
Flare

Release 1.0

Tilman Oestereich, Andreas H. Kelch

Jun 19, 2023

FLARE

1	About	3
	Python Module Index	171
	Index	173



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")
    })
  </script>
```

(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 Widgets

In this tutorial, we will introduce the widgets of flare.

Reusable UI components

Flare widgets are UI components that encapsulate one HTML element. They render their content using a template, other widgets or a mixture of both. When using templates, placeholders can be used to fill in dynamic data, elements can be qualified with a name to make them accessible, and events - such as a click - can have a handlers registered to.

Note that the template rendering will be performed only once, initially. After that, any changes to the content of the widget need to be performed programmatically. This should be kept in mind, especially when deciding where to draw the line between putting more complexity in the template and creating sub widgets.

Simple Widget

To get acquainted with widgets, we will create a simple one that just contains some static HTML.

```
from flare import html5

class SimpleWidget(html5.Div):
    def __init__(self):
        super().__init__(
            # language=HTML
            """
            <span style="color: red;">I am a </span><span style="color: blue;">Widget!</
            ↪span>
            """
        )
```

As we can see, our SimpleWidget contains just two basic HTML `<span>`'s, wrapped in a `<div>` (as denoted by the derivation from `html5.Div`). That means, that our widget fundamentally is a `<div>`, and can be treated as such. For example, it can now simply be put into the DOM somewhere. Try it out by just appending it to the `<body>`:

```
html5.Body().appendChild(SimpleWidget())
```

Placeholders

Static widgets like the one we just created do have their uses, but most of the time, you will want to display some dynamic data as well. For this, the template provided in the super constructor call can be enriched with placeholders, which are then replaced with the given data when the template is being rendered.

```
class ParametrizedWidget(html5.Div):
    def __init__(self, number: int):
        super().__init__(
            # language=HTML
            """
            My parameter is <span style="color: red;">{{number}}</span>
            """, number=number
        )
```

(continues on next page)

(continued from previous page)

```
html5.Body().appendChild(ParametrizedWidget(5))
```

Now our widget expects a “number” parameter, which is then passed into the template. However, you need to tell the template, which parameter is to be supplied with what value, which is a bit awkward. A much better approach is to expect a dict in your constructor signature, and unpack it when passing it to the super constructor.

```
class ParametrizedWidget(html5.Div):
    def __init__(self, parameters: dict):
        super().__init__(
            # language=HTML
            """
            My parameter is <span style="color: red;">{{number}}</span>
            """, **parameters
        )
```

```
html5.Body().appendChild(ParametrizedWidget({"number": 5}))
```

Note that the rendering of the template happens only once. If the parameters change after that, there is no built in reactivity; you have to handle these cases yourself. Let’s look into that now.

Placeholders

What we’re gonna build now is a widget that reacts to an event by increasing a number and displaying it. We will create a counter widget that displays a number, with a button that increases the number whenever it is clicked.

```
class CounterWidget(html5.Div):
    value = 0

    def __init__(self):
        super().__init__(
            # language=HTML
            """
            Counter: <span [name]="valueDisplay">{{value}}</span> <button @click=
            ↪ "increase">Increase!</button>
            """, value=self.value
        )

    def increase(self):
        self.value += 1
        self.valueDisplay.replaceChild(self.value)
```

First, let’s take a look at the `[name]` attribute. This attribute registers the element on which it is defined as a field of your widget class. As a result of that, we can simply access the `<span>` that contains the number in the `increase` method with the field name given by the `[name]` attribute value, which in this case is `valueDisplay`.

Next, we are registering the `increase` method as a handler on the click event of the button, by using the `@click` attribute on the `<button>` element.

What the `increase` method then does, is increase value by one. But since, as stated, this will not do anything by itself, it also updates the content of the `<span>` called `valueDisplay`, by replacing its content with the new value.

Conditional elements

When working with widgets, you often want to exclude elements from being rendered in certain conditions. As an example, your widget might combine a picture and a text, but you want to support the case that just one of both is provided.

You do not need to manipulate the DOM manually after rendering to achieve this. You can use the `flare-if` attribute instead. If the expression you provide as its value is not truthy, the element on which it is placed is excluded from rendering.

Let's build that widget that combines a picture and text.

```
class ImageAndTextWidget(html5.Div):
    def __init__(self, parameters: dict):
        super().__init__(
            # language=HTML
            """
            <div flare-if="pictureUrl" style="text-align: center;">
                
            </div>
            <p flare-if="text">{{text}}</p>
            """, **parameters
        )

html5.Body().appendChild(ImageAndTextWidget(
    {
        "pictureUrl": "https://upload.wikimedia.org/wikipedia/commons/thumb/4/46/A_
↪kitten_on_the_lawn_%28Pixabay%29.jpg/640px-A_kitten_on_the_lawn_%28Pixabay%29.jpg",
        "text": "Look! Kitty likes the flowers!"
    }
))
```

Running this, you get a picture of a kitten with some text below. If you now play around with it, you will notice that by omitting the `pictureUrl`, you do not get a broken image, but the entire `<div>` that contains the image is gone. The same goes for the `<p>` if you provide no text.

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):
        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):
        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.

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

<code>TextNode</code>	Represents a piece of text inside the DOM.
<code>_WidgetClassWrapper</code>	Built-in mutable sequence.
<code>_WidgetDataWrapper</code>	dict() -> new empty dictionary
<code>_WidgetStyleWrapper</code>	dict() -> new empty dictionary
<code>Widget</code>	
<code>_attrLabel</code>	
<code>_attrCharset</code>	

continues on next page

¹ Created with sphinx-autoapi

Table 1 – continued from previous page

<code>_attrCite</code>
<code>_attrDatetime</code>
<code>_attrForm</code>
<code>_attrAlt</code>
<code>_attrAutofocus</code>
<code>_attrDisabled</code>
<code>_attrChecked</code>
<code>_attrIndeterminate</code>
<code>_attrName</code>
<code>_attrValue</code>
<code>_attrAutocomplete</code>
<code>_attrRequired</code>
<code>_attrMultiple</code>
<code>_attrSize</code>
<code>_attrFor</code>
<code>_attrInputs</code>
<code>_attrFormhead</code>
<code>_attrHref</code>
<code>_attrTarget</code>
<code>_attrType</code>
<code>_attrMedia</code>
<code>_attrDimensions</code>
<code>_attrUsemap</code>
<code>_attrMultimedia</code>
<code>_attrRel</code>
<code>_attrSrc</code>

continues on next page

Table 1 – continued from previous page

<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>
<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>

continues on next page

Table 1 – continued from previous page

<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>
<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>

continues on next page

Table 1 – continued from previous page

<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>
<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>

continues on next page

Table 1 – continued from previous page

<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>
<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>

continues on next page

Table 1 – continued from previous page

<i>Template</i>	
<i>HtmlAst</i>	Abstract syntax tree element used by <code>parseHTML()</code> .
Functions	
<i>domCreateAttribute</i> (tag[, ns])	Creates a new HTML/SVG/... attribute.
<i>domCreateElement</i> (tag[, ns])	Creates a new HTML/SVG/... tag.
<i>domCreateTextNode</i> ([txt])	
<i>domGetElementById</i> (idTag)	
<i>domElementFromPoint</i> (x, y)	
<i>domGetElementsByName</i> (tag)	
<i>domParseString</i> (string[, mimetype])	Parses the given string with the optionally given mime-type using JavaScript's DOM parser.
<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])	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])	Parses a value as int.
<i>parseFloat</i> (s[, ret])	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)	

continues on next page

Table 2 – continued from previous page

<i>isShift</i> (event)	
<i>isMeta</i> (event)	
<i>registerTag</i> (tagName, widgetClass[, override])	
<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])	Generates a dictionary of all to the <code>html5</code> -library known tags and their associated objects and attributes.
<i>parseHTML</i> (→ <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</code> -decorator.
<i>fromHTML</i> (→ [Widget])	Parses the provided HTML code according to the objects defined in the <code>html5</code> -library.

Attributes

<i>htmlExpressionEvaluator</i>
<i>document</i>
<i>__domParser</i>
<i>_body</i>
<i>_head</i>
<i>__tags</i>
<i>__reVarReplacer</i>

`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.domParseString(string, mimetype='text/html')`

Parses the given string with the optionally given mimetype using JavaScript's DOM parser. :param string: Any XML/HTML string processable by DOMParser. :param mimetype: The mimetype to use.

Returns

Returns the parsed document DOM.

`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(txt)`

`_getText()`

`__str__()`

Return str(self).

`onAttach()`

`onDetach()`

`_setDisabled(disabled)`

`_getDisabled()`

`children()`

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(value)`

```

_updateElem()
append(p_object)
    Append object to the end of the list.
clear()
    Remove all items from list.
remove(value)
    Remove first occurrence of value.
    Raises ValueError if the value is not present.
extend(iterable)
    Extend list by appending elements from the iterable.
insert(index, p_object)
    Insert object before index.
pop(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__(key, value)
        Set self[key] to value.

    update(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__(*key, value*)

Set self[key] to value.

update(*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(*args)

unsinkEvent(*args)

addEventListener(*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(*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()

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

On disabled elements, events are not triggered anymore.

enable()

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

_getTargetfuncName(*key, type*)

__getitem__(*key*)

__setitem__(*key, value*)

__str__()

Return str(self).

`__iter__()`

`_getData()`

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()`

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(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()`

Advisory information associated with the element.

Returns

str

`_setTitle(val)`

Advisory information associated with the element.

Parameters

`val` – str

`_getTabIndex()`

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(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()`

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

Returns

True | False

_setSpellcheck(*val*)

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

Parameters

val – True | False

_getLang()

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(*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()

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

Returns

True | False

_setHidden(*val*)

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

Parameters

val – True | False

_getDisabled()**_setDisabled(*disable*)****_getDropzone()**

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(*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()

Specifies whether the element is draggable.

Returns

True | False | “auto”

_setDraggable(*val*)

Specifies whether the element is draggable.

Parameters**val** – True | False | “auto”**_getDir()**

Specifies the elements text directionality.

Returns

ltr | rtl | auto

_setDir(val)

Specifies the elements text directionality.

Parameters**val** – ltr | rtl | auto**_getContextmenu()**

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

Returns**_setContextMenu(val)**

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

Parameters**val** –**_getContentEditable()**

Specifies whether the contents of the element are editable.

Returns

True | False

_setContentEditable(val)

Specifies whether the contents of the element are editable.

Parameters**val** – True | False**_getAccesskey()**

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(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()

Specifies a unique id for an element.

Parameters**self** –

Returns

_setId(*val*)

Specifies a unique id for an element.

Parameters

- **self** –
- **val** –

_getClass()

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

Returns

_setClass(*value*)

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

Parameters

- **self** –
- **value** –

@raise ValueError:

_getStyle()

The style attribute specifies an inline style for an element.

Parameters

- self** –

Returns

_getRole()

Specifies a role for an element.

@param self: @return:

_setRole(*val*)

Specifies a role for an element.

@param self: @param val:

hide()

Hide element, if shown.

Returns

show()

Show element, if hidden.

Returns

isHidden()

Checks if a widget is hidden.

Returns

True if hidden, False otherwise.

isVisible()

Checks if a widget is visible.

Returns

True if visible, False otherwise.

onBind(*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()**onDetach()****__collectChildren(**args*, ***kwargs*)**

Internal function for collecting children from args.

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

insertBefore(*insert*, *child*, *kwargs*)****insertAfter(*insert*, *child*, ***kwargs*)****prependChild(**args*, ***kwargs*)****appendChild(**args*, ***kwargs*)****replaceChild(**args*, ***kwargs*)****removeChild(*child*)****removeAllChildren()**

Removes all child widgets of the current widget.

isParentOf(*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(*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(*className*)

Determine whether the current widget is assigned the given class.

Parameters

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

addClass(*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(*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(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(event)**onChange(event)****onContextMenu(event)****onFocus(event)****onFocusIn(event)****onFocusOut(event)****onFormChange(event)****onFormInput(event)****onInput(event)****onInvalid(event)****onReset(event)****onSelect(event)****onSubmit(event)****onKeyDown(event)**

onKeyPress(*event*)
onKeyUp(*event*)
onClick(*event*, *wdg=None*)
onDbClick(*event*)
onDrag(*event*)
onDragEnd(*event*)
onDragEnter(*event*)
onDragLeave(*event*)
onDragOver(*event*)
onDragStart(*event*)
onDrop(*event*)
onMouseDown(*event*)
onMouseMove(*event*)
onMouseOut(*event*)
onMouseOver(*event*)
onMouseUp(*event*)
onMouseWheel(*event*)
onScroll(*event*)
onTouchStart(*event*)
onTouchEnd(*event*)
onTouchMove(*event*)
onTouchCancel(*event*)
focus()
blur()
parent()
children(*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 typelist | [Widget](#) | None**sortChildren**(*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(*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()**_setLabel**(*val*)**class** flare.html5.core._attrCharset

Bases: object

_getCharset()**_setCharset**(*val*)**class** flare.html5.core._attrCite

Bases: object

_getCite()**_setCite**(*val*)**class** flare.html5.core._attrDatetime

Bases: object

_getDatetime()**_setDatetime**(*val*)**class** flare.html5.core._attrForm

Bases: object

_getForm()

```

    _setForm(val)
class flare.html5.core._attrAlt
  Bases: object
    _getAlt()
    _setAlt(val)
class flare.html5.core._attrAutofocus
  Bases: object
    _getAutofocus()
    _setAutofocus(val)
class flare.html5.core._attrDisabled
  Bases: object
class flare.html5.core._attrChecked
  Bases: object
    _getChecked()
    _setChecked(val)
class flare.html5.core._attrIndeterminate
  Bases: object
    _getIndeterminate()
    _setIndeterminate(val)
class flare.html5.core._attrName
  Bases: object
    _getName()
    _setName(val)
class flare.html5.core._attrValue
  Bases: object
    _getValue()
    _setValue(val)
class flare.html5.core._attrAutocomplete
  Bases: object
    _getAutocomplete()
    _setAutocomplete(val)
class flare.html5.core._attrRequired
  Bases: object
    _getRequired()
    _setRequired(val)

```

```
class flare.html5.core._attrMultiple
```

```
  Bases: object
```

```
  _getMultiple()
```

```
  _setMultiple(val)
```

```
class flare.html5.core._attrSize
```

```
  Bases: object
```

```
  _getSize()
```

```
  _setSize(val)
```

```
class flare.html5.core._attrFor
```

```
  Bases: object
```

```
  _getFor()
```

```
  _setFor(val)
```

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

```
  Bases: _attrRequired
```

```
  _getMaxlength()
```

```
  _setMaxlength(val)
```

```
  _getPlaceholder()
```

```
  _setPlaceholder(val)
```

```
  _getReadonly()
```

```
  _setReadonly(val)
```

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

```
  Bases: object
```

```
  _getFormaction()
```

```
  _setFormaction(val)
```

```
  _getFormenctype()
```

```
  _setFormenctype(val)
```

```
  _getFormmethod()
```

```
  _setFormmethod(val)
```

```
  _getFormtarget()
```

```
  _setFormtarget(val)
```

```
  _getFormnovalidate()
```

```
  _setFormnovalidate(val)
```

```

class flare.html5.core._attrHref
    Bases: object
    _getHref()
        Url of a Page.

        Parameters
        self –

    _setHref(val)
        Url of a Page.

        Parameters
        val – URL

    _getHreflang()

    _setHreflang(val)

class flare.html5.core._attrTarget
    Bases: object
    _getTarget()

    _setTarget(val)

class flare.html5.core._attrType
    Bases: object
    _getType()

    _setType(val)

class flare.html5.core._attrMedia
    Bases: \_attrType
    _getMedia()

    _setMedia(val)

class flare.html5.core._attrDimensions
    Bases: object
    _getWidth()

    _setWidth(val)

    _getHeight()

    _setHeight(val)

class flare.html5.core._attrUsemap
    Bases: object
    _getUsemap()

    _setUsemap(val)

class flare.html5.core._attrMultimedia
    Bases: object

```

```
_getAutoplay()
_setAutoplay(val)
_getPlaysinline()
_setPlaysinline(val)
_getControls()
_setControls(val)
_getLoop()
_setLoop(val)
_getMuted()
_setMuted(val)
_getPreload()
_setPreload(val)

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

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

class flare.html5.core.A(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrHref, _attrTarget, _attrMedia, _attrRel, _attrName
    _tagName = 'a'
    _getDownload()
        The download attribute specifies the path to a download.
        Returns
            filename
    _setDownload(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()
    _setCoords(val)
    _getShape()
    _setShape(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()
    _setBlockquote(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()
    _setIcon(val)
    _getRadiogroup()
    _setRadiogroup(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()
```

```
    _setOpen(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()
    _setNovalidate(val)

    _getAction()
    _setAction(val)

    _getMethod()
    _setMethod(val)

    _getEnctype()
    _setEnctype(val)

    _getAccept_attrCharset()
    _setAccept_attrCharset(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()
    _setAccept(val)

    _getList()
    _setList(val)
```

```

    _getMax()
    _setMax(val)
    _getMin()
    _setMin(val)
    _getPattern()
    _setPattern(val)
    _getStep()
    _setStep(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()
    _setSelected(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()
    _getOptions()

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

```

```
    _getRows()
    _setRows(val)
    _getWrap()
    _setWrap(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()
    _setSandbox(val)
    _getSrcdoc()
    _setSrcdoc(val)
    _getSeamless()
    _setSeamless(val)

class flare.html5.core.Img(src=None, *args, **kwargs)
    Bases: Widget, _attrSrc, _attrDimensions, _attrUsemap, _attrAlt
    _tagName = 'img'
    _leafTag = True
    _getCrossorigin()
    _setCrossorigin(val)
    _getIsmap()
    _setIsmap(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()
    _setChallenge(val)
```

```

    _getKeytype()
    _setKeytype(val)

class flare.html5.core.Link(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrHref, _attrMedia, _attrRel
    _tagName = 'link'
    _leafTag = True
    _getSizes()
    _setSizes(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()

    _setContent(val)

class flare.html5.core.Meter(*args, appendTo=None, style=None, **kwargs)
    Bases: Form, _attrValue
    _tagName = 'meter'

    _getHigh()

    _setHigh(val)

    _getLow()

    _setLow(val)

    _getMax()

    _setMax(val)

    _getMin()

    _setMin(val)

    _getOptimum()

    _setOptimum(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, _attrUsemmap
    _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()

    _setMax(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()
    _setAsync(val)
    _getDefer()
    _setDefer(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()
    _setOpen(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()
    _setScoped(val)
class flare.html5.core.Tr(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = 'tr'
    _getRowspan()
    _setRowspan(span)

```

```
class flare.html5.core.Td(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = 'td'
    _getColspan()
    _setColspan(span)
    _getRowspan()
    _setRowspan(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__(item)
    __setitem__(key, value)

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

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

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()
    _setKind(val)
    _getSrclang()
    _setSrclang(val)
    _getDefault()
    _setDefault(val)
class flare.html5.core.Video(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrSrc, _attrDimensions, _attrMultimedia
    _tagName = 'video'
    _getPoster()
    _setPoster(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.

Examples

```
"""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

<code>_attrSvgViewBox</code>
<code>_attrSvgDimensions</code>
<code>_attrSvgPoints</code>
<code>_attrSvgTransform</code>
<code>_attrSvgXlink</code>
<code>_attrSvgStyles</code>
<code>SvgWidget</code>
<code>Svg</code>
<code>SvgCircle</code>
<code>SvgEllipse</code>
<code>SvgG</code>
<code>SvgImage</code>
<code>SvgLine</code>
<code>SvgPath</code>
<code>SvgPolygon</code>
<code>SvgPolyline</code>
<code>SvgRect</code>
<code>SvgText</code>

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

```
    Bases: object
```

```
    _getViewbox()
```

```
    _setViewbox(val)
```

```
    _getPreserveaspectratio()
```

```
    _setPreserveaspectratio(val)
```

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

```
  Bases: object
```

```
  _getWidth()
```

```
  _setWidth(val)
```

```
  _getHeight()
```

```
  _setHeight(val)
```

```
  _getX()
```

```
  _setX(val)
```

```
  _getY()
```

```
  _setY(val)
```

```
  _getR()
```

```
  _setR(val)
```

```
  _getRx()
```

```
  _setRx(val)
```

```
  _getRy()
```

```
  _setRy(val)
```

```
  _getCx()
```

```
  _setCx(val)
```

```
  _getCy()
```

```
  _setCy(val)
```

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

```
  Bases: object
```

```
  _getPoints()
```

```
  _setPoints(val)
```

```
  _getX1()
```

```
  _setX1(val)
```

```
  _getY1()
```

```
  _setY1(val)
```

```
  _getX2()
```

```
  _setX2(val)
```

```
  _getY2()
```

```
  _setY2(val)
```

```
class flare.html5.svg._attrSvgTransform
    Bases: object
    _getTransform()
    _setTransform(val)

class flare.html5.svg._attrSvgXlink
    Bases: object
    _getXlinkhref()
    _setXlinkhref(val)

class flare.html5.svg._attrSvgStyles
    Bases: object
    _getFill()
    _setFill(val)
    _getStroke()
    _setStroke(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()
    _setVersion(val)
    _getXmlns()
    _setXmlns(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()
```

```

    _setSvgTransform(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()
    _setD(val)
    _getPathLength()
    _setPathLength(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>	

continues on next page

Table 3 – continued from previous page

<code>_attrLabel</code>
<code>_attrCharset</code>
<code>_attrCite</code>
<code>_attrDatetime</code>
<code>_attrForm</code>
<code>_attrAlt</code>
<code>_attrAutofocus</code>
<code>_attrDisabled</code>
<code>_attrChecked</code>
<code>_attrIndeterminate</code>
<code>_attrName</code>
<code>_attrValue</code>
<code>_attrAutocomplete</code>
<code>_attrRequired</code>
<code>_attrMultiple</code>
<code>_attrSize</code>
<code>_attrFor</code>
<code>_attrInputs</code>
<code>_attrFormhead</code>
<code>_attrHref</code>
<code>_attrTarget</code>
<code>_attrType</code>
<code>_attrMedia</code>
<code>_attrDimensions</code>
<code>_attrUsemap</code>
<code>_attrMultimedia</code>

continues on next page

Table 3 – continued from previous page

<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>
<i>Caption</i>
<i>Cite</i>
<i>Code</i>
<i>Datalist</i>
<i>Dfn</i>
<i>Div</i>
<i>Em</i>

continues on next page

Table 3 – continued from previous page

<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>
<i>Rq</i>
<i>Rt</i>
<i>Ruby</i>
<i>S</i>
<i>Samp</i>
<i>Section</i>
<i>Small</i>
<i>Strong</i>

continues on next page

Table 3 – continued from previous page

<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>
<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>

continues on next page

Table 3 – continued from previous page

<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>
<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>

continues on next page

Table 3 – continued from previous page

<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])	Creates a new HTML/SVG/... attribute.
<i>domCreateElement</i> (tag[, ns])	Creates a new HTML/SVG/... tag.
<i>domCreateTextNode</i> ([txt])	
<i>domGetElementById</i> (idTag)	
<i>domElementFromPoint</i> (x, y)	
<i>domGetElementsByName</i> (tag)	
<i>domParseString</i> (string[, mimetype])	Parses the given string with the optionally given mime-type using JavaScript's DOM parser.
<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])	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])	Parses a value as int.
<i>parseFloat</i> (s[, ret])	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)	

continues on next page

Table 4 – continued from previous page

<i>isReturn</i> (event)	
<i>isControl</i> (event)	
<i>isShift</i> (event)	
<i>isMeta</i> (event)	
<i>registerTag</i> (tagName, widgetClass[, override])	
<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])	Generates a dictionary of all to the <code>html5</code> -library known tags and their associated objects and attributes.
<i>parseHTML</i> (→ <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</code> -decorator.
<i>fromHTML</i> (→ [Widget])	Parses the provided HTML code according to the objects defined in the <code>html5</code> -library.

Attributes

<i>htmlExpressionEvaluator</i>
<i>document</i>
<i>__domParser</i>
<i>_body</i>
<i>_head</i>
<i>__tags</i>
<i>__reVarReplacer</i>

`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.domParseString(string, mimetype='text/html')
```

Parses the given string with the optionally given mimetype using JavaScript's DOM parser. :param string: Any XML/HTML string processable by DOMParser. :param mimetype: The mimetype to use.

