- Optgroup (class in *flare.html5.core*), 60
- Option (class in *flare.html5*), 100
- Option (class in *flare.html5.core*), 60

Output (class in flare.html5), 101
Output (class in flare.html5.core), 61

P

P (class in flare.html5), 98
P (class in flare.html5.core), 58
Param (class in flare.html5), 104
Param (class in flare.html5.core), 64
params (in module flare.views.view), 110
parent() (flare.html5.core.Widget method), 49
parent() (flare.html5.Widget method), 89
parseFloat() (in module flare.html5), 107
parseFloat() (in module flare.html5.core), 67
parseFloat() (in module flare.utils), 163
parseHTML() (in module flare.html5), 108
parseHTML() (in module flare.html5.core), 68
parseInt() (in module flare.html5), 107
parseInt() (in module flare.html5.core), 67
parseInt() (in module flare.utils), 162
PasswordBone (class in flare.viur.bones.password), 119
PasswordEditWidget (class in flare.viur.bones.password), 119
Plan (class in flare.cache), 144
pop() (flare.html5._WidgetClassWrapper method), 81
pop() (flare.html5.core._WidgetClassWrapper method), 41
Popout (class in flare.popout), 158
PopoutItem (class in flare.popout), 158
Popup (class in flare.popup), 158
prefix (flare.network.NetworkService attribute), 155
prepareCol() (flare.html5.core.Table method), 66
prepareCol() (flare.html5.Table method), 106
prepareCol() (flare.ignite.Table method), 151
prepareGrid() (flare.html5.core.Table method), 66
prepareGrid() (flare.html5.Table method), 106
prepareLogger() (in module flare.log), 153
prepareRow() (flare.html5.core.Table method), 66
prepareRow() (flare.html5.Table method), 106
prepareRow() (flare.ignite.Table method), 151
prependChild() (flare.html5.core.Widget method), 47
prependChild() (flare.html5.Widget method), 87
PriorityQueue (class in flare.priorityqueue), 160
PriorityQueue (class in flare.viur), 141
processSkelQueue() (in module flare.network), 154
Progress (class in flare.html5), 104
Progress (class in flare.html5.core), 64
Progress (class in flare.ignite), 150
Prompt (class in flare.popup), 159

Q

Q (class in flare.html5), 104
Q (class in flare.html5.core), 64

R

Radio (class in flare.ignite), 150
radioButtonDialog (class in flare.popup), 160
RawBone (class in flare.viur.bones.raw), 120
RawEditWidget (class in flare.viur.bones.raw), 120
RawViewWidget (class in flare.viur.bones.raw), 120
ReadFromClientErrorSeverity (class in flare.viur.bones.base), 112
RecordBone (class in flare.viur.bones.record), 121
RecordEditWidget (class in flare.viur.bones.record), 121
RecordViewWidget (class in flare.viur.bones.record), 121
reevaluate() (flare.viur.widgets.file.Search method), 131
register() (flare.event.EventDispatcher method), 145
register() (flare.observable.StateHandler method), 157
register() (flare.views.StateHandler method), 111
registerChangeListener() (flare.network.NetworkService static method), 155
registerField() (flare.viur.forms.ViurForm method), 139
registerTag() (in module flare.html5), 108
registerTag() (in module flare.html5.core), 68
registerViews() (in module flare.views.helpers), 110
RelationalBone (class in flare.viur.bones.relational), 123
RelationalEditWidget (class in flare.viur.bones.relational), 122
RelationalMultiEditWidget (class in flare.viur.bones.relational), 123
RelationalViewWidget (class in flare.viur.bones.relational), 123
reload() (flare.handler.ListHandler method), 146
reloadList() (flare.viur.widgets.list.ListSelection method), 134
remove() (flare.html5._WidgetClassWrapper method), 81
remove() (flare.html5.core._WidgetClassWrapper method), 41
removeAllChildren() (flare.html5.core.Widget method), 47
removeAllChildren() (flare.html5.Widget method), 87
removeChangeListener() (flare.network.NetworkService static method), 155
removeChild() (flare.html5.core.Widget method), 47
removeChild() (flare.html5.Widget method), 87
removeClass() (flare.html5.core.Widget method), 47
removeClass() (flare.html5.Widget method), 87
removeEventListener() (flare.html5.core.Widget method), 42