Returns

Returns the parsed document DOM.

```
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(txt)
```

```
_getText()
```

```
__str__()
```

Return str(self).

```
onAttach()
```

```
onDetach()
```

```
_setDisabled(disabled)
```

```
_getDisabled()
```

```
children()
```

```
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(*value*)

_updateElem()

append(*p_object*)

Append object to the end of the list.

clear()

Remove all items from list.

remove(*value*)

Remove first occurrence of value.

Raises ValueError if the value is not present.

extend(*iterable*)

Extend list by appending elements from the iterable.

insert(*index*, *p_object*)

Insert object before index.

pop(*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__(*key*, *value*)

Set self[key] to value.

update(*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__(key, value)

Set self[key] to value.

update(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(*args)

unsinkEvent(*args)

addEventListener(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(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()

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

On disabled elements, events are not triggered anymore.

enable()

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

_getTargetfuncName(key, type)

__getitem__(key)

__setitem__(*key*, *value*)

__str__()

Return str(self).

__iter__()

_getData()

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()

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(*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()

Advisory information associated with the element.

Returns

str

_setTitle(*val*)

Advisory information associated with the element.

Parameters

val – str

_getTabIndex()

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(*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()

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

Returns

True | False

_setSpellcheck(*val*)

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

Parameters

val – True | False

_getLang()

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(*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()

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

Returns

True | False

_setHidden(*val*)

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

Parameters

val – True | False

_getDisabled()**_setDisabled(*disable*)****_getDropzone()**

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(*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()

Specifies whether the element is draggable.

Returns

True | False | “auto”

_setDraggable(*val*)

Specifies whether the element is draggable.

Parameters

val – True | False | “auto”

_getDir()

Specifies the elements text directionality.

Returns

ltr | rtl | auto

_setDir(*val*)

Specifies the elements text directionality.

Parameters

val – ltr | rtl | auto

_getContextmenu()

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

Returns**_setContextmenu(*val*)**

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

Parameters

val –

_getContenteditable()

Specifies whether the contents of the element are editable.

Returns

True | False

_setContenteditable(*val*)

Specifies whether the contents of the element are editable.

Parameters

val – True | False

_getAccesskey()

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(*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()

Specifies a unique id for an element.

Parameters

- self** –

Returns**_setId(*val*)**

Specifies a unique id for an element.

Parameters

- **self** –

- **val** –

_getClass()

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

Returns**_setClass(*value*)**

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

Parameters

- **self** –

- **value** –

@raise ValueError:

_getStyle()

The style attribute specifies an inline style for an element.

Parameters

- self** –

Returns**_getRole()**

Specifies a role for an element.

@param self: @return:

_setRole(*val*)

Specifies a role for an element.

@param self: @param val:

hide()

Hide element, if shown.

Returns**show()**

Show element, if hidden.

Returns

isHidden()

Checks if a widget is hidden.

Returns

True if hidden, False otherwise.

isVisible()

Checks if a widget is visible.

Returns

True if visible, False otherwise.

onBind(*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()**onDetach()****__collectChildren(**args*, ***kwargs*)**

Internal function for collecting children from args.

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

insertBefore(*insert*, *child*, *kwargs*)****insertAfter(*insert*, *child*, ***kwargs*)****prependChild(**args*, ***kwargs*)****appendChild(**args*, ***kwargs*)****replaceChild(**args*, ***kwargs*)****removeChild(*child*)****removeAllChildren()**

Removes all child widgets of the current widget.

isParentOf(*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(*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(*className*)

Determine whether the current widget is assigned the given class.

Parameters

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

addClass(**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(**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(*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(*event*)**onChange**(*event*)**onContextMenu**(*event*)**onFocus**(*event*)**onFocusIn**(*event*)**onFocusOut**(*event*)**onFormChange**(*event*)**onFormInput**(*event*)**onInput**(*event*)**onInvalid**(*event*)**onReset**(*event*)

onSelect(*event*)
onSubmit(*event*)
onKeyDown(*event*)
onKeyPress(*event*)
onKeyUp(*event*)
onClick(*event*, *wdg=None*)
onDbClick(*event*)
onDrag(*event*)
onDragEnd(*event*)
onDragEnter(*event*)
onDragLeave(*event*)
onDragOver(*event*)
onDragStart(*event*)
onDrop(*event*)
onMouseDown(*event*)
onMouseMove(*event*)
onMouseOut(*event*)
onMouseOver(*event*)
onMouseUp(*event*)
onMouseWheel(*event*)
onScroll(*event*)
onTouchStart(*event*)
onTouchEnd(*event*)
onTouchMove(*event*)
onTouchCancel(*event*)
focus()
blur()
parent()

children(*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(*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(*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()

_setLabel(*val*)

class flare.html5._attrCharset

Bases: object

_getCharset()

_setCharset(*val*)

class flare.html5._attrCite

Bases: object

_getCite()

_setCite(*val*)

```
class flare.html5._attrDatetime
  Bases: object
  _getDatetime()
  _setDatetime(val)

class flare.html5._attrForm
  Bases: object
  _getForm()
  _setForm(val)

class flare.html5._attrAlt
  Bases: object
  _getAlt()
  _setAlt(val)

class flare.html5._attrAutofocus
  Bases: object
  _getAutofocus()
  _setAutofocus(val)

class flare.html5._attrDisabled
  Bases: object

class flare.html5._attrChecked
  Bases: object
  _getChecked()
  _setChecked(val)

class flare.html5._attrIndeterminate
  Bases: object
  _getIndeterminate()
  _setIndeterminate(val)

class flare.html5._attrName
  Bases: object
  _getName()
  _setName(val)

class flare.html5._attrValue
  Bases: object
  _getValue()
  _setValue(val)
```

```
class flare.html5._attrAutocomplete
```

```
  Bases: object
```

```
  _getAutocomplete()
```

```
  _setAutocomplete(val)
```

```
class flare.html5._attrRequired
```

```
  Bases: object
```

```
  _getRequired()
```

```
  _setRequired(val)
```

```
class flare.html5._attrMultiple
```

```
  Bases: object
```

```
  _getMultiple()
```

```
  _setMultiple(val)
```

```
class flare.html5._attrSize
```

```
  Bases: object
```

```
  _getSize()
```

```
  _setSize(val)
```

```
class flare.html5._attrFor
```

```
  Bases: object
```

```
  _getFor()
```

```
  _setFor(val)
```

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

```
  Bases: _attrRequired
```

```
  _getMaxlength()
```

```
  _setMaxlength(val)
```

```
  _getPlaceholder()
```

```
  _setPlaceholder(val)
```

```
  _getReadonly()
```

```
  _setReadonly(val)
```

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

```
  Bases: object
```

```
  _getFormaction()
```

```
  _setFormaction(val)
```

```
  _getFormenctype()
```

```
  _setFormenctype(val)
```

```
    _getFormmethod()
    _setFormmethod(val)
    _getFormtarget()
    _setFormtarget(val)
    _getFormnovalidate()
    _setFormnovalidate(val)
class flare.html5._attrHref
    Bases: object
    _getHref()
        Url of a Page.
        Parameters
        self –
    _setHref(val)
        Url of a Page.
        Parameters
        val – URL
    _getHreflang()
    _setHreflang(val)
class flare.html5._attrTarget
    Bases: object
    _getTarget()
    _setTarget(val)
class flare.html5._attrType
    Bases: object
    _getType()
    _setType(val)
class flare.html5._attrMedia
    Bases: _attrType
    _getMedia()
    _setMedia(val)
class flare.html5._attrDimensions
    Bases: object
    _getWidth()
    _setWidth(val)
    _getHeight()
```

```

    _setHeight(val)
class flare.html5._attrUsemap
    Bases: object
    _getUsemap()
    _setUsemap(val)
class flare.html5._attrMultimedia
    Bases: object
    _getAutoplay()
    _setAutoplay(val)
    _getPlaysinline()
    _setPlaysinline(val)
    _getControls()
    _setControls(val)
    _getLoop()
    _setLoop(val)
    _getMuted()
    _setMuted(val)
    _getPreload()
    _setPreload(val)
class flare.html5._attrRel
    Bases: object
    _getRel()
    _setRel(val)
class flare.html5._attrSrc
    Bases: object
    _getSrc()
    _setSrc(val)
class flare.html5.A(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrHref, _attrTarget, _attrMedia, _attrRel, _attrName
    _tagName = 'a'
    _getDownload()
        The download attribute specifies the path to a download.

        Returns
            filename

```

_setDownload(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()

_setCoords(val)

_getShape()

_setShape(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()

_setBlockquote(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()

_setIcon(val)

```

    _getRadiogroup()
    _setRadiogroup(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()
    _setOpen(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()
    _setNovalidate(val)
    _getAction()
    _setAction(val)
    _getMethod()
    _setMethod(val)
    _getEnctype()
    _setEnctype(val)
    _getAccept_attrCharset()
    _setAccept_attrCharset(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()
    _setAccept(val)
    _getList()
    _setList(val)
    _getMax()
    _setMax(val)
    _getMin()
    _setMin(val)
    _getPattern()
    _setPattern(val)
    _getStep()
    _setStep(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()
    _setSelected(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()
    _getOptions()
```

```

class flare.html5.Textarea(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrDisabled, _attrForm, _attrAutofocus, _attrName, _attrInputs, _attrValue
    _tagName = 'textarea'

    _getCols()
    _setCols(val)

    _getRows()
    _setRows(val)

    _getWrap()
    _setWrap(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()
    _setSandbox(val)

    _getSrcdoc()
    _setSrcdoc(val)

    _getSeamless()
    _setSeamless(val)

class flare.html5.Img(src=None, *args, **kwargs)
    Bases: Widget, _attrSrc, _attrDimensions, _attrUsemap, _attrAlt
    _tagName = 'img'

    _leafTag = True

    _getCrossorigin()
    _setCrossorigin(val)

    _getIsmap()
    _setIsmap(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()
    _setChallenge(val)
    _getKeytype()
    _setKeytype(val)

class flare.html5.Link(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrHref, _attrMedia, _attrRel
    _tagName = 'link'
    _leafTag = True
    _getSizes()
    _setSizes(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()
    _setContent(val)

class flare.html5.Meter(*args, appendTo=None, style=None, **kwargs)
    Bases: Form, _attrValue
    _tagName = 'meter'
    _getHigh()
    _setHigh(val)
    _getLow()
    _setLow(val)
    _getMax()
    _setMax(val)
    _getMin()
    _setMin(val)
    _getOptimum()
    _setOptimum(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()

    _setMax(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()

    _setAsync(val)

    _getDefer()

    _setDefer(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()

    _setOpen(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()

    _setScoped(val)
```

```

class flare.html5.Tr(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = 'tr'
    _getRowspan()
    _setRowspan(span)

class flare.html5.Td(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget
    _tagName = 'td'
    _getColspan()
    _setColspan(span)
    _getRowspan()
    _setRowspan(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__(item)
    __setitem__(key, value)

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

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

```

```
clear()

_getCell()

getRowCount()

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()

    _setKind(val)

    _getSrclang()

    _setSrclang(val)

    _getDefault()

    _setDefault(val)

class flare.html5.Video(*args, appendTo=None, style=None, **kwargs)
    Bases: Widget, _attrSrc, _attrDimensions, _attrMultimedia
    _tagName = 'video'

    _getPoster()

    _setPoster(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.

Examples

```
'''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, moduleName, actionName[, name, data])	
<i>addView</i> (view[, name])	Add a View and make it available.
<i>updateDefaultView</i> (name)	
<i>removeView</i> (name[, targetView])	
<i>registerViews</i> (root, path)	Add all Views in a folder.
<i>zip_listdir</i> (zip_file, target_dir)	

Attributes

<i>sitepackagespath</i>

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

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

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

```
    loadView()
```

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

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

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

```
    initWidget()
```

```
    onDetach()
```

Package Contents

Classes

StateHandler

Attributes

conf

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

updateState(*key*, *value*)

getState(*key*, *empty=None*)

register(*key*, *widget*)

unregister(*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()
```

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

```
    updateWidget()
```

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

```
    unserialize(value=None)
```

```
        Unserialize the widget value.
```

```
    serialize()
```

```
        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(value=None)
    Unserialize the widget value.

serialize()
    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()

    onDragStart(event)

    onDragOver(event)

    onDragLeave(event)

    onDragEnd(event)

    onDrop(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()

    onRemoveBtnClick()

    addEntry(value=None)

    unserialize(value)

    serialize()

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

    unserialize(value)

    serialize()

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(sender)

```

unserialize(*value*)

serialize()

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

viewWidget(*value=None*)

labelWidget()

tooltipWidget()

errorWidget()

boneWidget(**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()
    Function for creating the Widget or multiple Widgets that represent the bone.

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

unserialize(value=None)
    Unserialize the widget value.

serialize()
    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(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

<i>ColorEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>ColorViewWidget</i>	Base class for a bone-compliant view widget implementation using a div.
<i>ColorBone</i>	

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

updateWidget()

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

onUnsetBtnClick()**serialize()**

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(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**

<i>DateEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>DateViewWidget</i>	Base class for a bone-compliant view widget implementation using a div.
<i>DateBone</i>	

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()

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

updateWidget()

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

```
unserialize(value=None)
    Unserialize the widget value.

serialize()
    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(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()
        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(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()
```

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

```
updateWidget()
```

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

```
setValue(value)
```

```
onChange(event)
```

unserialize(*value=None*)
Unserialize the widget value.

serialize()
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(*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()
Function for creating the Widget or multiple Widgets that represent the bone.

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

serialize()
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()

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

updateWidget()

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(*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

<code>RecordEditWidget</code>	Base class for a bone-compliant edit widget implementation using an input field.
<code>RecordViewWidget</code>	Base class for a bone-compliant view widget implementation using a div.
<code>RecordBone</code>	

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()

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

updateWidget()

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

unserialize(*value=None*)

Unserialize the widget value.

serialize()

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(*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()

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

updateWidget()

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

updateString()

onChange(event)

unserialize(value=None)

Unserialize the widget value.

serialize()

Serialize the widget value.

onSelectBtnClick()

onDeleteBtnClick()

class flare.viur.bones.relational.**RelationalViewWidget**(bone, language=None, **kwargs)

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

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

unserialize(value=None)

serialize()

class flare.viur.bones.relational.**RelationalMultiEditWidget**(*args, **kwargs)

Bases: [flare.viur.bones.base.BaseMultiEditWidget](#)

Class for encapsulating multiple bones inside a container.

onAddBtnClick()

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

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

    onChange(event)

    startUpload(file)

    onDragEnter(event)

    onDragOver(event)

    onDragLeave(event)

    onDrop(event)

    onUploadSuccess(uploader, entry)

    onUploadFailed(uploader, errorCode)

    unserialize(value=None)
        Unserialize the widget value.

    onDeleteBtnClick()

class flare.viur.bones.relational.FileViewWidget(bone, language=None, **kwargs)
    Bases: RelationalViewWidget

    unserialize(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(event)
    startUpload(file)
    onDragEnter(event)
    onDragOver(event)
    onDragLeave(event)
    onDrop(event)
    onUploadSuccess(uploader, entry)
    onUploadFailed(uploader, errorCode)
    addEntry(value=None)
    unserialize(value)
    serialize()

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

<code>SelectMultipleEditWidget</code>	Base class for a bone-compliant edit widget implementation using an input field.
<code>SelectSingleEditWidget</code>	Base class for a bone-compliant edit widget implementation using an input field.
<code>SelectViewWidget</code>	Base class for a bone-compliant view widget implementation using a div.
<code>SelectMultipleBone</code>	
<code>SelectSingleBone</code>	

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()

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

updateWidget()

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

unserialize(*value=None*)

Unserialize the widget value.

serialize()

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()

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

updateWidget()

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

unserialize(*value=None*)
Unserialize the widget value.

serialize()
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(*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()

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

updateWidget()

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

unserialize(value=None)

Unserialize the widget value.

serialize()

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**

<i>StringEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>StringViewWidget</i>	Base class for a bone-compliant view widget implementation using a div.
<i>StringBone</i>	

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()

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

updateWidget()

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

onChange(event)**onKeyUp(event)****renderTimeout()****updateLength()**

```
unserialize(value=None)
    Unserialize the widget value.

serialize()
    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(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

<i>TextEditWidget</i>	Base class for a bone-compliant edit widget implementation using an input field.
<i>TextViewWidget</i>	Base class for a bone-compliant view widget implementation using a div.
<i>TextBone</i>	

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

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

    _setDisabled(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(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

<i>Search</i>	
<i>FileImagePopup</i>	
<i>FilePreviewImage</i>	
<i>Uploader</i>	Uploads a file to the server while providing visual feedback of the progress.
<i>FileLeafWidget</i>	
<i>FileNodeWidget</i>	
<i>FileWidget</i>	Base Widget that renders a tree.

Functions

```
getImagePreview(data[, cropped, size])
```

```
flare.viur.widgets.file.getImagePreview(data, cropped=False, size=150)
```

```
class flare.viur.widgets.file.Search(*args, **kwargs)
```

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

```
    doSearch(*args, **kwargs)
```

```
    resetSearch()
```

```
    onKeyDown(event)
```

```
    resetLoadingState()
```

```
    reevaluate()
```

```
    focus()
```

```
class flare.viur.widgets.file.FileImagePopup(preview, *args, **kwargs)
```

```
    Bases: flare.popup.Popup
```

```
    onClick(event)
```

```
    onDownloadBtnClick(sender=None)
```

```
class flare.viur.widgets.file.FilePreviewImage(file=None, size=150, *args, **kwargs)
```

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

```
    setFile(file)
```

```
    download()
```

```
    onClick(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(req)
```

Internal callback - the actual upload url (retrieved by calling /file/getUploadURL) is known.

```
    onLoad(*args, **kwargs)
```

Internal callback - The state of our upload changed.

```
    onUploadAdded(req)
```

```
    onProgress(event)
```

Internal callback - further bytes have been transmitted.

```
    onSuccess(*args, **kwargs)
```

Internal callback - The upload succeeded.

onFailed(*errorCode*, *args, **kwargs)

replaceWithMessage(*message*, *isSuccess*)

class flare.viur.widgets.file.**FileLeafWidget**(*module*, *data*, *structure*, *widget*, *args, **kwargs)

Bases: *flare.viur.widgets.tree.TreeLeafWidget*

EntryIcon()

Leafs have a different Icon.

setStyle()

Leaf have a different color.

class flare.viur.widgets.file.**FileNodeWidget**(*module*, *data*, *structure*, *widget*, *args, **kwargs)

Bases: *flare.viur.widgets.tree.TreeNodeWidget*

setStyle()

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()

onStartSearch(*searchStr*, *args, **kwargs)

getChildKey(*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, id, *args, **kwargs*)

Bases: *flare.button.Button*

Extended version for a button with a text and icon, which binds itself to an event function.

onClick(*sender=None*)

onSelectionActivated(*selectWdg, selection*)

static isSuitableFor(*modul, handler, actionName*)

resetLoadingState()

class flare.viur.widgets.htmleditor.**HtmlEditor**(*args, **kwargs)

Bases: *flare.html5.Textarea*

initSources = **False**

_attachSummernote(*retry=0*)

onAttach()

onDetach()

onEditorChange(*e, *args, **kwargs*)

_getValue()

_setValue(*val*)

enable()

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

disable()

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

<i>ListWidget</i>	Provides the interface to list-applications.
<i>SkellistItem</i>	Extended version for a button with a text and icon, which binds itself to an event function.
<i>ListSelection</i>	

```
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(callback, multi=True, allow=None)
```

Configures the widget as selector for a relationalBone and shows it.