- removeEventListener() (*flare.html5.Widget* method), 82
- removeView() (in module *flare.views.helpers*), 110
- renderTimeout() (*flare.viur.bones.string.StringEditWidget* method), 128
- replaceChild() (*flare.html5.core.Widget* method), 47
- replaceChild() (*flare.html5.Widget* method), 87
- replaceSVG() (*flare.icons.SvgIcon* method), 149
- replaceWithMessage()
(*flare.viur.widgets.file.Uploader* method), 131
- request() (*flare.Cache* method), 164
- request() (*flare.cache.Cache* method), 144
- request() (*flare.handler.SyncHandler* static method), 146
- request() (*flare.network.NetworkService* static method), 155
- requestClients() (*flare.viur.widgets.list.ListSelection* method), 134
- requestData() (*flare.handler.requestHandler* method), 146
- requestFallBack() (*flare.icons.SvgIcon* method), 149
- requestGroup (class in *flare.network*), 156
- requestHandler (class in *flare.handler*), 146
- requestNext() (*flare.handler.ListHandler* method), 146
- requestSuccess() (*flare.handler.ListHandler* method), 146
- requestSuccess() (*flare.handler.requestHandler* method), 146
- require() (*flare.Cache* method), 164
- require() (*flare.cache.Cache* method), 144
- resetIcon() (*flare.button.Button* method), 143
- resetLoadingState() (*flare.viur.widgets.file.Search* method), 131
- resetLoadingState()
(*flare.viur.widgets.htmleditor.TextInsertImageAction* method), 133
- resetSearch() (*flare.viur.widgets.file.Search* method), 130
- retryCodes (*flare.network.NetworkService* attribute), 155
- retryDelay (*flare.network.NetworkService* attribute), 155
- retryMax (*flare.network.NetworkService* attribute), 155
- RowWrapper (class in *flare.html5*), 106
- RowWrapper (class in *flare.html5.core*), 66
- Rq (class in *flare.html5*), 98
- Rq (class in *flare.html5.core*), 58
- Rt (class in *flare.html5*), 98
- Rt (class in *flare.html5.core*), 58
- Ruby (class in *flare.html5*), 98
- Ruby (class in *flare.html5.core*), 58
- run() (*flare.cache.Plan* method), 144
- run() (*flare.network.DeferredCall* method), 154
- ## S
- S (class in *flare.html5*), 98
- S (class in *flare.html5.core*), 58
- SafeEval (class in *flare.safeeval*), 161
- safeEval() (*flare.safeeval.SafeEval* method), 161
- Samp (class in *flare.html5*), 98
- Samp (class in *flare.html5.core*), 58
- Script (class in *flare.html5*), 104
- Script (class in *flare.html5.core*), 64
- Search (class in *flare.viur.widgets.file*), 130
- searchWidget() (*flare.viur.widgets.file.FileWidget* method), 132
- Section (class in *flare.html5*), 98
- Section (class in *flare.html5.core*), 58
- Select (class in *flare.html5*), 101
- Select (class in *flare.html5.core*), 61
- Select (class in *flare.ignite*), 150
- select() (*flare.priorityqueue.PriorityQueue* method), 160
- select() (*flare.viur.PriorityQueue* method), 141
- SelectMultipleBone (class in *flare.viur.bones.select*), 127
- SelectMultipleEditWidget (class in *flare.viur.bones.select*), 126
- selectorAllow (*flare.viur.bones.relational.RelationalBone* attribute), 123
- selectorAllow (*flare.viur.bones.relational.TreeDirBone* attribute), 124
- selectorAllow (*flare.viur.bones.relational.TreeItemBone* attribute), 123
- SelectSingleBone (class in *flare.viur.bones.select*), 127
- SelectSingleEditWidget (class in *flare.viur.bones.select*), 126
- SelectViewWidget (class in *flare.viur.bones.select*), 127
- sendViurForm() (*flare.viur.forms.ViurFormSubmit* method), 140
- serialize() (*flare.viur.bones.base.BaseEditWidget* method), 112
- serialize() (*flare.viur.bones.base.BaseLanguageEditWidget* method), 114
- serialize() (*flare.viur.bones.base.BaseMultiEditWidget* method), 113
- serialize() (*flare.viur.bones.base.BaseMultiViewWidget* method), 113
- serialize() (*flare.viur.bones.base.BaseViewWidget* method), 113
- serialize() (*flare.viur.bones.boolean.BooleanEditWidget* method), 115
- serialize() (*flare.viur.bones.color.ColorEditWidget* method), 116
- serialize() (*flare.viur.bones.date.DateEditWidget* method), 117