```
onAcceptSelectionChanged(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()
```

```
onActiveSelectionChanged(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()
```

```
onRequestList(skellist)
```

```
onActiveSelectionChanged(event, *args, **kwargs)
```

```
activateSelection(widget)
```

```
reloadList()
```

```
buildListSelection()
```

```
onApplyfilterChanged(value, *args, **kwargs)
```

```
onAcceptSelectionChanged(event, *args, **kwargs)
```

```
onActiveButtonChanged(event, *args, **kwargs)
```

```
acceptSelection()
```

```
setContent(widget)
```

flare.viur.widgets.tree

Module Contents

Classes

<i>TreeItemWidget</i>	
<i>TreeLeafWidget</i>	
<i>TreeNodeWidget</i>	
<i>TreeWidget</i>	Base Widget that renders a tree.
<i>BrowserLeafWidget</i>	
<i>BrowserNodeWidget</i>	
<i>BreadcrumbNodeWidget</i>	
<i>TreeBrowserWidget</i>	Base Widget that renders a tree.

class flare.viur.widgets.tree.**TreeItemWidget**(*module, data, structure, widget, *args, **kwargs*)
Bases: *flare.html5.Li*

setStyle()
Is used to define the appearance of the element.

additionalDropAreas()
Drag and Drop areas.

markDraggedElement()
Mark the current dragged Element.

unmarkDraggedElement()

onDragStart(*event*)

onDragEnd(*event*)

onDragOver(*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(*event*)
Remove all drop indicating classes.

disableDragMarkers()

moveRequest(*params*)
Performs the move operation with the provided params.

onDrop(*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()

toggleArrow()

buildDescription()

Creates the visual representation of our entry.

onClick(*event*)

onDblClick(*event*)

toggleExpand()

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()

Leaf have a different color.

toggleArrow()

Leafes cant be toggled.

EntryIcon()

Leafs have a different Icon.

moveRequest(*params*)

Performs the move operation with the provided params.

onDragOver(*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.

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

requestStructure()

receivedStructure(*resp*)

setSelector(*callback*, *multi=True*, *allow=None*)

Configures the widget as selector for a relationalBone and shows it.

selectorReturn()

Returns the current selection to the callback configured with *setSelector*.

onKeyDown(*event*)

onKeyUp(*event*)

getActions()

Returns a list of actions that are being set for the ActionBar. Override this to provide additional actions.

clearSelection()

Empties the current selection.

extendSelection(*element*)

Extends the current selection to element.

This is normally done by clicking or tabbing on an element.

activateSelection(*element*)

Activates the current selection or element.

An activation mostly is an action like selecting or editing an item. This is normally done by double-clicking an element.

requestChildren(*element*)

_showErrorMsg(*req=None*, *code=None*)

Removes all currently visible elements and displays an error message

onDataChanged(*module*, **args*, ***kwargs*)

onAttach()

onDetach()

itemForKey(*key*, *elem=None*)

Returns the HierarchyWidget displaying the entry with the given key. :param key: The key (id) of the item.
:type key: str :returns: HierarchyItem

onSetDefaultRootNode(*req*)

We requested the list of rootNodes for that module and that request just finished. Parse the response and set our rootNode to the first rootNode received.

setRootNode(*rootNode*, *node=None*)

Set the currently displayed hierarchy to 'rootNode'. :param rootNode: Key of the rootNode which children we shall display :type rootNode: str

reloadData()

Reload the data were displaying.

loadNode(*node*, *skelType=None*, *cursor=None*, *overrideParams=None*)

Fetch the (direct) children of the given node. Once the list is received, append them to their parent node.
:param node: Key of the node to fetch :type node: str

_onRequestSucceeded(*req*)

The NetworkRequest for a (sub)node finished. Create a new HierarchyItem for each entry received and add them to our view

onDrop(*event*)

We got a drop event. Make that item a direct child of our rootNode

onDragOver(*event*)

Allow dropping children on the rootNode

getChildKey(*widget*)

Order by sortindex

static canHandle(*moduleName, moduleInfo*)

class flare.viur.widgets.tree.**BrowserLeafWidget**(*module, data, structure, widget, *args, **kwargs*)

Bases: *TreeLeafWidget*

setStyle()

Leaf have a different color.

class flare.viur.widgets.tree.**BrowserNodeWidget**(*module, data, structure, widget, *args, **kwargs*)

Bases: *TreeNodeWidget*

setStyle()

Is used to define the appearance of the element.

class flare.viur.widgets.tree.**BreadcrumbNodeWidget**(*module, data, structure, widget, *args, **kwargs*)

Bases: *TreeNodeWidget*

setStyle()

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

reloadData()

Reload the data were displaying.

rebuildPath()

Rebuild the displayed path-list.

onPathRequestSucceded(*req*)

Rebuild the displayed path-list according to request data

activateSelection(*element*)

Activates the current selection or element.

An activation mostly is an action like selecting or editing an item. This is normally done by double-clicking an element.

static canHandle(*module, moduleInfo*)

Submodules

`flare.viur.formatString`

Module Contents

Functions

<code>formatString</code> (format, data[, structure, language])	Central entryPoint
<code>formatStringHandler</code> ($\rightarrow$ str)	
<code>displayStringHandler</code> ($\rightarrow$ [flare.html5.Widget])	
<code>evalStringHandler</code> (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') $\rightarrow$ str

`flare.viur.formatString.displayStringHandler`(display: str, value: Dict, structure: Dict, language: str = 'de') $\rightarrow$ [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

<code>collectBoneErrors</code> (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

<code>ViurForm</code>	Handles an input form for a VIUR skeleton.
<code>ViurFormBone</code>	
<code>ViurFormSubmit</code>	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(event)

onBoneChange(bone)

_setModuleName(val)

_setActionname(val)

_setFormname(val)

buildForm()

Builds a form with save button.

buildInternalForm()

Builds only the form.

registerField(*key*, *widget*)

update()

Updates current form view state regarding conditional input fields.

submitForm()

unserialize(*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(*all*=*False*) → *Dict*

Serializes all bone's values into a dict to be sent to ViUR or the be evaluated.

actionSuccess(*req*)

handleErrors()

createFormSuccessMessage()

createFormErrorMessage()

actionFailed(*req*, **args*, ***kwargs*)

onFormSuccess(*event*)

onSubmitStatusChanged(*value*, **args*, ***kwargs*)

class flare.viur.forms.**ViurFormBone**(*boneName*=*None*, *form*=*None*, *defaultvalue*=*None*, *hidden*=*False*, *filter*=*None*)

Bases: *flare.html5.Div*

onAttach()

onChange(*event*, **args*, ***kwargs*)

unserialize(*data*=*None*)

serialize()

_setBonename(*val*)

_setLabel(*val*)

_setPlaceholder(*val*)

_setHide(*val*)

_setValue(*val*)

setInvalid(*errors*=*None*)

setValid()

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()
sendViurForm(sender=None)
onSubmitStatusChanged(value, *args, **kwargs)
```

`flare.viur.formtooltip`

Module Contents

Classes

<i>ToolTip</i>	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(event)
```

```
_setDisabled(disabled)
```

Package Contents

Classes

<i>PriorityQueue</i>

Functions

<i>formatString</i> (format, data[, structure, language])	Central entryPoint
<i>displayStringHandler</i> (→ [flare.html5.Widget])	

Attributes

<i>conf</i>	# A value displayed as "empty value"
<i>BoneSelector</i>	
<i>ModuleWidgetSelector</i>	
<i>DisplayDelegateSelector</i>	

```

class flare.viur.PriorityQueue
    Bases: object
    insert(priority, validateFunc, generator)
    select(*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(*event*, *args, **kwargs)

addButton(*name*, *btnStr*)

buttonClicked(*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(*event*, *args, **kwargs)

class flare.widgets.buttonbar.**ButtonBarSearch**

Bases: *flare.html5.Div*

applyFilter(*widget*)

onApplyfilterChanged(*event*, *args, **kwargs)

onActiveButtonChanged(*event*, *args, **kwargs)

Submodules

flare.button

Flare-styled button Widgets.

Module Contents

Classes

Button

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(*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(*event*, *widget=None*)

resetIcon()

update()

```

    _setIcon(icon)
    _getIcon()
    _setText(text)
    _getText()

```

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

Cache

Plan

```
class flare.cache.Cache
```

```
    Bases: object
```

```
    updateStructure(module, structure)
```

```
    update(module, key, data, structure=None)
```

```
    lookup(module, key='current')
```

```
    struct(module)
```

```
    start(plan, finishHandler=None, failureHandler=None)
```

```
    finish(plan)
```

```
    require(*args)
```

```
    invalidate(*args)
```

```
    onDataChange(module, key=None, **kwargs)
```

```
    request(*args, finishHandler=None, failureHandler=None)
```

```
class flare.cache.Plan(module, action, params=None, follow=None, alias='current', local=True,
                      force=False)
```

```
    Bases: object
```

```
    run(cache)
```

```
    finish(cache)
```

`_onRequestSuccess(req)`

`_onRequestFailure(req, code)`

`flare.config`

Flare configuration.

Module Contents

Functions

<code>updateConf(other)</code>	Merges other into conf.
--------------------------------	-------------------------

Attributes

<code>conf</code>

`flare.config.updateConf(other: Dict)`

Merges other into conf.

`flare.config.conf`

`flare.debug`

still WIP

Module Contents

Functions

<code>debug([element])</code>	Debug popup
<code>debugElement(element)</code>	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***EventDispatcher*

Base class for event notifier.

```
class flare.event.EventDispatcher(name)
```

Bases: object

Base class for event notifier.

```
_genTargetFuncName()
```

Return the name of the function called on the receiving object.

```
register(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(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(*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(*args, **kwargs)
requestSuccess(req)
_requestFailed(req, *args, **kwargs)
onListStatusChanged(event, *args, **kwargs)
getDescrFromValue(definition, val)
buildSelectDescr(skel, structure)

class flare.handler.ListHandler(module, action, params=(), eventName='listUpdated', secure=False)
    Bases: requestHandler
    reload()
    filter(filterparams)
    getCurrentAmount()
    requestNext()
    requestSuccess(req)

class flare.handler.SyncHandler
    Bases: object
    static request(url, params=None, jsonResult=None)
    genReqStr(params)
    _request(url, params)
    onCompletion(text)
    onError(text, code)
```

[flare.i18n](#)

Internationalization tools to easily implement multi-language applications.

Module Contents

Functions

<i>buildTranslations</i> (pathToFolder)	
<i>translate</i> (key[, fallback])	Tries to translate the given string in the currently selected language.
<i>addTranslation</i> (lang, a[, b])	Adds or updates new translations.
<i>setLanguage</i> (lang)	Sets the current language to lang.
<i>getLanguage</i> ()	Returns the current language.

Attributes

<code>_currentLanguage</code>
<code>_currentLanguage</code>
<code>_currentLanguage</code>
<code>_currentLanguage</code>
<code>_runtimeTranslations</code>
<code>_lngMap</code>

`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(value)

_setTitle(val)

Advisory information associated with the element.

Parameters

val – str

getIcon()

replaceSVG(icondata)

requestFallBack(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(value)

_setTitle(val)

Advisory information associated with the element.

Parameters

val – str

_setFallback(val)

onError()

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(badge)

_getBadge()

flare.ignite

Flare-specific form Widgets with specialized classes and behavior.

Module Contents

Classes

<i>Label</i>
<i>Input</i>
<i>Switch</i>
<i>Check</i>
<i>Radio</i>
<i>Select</i>
<i>Textarea</i>
<i>Progress</i>
<i>Item</i>
<i>Table</i>

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(value)

_getChecked()

```
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(row)

    prepareCol(row, col)

    fastGrid(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(event)

    onFocus(event)

    onDetach()
```

flare.intersectionObserver

Module Contents

Classes

<i>IntersectionObserver</i>	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(*widget*)
unobserve(*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> (→ None)	Call this before first usage of logging or getLogger().
<i>getLogger</i> (→ Any)	Creates a child logger of our 'root' logger with a name.

Attributes

loggers

```
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() → 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(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

Tools for handling Ajax/fetch-network 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>fetch_json</i> (url, callback, **kwargs)	Wrapper that performs a fetch request (with all parameters related to [pyfetch])(https://pyodide.org/en/stable/usage/api/python-api.html#pyodide.http.pyfetch).
<i>NiceError</i> (req, code[, params, then])	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)	
<i>getUrlHashAsObject</i> (urlHash)	
<i>setUrlHash</i> (hash[, param])	

Attributes

<i>skeyRequestQueue</i>

flare.network.**fetch_json**(url, callback, **kwargs)

Wrapper that performs a fetch request (with all parameters related to [pyfetch])(<https://pyodide.org/en/stable/usage/api/python-api.html#pyodide.http.pyfetch>).

Parameters

- **url** – URL to fetch from.

- **then** – Callback for getting the JSON object as parameter, status-code and status-text as parameters. JSON is None in case of an error.
- **catch** – Optional callback for failure, getting the [FetchResponse](https://pyodide.org/en/stable/usage/api/python-api.html#pyodide.http.FetchResponse) as parameter.
- **kwargs** – Any kwargs being passed to pyfetch.

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()

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(**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()

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(*url, params, skey*)

Internal function performing the actual AJAX request.

onCompletion(*text*)

Internal hook for the AJAX call.

onError(*text, code*)

Internal hook for the AJAX call.

onTimeout(*text*)

Internal hook for the AJAX call.

clear()

onFinished(*success*)

class flare.network.**requestGroup**(*callback=None*)

addRequest(*request*)

call()

onFinished(*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

<i>ObservableValue</i>
<i>StateHandler</i>

class flare.observable.**ObservableValue**(*key, value=None*)

Bases: object

value

setValue(*value*)

class flare.observable.**StateHandler**(*initialize=(), widget=None*)

```

updateState(key, value)

getState(key, empty=None)

register(key, widget)

unregister(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(icon)
```

```
_getIcon()
```

```
_setText(text)
```

```
_getText()
```

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()

onDetach()

onDocumentKeyDown(event)

close()

onClose()