[serialize\(\)](#) (*flare.viur.bones.numeric.NumericEditWidget* [setValue\(\)](#) (*flare.observable.ObservableValue* method), method), 118 [157](#)
[serialize\(\)](#) (*flare.viur.bones.password.PasswordEditWidget* [setValue\(\)](#) (*flare.viur.bones.numeric.NumericEditWidget* method), 119 [method](#)), 118
[serialize\(\)](#) (*flare.viur.bones.record.RecordEditWidget* [show\(\)](#) (*flare.html5.core.Widget* method), 46 [method](#)), 121 [show\(\)](#) (*flare.html5.Widget* method), 86
[serialize\(\)](#) (*flare.viur.bones.relational.FileMultiEditDirectWidget* [sinkEvent\(\)](#) (*flare.html5.core.Widget* method), 42 [method](#)), 125 [sinkEvent\(\)](#) (*flare.html5.Widget* method), 82
[serialize\(\)](#) (*flare.viur.bones.relational.RelationalEditWidget* [listPackagesPath\(\)](#) (in module *flare.views.helpers*), 110 [method](#)), 123 [SkellistItem](#) (class in *flare.viur.widgets.list*), 134
[serialize\(\)](#) (*flare.viur.bones.relational.RelationalViewWidget* [widgetType](#) (*flare.viur.widgets.tree.TreeLeafWidget* attribute), 136 [method](#)), 123 [widgetType](#) (*flare.viur.widgets.tree.TreeNodeWidget* attribute), 136
[serialize\(\)](#) (*flare.viur.bones.select.SelectMultipleEditWidget* [keyRequestQueue](#) (in module *flare.network*), 154 [method](#)), 126 [Small](#) (class in *flare.html5*), 98
[serialize\(\)](#) (*flare.viur.bones.select.SelectSingleEditWidget* [Small](#) (class in *flare.html5.core*), 58 [method](#)), 127 [sortChildren\(\)](#) (*flare.html5.core.Widget* method), 49
[serialize\(\)](#) (*flare.viur.bones.spatial.SpatialEditWidget* [sortChildren\(\)](#) (*flare.html5.Widget* method), 89 [method](#)), 128 [Source](#) (class in *flare.html5*), 104
[serialize\(\)](#) (*flare.viur.bones.string.StringEditWidget* [Source](#) (class in *flare.html5.core*), 64 [method](#)), 128 [Span](#) (class in *flare.html5*), 104
[serialize\(\)](#) (*flare.viur.forms.ViurForm* method), 139 [Span](#) (class in *flare.html5.core*), 64
[serialize\(\)](#) (*flare.viur.forms.ViurFormBone* method), 139 [SpatialBone](#) (class in *flare.viur.bones.spatial*), 128
[set\(\)](#) (*flare.html5._WidgetClassWrapper* method), 81 [SpatialEditWidget](#) (class in *flare.viur.bones.spatial*), 127
[set\(\)](#) (*flare.html5.core._WidgetClassWrapper* method), 41 [start\(\)](#) (*flare.Cache* method), 164
[setContent\(\)](#) (*flare.viur.widgets.list.ListSelection* [start\(\)](#) (*flare.cache.Cache* method), 144 [method](#)), 134 [startUpload\(\)](#) (*flare.viur.bones.relational.FileEditDirectWidget* method), 124
[setFile\(\)](#) (*flare.viur.widgets.file.FilePreviewImage* [startUpload\(\)](#) (*flare.viur.bones.relational.FileMultiEditDirectWidget* method), 131 [method](#)), 124
[setInvalid\(\)](#) (*flare.viur.forms.ViurFormBone* method), 140 [StateHandler](#) (class in *flare.observable*), 157
[setLanguage\(\)](#) (in module *flare.i18n*), 148 [StateHandler](#) (class in *flare.views*), 111
[setSelector\(\)](#) (*flare.viur.widgets.list.ListWidget* [StringBone](#) (class in *flare.viur.bones.string*), 129 [method](#)), 133 [StringEditWidget](#) (class in *flare.viur.bones.string*), 128
[setSelector\(\)](#) (*flare.viur.widgets.tree.TreeWidget* [StringViewWidget](#) (class in *flare.viur.bones.string*), 129 [method](#)), 136 [Strong](#) (class in *flare.html5*), 99
[setStyle\(\)](#) (*flare.viur.widgets.file.FileLeafWidget* [Strong](#) (class in *flare.html5.core*), 59 [method](#)), 131 [struct\(\)](#) (*flare.Cache* method), 164
[setStyle\(\)](#) (*flare.viur.widgets.file.FileNodeWidget* [struct\(\)](#) (*flare.cache.Cache* method), 144 [method](#)), 132 [Style](#) (class in *flare.html5*), 105
[setStyle\(\)](#) (*flare.viur.widgets.tree.BreadcrumbNodeWidget* [Style](#) (class in *flare.html5.core*), 65 [method](#)), 136 [style](#) (*flare.html5.core.Widget* attribute), 42
[setStyle\(\)](#) (*flare.viur.widgets.tree.BrowserLeafWidget* [style](#) (*flare.html5.Widget* attribute), 82 [method](#)), 136 [style](#) (*flare.popout.Popout* attribute), 158
[setStyle\(\)](#) (*flare.viur.widgets.tree.BrowserNodeWidget* [style](#) (*flare.popout.PopoutItem* attribute), 158 [method](#)), 136 [style](#) (*flare.viur.bones.base.BaseEditWidget* attribute), 112
[setStyle\(\)](#) (*flare.viur.widgets.tree.TreeItemWidget* [style](#) (*flare.viur.bones.base.BaseMultiEditWidget* attribute), 113 [method](#)), 135
[setStyle\(\)](#) (*flare.viur.widgets.tree.TreeLeafWidget* [style](#) (*flare.viur.bones.base.BaseMultiEditWidget* attribute), 113 [method](#)), 136
[setUrlHash\(\)](#) (in module *flare.network*), 156
[setValid\(\)](#) (*flare.viur.forms.ViurFormBone* method), 140

- style (flare.viur.bones.base.BaseMultiEditWidgetEntry attribute), 113
- style (flare.viur.bones.base.BaseViewWidget attribute), 112
- style (flare.viur.bones.boolean.BooleanEditWidget attribute), 114
- style (flare.viur.bones.color.ColorEditWidget attribute), 115
- style (flare.viur.bones.date.DateEditWidget attribute), 116
- style (flare.viur.bones.numeric.NumericEditWidget attribute), 118
- style (flare.viur.bones.password.PasswordEditWidget attribute), 119
- style (flare.viur.bones.raw.RawEditWidget attribute), 120
- style (flare.viur.bones.record.RecordEditWidget attribute), 121
- style (flare.viur.bones.record.RecordViewWidget attribute), 121
- style (flare.viur.bones.relational.FileEditDirectWidget attribute), 124
- style (flare.viur.bones.relational.FileEditWidget attribute), 125
- style (flare.viur.bones.relational.FileMultiEditDirectWidget attribute), 124
- style (flare.viur.bones.relational.RelationalEditWidget attribute), 122
- style (flare.viur.bones.relational.RelationalViewWidget attribute), 123
- style (flare.viur.bones.select.SelectMultipleEditWidget attribute), 126
- style (flare.viur.bones.string.StringEditWidget attribute), 128
- style (flare.viur.bones.text.TextEditWidget attribute), 129
- Sub (class in flare.html5), 99
- Sub (class in flare.html5.core), 59
- submitForm() (flare.viur.forms.ViurForm method), 139
- Summary (class in flare.html5), 105
- Summary (class in flare.html5.core), 65
- summernoteEditor (in module flare.viur.widgets.htmleditor), 132
- Summery (class in flare.html5), 99
- Summery (class in flare.html5.core), 59
- Sup (class in flare.html5), 99
- Sup (class in flare.html5.core), 59
- Svg (class in flare.html5.svg), 71
- SvgCircle (class in flare.html5.svg), 71
- SvgEllipse (class in flare.html5.svg), 71
- SvgG (class in flare.html5.svg), 71
- SvgIcon (class in flare.icons), 148
- SvgImage (class in flare.html5.svg), 72
- SvgLine (class in flare.html5.svg), 72
- SvgPath (class in flare.html5.svg), 72
- SvgPolygon (class in flare.html5.svg), 72
- SvgPolyline (class in flare.html5.svg), 72
- SvgRect (class in flare.html5.svg), 72
- SvgText (class in flare.html5.svg), 72
- SvgWidget (class in flare.html5.svg), 71
- Switch (class in flare.ignite), 150
- SyncHandler (class in flare.handler), 146
- ## T
- Table (class in flare.html5), 106
- Table (class in flare.html5.core), 66
- Table (class in flare.ignite), 150
- tag() (in module flare.html5), 108
- tag() (in module flare.html5.core), 68
- Tbody (class in flare.html5), 105
- Tbody (class in flare.html5.core), 65
- Td (class in flare.html5), 105
- Td (class in flare.html5.core), 65
- Template (class in flare.html5), 106
- Template (class in flare.html5.core), 66
- Textarea (class in flare.html5), 101
- Textarea (class in flare.html5.core), 61
- Textarea (class in flare.ignite), 150
- TextareaDialog (class in flare.popup), 159
- TextBone (class in flare.viur.bones.text), 130
- TextEditWidget (class in flare.viur.bones.text), 129
- TextInsertImageAction (class in flare.viur.widgets.htmleditor), 132
- TextNode (class in flare.html5), 80
- TextNode (class in flare.html5.core), 40
- textToHtml() (in module flare.html5), 107
- textToHtml() (in module flare.html5.core), 67
- textToHtml() (in module flare.utils), 162
- TextViewWidget (class in flare.viur.bones.text), 129
- Th (class in flare.html5), 105
- Th (class in flare.html5.core), 65
- Thead (class in flare.html5), 105
- Thead (class in flare.html5.core), 65
- Time (class in flare.html5), 106
- Time (class in flare.html5.core), 66
- toggleArrow() (flare.viur.widgets.tree.TreeItemWidget method), 135
- toggleArrow() (flare.viur.widgets.tree.TreeLeafWidget method), 136
- toggleClass() (flare.html5.core.Widget method), 47
- toggleClass() (flare.html5.Widget method), 87
- toggleExpand() (flare.viur.widgets.tree.TreeItemWidget method), 136
- ToolTip (class in flare.viur.formtooltip), 140
- tooltipWidget() (flare.viur.bones.base.BaseBone method), 114
- Tr (class in flare.html5), 105
- Tr (class in flare.html5.core), 65