```
class flare.popup.Prompt(text, value="", successHandler=None, abortHandler=None, successLbl=None,
                        abortLbl=None, placeholder="", *args, **kwargs)
```

Bases: *Popup*

onKeyDown(event)

onKeyUp(event)

onDocumentKeyDown(event)

onOkay(*args, **kwargs)

onCancel(*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()

onOkBtnClick()

onKeyDown(*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(*event*)

onDocumentKeyDown(*event*)

drop()

onYesClicked(**args*, ***kwargs*)

onNoClicked(**args*, ***kwargs*)

class flare.popup.**TextareaDialog**(*text*, *value=""*, *successHandler=None*, *abortHandler=None*, *successLbl=None*, *abortLbl=None*, **args*, ***kwargs*)

Bases: [Popup](#)

onDocumentKeyDown(*event*)

onOkay(**args*, ***kwargs*)

onCancel(**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(**args*, ***kwargs*)

onCancel(**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(priority, validateFunc, generator)
```

```
    select(*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: None | Dict[str, Any] = None)
```

```
    Safely evaluate an expression from an untrusted party.
```

```
    _BoolOp(node, names)
```

```
        Handling ast.BoolOp in a Pythonic style.
```

```
    callNode(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(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(node, names)
```

```
    execute(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(*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(*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**

<i>unescape</i>(val[, maxLength])	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])	Parses a value as int.
<i>parseFloat</i>(s[, ret])	Parses a value as float.
<i>createWorker</i>(pythonCode[, callback, errorCallback, ...])	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

<i>updateConf</i> (other)	Merges other into conf.
<i>_html5WidgetSetHidden</i> (widget, hidden)	
<i>_html5WidgetGetHidden</i> (widget)	
<i>_html5WidgetSetDisabled</i> (widget, disabled)	
<i>loadProjectConf</i> (aconf)	Update the default flare config with project configuration.
<i>bindApp</i> (app, injectdata)	Add the app instance as "app" to global config.

Attributes

conf

class flare.Cache

Bases: object

updateStructure(module, structure)

update(module, key, data, structure=None)

lookup(module, key='current')

struct(module)

start(plan, finishHandler=None, failureHandler=None)

finish(plan)

require(*args)

invalidate(*args)

onDataChanged(module, key=None, **kwargs)

request(*args, finishHandler=None, failureHandler=None)

flare.conf

`flare.updateConf(other: Dict)`

Merges other into conf.

`flare._html5WidgetSetHidden(widget, hidden)`

`flare._html5WidgetGetHidden(widget)`

`flare._html5WidgetSetDisabled(widget, disabled)`

`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, 34
- flare.button, 148
- flare.cache, 149
- flare.config, 150
- flare.debug, 150
- flare.event, 151
- flare.handler, 151
- flare.html5, 34
- flare.html5.core, 34
- flare.html5.svg, 71
- flare.i18n, 152
- flare.icons, 154
- flare.ignite, 155
- flare.input, 156
- flare.intersectionObserver, 157
- flare.log, 157
- flare.network, 159
- flare.observable, 162
- flare.popout, 163
- flare.popup, 164
- flare.priorityqueue, 165
- flare.safeeval, 166
- flare.translations, 112
- flare.translations.de, 112
- flare.translations.en, 113
- flare.utils, 167
- flare.views, 113
- flare.views.helpers, 113
- flare.views.view, 114
- flare.viur, 115
- flare.viur.bones, 115
- flare.viur.bones.base, 115
- flare.viur.bones.boolean, 118
- flare.viur.bones.color, 119
- flare.viur.bones.date, 120
- flare.viur.bones.email, 121
- flare.viur.bones.numeric, 122
- flare.viur.bones.password, 123
- flare.viur.bones.raw, 124
- flare.viur.bones.record, 125
- flare.viur.bones.relational, 126

- flare.viur.bones.select, 130
- flare.viur.bones.spatial, 131
- flare.viur.bones.string, 132
- flare.viur.bones.text, 133
- flare.viur.formatString, 143
- flare.viur.formconf, 143
- flare.viur.formerrors, 144
- flare.viur.forms, 144
- flare.viur.formtooltip, 146
- flare.viur.widgets, 134
- flare.viur.widgets.file, 134
- flare.viur.widgets.htmleditor, 136
- flare.viur.widgets.list, 137
- flare.viur.widgets.tree, 139
- flare.widgets, 147
- flare.widgets.buttonbar, 147

W

- webworker_scripts, 170

Symbols

- `__BoolOp()` (*flare.safeeval.SafeEval* method), 166
- `__Del` (*class in flare.html5*), 99
- `__Del` (*class in flare.html5.core*), 57
- `__WidgetClassWrapper` (*class in flare.html5*), 83
- `__WidgetClassWrapper` (*class in flare.html5.core*), 42
- `__WidgetDataWrapper` (*class in flare.html5*), 84
- `__WidgetDataWrapper` (*class in flare.html5.core*), 43
- `__WidgetStyleWrapper` (*class in flare.html5*), 84
- `__WidgetStyleWrapper` (*class in flare.html5.core*), 43
- `__collectChildren()` (*flare.html5.Widget* method), 90
- `__collectChildren()` (*flare.html5.core.Widget* method), 49
- `__domParser` (*in module flare.html5*), 83
- `__domParser` (*in module flare.html5.core*), 42
- `__getitem__()` (*flare.html5.ColWrapper* method), 109
- `__getitem__()` (*flare.html5.RowWrapper* method), 109
- `__getitem__()` (*flare.html5.Widget* method), 85
- `__getitem__()` (*flare.html5.core.ColWrapper* method), 68
- `__getitem__()` (*flare.html5.core.RowWrapper* method), 68
- `__getitem__()` (*flare.html5.core.Widget* method), 44
- `__iter__()` (*flare.html5.Widget* method), 86
- `__iter__()` (*flare.html5.core.Widget* method), 44
- `__reVarReplacer` (*in module flare.html5*), 111
- `__reVarReplacer` (*in module flare.html5.core*), 70
- `__setitem__()` (*flare.html5.ColWrapper* method), 109
- `__setitem__()` (*flare.html5.Widget* method), 85
- `__setitem__()` (*flare.html5._WidgetDataWrapper* method), 84
- `__setitem__()` (*flare.html5._WidgetStyleWrapper* method), 85
- `__setitem__()` (*flare.html5.core.ColWrapper* method), 68
- `__setitem__()` (*flare.html5.core.Widget* method), 44
- `__setitem__()` (*flare.html5.core._WidgetDataWrapper* method), 43
- `__setitem__()` (*flare.html5.core._WidgetStyleWrapper* method), 44
- `__str__()` (*flare.html5.TextNode* method), 83
- `__str__()` (*flare.html5.Widget* method), 86
- `__str__()` (*flare.html5.core.TextNode* method), 42
- `__str__()` (*flare.html5.core.Widget* method), 44
- `__tags` (*in module flare.html5*), 111
- `__tags` (*in module flare.html5.core*), 70
- `__addEntriesFromSelection()` (*flare.viur.bones.relational.RelationalMultiEditWidget* method), 127
- `__attachSummernote()` (*flare.viur.widgets.htmleditor.HtmlEditor* method), 137
- `__attrAlt` (*class in flare.html5*), 94
- `__attrAlt` (*class in flare.html5.core*), 53
- `__attrAutocomplete` (*class in flare.html5*), 94
- `__attrAutocomplete` (*class in flare.html5.core*), 53
- `__attrAutofocus` (*class in flare.html5*), 94
- `__attrAutofocus` (*class in flare.html5.core*), 53
- `__attrCharset` (*class in flare.html5*), 93
- `__attrCharset` (*class in flare.html5.core*), 52
- `__attrChecked` (*class in flare.html5*), 94
- `__attrChecked` (*class in flare.html5.core*), 53
- `__attrCite` (*class in flare.html5*), 93
- `__attrCite` (*class in flare.html5.core*), 52
- `__attrDatetime` (*class in flare.html5*), 93
- `__attrDatetime` (*class in flare.html5.core*), 52
- `__attrDimensions` (*class in flare.html5*), 96
- `__attrDimensions` (*class in flare.html5.core*), 55
- `__attrDisabled` (*class in flare.html5*), 94
- `__attrDisabled` (*class in flare.html5.core*), 53
- `__attrFor` (*class in flare.html5*), 95
- `__attrFor` (*class in flare.html5.core*), 54
- `__attrForm` (*class in flare.html5*), 94
- `__attrForm` (*class in flare.html5.core*), 52
- `__attrFormhead` (*class in flare.html5*), 95
- `__attrFormhead` (*class in flare.html5.core*), 54
- `__attrHref` (*class in flare.html5*), 96
- `__attrHref` (*class in flare.html5.core*), 54
- `__attrIndeterminate` (*class in flare.html5*), 94
- `__attrIndeterminate` (*class in flare.html5.core*), 53
- `__attrInputs` (*class in flare.html5*), 95
- `__attrInputs` (*class in flare.html5.core*), 54
- `__attrLabel` (*class in flare.html5*), 93
- `__attrLabel` (*class in flare.html5.core*), 52

- `_attrMedia` (class in `flare.html5`), 96
- `_attrMedia` (class in `flare.html5.core`), 55
- `_attrMultimedia` (class in `flare.html5`), 97
- `_attrMultimedia` (class in `flare.html5.core`), 55
- `_attrMultiple` (class in `flare.html5`), 95
- `_attrMultiple` (class in `flare.html5.core`), 53
- `_attrName` (class in `flare.html5`), 94
- `_attrName` (class in `flare.html5.core`), 53
- `_attrRel` (class in `flare.html5`), 97
- `_attrRel` (class in `flare.html5.core`), 56
- `_attrRequired` (class in `flare.html5`), 95
- `_attrRequired` (class in `flare.html5.core`), 53
- `_attrSize` (class in `flare.html5`), 95
- `_attrSize` (class in `flare.html5.core`), 54
- `_attrSrc` (class in `flare.html5`), 97
- `_attrSrc` (class in `flare.html5.core`), 56
- `_attrSvgDimensions` (class in `flare.html5.svg`), 72
- `_attrSvgPoints` (class in `flare.html5.svg`), 73
- `_attrSvgStyles` (class in `flare.html5.svg`), 74
- `_attrSvgTransform` (class in `flare.html5.svg`), 73
- `_attrSvgViewBox` (class in `flare.html5.svg`), 72
- `_attrSvgXlink` (class in `flare.html5.svg`), 74
- `_attrTarget` (class in `flare.html5`), 96
- `_attrTarget` (class in `flare.html5.core`), 55
- `_attrType` (class in `flare.html5`), 96
- `_attrType` (class in `flare.html5.core`), 55
- `_attrUsemap` (class in `flare.html5`), 97
- `_attrUsemap` (class in `flare.html5.core`), 55
- `_attrValue` (class in `flare.html5`), 94
- `_attrValue` (class in `flare.html5.core`), 53
- `_body` (in module `flare.html5`), 98
- `_body` (in module `flare.html5.core`), 57
- `_buildTags` (in module `flare.html5`), 112
- `_buildTags` (in module `flare.html5.core`), 70
- `_currentLanguage` (in module `flare.i18n`), 153
- `_formatCurrencyValue` (in module `flare.viur.bones.numeric`), 122
- `_genTargetFuncName` (`flare.event.EventDispatcher` method), 151
- `_getAccept` (`flare.html5.Input` method), 103
- `_getAccept` (`flare.html5.core.Input` method), 62
- `_getAccept_attrCharset` (`flare.html5.Form` method), 103
- `_getAccept_attrCharset` (`flare.html5.core.Form` method), 62
- `_getAccesskey` (`flare.html5.Widget` method), 88
- `_getAccesskey` (`flare.html5.core.Widget` method), 47
- `_getAction` (`flare.html5.Form` method), 103
- `_getAction` (`flare.html5.core.Form` method), 62
- `_getAlt` (`flare.html5._attrAlt` method), 94
- `_getAlt` (`flare.html5.core._attrAlt` method), 53
- `_getAsync` (`flare.html5.Script` method), 108
- `_getAsync` (`flare.html5.core.Script` method), 67
- `_getAutocomplete` (`flare.html5._attrAutocomplete` method), 95
- `_getAutocomplete` (`flare.html5.core._attrAutocomplete` method), 53
- `_getAutofocus` (`flare.html5._attrAutofocus` method), 94
- `_getAutofocus` (`flare.html5.core._attrAutofocus` method), 53
- `_getAutoplay` (`flare.html5._attrMultimedia` method), 97
- `_getAutoplay` (`flare.html5.core._attrMultimedia` method), 55
- `_getBadge` (`flare.icons.BadgeIcon` method), 155
- `_getBlockquote` (`flare.html5.Blockquote` method), 98
- `_getBlockquote` (`flare.html5.core.Blockquote` method), 57
- `_getCell` (`flare.html5.Table` method), 110
- `_getCell` (`flare.html5.core.Table` method), 68
- `_getChallenge` (`flare.html5.Keygen` method), 106
- `_getChallenge` (`flare.html5.core.Keygen` method), 64
- `_getCharset` (`flare.html5._attrCharset` method), 93
- `_getCharset` (`flare.html5.core._attrCharset` method), 52
- `_getChecked` (`flare.html5._attrChecked` method), 94
- `_getChecked` (`flare.html5.core._attrChecked` method), 53
- `_getChecked` (`flare.ignite.Switch` method), 155
- `_getCite` (`flare.html5._attrCite` method), 93
- `_getCite` (`flare.html5.core._attrCite` method), 52
- `_getClass` (`flare.html5.Widget` method), 89
- `_getClass` (`flare.html5.core.Widget` method), 48
- `_getCols` (`flare.html5.Textarea` method), 105
- `_getCols` (`flare.html5.core.Textarea` method), 63
- `_getColspan` (`flare.html5.Td` method), 109
- `_getColspan` (`flare.html5.core.Td` method), 68
- `_getContent` (`flare.html5.Meta` method), 107
- `_getContent` (`flare.html5.core.Meta` method), 66
- `_getContenteditable` (`flare.html5.Widget` method), 88
- `_getContenteditable` (`flare.html5.core.Widget` method), 47
- `_getContextmenu` (`flare.html5.Widget` method), 88
- `_getContextmenu` (`flare.html5.core.Widget` method), 47
- `_getControls` (`flare.html5._attrMultimedia` method), 97
- `_getControls` (`flare.html5.core._attrMultimedia` method), 56
- `_getCoords` (`flare.html5.Area` method), 98
- `_getCoords` (`flare.html5.core.Area` method), 57
- `_getCrossorigin` (`flare.html5.Img` method), 105
- `_getCrossorigin` (`flare.html5.core.Img` method), 64
- `_getCx` (`flare.html5.svg._attrSvgDimensions` method), 73

_getCy() (flare.html5.svg._attrSvgDimensions method), 73
 _getD() (flare.html5.svg.SvgPath method), 75
 _getData() (flare.html5.Widget method), 86
 _getData() (flare.html5.core.Widget method), 45
 _getDatetime() (flare.html5._attrDatetime method), 94
 _getDatetime() (flare.html5.core._attrDatetime method), 52
 _getDefault() (flare.html5.Track method), 110
 _getDefault() (flare.html5.core.Track method), 69
 _getDefaultValues() (in module flare.viur.bones.relational), 126
 _getDefer() (flare.html5.Script method), 108
 _getDefer() (flare.html5.core.Script method), 67
 _getDir() (flare.html5.Widget method), 88
 _getDir() (flare.html5.core.Widget method), 47
 _getDisabled() (flare.html5.TextNode method), 83
 _getDisabled() (flare.html5.Widget method), 87
 _getDisabled() (flare.html5.core.TextNode method), 42
 _getDisabled() (flare.html5.core.Widget method), 46
 _getDownload() (flare.html5.A method), 97
 _getDownload() (flare.html5.core.A method), 56
 _getDraggable() (flare.html5.Widget method), 87
 _getDraggable() (flare.html5.core.Widget method), 46
 _getDropzone() (flare.html5.Widget method), 87
 _getDropzone() (flare.html5.core.Widget method), 46
 _getEnctype() (flare.html5.Form method), 103
 _getEnctype() (flare.html5.core.Form method), 62
 _getFill() (flare.html5.svg._attrSvgStyles method), 74
 _getFor() (flare.html5._attrFor method), 95
 _getFor() (flare.html5.core._attrFor method), 54
 _getForm() (flare.html5._attrForm method), 94
 _getForm() (flare.html5.core._attrForm method), 52
 _getFormaction() (flare.html5._attrFormhead method), 95
 _getFormaction() (flare.html5.core._attrFormhead method), 54
 _getFormenctype() (flare.html5._attrFormhead method), 95
 _getFormenctype() (flare.html5.core._attrFormhead method), 54
 _getFormmethod() (flare.html5._attrFormhead method), 95
 _getFormmethod() (flare.html5.core._attrFormhead method), 54
 _getFormnovalidate() (flare.html5._attrFormhead method), 96
 _getFormnovalidate() (flare.html5.core._attrFormhead method), 54
 _getFormtarget() (flare.html5._attrFormhead method), 96
 _getFormtarget() (flare.html5.core._attrFormhead method), 54
 _getHeight() (flare.html5._attrDimensions method), 96
 _getHeight() (flare.html5.core._attrDimensions method), 55
 _getHeight() (flare.html5.svg._attrSvgDimensions method), 73
 _getHidden() (flare.html5.Widget method), 87
 _getHidden() (flare.html5.core.Widget method), 46
 _getHigh() (flare.html5.Meter method), 107
 _getHigh() (flare.html5.core.Meter method), 66
 _getHref() (flare.html5._attrHref method), 96
 _getHref() (flare.html5.core._attrHref method), 55
 _getHreflang() (flare.html5._attrHref method), 96
 _getHreflang() (flare.html5.core._attrHref method), 55
 _getIcon() (flare.button.Button method), 149
 _getIcon() (flare.html5.Command method), 98
 _getIcon() (flare.html5.core.Command method), 57
 _getIcon() (flare.popout.Popout method), 163
 _getId() (flare.html5.Widget method), 89
 _getId() (flare.html5.core.Widget method), 47
 _getIndeterminate() (flare.html5._attrIndeterminate method), 94
 _getIndeterminate() (flare.html5.core._attrIndeterminate method), 53
 _getIsmap() (flare.html5.Img method), 105
 _getIsmap() (flare.html5.core.Img method), 64
 _getKeytype() (flare.html5.Keygen method), 106
 _getKeytype() (flare.html5.core.Keygen method), 64
 _getKind() (flare.html5.Track method), 110
 _getKind() (flare.html5.core.Track method), 69
 _getLabel() (flare.html5._attrLabel method), 93
 _getLabel() (flare.html5.core._attrLabel method), 52
 _getLang() (flare.html5.Widget method), 87
 _getLang() (flare.html5.core.Widget method), 46
 _getList() (flare.html5.Input method), 104
 _getList() (flare.html5.core.Input method), 62
 _getLoop() (flare.html5._attrMultimedia method), 97
 _getLoop() (flare.html5.core._attrMultimedia method), 56
 _getLow() (flare.html5.Meter method), 107
 _getLow() (flare.html5.core.Meter method), 66
 _getMax() (flare.html5.Input method), 104
 _getMax() (flare.html5.Meter method), 107
 _getMax() (flare.html5.Progress method), 108
 _getMax() (flare.html5.core.Input method), 62
 _getMax() (flare.html5.core.Meter method), 66
 _getMax() (flare.html5.core.Progress method), 66
 _getMaxlength() (flare.html5._attrInputs method), 95
 _getMaxlength() (flare.html5.core._attrInputs method), 54
 _getMedia() (flare.html5._attrMedia method), 96

_getMedia() (flare.html5.core._attrMedia method), 55
 _getMethod() (flare.html5.Form method), 103
 _getMethod() (flare.html5.core.Form method), 62
 _getMin() (flare.html5.Input method), 104
 _getMin() (flare.html5.Meter method), 107
 _getMin() (flare.html5.core.Input method), 63
 _getMin() (flare.html5.core.Meter method), 66
 _getMultiple() (flare.html5._attrMultiple method), 95
 _getMultiple() (flare.html5.core._attrMultiple method), 54
 _getMuted() (flare.html5._attrMultimedia method), 97
 _getMuted() (flare.html5.core._attrMultimedia method), 56
 _getName() (flare.html5._attrName method), 94
 _getName() (flare.html5.core._attrName method), 53
 _getNovalidate() (flare.html5.Form method), 103
 _getNovalidate() (flare.html5.core.Form method), 62
 _getOpen() (flare.html5.Details method), 108
 _getOpen() (flare.html5.Dialog method), 99
 _getOpen() (flare.html5.core.Details method), 67
 _getOpen() (flare.html5.core.Dialog method), 58
 _getOptimum() (flare.html5.Meter method), 107
 _getOptimum() (flare.html5.core.Meter method), 66
 _getOptions() (flare.html5.Select method), 104
 _getOptions() (flare.html5.core.Select method), 63
 _getPathLength() (flare.html5.svg.SvgPath method), 75
 _getPattern() (flare.html5.Input method), 104
 _getPattern() (flare.html5.core.Input method), 63
 _getPlaceholder() (flare.html5._attrInputs method), 95
 _getPlaceholder() (flare.html5.core._attrInputs method), 54
 _getPlaysinline() (flare.html5._attrMultimedia method), 97
 _getPlaysinline() (flare.html5.core._attrMultimedia method), 56
 _getPoints() (flare.html5.svg._attrSvgPoints method), 73
 _getPoster() (flare.html5.Video method), 110
 _getPoster() (flare.html5.core.Video method), 69
 _getPreload() (flare.html5._attrMultimedia method), 97
 _getPreload() (flare.html5.core._attrMultimedia method), 56
 _getPreserveaspectratio() (flare.html5.svg._attrSvg viewBox method), 72
 _getR() (flare.html5.svg._attrSvgDimensions method), 73
 _getRadiogroup() (flare.html5.Command method), 98
 _getRadiogroup() (flare.html5.core.Command method), 57
 _getReadonly() (flare.html5._attrInputs method), 95
 _getReadonly() (flare.html5.core._attrInputs method), 54
 _getRel() (flare.html5._attrRel method), 97
 _getRel() (flare.html5.core._attrRel method), 56
 _getRequired() (flare.html5._attrRequired method), 95
 _getRequired() (flare.html5.core._attrRequired method), 53
 _getRole() (flare.html5.Widget method), 89
 _getRole() (flare.html5.core.Widget method), 48
 _getRows() (flare.html5.Textarea method), 105
 _getRows() (flare.html5.core.Textarea method), 63
 _getRowspan() (flare.html5.Td method), 109
 _getRowspan() (flare.html5.Tr method), 109
 _getRowspan() (flare.html5.core.Td method), 68
 _getRowspan() (flare.html5.core.Tr method), 67
 _getRx() (flare.html5.svg._attrSvgDimensions method), 73
 _getRy() (flare.html5.svg._attrSvgDimensions method), 73
 _getSandbox() (flare.html5.Iframe method), 105
 _getSandbox() (flare.html5.core.Iframe method), 64
 _getScoped() (flare.html5.Style method), 108
 _getScoped() (flare.html5.core.Style method), 67
 _getSeamless() (flare.html5.Iframe method), 105
 _getSeamless() (flare.html5.core.Iframe method), 64
 _getSelected() (flare.html5.Option method), 104
 _getSelected() (flare.html5.core.Option method), 63
 _getSelectedIndex() (flare.html5.Select method), 104
 _getSelectedIndex() (flare.html5.core.Select method), 63
 _getShape() (flare.html5.Area method), 98
 _getShape() (flare.html5.core.Area method), 57
 _getSize() (flare.html5._attrSize method), 95
 _getSize() (flare.html5.core._attrSize method), 54
 _getSizes() (flare.html5.Link method), 106
 _getSizes() (flare.html5.core.Link method), 65
 _getSpellcheck() (flare.html5.Widget method), 86
 _getSpellcheck() (flare.html5.core.Widget method), 45
 _getSrc() (flare.html5._attrSrc method), 97
 _getSrc() (flare.html5.core._attrSrc method), 56
 _getSrcdoc() (flare.html5.Iframe method), 105
 _getSrcdoc() (flare.html5.core.Iframe method), 64
 _getSrclang() (flare.html5.Track method), 110
 _getSrclang() (flare.html5.core.Track method), 69
 _getStep() (flare.html5.Input method), 104
 _getStep() (flare.html5.core.Input method), 63
 _getStroke() (flare.html5.svg._attrSvgStyles method), 74
 _getStyle() (flare.html5.Widget method), 89
 _getStyle() (flare.html5.core.Widget method), 48
 _getSvgTransform() (flare.html5.svg.SvgG method), 74
 _getTabIndex() (flare.html5.Widget method), 86

_getTabIndex() (flare.html5.core.Widget method), 45
 _getTarget() (flare.html5._attrTarget method), 96
 _getTarget() (flare.html5.core._attrTarget method), 55
 _getTargetfuncName() (flare.html5.Widget method), 85
 _getTargetfuncName() (flare.html5.core.Widget method), 44
 _getText() (flare.button.Button method), 149
 _getText() (flare.html5.TextNode method), 83
 _getText() (flare.html5.core.TextNode method), 42
 _getText() (flare.popout.Popout method), 163
 _getTitle() (flare.html5.Widget method), 86
 _getTitle() (flare.html5.core.Widget method), 45
 _getTransform() (flare.html5.svg._attrSvgTransform method), 74
 _getTranslate() (flare.html5.Widget method), 86
 _getTranslate() (flare.html5.core.Widget method), 45
 _getType() (flare.html5._attrType method), 96