- Track (class in flare.html5), 106
- Track (class in flare.html5.core), 66
- translate() (in module flare.i18n), 148
- TreeBrowserWidget (class in flare.viur.widgets.tree), 137
- TreeDirBone (class in flare.viur.bones.relational), 123
- TreeItemBone (class in flare.viur.bones.relational), 123
- TreeItemWidget (class in flare.viur.widgets.tree), 135
- TreeLeafWidget (class in flare.viur.widgets.tree), 136
- TreeNodeWidget (class in flare.viur.widgets.tree), 136
- TreeWidget (class in flare.viur.widgets.tree), 136
- U**
- U (class in flare.html5), 99
- U (class in flare.html5.core), 59
- Ul (class in flare.html5), 102
- Ul (class in flare.html5.core), 62
- unescape() (in module flare.html5), 107
- unescape() (in module flare.html5.core), 67
- unescape() (in module flare.utils), 162
- unmarkDraggedElement() (flare.viur.widgets.tree.TreeItemWidget method), 135
- unobserve() (flare.intersectionObserver.IntersectionObserver method), 151
- unregister() (flare.event.EventDispatcher method), 145
- unregister() (flare.observable.StateHandler method), 157
- unregister() (flare.views.StateHandler method), 111
- unserialize() (flare.viur.bones.base.BaseEditWidget method), 112
- unserialize() (flare.viur.bones.base.BaseLanguageEditWidget method), 113
- unserialize() (flare.viur.bones.base.BaseMultiEditWidget method), 113
- unserialize() (flare.viur.bones.base.BaseMultiViewWidget method), 113
- unserialize() (flare.viur.bones.base.BaseViewWidget method), 113
- unserialize() (flare.viur.bones.boolean.BooleanEditWidget method), 115
- unserialize() (flare.viur.bones.boolean.BooleanViewWidget method), 115
- unserialize() (flare.viur.bones.color.ColorViewWidget method), 116
- unserialize() (flare.viur.bones.date.DateEditWidget method), 116
- unserialize() (flare.viur.bones.date.DateViewWidget method), 117
- unserialize() (flare.viur.bones.email.EmailViewWidget method), 117
- unserialize() (flare.viur.bones.numeric.NumericEditWidget method), 118
- unserialize() (flare.viur.bones.numeric.NumericViewWidget method), 119
- unserialize() (flare.viur.bones.raw.RawViewWidget method), 120
- unserialize() (flare.viur.bones.record.RecordEditWidget method), 121
- unserialize() (flare.viur.bones.record.RecordViewWidget method), 121
- unserialize() (flare.viur.bones.relational.FileEditDirectWidget method), 124
- unserialize() (flare.viur.bones.relational.FileEditWidget method), 125
- unserialize() (flare.viur.bones.relational.FileMultiEditDirectWidget method), 125
- unserialize() (flare.viur.bones.relational.FileViewWidget method), 124
- unserialize() (flare.viur.bones.relational.RelationalEditWidget method), 123
- unserialize() (flare.viur.bones.relational.RelationalViewWidget method), 123
- unserialize() (flare.viur.bones.select.SelectMultipleEditWidget method), 126
- unserialize() (flare.viur.bones.select.SelectSingleEditWidget method), 126
- unserialize() (flare.viur.bones.select.SelectViewWidget method), 127
- unserialize() (flare.viur.bones.spatial.SpatialEditWidget method), 128
- unserialize() (flare.viur.bones.string.StringEditWidget method), 128
- unserialize() (flare.viur.bones.string.StringViewWidget method), 129
- unserialize() (flare.viur.bones.text.TextViewWidget method), 129
- unserialize() (flare.viur.forms.ViurForm method), 139
- unserialize() (flare.viur.forms.ViurFormBone method), 139
- unsinkEvent() (flare.html5.core.Widget method), 42
- unsinkEvent() (flare.html5.Widget method), 82
- update() (flare.button.Button method), 143
- update() (flare.Cache method), 164
- update() (flare.cache.Cache method), 144
- update() (flare.html5._WidgetDataWrapper method), 82
- update() (flare.html5._WidgetStyleWrapper method), 82
- update() (flare.html5.core._WidgetDataWrapper method), 42
- update() (flare.html5.core._WidgetStyleWrapper method), 42
- update() (flare.viur.forms.ViurForm method), 139
- updateConf() (in module flare), 164
- updateConf() (in module flare.config), 144
- updateDefaultView() (in module flare.views.helpers), 110

[updateLength\(\)](#) (*flare.viur.bones.string.StringEditWidget* method), 128
[updateState\(\)](#) (*flare.observable.StateHandler* method), 157
[updateState\(\)](#) (*flare.views.StateHandler* method), 111
[updateString\(\)](#) (*flare.viur.bones.relational.RelationalEditWidget* method), 123
[updateStructure\(\)](#) (*flare.Cache* method), 164
[updateStructure\(\)](#) (*flare.cache.Cache* method), 144
[updateWidget\(\)](#) (*flare.viur.bones.base.BaseEditWidget* method), 112
[updateWidget\(\)](#) (*flare.viur.bones.boolean.BooleanEditWidget* method), 115
[updateWidget\(\)](#) (*flare.viur.bones.color.ColorEditWidget* method), 115
[updateWidget\(\)](#) (*flare.viur.bones.date.DateEditWidget* method), 116
[updateWidget\(\)](#) (*flare.viur.bones.email.EmailEditWidget* method), 117
[updateWidget\(\)](#) (*flare.viur.bones.numeric.NumericEditWidget* method), 118
[updateWidget\(\)](#) (*flare.viur.bones.password.PasswordEditWidget* method), 119
[updateWidget\(\)](#) (*flare.viur.bones.raw.RawEditWidget* method), 120
[updateWidget\(\)](#) (*flare.viur.bones.record.RecordEditWidget* method), 121
[updateWidget\(\)](#) (*flare.viur.bones.relational.FileEditDirectWidget* method), 124
[updateWidget\(\)](#) (*flare.viur.bones.relational.RelationalEditWidget* method), 122
[updateWidget\(\)](#) (*flare.viur.bones.select.SelectMultipleEditWidget* method), 126
[updateWidget\(\)](#) (*flare.viur.bones.select.SelectSingleEditWidget* method), 126
[updateWidget\(\)](#) (*flare.viur.bones.spatial.SpatialEditWidget* method), 127
[updateWidget\(\)](#) (*flare.viur.bones.string.StringEditWidget* method), 128
[updateWidget\(\)](#) (*flare.viur.bones.text.TextEditWidget* method), 129
[Uploader](#) (class in *flare.viur.widgets.file*), 131
[urlForArgs\(\)](#) (*flare.network.NetworkService* static method), 155