 _getType() (flare.html5.core._attrType method), 55
 _getUsemap() (flare.html5._attrUsemap method), 97
 _getUsemap() (flare.html5.core._attrUsemap method), 55
 _getValue() (flare.html5._attrValue method), 94
 _getValue() (flare.html5.core._attrValue method), 53
 _getValue() (flare.viur.widgets.htmleditor.HtmlEditor method), 137
 _getVersion() (flare.html5.svg.Svg method), 74
 _getViewbox() (flare.html5.svg._attrSvgViewBox method), 72
 _getWidth() (flare.html5._attrDimensions method), 96
 _getWidth() (flare.html5.core._attrDimensions method), 55
 _getWidth() (flare.html5.svg._attrSvgDimensions method), 73
 _getWrap() (flare.html5.Textarea method), 105
 _getWrap() (flare.html5.core.Textarea method), 64
 _getX() (flare.html5.svg._attrSvgDimensions method), 73
 _getX1() (flare.html5.svg._attrSvgPoints method), 73
 _getX2() (flare.html5.svg._attrSvgPoints method), 73
 _getXlinkhref() (flare.html5.svg._attrSvgXlink method), 74
 _getXmlns() (flare.html5.svg.Svg method), 74
 _getY() (flare.html5.svg._attrSvgDimensions method), 73
 _getY1() (flare.html5.svg._attrSvgPoints method), 73
 _getY2() (flare.html5.svg._attrSvgPoints method), 73
 _head (in module flare.html5), 105
 _head (in module flare.html5.core), 64
 _html5WidgetGetHidden() (in module flare), 170
 _html5WidgetSetDisabled() (in module flare), 170
 _html5WidgetSetHidden() (in module flare), 170
 _leafTag (flare.html5.Area attribute), 98
 _leafTag (flare.html5.Br attribute), 99
 _leafTag (flare.html5.Embed attribute), 100
 _leafTag (flare.html5.Hr attribute), 101
 _leafTag (flare.html5.Img attribute), 105
 _leafTag (flare.html5.Input attribute), 103
 _leafTag (flare.html5.Link attribute), 106
 _leafTag (flare.html5.Meta attribute), 107
 _leafTag (flare.html5.Param attribute), 107
 _leafTag (flare.html5.Source attribute), 108
 _leafTag (flare.html5.Track attribute), 110
 _leafTag (flare.html5.Widget attribute), 85
 _leafTag (flare.html5.core.Area attribute), 56
 _leafTag (flare.html5.core.Br attribute), 58
 _leafTag (flare.html5.core.Embed attribute), 59
 _leafTag (flare.html5.core.Hr attribute), 60
 _leafTag (flare.html5.core.Img attribute), 64
 _leafTag (flare.html5.core.Input attribute), 62
 _leafTag (flare.html5.core.Link attribute), 65
 _leafTag (flare.html5.core.Meta attribute), 65
 _leafTag (flare.html5.core.Param attribute), 66
 _leafTag (flare.html5.core.Source attribute), 67
 _leafTag (flare.html5.core.Track attribute), 69
 _leafTag (flare.html5.core.Widget attribute), 44
 _leafTag (flare.icons.Icon attribute), 154
 _leafTag (flare.icons.SvgIcon attribute), 154
 _lngMap (in module flare.i18n), 153
 _namespace (flare.html5.Widget attribute), 85
 _namespace (flare.html5.core.Widget attribute), 44
 _namespace (flare.html5.svg.SvgWidget attribute), 74
 _onRequestFailure() (flare.cache.Plan method), 150
 _onRequestSucceeded() (flare.viur.widgets.tree.TreeWidget method), 141
 _onRequestSuccess() (flare.cache.Plan method), 149
 _request() (flare.handler.SyncHandler method), 152
 _requestFailed() (flare.handler.requestHandler method), 152
 _runtimeTranslations (in module flare.i18n), 153
 _setAccept() (flare.html5.Input method), 104
 _setAccept() (flare.html5.core.Input method), 62
 _setAccept_attrCharset() (flare.html5.Form method), 103
 _setAccept_attrCharset() (flare.html5.core.Form method), 62
 _setAccesskey() (flare.html5.Widget method), 88
 _setAccesskey() (flare.html5.core.Widget method), 47
 _setAction() (flare.html5.Form method), 103
 _setAction() (flare.html5.core.Form method), 62
 _setActionname() (flare.viur.forms.ViurForm method), 144
 _setAlt() (flare.html5._attrAlt method), 94
 _setAlt() (flare.html5.core._attrAlt method), 53
 _setAsync() (flare.html5.Script method), 108
 _setAsync() (flare.html5.core.Script method), 67

_setAutocomplete() (flare.html5._attrAutocomplete method), 95
 _setAutocomplete() (flare.html5.core._attrAutocomplete method), 53
 _setAutofocus() (flare.html5._attrAutofocus method), 94
 _setAutofocus() (flare.html5.core._attrAutofocus method), 53
 _setAutoplay() (flare.html5._attrMultimedia method), 97
 _setAutoplay() (flare.html5.core._attrMultimedia method), 56
 _setBadge() (flare.icons.BadgeIcon method), 155
 _setBlockquote() (flare.html5.Blockquote method), 98
 _setBlockquote() (flare.html5.core.Blockquote method), 57
 _setBonename() (flare.viur.forms.ViurFormBone method), 145
 _setChallenge() (flare.html5.Keygen method), 106
 _setChallenge() (flare.html5.core.Keygen method), 64
 _setCharset() (flare.html5._attrCharset method), 93
 _setCharset() (flare.html5.core._attrCharset method), 52
 _setChecked() (flare.html5._attrChecked method), 94
 _setChecked() (flare.html5.core._attrChecked method), 53
 _setChecked() (flare.ignite.Switch method), 155
 _setCite() (flare.html5._attrCite method), 93
 _setCite() (flare.html5.core._attrCite method), 52
 _setClass() (flare.html5.Widget method), 89
 _setClass() (flare.html5.core.Widget method), 48
 _setCols() (flare.html5.Textarea method), 105
 _setCols() (flare.html5.core.Textarea method), 63
 _setColspan() (flare.html5.Td method), 109
 _setColspan() (flare.html5.core.Td method), 68
 _setContent() (flare.html5.Meta method), 107
 _setContent() (flare.html5.core.Meta method), 66
 _setContenteditable() (flare.html5.Widget method), 88
 _setContenteditable() (flare.html5.core.Widget method), 47
 _setContextmenu() (flare.html5.Widget method), 88
 _setContextmenu() (flare.html5.core.Widget method), 47
 _setControls() (flare.html5._attrMultimedia method), 97
 _setControls() (flare.html5.core._attrMultimedia method), 56
 _setCoords() (flare.html5.Area method), 98
 _setCoords() (flare.html5.core.Area method), 57
 _setCrossorigin() (flare.html5.Img method), 105
 _setCrossorigin() (flare.html5.core.Img method), 64
 _setCx() (flare.html5.svg._attrSvgDimensions method), 73
 _setCy() (flare.html5.svg._attrSvgDimensions method), 73
 _setD() (flare.html5.svg.SvgPath method), 75
 _setDatetime() (flare.html5._attrDatetime method), 94
 _setDatetime() (flare.html5.core._attrDatetime method), 52
 _setDefault() (flare.html5.Track method), 110
 _setDefault() (flare.html5.core.Track method), 69
 _setDefer() (flare.html5.Script method), 108
 _setDefer() (flare.html5.core.Script method), 67
 _setDir() (flare.html5.Widget method), 88
 _setDir() (flare.html5.core.Widget method), 47
 _setDisabled() (flare.html5.TextNode method), 83
 _setDisabled() (flare.html5.Widget method), 87
 _setDisabled() (flare.html5.core.TextNode method), 42
 _setDisabled() (flare.html5.core.Widget method), 46
 _setDisabled() (flare.viur.bones.text.TextEditWidget method), 133
 _setDisabled() (flare.viur.formtooltip.ToolTip method), 146
 _setDownload() (flare.html5.A method), 97
 _setDownload() (flare.html5.core.A method), 56
 _setDraggable() (flare.html5.Widget method), 88
 _setDraggable() (flare.html5.core.Widget method), 46
 _setDropzone() (flare.html5.Widget method), 87
 _setDropzone() (flare.html5.core.Widget method), 46
 _setEnctype() (flare.html5.Form method), 103
 _setEnctype() (flare.html5.core.Form method), 62
 _setFallback() (flare.icons.Icon method), 154
 _setFill() (flare.html5.svg._attrSvgStyles method), 74
 _setFor() (flare.html5._attrFor method), 95
 _setFor() (flare.html5.core._attrFor method), 54
 _setForm() (flare.html5._attrForm method), 94
 _setForm() (flare.html5.core._attrForm method), 52
 _setFormaction() (flare.html5._attrFormhead method), 95
 _setFormaction() (flare.html5.core._attrFormhead method), 54
 _setFormenctype() (flare.html5._attrFormhead method), 95
 _setFormenctype() (flare.html5.core._attrFormhead method), 54
 _setFormmethod() (flare.html5._attrFormhead method), 96
 _setFormmethod() (flare.html5.core._attrFormhead method), 54
 _setFormname() (flare.viur.forms.ViurForm method), 144
 _setFormnovalidate() (flare.html5._attrFormhead method), 96
 _setFormnovalidate() (flare.html5.core._attrFormhead method), 54

_setFormtarget() (flare.html5._attrFormhead method), 96
 _setFormtarget() (flare.html5.core._attrFormhead method), 54
 _setHeight() (flare.html5._attrDimensions method), 96
 _setHeight() (flare.html5.core._attrDimensions method), 55
 _setHeight() (flare.html5.svg._attrSvgDimensions method), 73
 _setHidden() (flare.html5.Widget method), 87
 _setHidden() (flare.html5.core.Widget method), 46
 _setHide() (flare.viur.forms.ViurFormBone method), 145
 _setHigh() (flare.html5.Meter method), 107
 _setHigh() (flare.html5.core.Meter method), 66
 _setHref() (flare.html5._attrHref method), 96
 _setHref() (flare.html5.core._attrHref method), 55
 _setHreflang() (flare.html5._attrHref method), 96
 _setHreflang() (flare.html5.core._attrHref method), 55
 _setIcon() (flare.button.Button method), 148
 _setIcon() (flare.html5.Command method), 98
 _setIcon() (flare.html5.core.Command method), 57
 _setIcon() (flare.popout.Popout method), 163
 _setId() (flare.html5.Widget method), 89
 _setId() (flare.html5.core.Widget method), 48
 _setIndeterminate() (flare.html5._attrIndeterminate method), 94
 _setIndeterminate() (flare.html5.core._attrIndeterminate method), 53
 _setIsmap() (flare.html5.Img method), 105
 _setIsmap() (flare.html5.core.Img method), 64
 _setKeytype() (flare.html5.Keygen method), 106
 _setKeytype() (flare.html5.core.Keygen method), 65
 _setKind() (flare.html5.Track method), 110
 _setKind() (flare.html5.core.Track method), 69
 _setLabel() (flare.html5._attrLabel method), 93
 _setLabel() (flare.html5.core._attrLabel method), 52
 _setLabel() (flare.viur.forms.ViurFormBone method), 145
 _setLang() (flare.html5.Widget method), 87
 _setLang() (flare.html5.core.Widget method), 46
 _setList() (flare.html5.Input method), 104
 _setList() (flare.html5.core.Input method), 62
 _setLoop() (flare.html5._attrMultimedia method), 97
 _setLoop() (flare.html5.core._attrMultimedia method), 56
 _setLow() (flare.html5.Meter method), 107
 _setLow() (flare.html5.core.Meter method), 66
 _setMax() (flare.html5.Input method), 104
 _setMax() (flare.html5.Meter method), 107
 _setMax() (flare.html5.Progress method), 108
 _setMax() (flare.html5.core.Input method), 63
 _setMax() (flare.html5.core.Meter method), 66
 _setMax() (flare.html5.core.Progress method), 66
 _setMaxlength() (flare.html5._attrInputs method), 95
 _setMaxlength() (flare.html5.core._attrInputs method), 54
 _setMedia() (flare.html5._attrMedia method), 96
 _setMedia() (flare.html5.core._attrMedia method), 55
 _setMethod() (flare.html5.Form method), 103
 _setMethod() (flare.html5.core.Form method), 62
 _setMin() (flare.html5.Input method), 104
 _setMin() (flare.html5.Meter method), 107
 _setMin() (flare.html5.core.Input method), 63
 _setMin() (flare.html5.core.Meter method), 66
 _setModulename() (flare.viur.forms.ViurForm method), 144
 _setMultiple() (flare.html5._attrMultiple method), 95
 _setMultiple() (flare.html5.core._attrMultiple method), 54
 _setMuted() (flare.html5._attrMultimedia method), 97
 _setMuted() (flare.html5.core._attrMultimedia method), 56
 _setName() (flare.html5._attrName method), 94
 _setName() (flare.html5.core._attrName method), 53
 _setNovalidate() (flare.html5.Form method), 103
 _setNovalidate() (flare.html5.core.Form method), 62
 _setOpen() (flare.html5.Details method), 108
 _setOpen() (flare.html5.Dialog method), 99
 _setOpen() (flare.html5.core.Details method), 67
 _setOpen() (flare.html5.core.Dialog method), 58
 _setOptimum() (flare.html5.Meter method), 107
 _setOptimum() (flare.html5.core.Meter method), 66
 _setPathLength() (flare.html5.svg.SvgPath method), 75
 _setPattern() (flare.html5.Input method), 104
 _setPattern() (flare.html5.core.Input method), 63
 _setPlaceholder() (flare.html5._attrInputs method), 95
 _setPlaceholder() (flare.html5.core._attrInputs method), 54
 _setPlaceholder() (flare.viur.forms.ViurFormBone method), 145
 _setPlaysinline() (flare.html5._attrMultimedia method), 97
 _setPlaysinline() (flare.html5.core._attrMultimedia method), 56
 _setPoints() (flare.html5.svg._attrSvgPoints method), 73
 _setPoster() (flare.html5.Video method), 110
 _setPoster() (flare.html5.core.Video method), 69
 _setPreload() (flare.html5._attrMultimedia method), 97
 _setPreload() (flare.html5.core._attrMultimedia method), 56
 _setPreserveaspectratio()

(flare.html5.svg._attrSvgViewBox method), 72
 _setR() (flare.html5.svg._attrSvgDimensions method), 73
 _setRadiogroup() (flare.html5.Command method), 99
 _setRadiogroup() (flare.html5.core.Command method), 57
 _setReadOnly() (flare.html5._attrInputs method), 95
 _setReadOnly() (flare.html5.core._attrInputs method), 54
 _setRel() (flare.html5._attrRel method), 97
 _setRel() (flare.html5.core._attrRel method), 56
 _setRequired() (flare.html5._attrRequired method), 95
 _setRequired() (flare.html5.core._attrRequired method), 53
 _setRole() (flare.html5.Widget method), 89
 _setRole() (flare.html5.core.Widget method), 48
 _setRows() (flare.html5.Textarea method), 105
 _setRows() (flare.html5.core.Textarea method), 64
 _setRowspan() (flare.html5.Td method), 109
 _setRowspan() (flare.html5.Tr method), 109
 _setRowspan() (flare.html5.core.Td method), 68
 _setRowspan() (flare.html5.core.Tr method), 67
 _setRx() (flare.html5.svg._attrSvgDimensions method), 73
 _setRy() (flare.html5.svg._attrSvgDimensions method), 73
 _setSandbox() (flare.html5.Iframe method), 105
 _setSandbox() (flare.html5.core.Iframe method), 64
 _setScoped() (flare.html5.Style method), 108
 _setScoped() (flare.html5.core.Style method), 67
 _setSeamless() (flare.html5.Iframe method), 105
 _setSeamless() (flare.html5.core.Iframe method), 64
 _setSelected() (flare.html5.Option method), 104
 _setSelected() (flare.html5.core.Option method), 63
 _setShape() (flare.html5.Area method), 98
 _setShape() (flare.html5.core.Area method), 57
 _setSize() (flare.html5._attrSize method), 95
 _setSize() (flare.html5.core._attrSize method), 54
 _setSizes() (flare.html5.Link method), 106
 _setSizes() (flare.html5.core.Link method), 65
 _setSpellcheck() (flare.html5.Widget method), 87
 _setSpellcheck() (flare.html5.core.Widget method), 45
 _setSrc() (flare.html5._attrSrc method), 97
 _setSrc() (flare.html5.core._attrSrc method), 56
 _setSrcdoc() (flare.html5.Iframe method), 105
 _setSrcdoc() (flare.html5.core.Iframe method), 64
 _setSrclang() (flare.html5.Track method), 110
 _setSrclang() (flare.html5.core.Track method), 69
 _setStep() (flare.html5.Input method), 104
 _setStep() (flare.html5.core.Input method), 63
 _setStroke() (flare.html5.svg._attrSvgStyles method), 74
 _setSvgTransform() (flare.html5.svg.SvgG method), 74
 _setTabIndex() (flare.html5.Widget method), 86
 _setTabIndex() (flare.html5.core.Widget method), 45
 _setTarget() (flare.html5._attrTarget method), 96
 _setTarget() (flare.html5.core._attrTarget method), 55
 _setText() (flare.button.Button method), 149
 _setText() (flare.html5.TextNode method), 83
 _setText() (flare.html5.core.TextNode method), 42
 _setText() (flare.popout.Popout method), 163
 _setTitle() (flare.html5.Widget method), 86
 _setTitle() (flare.html5.core.Widget method), 45
 _setTitle() (flare.icons.Icon method), 154
 _setTitle() (flare.icons.SvgIcon method), 154
 _setTransform() (flare.html5.svg._attrSvgTransform method), 74
 _setTranslate() (flare.html5.Widget method), 86
 _setTranslate() (flare.html5.core.Widget method), 45
 _setType() (flare.html5._attrType method), 96
 _setType() (flare.html5.core._attrType method), 55
 _setUsemaph() (flare.html5._attrUsemaph method), 97
 _setUsemaph() (flare.html5.core._attrUsemaph method), 55
 _setValue() (flare.html5._attrValue method), 94
 _setValue() (flare.html5.core._attrValue method), 53
 _setValue() (flare.icons.Icon method), 154
 _setValue() (flare.icons.SvgIcon method), 154
 _setValue() (flare.viur.forms.ViurFormBone method), 145
 _setValue() (flare.viur.widgets.htmleditor.HtmlEditor method), 137
 _setVersion() (flare.html5.svg.Svg method), 74
 _setViewbox() (flare.html5.svg._attrSvgViewBox method), 72
 _setWidth() (flare.html5._attrDimensions method), 96
 _setWidth() (flare.html5.core._attrDimensions method), 55
 _setWidth() (flare.html5.svg._attrSvgDimensions method), 73
 _setWrap() (flare.html5.Textarea method), 105
 _setWrap() (flare.html5.core.Textarea method), 64
 _setX() (flare.html5.svg._attrSvgDimensions method), 73
 _setX1() (flare.html5.svg._attrSvgPoints method), 73
 _setX2() (flare.html5.svg._attrSvgPoints method), 73
 _setXlinkhref() (flare.html5.svg._attrSvgXlink method), 74
 _setXmlns() (flare.html5.svg.Svg method), 74
 _setY() (flare.html5.svg._attrSvgDimensions method), 73
 _setY1() (flare.html5.svg._attrSvgPoints method), 73
 _setY2() (flare.html5.svg._attrSvgPoints method), 73
 _showErrorMsg() (flare.viur.widgets.tree.TreeWidget method), 141

_tagName (flare.html5.A attribute), 97
 _tagName (flare.html5.Abbbr attribute), 99
 _tagName (flare.html5.Address attribute), 99
 _tagName (flare.html5.Area attribute), 98
 _tagName (flare.html5.Article attribute), 99
 _tagName (flare.html5.Aside attribute), 99
 _tagName (flare.html5.Audio attribute), 98
 _tagName (flare.html5.B attribute), 99
 _tagName (flare.html5.Bdi attribute), 99
 _tagName (flare.html5.Bdo attribute), 98
 _tagName (flare.html5.Blockquote attribute), 98
 _tagName (flare.html5.Br attribute), 99
 _tagName (flare.html5.Button attribute), 103
 _tagName (flare.html5.Canvas attribute), 98
 _tagName (flare.html5.Caption attribute), 99
 _tagName (flare.html5.Cite attribute), 100
 _tagName (flare.html5.Code attribute), 100
 _tagName (flare.html5.Command attribute), 98
 _tagName (flare.html5.Datalist attribute), 100
 _tagName (flare.html5.Dd attribute), 106
 _tagName (flare.html5.Details attribute), 108
 _tagName (flare.html5.Dfn attribute), 100
 _tagName (flare.html5.Dialog attribute), 99
 _tagName (flare.html5.Div attribute), 100
 _tagName (flare.html5.Dl attribute), 106
 _tagName (flare.html5.Dt attribute), 106
 _tagName (flare.html5.Em attribute), 100
 _tagName (flare.html5.Embed attribute), 100
 _tagName (flare.html5.Fieldset attribute), 103
 _tagName (flare.html5.Figcaption attribute), 100
 _tagName (flare.html5.Figure attribute), 100
 _tagName (flare.html5.Footer attribute), 100
 _tagName (flare.html5.Form attribute), 103
 _tagName (flare.html5.H1 attribute), 100
 _tagName (flare.html5.H2 attribute), 101
 _tagName (flare.html5.H3 attribute), 101
 _tagName (flare.html5.H4 attribute), 101
 _tagName (flare.html5.H5 attribute), 101
 _tagName (flare.html5.H6 attribute), 101
 _tagName (flare.html5.Header attribute), 100
 _tagName (flare.html5.Hr attribute), 101
 _tagName (flare.html5.I attribute), 101
 _tagName (flare.html5.Iframe attribute), 105
 _tagName (flare.html5.Img attribute), 105
 _tagName (flare.html5.Input attribute), 103
 _tagName (flare.html5.Ins attribute), 105
 _tagName (flare.html5.Kdb attribute), 101
 _tagName (flare.html5.Keygen attribute), 106
 _tagName (flare.html5.Label attribute), 104
 _tagName (flare.html5.Legend attribute), 101
 _tagName (flare.html5.Li attribute), 106
 _tagName (flare.html5.Link attribute), 106
 _tagName (flare.html5.Map attribute), 106
 _tagName (flare.html5.Mark attribute), 101
 _tagName (flare.html5.Menu attribute), 107
 _tagName (flare.html5.Meta attribute), 107
 _tagName (flare.html5.Meter attribute), 107
 _tagName (flare.html5.Nav attribute), 107
 _tagName (flare.html5.Noscript attribute), 101
 _tagName (flare.html5.Object attribute), 107
 _tagName (flare.html5.Ol attribute), 106
 _tagName (flare.html5.Optgroup attribute), 104
 _tagName (flare.html5.Option attribute), 104
 _tagName (flare.html5.Output attribute), 104
 _tagName (flare.html5.P attribute), 102
 _tagName (flare.html5.Param attribute), 107
 _tagName (flare.html5.Progress attribute), 107
 _tagName (flare.html5.Q attribute), 108
 _tagName (flare.html5.Rq attribute), 102
 _tagName (flare.html5.Rt attribute), 102
 _tagName (flare.html5.Ruby attribute), 102
 _tagName (flare.html5.S attribute), 102
 _tagName (flare.html5.Samp attribute), 102
 _tagName (flare.html5.Script attribute), 108
 _tagName (flare.html5.Section attribute), 102
 _tagName (flare.html5.Select attribute), 104
 _tagName (flare.html5.Small attribute), 102
 _tagName (flare.html5.Source attribute), 108
 _tagName (flare.html5.Span attribute), 108
 _tagName (flare.html5.Strong attribute), 102
 _tagName (flare.html5.Style attribute), 108
 _tagName (flare.html5.Sub attribute), 102
 _tagName (flare.html5.Summary attribute), 108
 _tagName (flare.html5.Summary attribute), 102
 _tagName (flare.html5.Sup attribute), 102
 _tagName (flare.html5.Table attribute), 109
 _tagName (flare.html5.Tbody attribute), 109
 _tagName (flare.html5.Td attribute), 109
 _tagName (flare.html5.Template attribute), 110
 _tagName (flare.html5.Textarea attribute), 105
 _tagName (flare.html5.Th attribute), 109
 _tagName (flare.html5.Thead attribute), 109
 _tagName (flare.html5.Time attribute), 110
 _tagName (flare.html5.Tr attribute), 109
 _tagName (flare.html5.Track attribute), 110
 _tagName (flare.html5.U attribute), 103
 _tagName (flare.html5.Ul attribute), 106
 _tagName (flare.html5.Var attribute), 103
 _tagName (flare.html5.Video attribute), 110
 _tagName (flare.html5.Wbr attribute), 103
 _tagName (flare.html5.Widget attribute), 85
 _tagName (flare.html5._Del attribute), 99
 _tagName (flare.html5.core.A attribute), 56
 _tagName (flare.html5.core.Abbbr attribute), 58
 _tagName (flare.html5.core.Address attribute), 58
 _tagName (flare.html5.core.Area attribute), 56
 _tagName (flare.html5.core.Article attribute), 58
 _tagName (flare.html5.core.Aside attribute), 58

_tagName (flare.html5.core.Audio attribute), 57
 _tagName (flare.html5.core.B attribute), 58
 _tagName (flare.html5.core.Bdi attribute), 58
 _tagName (flare.html5.core.Bdo attribute), 57
 _tagName (flare.html5.core.Blockquote attribute), 57
 _tagName (flare.html5.core.Br attribute), 58
 _tagName (flare.html5.core.Button attribute), 62
 _tagName (flare.html5.core.Canvas attribute), 57
 _tagName (flare.html5.core.Caption attribute), 58
 _tagName (flare.html5.core.Cite attribute), 58
 _tagName (flare.html5.core.Code attribute), 58
 _tagName (flare.html5.core.Command attribute), 57
 _tagName (flare.html5.core.Datalist attribute), 59
 _tagName (flare.html5.core.Dd attribute), 65
 _tagName (flare.html5.core.Details attribute), 67
 _tagName (flare.html5.core.Dfn attribute), 59
 _tagName (flare.html5.core.Dialog attribute), 58
 _tagName (flare.html5.core.Div attribute), 59