[viewWidget\(\)](#) (*flare.viur.bones.base.BaseBone* method), 114
[viewWidgetFactory](#) (*flare.viur.bones.base.BaseBone* attribute), 114
[viewWidgetFactory](#) (*flare.viur.bones.boolean.BooleanBone* attribute), 115
[viewWidgetFactory](#) (*flare.viur.bones.color.ColorBone* attribute), 116
[viewWidgetFactory](#) (*flare.viur.bones.date.DateBone* attribute), 117
[viewWidgetFactory](#) (*flare.viur.bones.email.EmailBone* attribute), 118
[viewWidgetFactory](#) (*flare.viur.bones.numeric.NumericBone* attribute), 119
[viewWidgetFactory](#) (*flare.viur.bones.raw.RawBone* attribute), 120
[viewWidgetFactory](#) (*flare.viur.bones.record.RecordBone* attribute), 121
[viewWidgetFactory](#) (*flare.viur.bones.relational.FileBone* attribute), 125
[viewWidgetFactory](#) (*flare.viur.bones.relational.FileDirectBone* attribute), 125
[viewWidgetFactory](#) (*flare.viur.bones.relational.RelationalBone* attribute), 123
[viewWidgetFactory](#) (*flare.viur.bones.select.SelectMultipleBone* attribute), 127
[viewWidgetFactory](#) (*flare.viur.bones.string.StringBone* attribute), 129
[viewWidgetFactory](#) (*flare.viur.bones.text.TextBone* attribute), 130
[ViurForm](#) (class in *flare.viur.forms*), 138
[ViurFormBone](#) (class in *flare.viur.forms*), 139
[ViurFormSubmit](#) (class in *flare.viur.forms*), 140

W

[warn\(\)](#) (*webworker_scripts.weblog* static method), 165
[Wbr](#) (class in *flare.html5*), 99
[Wbr](#) (class in *flare.html5.core*), 59
[weblog](#) (class in *webworker_scripts*), 165
[webworker_scripts](#) module, 165
[Widget](#) (class in *flare.html5*), 82
[Widget](#) (class in *flare.html5.core*), 42

Z

[zip_listdir\(\)](#) (in module *flare.views.helpers*), 110

V

[value](#) (*flare.observable.ObservableValue* attribute), 157
[Var](#) (class in *flare.html5*), 99
[Var](#) (class in *flare.html5.core*), 59
[Video](#) (class in *flare.html5*), 106
[Video](#) (class in *flare.html5.core*), 66
[View](#) (class in *flare.views.view*), 110
[ViewWidget](#) (class in *flare.views.view*), 110