 _tagName (flare.html5.core.Dl attribute), 65
 _tagName (flare.html5.core.Dt attribute), 65
 _tagName (flare.html5.core.Em attribute), 59
 _tagName (flare.html5.core.Embed attribute), 59
 _tagName (flare.html5.core.Fieldset attribute), 62
 _tagName (flare.html5.core.Figcaption attribute), 59
 _tagName (flare.html5.core.Figure attribute), 59
 _tagName (flare.html5.core.Footer attribute), 59
 _tagName (flare.html5.core.Form attribute), 62
 _tagName (flare.html5.core.H1 attribute), 59
 _tagName (flare.html5.core.H2 attribute), 59
 _tagName (flare.html5.core.H3 attribute), 60
 _tagName (flare.html5.core.H4 attribute), 60
 _tagName (flare.html5.core.H5 attribute), 60
 _tagName (flare.html5.core.H6 attribute), 60
 _tagName (flare.html5.core.Header attribute), 59
 _tagName (flare.html5.core.Hr attribute), 60
 _tagName (flare.html5.core.I attribute), 60
 _tagName (flare.html5.core.Iframe attribute), 64
 _tagName (flare.html5.core.Img attribute), 64
 _tagName (flare.html5.core.Input attribute), 62
 _tagName (flare.html5.core.Ins attribute), 64
 _tagName (flare.html5.core.Kdb attribute), 60
 _tagName (flare.html5.core.Keygen attribute), 64
 _tagName (flare.html5.core.Label attribute), 63
 _tagName (flare.html5.core.Legend attribute), 60
 _tagName (flare.html5.core.Li attribute), 65
 _tagName (flare.html5.core.Link attribute), 65
 _tagName (flare.html5.core.Map attribute), 65
 _tagName (flare.html5.core.Mark attribute), 60
 _tagName (flare.html5.core.Menu attribute), 65
 _tagName (flare.html5.core.Meta attribute), 65
 _tagName (flare.html5.core.Meter attribute), 66
 _tagName (flare.html5.core.Nav attribute), 66
 _tagName (flare.html5.core.Noscript attribute), 60
 _tagName (flare.html5.core.Object attribute), 66
 _tagName (flare.html5.core.Ol attribute), 65
 _tagName (flare.html5.core.Optgroup attribute), 63
 _tagName (flare.html5.core.Option attribute), 63
 _tagName (flare.html5.core.Output attribute), 63
 _tagName (flare.html5.core.P attribute), 60
 _tagName (flare.html5.core.Param attribute), 66
 _tagName (flare.html5.core.Progress attribute), 66
 _tagName (flare.html5.core.Q attribute), 66
 _tagName (flare.html5.core.Rq attribute), 60
 _tagName (flare.html5.core.Rt attribute), 61
 _tagName (flare.html5.core.Ruby attribute), 61
 _tagName (flare.html5.core.S attribute), 61
 _tagName (flare.html5.core.Samp attribute), 61
 _tagName (flare.html5.core.Script attribute), 67
 _tagName (flare.html5.core.Section attribute), 61
 _tagName (flare.html5.core.Select attribute), 63
 _tagName (flare.html5.core.Small attribute), 61
 _tagName (flare.html5.core.Source attribute), 67
 _tagName (flare.html5.core.Span attribute), 67
 _tagName (flare.html5.core.Strong attribute), 61
 _tagName (flare.html5.core.Style attribute), 67
 _tagName (flare.html5.core.Sub attribute), 61
 _tagName (flare.html5.core.Summary attribute), 67
 _tagName (flare.html5.core.Summary attribute), 61
 _tagName (flare.html5.core.Sup attribute), 61
 _tagName (flare.html5.core.Table attribute), 68
 _tagName (flare.html5.core.Tbody attribute), 68
 _tagName (flare.html5.core.Td attribute), 68
 _tagName (flare.html5.core.Template attribute), 69
 _tagName (flare.html5.core.Textarea attribute), 63
 _tagName (flare.html5.core.Th attribute), 68
 _tagName (flare.html5.core.Thead attribute), 68
 _tagName (flare.html5.core.Time attribute), 68
 _tagName (flare.html5.core.Tr attribute), 67
 _tagName (flare.html5.core.Track attribute), 69
 _tagName (flare.html5.core.U attribute), 61
 _tagName (flare.html5.core.Ul attribute), 65
 _tagName (flare.html5.core.Var attribute), 61
 _tagName (flare.html5.core.Video attribute), 69
 _tagName (flare.html5.core.Wbr attribute), 62
 _tagName (flare.html5.core.Widget attribute), 44
 _tagName (flare.html5.core._Del attribute), 57
 _tagName (flare.html5.svg.Svg attribute), 74
 _tagName (flare.html5.svg.SvgCircle attribute), 74
 _tagName (flare.html5.svg.SvgEllipse attribute), 74
 _tagName (flare.html5.svg.SvgG attribute), 74
 _tagName (flare.html5.svg.SvgImage attribute), 75
 _tagName (flare.html5.svg.SvgLine attribute), 75
 _tagName (flare.html5.svg.SvgPath attribute), 75
 _tagName (flare.html5.svg.SvgPolygon attribute), 75
 _tagName (flare.html5.svg.SvgPolyline attribute), 75
 _tagName (flare.html5.svg.SvgRect attribute), 75
 _tagName (flare.html5.svg.SvgText attribute), 75

`_updateElem()` (*flare.html5._WidgetClassWrapper method*), 84
`_updateElem()` (*flare.html5.core._WidgetClassWrapper method*), 42

A

`A` (*class in flare.html5*), 97
`A` (*class in flare.html5.core*), 56
`Abbr` (*class in flare.html5*), 99
`Abbr` (*class in flare.html5.core*), 58
`acceptSelection()` (*flare.viur.widgets.list.ListSelection method*), 138
`actionFailed()` (*flare.viur.forms.ViurForm method*), 145
`actionSuccess()` (*flare.viur.forms.ViurForm method*), 145
`activateSelection()` (*flare.viur.widgets.list.ListSelection method*), 138
`activateSelection()` (*flare.viur.widgets.tree.TreeBrowserWidget method*), 142
`activateSelection()` (*flare.viur.widgets.tree.TreeWidget method*), 141
`addButton()` (*flare.widgets.buttonbar.ButtonBar method*), 148
`addClass()` (*flare.html5.core.Widget method*), 49
`addClass()` (*flare.html5.Widget method*), 91
`addEntry()` (*flare.viur.bones.base.BaseMultiEditWidget method*), 117
`addEntry()` (*flare.viur.bones.relational.FileMultiEditDirectWidget method*), 129
`addEventListener()` (*flare.html5.core.Widget method*), 44
`addEventListener()` (*flare.html5.Widget method*), 85
`additionalDropAreas()` (*flare.viur.widgets.tree.TreeItemWidget method*), 139
`addRequest()` (*flare.network.requestGroup method*), 162
`Address` (*class in flare.html5*), 99
`Address` (*class in flare.html5.core*), 58
`addTranslation()` (*in module flare.i18n*), 153
`addView()` (*in module flare.views.helpers*), 113
`Alert` (*class in flare.popup*), 164
`append()` (*flare.html5._WidgetClassWrapper method*), 84
`append()` (*flare.html5.core._WidgetClassWrapper method*), 43
`appendChild()` (*flare.html5.core.Widget method*), 49
`appendChild()` (*flare.html5.Widget method*), 90
`applyFilter()` (*flare.widgets.buttonbar.ButtonBarSearch method*), 148

`Area` (*class in flare.html5*), 98
`Area` (*class in flare.html5.core*), 56
`Article` (*class in flare.html5*), 99
`Article` (*class in flare.html5.core*), 58
`Aside` (*class in flare.html5*), 99
`Aside` (*class in flare.html5.core*), 58
`Audio` (*class in flare.html5*), 98
`Audio` (*class in flare.html5.core*), 57
`autoIdCounter` (*flare.html5.core.Label attribute*), 63
`autoIdCounter` (*flare.html5.Label attribute*), 104

B

`B` (*class in flare.html5*), 99
`B` (*class in flare.html5.core*), 58
`BadgeIcon` (*class in flare.icons*), 154
`BaseBone` (*class in flare.viur.bones.base*), 118
`BaseEditWidget` (*class in flare.viur.bones.base*), 116
`BaseLanguageEditWidget` (*class in flare.viur.bones.base*), 117
`BaseMultiEditWidget` (*class in flare.viur.bones.base*), 117
`BaseMultiEditWidgetEntry` (*class in flare.viur.bones.base*), 117
`BaseMultiViewWidget` (*class in flare.viur.bones.base*), 117
`BaseViewWidget` (*class in flare.viur.bones.base*), 116
`Bdi` (*class in flare.html5*), 99
`Bdi` (*class in flare.html5.core*), 58
`Bdo` (*class in flare.html5*), 98
`Bdo` (*class in flare.html5.core*), 57
`bindApp()` (*in module flare*), 170
`Blockquote` (*class in flare.html5*), 98
`Blockquote` (*class in flare.html5.core*), 57
`blur()` (*flare.html5.core.Widget method*), 51
`blur()` (*flare.html5.Widget method*), 92
`Body()` (*in module flare.html5*), 98
`Body()` (*in module flare.html5.core*), 57
`BodyCls` (*class in flare.html5*), 98
`BodyCls` (*class in flare.html5.core*), 57
`BoneSelector` (*in module flare.viur*), 147
`boneWidget()` (*flare.viur.bones.base.BaseBone method*), 118
`BooleanBone` (*class in flare.viur.bones.boolean*), 119
`BooleanEditWidget` (*class in flare.viur.bones.boolean*), 118
`BooleanViewWidget` (*class in flare.viur.bones.boolean*), 119
`Br` (*class in flare.html5*), 99
`Br` (*class in flare.html5.core*), 58
`BreadcrumbNodeWidget` (*class in flare.viur.widgets.tree*), 142
`BrowserLeafWidget` (*class in flare.viur.widgets.tree*), 142

- BrowserNodeWidget (class in *flare.viur.widgets.tree*), 142
 buildDescription() (*flare.viur.widgets.tree.TreeItemWidget* method), 140
 buildForm() (*flare.viur.forms.ViurForm* method), 144
 buildInternalForm() (*flare.viur.forms.ViurForm* method), 144
 buildListSelection() (*flare.viur.widgets.list.ListSelection* method), 138
 buildSelectDescr() (*flare.handler.requestHandler* method), 152
 buildTranslations() (in module *flare.i18n*), 153
 buildWidget() (*flare.viur.widgets.list.SkellistItem* method), 138
 Button (class in *flare.button*), 148
 Button (class in *flare.html5*), 103
 Button (class in *flare.html5.core*), 62
 ButtonBar (class in *flare.widgets.buttonbar*), 148
 ButtonBarButton (class in *flare.widgets.buttonbar*), 148
 ButtonBarSearch (class in *flare.widgets.buttonbar*), 148
 buttonClicked() (*flare.widgets.buttonbar.ButtonBar* method), 148
- ## C
- Cache (class in *flare*), 169
 Cache (class in *flare.cache*), 149
 call() (*flare.network.requestGroup* method), 162
 callNode() (*flare.safeeval.SafeEval* method), 166
 canHandle() (*flare.viur.widgets.file.FileWidget* static method), 136
 canHandle() (*flare.viur.widgets.list.ListWidget* static method), 138
 canHandle() (*flare.viur.widgets.tree.TreeBrowserWidget* static method), 142
 canHandle() (*flare.viur.widgets.tree.TreeWidget* static method), 142
 Canvas (class in *flare.html5*), 98
 Canvas (class in *flare.html5.core*), 57
 Caption (class in *flare.html5*), 99
 Caption (class in *flare.html5.core*), 58
 changeListeners (*flare.network.NetworkService* attribute), 160
 Check (class in *flare.ignite*), 155
 checkFor() (*flare.viur.bones.boolean.BooleanBone* static method), 119
 checkFor() (*flare.viur.bones.color.ColorBone* static method), 120
 checkFor() (*flare.viur.bones.date.DateBone* static method), 121
 checkFor() (*flare.viur.bones.email.EmailBone* static method), 122
 checkFor() (*flare.viur.bones.numeric.NumericBone* static method), 123
 checkFor() (*flare.viur.bones.password.PasswordBone* static method), 124
 checkFor() (*flare.viur.bones.raw.RawBone* static method), 124
 checkFor() (*flare.viur.bones.record.RecordBone* static method), 125
 checkFor() (*flare.viur.bones.relational.FileBone* static method), 129
 checkFor() (*flare.viur.bones.relational.FileDirectBone* static method), 129
 checkFor() (*flare.viur.bones.relational.HierarchyBone* static method), 127
 checkFor() (*flare.viur.bones.relational.RelationalBone* static method), 127
 checkFor() (*flare.viur.bones.relational.TreeDirBone* static method), 128
 checkFor() (*flare.viur.bones.relational.TreeItemBone* static method), 127
 checkFor() (*flare.viur.bones.select.SelectMultipleBone* static method), 131
 checkFor() (*flare.viur.bones.select.SelectSingleBone* static method), 131
 checkFor() (*flare.viur.bones.spatial.SpatialBone* static method), 132
 checkFor() (*flare.viur.bones.string.StringBone* static method), 133
 checkFor() (*flare.viur.bones.text.TextBone* static method), 134
 children() (*flare.html5.core.TextNode* method), 42
 children() (*flare.html5.core.Widget* method), 51
 children() (*flare.html5.TextNode* method), 83
 children() (*flare.html5.Widget* method), 92
 Cite (class in *flare.html5*), 99
 Cite (class in *flare.html5.core*), 58
 clear() (*flare.html5._WidgetClassWrapper* method), 84
 clear() (*flare.html5.core._WidgetClassWrapper* method), 43
 clear() (*flare.html5.core.Table* method), 68
 clear() (*flare.html5.Table* method), 109
 clear() (*flare.network.NetworkService* method), 162
 clearSelection() (*flare.viur.widgets.tree.TreeWidget* method), 141
 close() (*flare.popup.Popup* method), 164
 Code (class in *flare.html5*), 100
 Code (class in *flare.html5.core*), 58
 collectBoneErrors() (in module *flare.viur.formerrors*), 144
 ColorBone (class in *flare.viur.bones.color*), 120
 ColorEditWidget (class in *flare.viur.bones.color*), 119
 ColorViewWidget (class in *flare.viur.bones.color*), 120
 ColWrapper (class in *flare.html5*), 109
 ColWrapper (class in *flare.html5.core*), 68

- Command (class in flare.html5), 98
 Command (class in flare.html5.core), 57
 compareNode() (flare.safeeval.SafeEval method), 166
 compile() (flare.safeeval.SafeEval method), 167
 conf (in module flare), 169
 conf (in module flare.config), 150
 conf (in module flare.views), 115
 conf (in module flare.viur), 147
 conf (in module flare.viur.formconf), 143
 Confirm (class in flare.popup), 165
 createFormErrorMessage()
 (flare.viur.forms.ViurForm method), 145
 createFormSuccessMessage()
 (flare.viur.forms.ViurForm method), 145
 createWidget() (flare.viur.bones.base.BaseEditWidget
 method), 116
 createWidget() (flare.viur.bones.boolean.BooleanEditWidget
 method), 118
 createWidget() (flare.viur.bones.color.ColorEditWidget
 method), 119
 createWidget() (flare.viur.bones.date.DateEditWidget
 method), 120
 createWidget() (flare.viur.bones.numeric.NumericEditWidget
 method), 122
 createWidget() (flare.viur.bones.password.PasswordEditWidget
 method), 123
 createWidget() (flare.viur.bones.raw.RawEditWidget
 method), 124
 createWidget() (flare.viur.bones.record.RecordEditWidget
 method), 125
 createWidget() (flare.viur.bones.relational.FileEditWidget
 method), 128
 createWidget() (flare.viur.bones.relational.FileEditWidget
 method), 129
 createWidget() (flare.viur.bones.relational.RelationalEditWidget
 method), 126
 createWidget() (flare.viur.bones.select.SelectMultipleEditWidget
 method), 130
 createWidget() (flare.viur.bones.select.SelectSingleEditWidget
 method), 130
 createWidget() (flare.viur.bones.spatial.SpatialEditWidget
 method), 131
 createWidget() (flare.viur.bones.string.StringEditWidget
 method), 132
 createWidget() (flare.viur.bones.text.TextEditWidget
 method), 133
 createWorker() (in module flare.utils), 168

D
 Datalist (class in flare.html5), 100
 Datalist (class in flare.html5.core), 59
 DateBone (class in flare.viur.bones.date), 121
 DateEditWidget (class in flare.viur.bones.date), 120
 DateViewWidget (class in flare.viur.bones.date), 121
 Dd (class in flare.html5), 106
 Dd (class in flare.html5.core), 65
 debug() (in module flare.debug), 150
 debugElement() (in module flare.debug), 150
 decode() (flare.network.NetworkService static method),
 161
 defaultFailureHandler
 (flare.network.NetworkService attribute),
 160
 DeferredCall (class in flare.network), 160
 Details (class in flare.html5), 108
 Details (class in flare.html5.core), 67
 Dfn (class in flare.html5), 100
 Dfn (class in flare.html5.core), 59
 Dialog (class in flare.html5), 99
 Dialog (class in flare.html5.core), 57
 disable() (flare.html5.core.Widget method), 44
 disable() (flare.html5.Widget method), 85
 disable() (flare.viur.widgets.htmleditor.HtmlEditor
 method), 137
 disableDragMarkers()
 (flare.viur.widgets.tree.TreeItemWidget
 method), 139
 DisplayDelegateSelector (in module flare.viur), 147
 DisplayStringHandler() (in module flare.viur), 147
 displayStringHandler() (in module
 flare.viur.formatString), 143
 Div (class in flare.html5), 100
 Div (class in flare.html5.core), 59
 Dl (class in flare.html5), 106
 Dl (class in flare.html5.core), 65
 document (in module flare.html5), 82
 document (in module flare.html5.core), 41
 doesEventHitWidgetOrChildren() (in module
 flare.html5), 111
 doesEventHitWidgetOrChildren() (in module
 flare.html5.core), 69
 doesEventHitWidgetOrChildren() (in module
 flare.utils), 168
 doesEventHitWidgetOrParents() (in module
 flare.html5), 110
 doesEventHitWidgetOrParents() (in module
 flare.html5.core), 69
 doesEventHitWidgetOrParents() (in module
 flare.utils), 168
 doFetch() (flare.network.NetworkService method), 162
 domConvertEncodedText() (in module flare.html5), 83
 domConvertEncodedText() (in module
 flare.html5.core), 42
 domCreateAttribute() (in module flare.html5), 82
 domCreateAttribute() (in module flare.html5.core),
 41
 domCreateElement() (in module flare.html5), 82
 domCreateElement() (in module flare.html5.core), 41

[domCreateTextNode\(\)](#) (in module *flare.html5*), 83
[domCreateTextNode\(\)](#) (in module *flare.html5.core*), 41
[domElementFromPoint\(\)](#) (in module *flare.html5*), 83
[domElementFromPoint\(\)](#) (in module *flare.html5.core*), 42
[domGetElementById\(\)](#) (in module *flare.html5*), 83
[domGetElementById\(\)](#) (in module *flare.html5.core*), 41
[domGetElementsByTagName\(\)](#) (in module *flare.html5*), 83
[domGetElementsByTagName\(\)](#) (in module *flare.html5.core*), 42
[domParseString\(\)](#) (in module *flare.html5*), 83
[domParseString\(\)](#) (in module *flare.html5.core*), 42
[doSearch\(\)](#) (*flare.viur.widgets.file.Search* method), 135
[download\(\)](#) (*flare.viur.widgets.file.FilePreviewImage* method), 135
[drop\(\)](#) (*flare.popup.Alert* method), 164
[drop\(\)](#) (*flare.popup.Confirm* method), 165
[Dt](#) (class in *flare.html5*), 106
[Dt](#) (class in *flare.html5.core*), 65

E

[editWidget\(\)](#) (*flare.viur.bones.base.BaseBone* method), 118
[editWidgetFactory](#) (*flare.viur.bones.base.BaseBone* attribute), 118
[editWidgetFactory](#) (*flare.viur.bones.boolean.BooleanBone* attribute), 119
[editWidgetFactory](#) (*flare.viur.bones.color.ColorBone* attribute), 120
[editWidgetFactory](#) (*flare.viur.bones.date.DateBone* attribute), 121
[editWidgetFactory](#) (*flare.viur.bones.email.EmailBone* attribute), 122
[editWidgetFactory](#) (*flare.viur.bones.numeric.NumericBone* attribute), 123
[editWidgetFactory](#) (*flare.viur.bones.password.PasswordBone* attribute), 123
[editWidgetFactory](#) (*flare.viur.bones.raw.RawBone* attribute), 124
[editWidgetFactory](#) (*flare.viur.bones.record.RecordBone* attribute), 125
[editWidgetFactory](#) (*flare.viur.bones.relational.FileBone* attribute), 129
[editWidgetFactory](#) (*flare.viur.bones.relational.FileDirectBone* attribute), 129
[editWidgetFactory](#) (*flare.viur.bones.relational.RelationalBone* attribute), 127
[editWidgetFactory](#) (*flare.viur.bones.select.SelectMultipleBone* attribute), 131
[editWidgetFactory](#) (*flare.viur.bones.select.SelectSingleBone* attribute), 131
[editWidgetFactory](#) (*flare.viur.bones.spatial.SpatialBone* attribute), 132

[editWidgetFactory](#) (*flare.viur.bones.string.StringBone* attribute), 133
[editWidgetFactory](#) (*flare.viur.bones.text.TextBone* attribute), 134
[Em](#) (class in *flare.html5*), 100
[Em](#) (class in *flare.html5.core*), 59
[EmailBone](#) (class in *flare.viur.bones.email*), 121
[EmailEditWidget](#) (class in *flare.viur.bones.email*), 121
[EmailViewWidget](#) (class in *flare.viur.bones.email*), 121
[Embed](#) (class in *flare.html5*), 100
[Embed](#) (class in *flare.html5.core*), 59
[emit\(\)](#) (*flare.log.JSConsoleHandler* method), 158
[Empty](#) (*flare.viur.bones.base.ReadFromClientErrorSeverity* attribute), 116
[enable\(\)](#) (*flare.html5.core.Widget* method), 44
[enable\(\)](#) (*flare.html5.Widget* method), 85
[enable\(\)](#) (*flare.viur.widgets.htmleditor.HtmlEditor* method), 137
[entryFactory](#) (*flare.viur.bones.base.BaseMultiEditWidget* attribute), 117
[entryFactory](#) (*flare.viur.bones.relational.FileMultiEditDirectWidget* attribute), 128
[EntryIcon\(\)](#) (*flare.viur.widgets.file.FileLeafWidget* method), 136
[EntryIcon\(\)](#) (*flare.viur.widgets.tree.TreeItemWidget* method), 140
[EntryIcon\(\)](#) (*flare.viur.widgets.tree.TreeLeafWidget* method), 140
[entryTemplate](#) (*flare.viur.bones.select.SelectMultipleEditWidget* attribute), 130
[entryTemplate](#) (*flare.viur.bones.select.SelectSingleEditWidget* attribute), 130
[error\(\)](#) (*webworker_scripts.weblog* static method), 170
[errorWidget\(\)](#) (*flare.viur.bones.base.BaseBone* method), 118
[evalStringHandler\(\)](#) (in module *flare.viur.formatString*), 143
[EventDispatcher](#) (class in *flare.event*), 151
[execute\(\)](#) (*flare.safeeval.SafeEval* method), 166
[extend\(\)](#) (*flare.html5._WidgetClassWrapper* method), 84
[extend\(\)](#) (*flare.html5.core._WidgetClassWrapper* method), 43
[extendSelection\(\)](#) (*flare.viur.widgets.tree.TreeWidget* method), 141

F

[fastGrid\(\)](#) (*flare.ignite.Table* method), 156
[FetchJson\(\)](#) (in module *flare.network*), 159
[Fieldset](#) (class in *flare.html5*), 103
[Fieldset](#) (class in *flare.html5.core*), 62
[Figcaption](#) (class in *flare.html5*), 100
[Figcaption](#) (class in *flare.html5.core*), 59
[Figure](#) (class in *flare.html5*), 100

Figure (class in flare.html5.core), 59
 FileBone (class in flare.viur.bones.relational), 129
 FileDirectBone (class in flare.viur.bones.relational), 129
 FileEditDirectWidget (class in flare.viur.bones.relational), 128
 FileEditWidget (class in flare.viur.bones.relational), 129
 FileImagePopup (class in flare.viur.widgets.file), 135
 FileLeafWidget (class in flare.viur.widgets.file), 136
 FileMultiEditDirectWidget (class in flare.viur.bones.relational), 128
 FileNodeWidget (class in flare.viur.widgets.file), 136
 FilePreviewImage (class in flare.viur.widgets.file), 135
 FileViewWidget (class in flare.viur.bones.relational), 128
 FileWidget (class in flare.viur.widgets.file), 136
 filter() (flare.handler.ListHandler method), 152
 finish() (flare.Cache method), 169
 finish() (flare.cache.Cache method), 149
 finish() (flare.cache.Plan method), 149
 fire() (flare.event.EventDispatcher method), 151
 flare
 module, 34
 flare.button
 module, 148
 flare.cache
 module, 149
 flare.config
 module, 150
 flare.debug
 module, 150
 flare.event
 module, 151
 flare.handler
 module, 151
 flare.html5
 module, 34
 flare.html5.core
 module, 34
 flare.html5.svg
 module, 71
 flare.i18n
 module, 152
 flare.icons
 module, 154
 flare.ignite
 module, 155
 flare.input
 module, 156
 flare.intersectionObserver
 module, 157
 flare.log
 module, 157
 flare.network
 module, 159
 flare.observable
 module, 162
 flare.popout
 module, 163
 flare.popup
 module, 164
 flare.priorityqueue
 module, 165
 flare.safeeval
 module, 166
 flare.translations
 module, 112
 flare.translations.de
 module, 112
 flare.translations.en
 module, 113
 flare.utils
 module, 167
 flare.views
 module, 113
 flare.views.helpers
 module, 113
 flare.views.view
 module, 114
 flare.viur
 module, 115
 flare.viur.bones
 module, 115
 flare.viur.bones.base
 module, 115
 flare.viur.bones.boolean
 module, 118
 flare.viur.bones.color
 module, 119
 flare.viur.bones.date
 module, 120
 flare.viur.bones.email
 module, 121
 flare.viur.bones.numeric
 module, 122
 flare.viur.bones.password
 module, 123
 flare.viur.bones.raw
 module, 124
 flare.viur.bones.record
 module, 125
 flare.viur.bones.relational
 module, 126
 flare.viur.bones.select
 module, 130
 flare.viur.bones.spatial
 module, 131

[flare.viur.bones.string module](#), 132
[flare.viur.bones.text module](#), 133
[flare.viur.formatString module](#), 143
[flare.viur.formconf module](#), 143
[flare.viur.formerrors module](#), 144
[flare.viur.forms module](#), 144
[flare.viur.formtooltip module](#), 146
[flare.viur.widgets module](#), 134
[flare.viur.widgets.file module](#), 134
[flare.viur.widgets.htmleditor module](#), 136
[flare.viur.widgets.list module](#), 137
[flare.viur.widgets.tree module](#), 139
[flare.widgets module](#), 147
[flare.widgets.buttonbar module](#), 147
[FlareLogRecord](#) (class in [flare.log](#)), 158
[focus\(\)](#) ([flare.html5.core.Widget](#) method), 51
[focus\(\)](#) ([flare.html5.Widget](#) method), 92
[focus\(\)](#) ([flare.viur.widgets.file.Search](#) method), 135
[Footer](#) (class in [flare.html5](#)), 100
[Footer](#) (class in [flare.html5.core](#)), 59
[Form](#) (class in [flare.html5](#)), 103
[Form](#) (class in [flare.html5.core](#)), 62
[formatString\(\)](#) (in module [flare.viur](#)), 147
[formatString\(\)](#) (in module [flare.viur.formatString](#)), 143
[formatStringHandler\(\)](#) (in module [flare.viur.formatString](#)), 143
[fromHTML\(\)](#) ([flare.html5.core.Widget](#) method), 52
[fromHTML\(\)](#) ([flare.html5.Widget](#) method), 93
[fromHTML\(\)](#) (in module [flare.html5](#)), 112
[fromHTML\(\)](#) (in module [flare.html5.core](#)), 71

G

[generateView\(\)](#) (in module [flare.views.helpers](#)), 113
[genReqStr\(\)](#) ([flare.handler.SyncHandler](#) method), 152
[genReqStr\(\)](#) ([flare.network.NetworkService](#) static method), 161
[getActions\(\)](#) ([flare.viur.widgets.tree.TreeWidget](#) method), 141

[getChildKey\(\)](#) ([flare.viur.widgets.file.FileWidget](#) method), 136
[getChildKey\(\)](#) ([flare.viur.widgets.tree.TreeWidget](#) method), 142
[getCurrentAmount\(\)](#) ([flare.handler.ListHandler](#) method), 152
[getDescrFromValue\(\)](#) ([flare.handler.requestHandler](#) method), 152
[getIcon\(\)](#) ([flare.icons.SvgIcon](#) method), 154
[getImagePreview\(\)](#) (in module [flare.viur.widgets.file](#)), 135
[getKey\(\)](#) (in module [flare.html5](#)), 111
[getKey\(\)](#) (in module [flare.html5.core](#)), 70
[getLanguage\(\)](#) (in module [flare.i18n](#)), 153
[getLogger\(\)](#) (in module [flare.log](#)), 158
[getMessage\(\)](#) ([flare.log.FlareLogRecord](#) method), 158
[getRowCount\(\)](#) ([flare.html5.core.Table](#) method), 68
[getRowCount\(\)](#) ([flare.html5.Table](#) method), 110
[getState\(\)](#) ([flare.observable.StateHandler](#) method), 163
[getState\(\)](#) ([flare.views.StateHandler](#) method), 115
[getUriHashAsObject\(\)](#) (in module [flare.network](#)), 162
[getUriHashAsString\(\)](#) (in module [flare.network](#)), 162

H

[H1](#) (class in [flare.html5](#)), 100
[H1](#) (class in [flare.html5.core](#)), 59
[H2](#) (class in [flare.html5](#)), 101
[H2](#) (class in [flare.html5.core](#)), 59
[H3](#) (class in [flare.html5](#)), 101
[H3](#) (class in [flare.html5.core](#)), 59
[H4](#) (class in [flare.html5](#)), 101
[H4](#) (class in [flare.html5.core](#)), 60
[H5](#) (class in [flare.html5](#)), 101
[H5](#) (class in [flare.html5.core](#)), 60
[H6](#) (class in [flare.html5](#)), 101
[H6](#) (class in [flare.html5.core](#)), 60
[handleErrors\(\)](#) ([flare.viur.forms.ViurForm](#) method), 145
[hasClass\(\)](#) ([flare.html5.core.Widget](#) method), 49
[hasClass\(\)](#) ([flare.html5.Widget](#) method), 90
[Head\(\)](#) (in module [flare.html5](#)), 105
[Head\(\)](#) (in module [flare.html5.core](#)), 64
[HeadCls](#) (class in [flare.html5](#)), 105
[HeadCls](#) (class in [flare.html5.core](#)), 64
[Header](#) (class in [flare.html5](#)), 100
[Header](#) (class in [flare.html5.core](#)), 59
[hide\(\)](#) ([flare.html5.core.Widget](#) method), 48
[hide\(\)](#) ([flare.html5.Widget](#) method), 89
[HierarchyBone](#) (class in [flare.viur.bones.relational](#)), 127
[host](#) ([flare.network.NetworkService](#) attribute), 160
[Hr](#) (class in [flare.html5](#)), 101
[Hr](#) (class in [flare.html5.core](#)), 60

- [HtmlAst \(class in flare.html5\)](#), 112
[HtmlAst \(class in flare.html5.core\)](#), 70
[HtmlEditor \(class in flare.viur.widgets.htmleditor\)](#), 137
[htmlExpressionEvaluator \(in module flare.html5\)](#), 82
[htmlExpressionEvaluator \(in module flare.html5.core\)](#), 41
[HttpRequest \(class in flare.network\)](#), 160
- I**
- [I \(class in flare.html5\)](#), 101
[I \(class in flare.html5.core\)](#), 60
[Icon \(class in flare.icons\)](#), 154
[Iframe \(class in flare.html5\)](#), 105
[Iframe \(class in flare.html5.core\)](#), 64
[Img \(class in flare.html5\)](#), 105
[Img \(class in flare.html5.core\)](#), 64
[info\(\) \(webworker_scripts.weblog static method\)](#), 170
[initSources \(flare.viur.widgets.htmleditor.HtmlEditor attribute\)](#), 137
[initWidget\(\) \(flare.views.view.ViewWidget method\)](#), 114
[Input \(class in flare.html5\)](#), 103
[Input \(class in flare.html5.core\)](#), 62
[Input \(class in flare.ignite\)](#), 155
[Input \(class in flare.input\)](#), 156
[Ins \(class in flare.html5\)](#), 105
[Ins \(class in flare.html5.core\)](#), 64
[insert\(\) \(flare.html5._WidgetClassWrapper method\)](#), 84
[insert\(\) \(flare.html5.core._WidgetClassWrapper method\)](#), 43
[insert\(\) \(flare.priorityqueue.PriorityQueue method\)](#), 166
[insert\(\) \(flare.viur.PriorityQueue method\)](#), 147
[insertAfter\(\) \(flare.html5.core.Widget method\)](#), 49
[insertAfter\(\) \(flare.html5.Widget method\)](#), 90
[insertBefore\(\) \(flare.html5.core.Widget method\)](#), 49
[insertBefore\(\) \(flare.html5.Widget method\)](#), 90
[IntersectionObserver \(class in flare.intersectionObserver\)](#), 157
[Invalid \(flare.viur.bones.base.ReadFromClientErrorSeverity attribute\)](#), 116
[invalidate\(\) \(flare.Cache method\)](#), 169
[invalidate\(\) \(flare.cache.Cache method\)](#), 149
[InvalidatesOther \(flare.viur.bones.base.ReadFromClientErrorSeverity attribute\)](#), 116
[InvalidBoneValueException](#), 147
[isArrowDown\(\) \(in module flare.html5\)](#), 111
[isArrowDown\(\) \(in module flare.html5.core\)](#), 70
[isArrowLeft\(\) \(in module flare.html5\)](#), 111
[isArrowLeft\(\) \(in module flare.html5.core\)](#), 70
[isArrowRight\(\) \(in module flare.html5\)](#), 111
[isArrowRight\(\) \(in module flare.html5.core\)](#), 70
[isArrowUp\(\) \(in module flare.html5\)](#), 111
[isArrowUp\(\) \(in module flare.html5.core\)](#), 70
[isChildOf\(\) \(flare.html5.core.Widget method\)](#), 49
[isChildOf\(\) \(flare.html5.Widget method\)](#), 90
[isControl\(\) \(in module flare.html5\)](#), 111
[isControl\(\) \(in module flare.html5.core\)](#), 70
[isEscape\(\) \(in module flare.html5\)](#), 111
[isEscape\(\) \(in module flare.html5.core\)](#), 70
[isHidden\(\) \(flare.html5.core.Widget method\)](#), 48
[isHidden\(\) \(flare.html5.Widget method\)](#), 89
[isMeta\(\) \(in module flare.html5\)](#), 111
[isMeta\(\) \(in module flare.html5.core\)](#), 70
[isOkay\(\) \(flare.network.NetworkService static method\)](#), 161
[isParentOf\(\) \(flare.html5.core.Widget method\)](#), 49
[isParentOf\(\) \(flare.html5.Widget method\)](#), 90
[isReturn\(\) \(in module flare.html5\)](#), 111
[isReturn\(\) \(in module flare.html5.core\)](#), 70
[isShift\(\) \(in module flare.html5\)](#), 111
[isShift\(\) \(in module flare.html5.core\)](#), 70
[isSuitableFor\(\) \(flare.viur.widgets.htmleditor.TextInsertImageAction static method\)](#), 137
[isVisible\(\) \(flare.html5.core.Widget method\)](#), 48
[isVisible\(\) \(flare.html5.Widget method\)](#), 90
[Item \(class in flare.ignite\)](#), 156
[itemForKey\(\) \(flare.viur.widgets.tree.TreeWidget method\)](#), 141
- J**
- [JSConsoleHandler \(class in flare.log\)](#), 158
[jsObserver \(flare.intersectionObserver.IntersectionObserver attribute\)](#), 157
- K**
- [Kdb \(class in flare.html5\)](#), 101
[Kdb \(class in flare.html5.core\)](#), 60
[Keygen \(class in flare.html5\)](#), 105
[Keygen \(class in flare.html5.core\)](#), 64
[kickoff\(\) \(flare.network.NetworkService method\)](#), 161
- L**
- [Label \(class in flare.html5\)](#), 104
[Label \(class in flare.html5.core\)](#), 63
[Label \(class in flare.ignite\)](#), 155
[labelWidget\(\) \(flare.viur.bones.base.BaseBone method\)](#), 118
[languageEditWidgetFactory \(flare.viur.bones.base.BaseBone attribute\)](#), 118
[languageViewWidgetFactory \(flare.viur.bones.base.BaseBone attribute\)](#), 118
[leafWidget \(flare.viur.widgets.file.FileWidget attribute\)](#), 136

`leafWidget` (*flare.viur.widgets.tree.TreeBrowserWidget* attribute), 142
`leafWidget` (*flare.viur.widgets.tree.TreeWidget* attribute), 140
`Legend` (*class in flare.html5*), 101
`Legend` (*class in flare.html5.core*), 60
`Li` (*class in flare.html5*), 106
`Li` (*class in flare.html5.core*), 65
`Link` (*class in flare.html5*), 106
`Link` (*class in flare.html5.core*), 65
`ListHandler` (*class in flare.handler*), 152
`listNode()` (*flare.safeeval.SafeEval* method), 166
`ListSelection` (*class in flare.viur.widgets.list*), 138
`ListWidget` (*class in flare.viur.widgets.list*), 138
`lngDe` (*in module flare.translations*), 113
`lngDe` (*in module flare.translations.de*), 112
`lngEn` (*in module flare.translations*), 113
`lngEn` (*in module flare.translations.en*), 113
`loadNode()` (*flare.viur.widgets.tree.TreeWidget* method), 141
`loadProjectConf()` (*in module flare*), 170
`loadView()` (*flare.views.view.View* method), 114
`log` (*in module webworker_scripts*), 170
`loggers` (*in module flare.log*), 158
`lookup()` (*flare.Cache* method), 169
`lookup()` (*flare.cache.Cache* method), 149

M

`Map` (*class in flare.html5*), 106
`Map` (*class in flare.html5.core*), 65
`Mark` (*class in flare.html5*), 101
`Mark` (*class in flare.html5.core*), 60
`markDraggedElement()`
 (*flare.viur.widgets.tree.TreeItemWidget* method), 139
`Menu` (*class in flare.html5*), 106
`Menu` (*class in flare.html5.core*), 65
`Meta` (*class in flare.html5*), 107
`Meta` (*class in flare.html5.core*), 65
`Meter` (*class in flare.html5*), 107
`Meter` (*class in flare.html5.core*), 66
`module`
 flare, 34
 flare.button, 148
 flare.cache, 149
 flare.config, 150
 flare.debug, 150
 flare.event, 151
 flare.handler, 151
 flare.html5, 34
 flare.html5.core, 34
 flare.html5.svg, 71
 flare.i18n, 152
 flare.icons, 154

flare.ignite, 155
 flare.input, 156
 flare.intersectionObserver, 157
 flare.log, 157
 flare.network, 159
 flare.observable, 162
 flare.popout, 163
 flare.popup, 164
 flare.priorityqueue, 165
 flare.safeeval, 166
 flare.translations, 112
 flare.translations.de, 112
 flare.translations.en, 113
 flare.utils, 167
 flare.views, 113
 flare.views.helpers, 113
 flare.views.view, 114
 flare.viur, 115
 flare.viur.bones, 115
 flare.viur.bones.base, 115
 flare.viur.bones.boolean, 118
 flare.viur.bones.color, 119
 flare.viur.bones.date, 120
 flare.viur.bones.email, 121
 flare.viur.bones.numeric, 122
 flare.viur.bones.password, 123
 flare.viur.bones.raw, 124
 flare.viur.bones.record, 125
 flare.viur.bones.relational, 126
 flare.viur.bones.select, 130
 flare.viur.bones.spatial, 131
 flare.viur.bones.string, 132
 flare.viur.bones.text, 133
 flare.viur.formatString, 143
 flare.viur.formconf, 143
 flare.viur.formerrors, 144
 flare.viur.forms, 144
 flare.viur.formtooltip, 146
 flare.viur.widgets, 134
 flare.viur.widgets.file, 134
 flare.viur.widgets.html5editor, 136
 flare.viur.widgets.list, 137
 flare.viur.widgets.tree, 139
 flare.widgets, 147
 flare.widgets.buttonbar, 147
 webworker_scripts, 170
`ModuleWidgetSelector` (*in module flare.viur*), 147
`moveRequest()` (*flare.viur.widgets.tree.TreeItemWidget* method), 139
`moveRequest()` (*flare.viur.widgets.tree.TreeLeafWidget* method), 140
`multiEditWidgetFactory`
 (*flare.viur.bones.base.BaseBone* attribute), 118

- multiEditWidgetFactory
 (flare.viur.bones.relational.FileDirectBone attribute), 129
 multiEditWidgetFactory
 (flare.viur.bones.relational.RelationalBone attribute), 127
 multiEditWidgetFactory
 (flare.viur.bones.select.SelectMultipleBone attribute), 131
 multiViewWidgetFactory
 (flare.viur.bones.base.BaseBone attribute), 118
- ## N
- Nav (class in flare.html5), 107
 Nav (class in flare.html5.core), 66
 NetworkService (class in flare.network), 160
 NiceError() (in module flare.network), 160
 NiceErrorAndThen() (in module flare.network), 160
 nodeWidget (flare.viur.widgets.file.FileWidget attribute), 136
 nodeWidget (flare.viur.widgets.tree.TreeBrowserWidget attribute), 142
 nodeWidget (flare.viur.widgets.tree.TreeWidget attribute), 140
 Noscript (class in flare.html5), 101
 Noscript (class in flare.html5.core), 60
 notifyChange() (flare.network.NetworkService static method), 160
 NotSet (flare.viur.bones.base.ReadFromClientErrorSeverity attribute), 116
 NumericBone (class in flare.viur.bones.numeric), 123
 NumericEditWidget (class in flare.viur.bones.numeric), 122
 NumericViewWidget (class in flare.viur.bones.numeric), 123
- ## O
- Object (class in flare.html5), 107
 Object (class in flare.html5.core), 66
 ObservableValue (class in flare.observable), 162
 observableWidgets (flare.intersectionObserver.IntersectionObserver attribute), 157
 observe() (flare.intersectionObserver.IntersectionObserver method), 157
 Ol (class in flare.html5), 106
 Ol (class in flare.html5.core), 65
 onAcceptSelectionChanged()
 (flare.viur.widgets.list.ListSelection method), 138
 onAcceptSelectionChanged()
 (flare.viur.widgets.list.ListWidget method), 138
 onActiveButtonChanged()
 (flare.viur.widgets.list.ListSelection method), 138
 onActiveButtonChanged()
 (flare.widgets.buttonbar.ButtonBar method), 148
 onActiveButtonChanged()
 (flare.widgets.buttonbar.ButtonBarButton method), 148
 onActiveButtonChanged()
 (flare.widgets.buttonbar.ButtonBarSearch method), 148
 onActiveSelectionChanged()
 (flare.viur.widgets.list.ListSelection method), 138
 onActiveSelectionChanged()
 (flare.viur.widgets.list.SkellistItem method), 138
 onActiveViewChanged() (flare.views.view.View method), 114
 onAddBtnClick() (flare.viur.bones.base.BaseMultiEditWidget method), 117
 onAddBtnClick() (flare.viur.bones.relational.RelationalMultiEditWidget method), 127
 onApplyfilterChanged()
 (flare.viur.widgets.list.ListSelection method), 138
 onApplyfilterChanged()
 (flare.widgets.buttonbar.ButtonBarSearch method), 148
 onAttach() (flare.html5.core.TextNode method), 42
 onAttach() (flare.html5.core.Widget method), 49
 onAttach() (flare.html5.TextNode method), 83
 onAttach() (flare.html5.Widget method), 90
 onAttach() (flare.popup.Popup method), 164
 onAttach() (flare.viur.forms.ViurFormBone method), 145
 onAttach() (flare.viur.forms.ViurFormSubmit method), 145
 onAttach() (flare.viur.widgets.htmleditor.HtmlEditor method), 137
 onAttach() (flare.viur.widgets.tree.TreeWidget method), 141
 onBind() (flare.button.Button method), 148
 onBind() (flare.html5.core.Widget method), 49
 onBind() (flare.html5.Widget method), 90
 onBlur() (flare.html5.core.Widget method), 50
 onBlur() (flare.html5.Widget method), 91
 onBoneChange() (flare.viur.forms.ViurForm method), 144
 onCancel() (flare.popup.Prompt method), 164
 onCancel() (flare.popup.radioButtonDialog method), 165
 onCancel() (flare.popup.TextareaDialog method), 165

[onChange\(\) \(flare.html5.core.Widget method\), 50](#)
[onChange\(\) \(flare.html5.Widget method\), 91](#)
[onChange\(\) \(flare.input.Input method\), 156](#)
[onChange\(\) \(flare.viur.bones.numeric.NumericEditWidget method\), 122](#)
[onChange\(\) \(flare.viur.bones.relational.FileEditDirectWidget method\), 128](#)
[onChange\(\) \(flare.viur.bones.relational.FileMultiEditDirectWidget method\), 128](#)
[onChange\(\) \(flare.viur.bones.relational.RelationalEditWidget method\), 127](#)
[onChange\(\) \(flare.viur.bones.string.StringEditWidget method\), 132](#)
[onChange\(\) \(flare.viur.forms.ViurForm method\), 144](#)
[onChange\(\) \(flare.viur.forms.ViurFormBone method\), 145](#)
[onClick\(\) \(flare.button.Button method\), 148](#)
[onClick\(\) \(flare.html5.core.Widget method\), 51](#)
[onClick\(\) \(flare.html5.Widget method\), 92](#)
[onClick\(\) \(flare.viur.formtooltip.ToolTip method\), 146](#)
[onClick\(\) \(flare.viur.widgets.file.FileImagePopup method\), 135](#)
[onClick\(\) \(flare.viur.widgets.file.FilePreviewImage method\), 135](#)
[onClick\(\) \(flare.viur.widgets.htmleditor.TextInsertImageArea method\), 137](#)
[onClick\(\) \(flare.viur.widgets.tree.TreeItemWidget method\), 140](#)
[onClose\(\) \(flare.popup.Popup method\), 164](#)
[onCompletion\(\) \(flare.handler.SyncHandler method\), 152](#)
[onCompletion\(\) \(flare.network.NetworkService method\), 162](#)
[onContextMenu\(\) \(flare.html5.core.Widget method\), 50](#)
[onContextMenu\(\) \(flare.html5.Widget method\), 91](#)
[onDataChanged\(\) \(flare.Cache method\), 169](#)
[onDataChanged\(\) \(flare.cache.Cache method\), 149](#)
[onDataChanged\(\) \(flare.viur.widgets.tree.TreeWidget method\), 141](#)
[onDbClick\(\) \(flare.html5.core.Widget method\), 51](#)
[onDbClick\(\) \(flare.html5.Widget method\), 92](#)
[onDbClick\(\) \(flare.viur.widgets.tree.TreeItemWidget method\), 140](#)
[onDeleteBtnClick\(\) \(flare.viur.bones.relational.FileEditDirectWidget method\), 128](#)
[onDeleteBtnClick\(\) \(flare.viur.bones.relational.RelationalEditWidget method\), 127](#)
[onDetach\(\) \(flare.html5.core.TextNode method\), 42](#)
[onDetach\(\) \(flare.html5.core.Widget method\), 49](#)
[onDetach\(\) \(flare.html5.TextNode method\), 83](#)
[onDetach\(\) \(flare.html5.Widget method\), 90](#)
[onDetach\(\) \(flare.input.Input method\), 156](#)
[onDetach\(\) \(flare.popup.Popup method\), 164](#)
[onDetach\(\) \(flare.views.view.ViewWidget method\), 114](#)
[onDetach\(\) \(flare.viur.widgets.htmleditor.HtmlEditor method\), 137](#)
[onDetach\(\) \(flare.viur.widgets.tree.TreeWidget method\), 141](#)
[onDocumentKeyDown\(\) \(flare.popup.Confirm method\), 165](#)
[onDocumentKeyDown\(\) \(flare.popup.Popup method\), 164](#)
[onDocumentKeyDown\(\) \(flare.popup.Prompt method\), 164](#)
[onDocumentKeyDown\(\) \(flare.popup.TextareaDialog method\), 165](#)
[onDownloadBtnClick\(\) \(flare.viur.widgets.file.FileImagePopup method\), 135](#)
[onDrag\(\) \(flare.html5.core.Widget method\), 51](#)
[onDrag\(\) \(flare.html5.Widget method\), 92](#)
[onDragEnd\(\) \(flare.html5.core.Widget method\), 51](#)
[onDragEnd\(\) \(flare.html5.Widget method\), 92](#)
[onDragEnd\(\) \(flare.viur.bones.base.BaseMultiEditWidgetEntry method\), 117](#)
[onDragEnd\(\) \(flare.viur.widgets.tree.TreeItemWidget method\), 139](#)
[onDragEnter\(\) \(flare.html5.core.Widget method\), 51](#)
[onDragEnter\(\) \(flare.html5.Widget method\), 92](#)
[onDragEnter\(\) \(flare.viur.bones.relational.FileEditDirectWidget method\), 128](#)
[onDragEnter\(\) \(flare.viur.bones.relational.FileMultiEditDirectWidget method\), 129](#)
[onDragLeave\(\) \(flare.html5.core.Widget method\), 51](#)
[onDragLeave\(\) \(flare.html5.Widget method\), 92](#)
[onDragLeave\(\) \(flare.viur.bones.base.BaseMultiEditWidgetEntry method\), 117](#)
[onDragLeave\(\) \(flare.viur.bones.relational.FileEditDirectWidget method\), 128](#)
[onDragLeave\(\) \(flare.viur.bones.relational.FileMultiEditDirectWidget method\), 129](#)
[onDragLeave\(\) \(flare.viur.widgets.tree.TreeItemWidget method\), 139](#)
[onDragOver\(\) \(flare.html5.core.Widget method\), 51](#)
[onDragOver\(\) \(flare.html5.Widget method\), 92](#)
[onDragOver\(\) \(flare.viur.bones.base.BaseMultiEditWidgetEntry method\), 117](#)
[onDragOver\(\) \(flare.viur.bones.relational.FileEditDirectWidget method\), 128](#)
[onDragOver\(\) \(flare.viur.bones.relational.FileMultiEditDirectWidget method\), 129](#)
[onDragOver\(\) \(flare.viur.widgets.tree.TreeItemWidget method\), 139](#)
[onDragOver\(\) \(flare.viur.widgets.tree.TreeLeafWidget method\), 140](#)
[onDragOver\(\) \(flare.viur.widgets.tree.TreeWidget method\), 142](#)
[onDragStart\(\) \(flare.html5.core.Widget method\), 51](#)

onDragStart() (flare.html5.Widget method), 92
 onDragStart() (flare.viur.bones.base.BaseMultiEditWidget method), 117
 onDragStart() (flare.viur.widgets.tree.TreeItemWidget method), 139
 onDrop() (flare.html5.core.Widget method), 51
 onDrop() (flare.html5.Widget method), 92
 onDrop() (flare.viur.bones.base.BaseMultiEditWidgetEntry method), 117
 onDrop() (flare.viur.bones.relational.FileEditDirectWidget method), 128
 onDrop() (flare.viur.bones.relational.FileMultiEditDirectWidget method), 129
 onDrop() (flare.viur.widgets.tree.TreeItemWidget method), 139
 onDrop() (flare.viur.widgets.tree.TreeWidget method), 141
 onEditorChange() (flare.viur.widgets.htmleditor.HtmlEditor method), 137
 onError() (flare.handler.SyncHandler method), 152
 onError() (flare.icons.Icon method), 154
 onError() (flare.network.NetworkService method), 162
 onFailed() (flare.viur.widgets.file.Uploader method), 135
 onFinished() (flare.network.NetworkService method), 162
 onFinished() (flare.network.requestGroup method), 162
 onFocus() (flare.html5.core.Widget method), 50
 onFocus() (flare.html5.Widget method), 91
 onFocus() (flare.input.Input method), 156
 onFocusIn() (flare.html5.core.Widget method), 50
 onFocusIn() (flare.html5.Widget method), 91
 onFocusOut() (flare.html5.core.Widget method), 50
 onFocusOut() (flare.html5.Widget method), 91
 onFormChange() (flare.html5.core.Widget method), 50
 onFormChange() (flare.html5.Widget method), 91
 onFormInput() (flare.html5.core.Widget method), 50
 onFormInput() (flare.html5.Widget method), 91
 onFormSuccess() (flare.viur.forms.ViurForm method), 145
 onInput() (flare.html5.core.Widget method), 50
 onInput() (flare.html5.Widget method), 91
 onInvalid() (flare.html5.core.Widget method), 50
 onInvalid() (flare.html5.Widget method), 91
 onKeyDown() (flare.html5.core.Widget method), 50
 onKeyDown() (flare.html5.Widget method), 92
 onKeyDown() (flare.popup.Alert method), 165
 onKeyDown() (flare.popup.Confirm method), 165
 onKeyDown() (flare.popup.Prompt method), 164
 onKeyDown() (flare.viur.widgets.file.Search method), 135
 onKeyDown() (flare.viur.widgets.tree.TreeWidget method), 141
 onKeyPress() (flare.html5.core.Widget method), 50
 onKeyPress() (flare.html5.Widget method), 92
 onKeyUp() (flare.html5.core.Widget method), 51
 onKeyUp() (flare.html5.Widget method), 92
 onKeyUp() (flare.popup.Prompt method), 164
 onKeyUp() (flare.viur.bones.string.StringEditWidget method), 132
 onKeyUp() (flare.viur.widgets.tree.TreeWidget method), 141
 onLangBtnClick() (flare.viur.bones.base.BaseLanguageEditWidget method), 117
 onListStatusChanged() (flare.handler.requestHandler method), 152
 onLoad() (flare.viur.widgets.file.Uploader method), 135
 onMouseDown() (flare.html5.core.Widget method), 51
 onMouseDown() (flare.html5.Widget method), 92
 onMouseMove() (flare.html5.core.Widget method), 51
 onMouseMove() (flare.html5.Widget method), 92
 onMouseOut() (flare.html5.core.Widget method), 51
 onMouseOut() (flare.html5.Widget method), 92
 onMouseOver() (flare.html5.core.Widget method), 51
 onMouseOver() (flare.html5.Widget method), 92
 onMouseUp() (flare.html5.core.Widget method), 51
 onMouseUp() (flare.html5.Widget method), 92
 onMouseWheel() (flare.html5.core.Widget method), 51
 onMouseWheel() (flare.html5.Widget method), 92
 onNoClicked() (flare.popup.Confirm method), 165
 onOkay() (flare.popup.Prompt method), 164
 onOkay() (flare.popup.radioButtonDialog method), 165
 onOkay() (flare.popup.TextareaDialog method), 165
 onOkBtnClick() (flare.popup.Alert method), 165
 onPathRequestSucceeded() (flare.viur.widgets.tree.TreeBrowserWidget method), 142
 onProgress() (flare.viur.widgets.file.Uploader method), 135
 onReadyStateChange() (flare.network.HTTPRequest method), 160
 onRemoveBtnClick() (flare.viur.bones.base.BaseMultiEditWidget method), 117
 onRemoveBtnClick() (flare.viur.bones.base.BaseMultiEditWidgetEntry method), 117
 onRequestList() (flare.viur.widgets.list.ListSelection method), 138
 onReset() (flare.html5.core.Widget method), 50
 onReset() (flare.html5.Widget method), 91
 onScroll() (flare.html5.core.Widget method), 51
 onScroll() (flare.html5.Widget method), 92
 onSelect() (flare.html5.core.Widget method), 50
 onSelect() (flare.html5.Widget method), 91
 onSelectBtnClick() (flare.viur.bones.relational.RelationalEditWidget method), 127
 onSelectionActivated() (flare.viur.widgets.htmleditor.TextInsertImageAction method), 137

onSetDefaultRootNode() (flare.viur.widgets.tree.TreeWidget method), 141
 onStartSearch() (flare.viur.widgets.file.FileWidget method), 136
 onSubmit() (flare.html5.core.Widget method), 50
 onSubmit() (flare.html5.Widget method), 92
 onSubmitStatusChanged() (flare.viur.forms.ViurForm method), 145
 onSubmitStatusChanged() (flare.viur.forms.ViurFormSubmit method), 146
 onSuccess() (flare.viur.widgets.file.Uploader method), 135
 onTimeout() (flare.network.NetworkService method), 162
 onTouchCancel() (flare.html5.core.Widget method), 51
 onTouchCancel() (flare.html5.Widget method), 92
 onTouchEnd() (flare.html5.core.Widget method), 51
 onTouchEnd() (flare.html5.Widget method), 92
 onTouchMove() (flare.html5.core.Widget method), 51
 onTouchMove() (flare.html5.Widget method), 92
 onTouchStart() (flare.html5.core.Widget method), 51
 onTouchStart() (flare.html5.Widget method), 92
 onUnsetBtnClick() (flare.viur.bones.color.ColorEditWidget method), 120
 onUploadAdded() (flare.viur.widgets.file.Uploader method), 135
 onUploadFailed() (flare.viur.bones.relational.FileEditDialog method), 128
 onUploadFailed() (flare.viur.bones.relational.FileMultiEditDialog method), 129
 onUploadSuccess() (flare.viur.bones.relational.FileEditDialog method), 128
 onUploadSuccess() (flare.viur.bones.relational.FileMultiEditDialog method), 129
 onUploadUrlAvailable() (flare.viur.widgets.file.Uploader method), 135
 onViewfocusedChanged() (flare.views.view.ViewWidget method), 114
 onYesClicked() (flare.popup.Confirm method), 165
 Optgroup (class in flare.html5), 104
 Optgroup (class in flare.html5.core), 63
 Option (class in flare.html5), 104
 Option (class in flare.html5.core), 63
 Output (class in flare.html5), 104
 Output (class in flare.html5.core), 63

P

P (class in flare.html5), 101
 P (class in flare.html5.core), 60
 Param (class in flare.html5), 107
 Param (class in flare.html5.core), 66

parent() (flare.html5.core.Widget method), 51
 parent() (flare.html5.Widget method), 92
 parseFloat() (in module flare.html5), 111
 parseFloat() (in module flare.html5.core), 70
 parseFloat() (in module flare.utils), 168
 parseHTML() (in module flare.html5), 112
 parseHTML() (in module flare.html5.core), 71
 parseInt() (in module flare.html5), 111
 parseInt() (in module flare.html5.core), 70
 parseInt() (in module flare.utils), 168
 PasswordBone (class in flare.viur.bones.password), 123
 PasswordEditWidget (class in flare.viur.bones.password), 123
 Plan (class in flare.cache), 149
 pop() (flare.html5._WidgetClassWrapper method), 84
 pop() (flare.html5.core._WidgetClassWrapper method), 43
 Popout (class in flare.popout), 163
 PopoutItem (class in flare.popout), 163
 Popup (class in flare.popup), 164
 prefix (flare.network.NetworkService attribute), 160
 prepareCol() (flare.html5.core.Table method), 68
 prepareCol() (flare.html5.Table method), 109
 prepareCol() (flare.ignite.Table method), 156
 prepareGrid() (flare.html5.core.Table method), 68
 prepareGrid() (flare.html5.Table method), 109
 prepareLogger() (in module flare.log), 158
 prepareRow() (flare.html5.core.Table method), 68
 prepareRow() (flare.html5.Table method), 109
 prepareRow() (flare.ignite.Table method), 156
 prependChild() (flare.html5.core.Widget method), 49
 prependChild() (flare.html5.Widget method), 90
 PriorityQueue (class in flare.priorityqueue), 165
 PriorityQueue (class in flare.viur), 147
 ProcessQueue (in module flare.network), 160
 Progress (class in flare.html5), 107
 Progress (class in flare.html5.core), 66
 Progress (class in flare.ignite), 156
 Prompt (class in flare.popup), 164

Q

Q (class in flare.html5), 108
 Q (class in flare.html5.core), 66

R

Radio (class in flare.ignite), 156
 radioButtonDialog (class in flare.popup), 165
 RawBone (class in flare.viur.bones.raw), 124
 RawEditWidget (class in flare.viur.bones.raw), 124
 RawViewWidget (class in flare.viur.bones.raw), 124
 ReadFromClientErrorSeverity (class in flare.viur.bones.base), 116
 rebuildPath() (flare.viur.widgets.tree.TreeBrowserWidget method), 142

receivedStructure() (flare.viur.widgets.tree.TreeWidget method), 140
 RecordBone (class in flare.viur.bones.record), 125
 RecordEditWidget (class in flare.viur.bones.record), 125
 RecordViewWidget (class in flare.viur.bones.record), 125
 reevaluate() (flare.viur.widgets.file.Search method), 135
 register() (flare.event.EventDispatcher method), 151
 register() (flare.observable.StateHandler method), 163
 register() (flare.views.StateHandler method), 115
 registerChangeListener() (flare.network.NetworkService static method), 161
 registerField() (flare.viur.forms.ViurForm method), 144
 registerTag() (in module flare.html5), 111
 registerTag() (in module flare.html5.core), 70
 registerViews() (in module flare.views.helpers), 114
 RelationalBone (class in flare.viur.bones.relational), 127
 RelationalEditWidget (class in flare.viur.bones.relational), 126
 RelationalMultiEditWidget (class in flare.viur.bones.relational), 127
 RelationalViewWidget (class in flare.viur.bones.relational), 127
 reload() (flare.handler.ListHandler method), 152
 reloadData() (flare.viur.widgets.tree.TreeBrowserWidget method), 142
 reloadData() (flare.viur.widgets.tree.TreeWidget method), 141
 reloadList() (flare.viur.widgets.list.ListSelection method), 138
 remove() (flare.html5._WidgetClassWrapper method), 84
 remove() (flare.html5.core._WidgetClassWrapper method), 43
 removeAllChildren() (flare.html5.core.Widget method), 49
 removeAllChildren() (flare.html5.Widget method), 90
 removeChangeListener() (flare.network.NetworkService static method), 161
 removeChild() (flare.html5.core.Widget method), 49
 removeChild() (flare.html5.Widget method), 90
 removeClass() (flare.html5.core.Widget method), 50
 removeClass() (flare.html5.Widget method), 91
 removeEventListener() (flare.html5.core.Widget method), 44
 removeEventListener() (flare.html5.Widget method), 85
 removeView() (in module flare.views.helpers), 114
 renderTimeout() (flare.viur.bones.string.StringEditWidget method), 132
 replaceChild() (flare.html5.core.Widget method), 49
 replaceChild() (flare.html5.Widget method), 90
 replaceSVG() (flare.icons.SvgIcon method), 154
 replaceWithMessage() (flare.viur.widgets.file.Uploader method), 136
 request() (flare.Cache method), 169
 request() (flare.cache.Cache method), 149
 request() (flare.handler.SyncHandler static method), 152
 request() (flare.network.NetworkService static method), 161
 requestChildren() (flare.viur.widgets.tree.TreeWidget method), 141
 requestClients() (flare.viur.widgets.list.ListSelection method), 138
 requestData() (flare.handler.requestHandler method), 151
 requestFallBack() (flare.icons.SvgIcon method), 154
 requestGroup (class in flare.network), 162
 requestHandler (class in flare.handler), 151
 requestNext() (flare.handler.ListHandler method), 152
 requestStructure() (flare.viur.widgets.tree.TreeWidget method), 140
 requestSuccess() (flare.handler.ListHandler method), 152
 requestSuccess() (flare.handler.requestHandler method), 152
 require() (flare.Cache method), 169
 require() (flare.cache.Cache method), 149
 resetIcon() (flare.button.Button method), 148
 resetLoadingState() (flare.viur.widgets.file.Search method), 135
 resetLoadingState() (flare.viur.widgets.htmleditor.TextInsertImageAction method), 137
 resetSearch() (flare.viur.widgets.file.Search method), 135
 retryCodes (flare.network.NetworkService attribute), 160
 retryDelay (flare.network.NetworkService attribute), 160
 retryMax (flare.network.NetworkService attribute), 160
 RowWrapper (class in flare.html5), 109
 RowWrapper (class in flare.html5.core), 68
 Rq (class in flare.html5), 102
 Rq (class in flare.html5.core), 60
 Rt (class in flare.html5), 102
 Rt (class in flare.html5.core), 61
 Ruby (class in flare.html5), 102

Ruby (class in *flare.html5.core*), 61
run() (*flare.cache.Plan* method), 149
run() (*flare.network.DeferredCall* method), 160

S

S (class in *flare.html5*), 102
S (class in *flare.html5.core*), 61
SafeEval (class in *flare.safeeval*), 166
safeEval() (*flare.safeeval.SafeEval* method), 167
Samp (class in *flare.html5*), 102
Samp (class in *flare.html5.core*), 61
Script (class in *flare.html5*), 108
Script (class in *flare.html5.core*), 67
Search (class in *flare.viur.widgets.file*), 135
searchWidget() (*flare.viur.widgets.file.FileWidget* method), 136
Section (class in *flare.html5*), 102
Section (class in *flare.html5.core*), 61
Select (class in *flare.html5*), 104
Select (class in *flare.html5.core*), 63
Select (class in *flare.ignite*), 156
select() (*flare.priorityqueue.PriorityQueue* method), 166
select() (*flare.viur.PriorityQueue* method), 147
SelectMultipleBone (class in *flare.viur.bones.select*), 131
SelectMultipleEditWidget (class in *flare.viur.bones.select*), 130
selectorAllow (*flare.viur.bones.relational.RelationalBone* attribute), 127
selectorAllow (*flare.viur.bones.relational.TreeDirBone* attribute), 128
selectorAllow (*flare.viur.bones.relational.TreeItemBone* attribute), 127
selectorReturn() (*flare.viur.widgets.tree.TreeWidget* method), 141
SelectSingleBone (class in *flare.viur.bones.select*), 131
SelectSingleEditWidget (class in *flare.viur.bones.select*), 130
SelectViewWidget (class in *flare.viur.bones.select*), 131
sendViurForm() (*flare.viur.forms.ViurFormSubmit* method), 146
serialize() (*flare.viur.bones.base.BaseEditWidget* method), 116
serialize() (*flare.viur.bones.base.BaseLanguageEditWidget* method), 118
serialize() (*flare.viur.bones.base.BaseMultiEditWidget* method), 117
serialize() (*flare.viur.bones.base.BaseMultiViewWidget* method), 117
serialize() (*flare.viur.bones.base.BaseViewWidget* method), 117
serialize() (*flare.viur.bones.boolean.BooleanEditWidget* method), 119
serialize() (*flare.viur.bones.color.ColorEditWidget* method), 120
serialize() (*flare.viur.bones.date.DateEditWidget* method), 121
serialize() (*flare.viur.bones.numeric.NumericEditWidget* method), 123
serialize() (*flare.viur.bones.password.PasswordEditWidget* method), 123
serialize() (*flare.viur.bones.record.RecordEditWidget* method), 125
serialize() (*flare.viur.bones.relational.FileMultiEditDirectWidget* method), 129
serialize() (*flare.viur.bones.relational.RelationalEditWidget* method), 127
serialize() (*flare.viur.bones.relational.RelationalViewWidget* method), 127
serialize() (*flare.viur.bones.select.SelectMultipleEditWidget* method), 130
serialize() (*flare.viur.bones.select.SelectSingleEditWidget* method), 131
serialize() (*flare.viur.bones.spatial.SpatialEditWidget* method), 132
serialize() (*flare.viur.bones.string.StringEditWidget* method), 133
serialize() (*flare.viur.forms.ViurForm* method), 145
serialize() (*flare.viur.forms.ViurFormBone* method), 145
set() (*flare.html5._WidgetClassWrapper* method), 83
set() (*flare.html5.core._WidgetClassWrapper* method), 42
setContent() (*flare.viur.widgets.list.ListSelection* method), 138
setFile() (*flare.viur.widgets.file.FilePreviewImage* method), 135
setInvalid() (*flare.viur.forms.ViurFormBone* method), 145
setLanguage() (in module *flare.i18n*), 153
setRootNode() (*flare.viur.widgets.tree.TreeWidget* method), 141
setSelector() (*flare.viur.widgets.list.ListWidget* method), 138
setSelector() (*flare.viur.widgets.tree.TreeWidget* method), 140
setStyle() (*flare.viur.widgets.file.FileLeafWidget* method), 136
setStyle() (*flare.viur.widgets.file.FileNodeWidget* method), 136
setStyle() (*flare.viur.widgets.tree.BreadcrumbNodeWidget* method), 142
setStyle() (*flare.viur.widgets.tree.BrowserLeafWidget* method), 142
setStyle() (*flare.viur.widgets.tree.BrowserNodeWidget*

- method*), 142
- `setStyle()` (*flare.viur.widgets.tree.TreeItemWidget method*), 139
- `setStyle()` (*flare.viur.widgets.tree.TreeLeafWidget method*), 140
- `setUrlHash()` (*in module flare.network*), 162
- `setValid()` (*flare.viur.forms.ViurFormBone method*), 145
- `setValue()` (*flare.observable.ObservableValue method*), 162
- `setValue()` (*flare.viur.bones.numeric.NumericEditWidget method*), 122
- `show()` (*flare.html5.core.Widget method*), 48
- `show()` (*flare.html5.Widget method*), 89
- `sinkEvent()` (*flare.html5.core.Widget method*), 44
- `sinkEvent()` (*flare.html5.Widget method*), 85
- `sitepackagespath` (*in module flare.views.helpers*), 113
- `SkellistItem` (*class in flare.viur.widgets.list*), 138
- `skelType` (*flare.viur.widgets.tree.TreeLeafWidget attribute*), 140
- `skelType` (*flare.viur.widgets.tree.TreeNodeWidget attribute*), 140
- `skeyRequestQueue` (*in module flare.network*), 160
- `Small` (*class in flare.html5*), 102
- `Small` (*class in flare.html5.core*), 61
- `sortChildren()` (*flare.html5.core.Widget method*), 52
- `sortChildren()` (*flare.html5.Widget method*), 93
- `Source` (*class in flare.html5*), 108
- `Source` (*class in flare.html5.core*), 67
- `Span` (*class in flare.html5*), 108
- `Span` (*class in flare.html5.core*), 67
- `SpatialBone` (*class in flare.viur.bones.spatial*), 132
- `SpatialEditWidget` (*class in flare.viur.bones.spatial*), 131
- `start()` (*flare.Cache method*), 169
- `start()` (*flare.cache.Cache method*), 149
- `startUpload()` (*flare.viur.bones.relational.FileEditDirectWidget method*), 128
- `startUpload()` (*flare.viur.bones.relational.FileMultiEditDirectWidget method*), 129
- `StateHandler` (*class in flare.observable*), 162
- `StateHandler` (*class in flare.views*), 115
- `StringBone` (*class in flare.viur.bones.string*), 133
- `StringEditWidget` (*class in flare.viur.bones.string*), 132
- `StringViewWidget` (*class in flare.viur.bones.string*), 133
- `Strong` (*class in flare.html5*), 102
- `Strong` (*class in flare.html5.core*), 61
- `struct()` (*flare.Cache method*), 169
- `struct()` (*flare.cache.Cache method*), 149
- `Style` (*class in flare.html5*), 108
- `Style` (*class in flare.html5.core*), 67
- `style` (*flare.html5.core.Widget attribute*), 44
- `style` (*flare.html5.Widget attribute*), 85
- `style` (*flare.popout.Popout attribute*), 163
- `style` (*flare.popout.PopoutItem attribute*), 163
- `style` (*flare.viur.bones.base.BaseEditWidget attribute*), 116
- `style` (*flare.viur.bones.base.BaseMultiEditWidget attribute*), 117
- `style` (*flare.viur.bones.base.BaseMultiEditWidgetEntry attribute*), 117
- `style` (*flare.viur.bones.base.BaseViewWidget attribute*), 116
- `style` (*flare.viur.bones.boolean.BooleanEditWidget attribute*), 118
- `style` (*flare.viur.bones.color.ColorEditWidget attribute*), 119
- `style` (*flare.viur.bones.date.DateEditWidget attribute*), 120
- `style` (*flare.viur.bones.numeric.NumericEditWidget attribute*), 122
- `style` (*flare.viur.bones.password.PasswordEditWidget attribute*), 123
- `style` (*flare.viur.bones.raw.RawEditWidget attribute*), 124
- `style` (*flare.viur.bones.record.RecordEditWidget attribute*), 125
- `style` (*flare.viur.bones.record.RecordViewWidget attribute*), 125
- `style` (*flare.viur.bones.relational.FileEditDirectWidget attribute*), 128
- `style` (*flare.viur.bones.relational.FileEditWidget attribute*), 129
- `style` (*flare.viur.bones.relational.FileMultiEditDirectWidget attribute*), 128
- `style` (*flare.viur.bones.relational.RelationalEditWidget attribute*), 126
- `style` (*flare.viur.bones.relational.RelationalViewWidget attribute*), 127
- `style` (*flare.viur.bones.select.SelectMultipleEditWidget attribute*), 130
- `style` (*flare.viur.bones.string.StringEditWidget attribute*), 132
- `style` (*flare.viur.bones.text.TextEditWidget attribute*), 133
- `Sub` (*class in flare.html5*), 102
- `Sub` (*class in flare.html5.core*), 61
- `submitForm()` (*flare.viur.forms.ViurForm method*), 145
- `Summary` (*class in flare.html5*), 108
- `Summary` (*class in flare.html5.core*), 67
- `summernoteEditor` (*in module flare.viur.widgets.htmleditor*), 137
- `Summery` (*class in flare.html5*), 102
- `Summery` (*class in flare.html5.core*), 61
- `Sup` (*class in flare.html5*), 102
- `Sup` (*class in flare.html5.core*), 61

Svg (class in flare.html5.svg), 74
 SvgCircle (class in flare.html5.svg), 74
 SvgEllipse (class in flare.html5.svg), 74
 SvgG (class in flare.html5.svg), 74
 SvgIcon (class in flare.icons), 154
 SvgImage (class in flare.html5.svg), 75
 SvgLine (class in flare.html5.svg), 75
 SvgPath (class in flare.html5.svg), 75
 SvgPolygon (class in flare.html5.svg), 75
 SvgPolyline (class in flare.html5.svg), 75
 SvgRect (class in flare.html5.svg), 75
 SvgText (class in flare.html5.svg), 75
 SvgWidget (class in flare.html5.svg), 74
 Switch (class in flare.ignite), 155
 SyncHandler (class in flare.handler), 152

T

Table (class in flare.html5), 109
 Table (class in flare.html5.core), 68
 Table (class in flare.ignite), 156
 tag() (in module flare.html5), 111
 tag() (in module flare.html5.core), 70
 Tbody (class in flare.html5), 109
 Tbody (class in flare.html5.core), 68
 Td (class in flare.html5), 109
 Td (class in flare.html5.core), 67
 Template (class in flare.html5), 110
 Template (class in flare.html5.core), 69
 Textarea (class in flare.html5), 104
 Textarea (class in flare.html5.core), 63
 Textarea (class in flare.ignite), 156
 TextareaDialog (class in flare.popup), 165
 TextBone (class in flare.viur.bones.text), 134
 TextEditWidget (class in flare.viur.bones.text), 133
 TextInsertImageAction (class in flare.viur.widgets.htmleditor), 137
 TextNode (class in flare.html5), 83
 TextNode (class in flare.html5.core), 42
 textToHtml() (in module flare.html5), 111
 textToHtml() (in module flare.html5.core), 69
 textToHtml() (in module flare.utils), 168
 TextViewWidget (class in flare.viur.bones.text), 133
 Th (class in flare.html5), 109
 Th (class in flare.html5.core), 68
 Thead (class in flare.html5), 109
 Thead (class in flare.html5.core), 68
 Time (class in flare.html5), 110
 Time (class in flare.html5.core), 68
 toggleArrow() (flare.viur.widgets.tree.TreeItemWidget method), 140
 toggleArrow() (flare.viur.widgets.tree.TreeLeafWidget method), 140
 toggleClass() (flare.html5.core.Widget method), 50
 toggleClass() (flare.html5.Widget method), 91

toggleExpand() (flare.viur.widgets.tree.TreeItemWidget method), 140
 ToolTip (class in flare.viur.formtooltip), 146
 tooltipWidget() (flare.viur.bones.base.BaseBone method), 118
 Tr (class in flare.html5), 108
 Tr (class in flare.html5.core), 67
 Track (class in flare.html5), 110
 Track (class in flare.html5.core), 69
 translate() (in module flare.i18n), 153
 TreeBrowserWidget (class in flare.viur.widgets.tree), 142
 TreeDirBone (class in flare.viur.bones.relational), 128
 TreeItemBone (class in flare.viur.bones.relational), 127
 TreeItemWidget (class in flare.viur.widgets.tree), 139
 TreeLeafWidget (class in flare.viur.widgets.tree), 140
 TreeNodeWidget (class in flare.viur.widgets.tree), 140
 TreeWidget (class in flare.viur.widgets.tree), 140

U

U (class in flare.html5), 102
 U (class in flare.html5.core), 61
 Ul (class in flare.html5), 106
 Ul (class in flare.html5.core), 65
 unescape() (in module flare.html5), 110
 unescape() (in module flare.html5.core), 69
 unescape() (in module flare.utils), 167
 unmarkDraggedElement() (flare.viur.widgets.tree.TreeItemWidget method), 139
 unobserve() (flare.intersectionObserver.IntersectionObserver method), 157
 unregister() (flare.event.EventDispatcher method), 151
 in unregister() (flare.observable.StateHandler method), 163
 unregister() (flare.views.StateHandler method), 115
 unserialize() (flare.viur.bones.base.BaseEditWidget method), 116
 unserialize() (flare.viur.bones.base.BaseLanguageEditWidget method), 117
 unserialize() (flare.viur.bones.base.BaseMultiEditWidget method), 117
 unserialize() (flare.viur.bones.base.BaseMultiViewWidget method), 117
 unserialize() (flare.viur.bones.base.BaseViewWidget method), 117
 unserialize() (flare.viur.bones.boolean.BooleanEditWidget method), 119
 unserialize() (flare.viur.bones.boolean.BooleanViewWidget method), 119
 unserialize() (flare.viur.bones.color.ColorViewWidget method), 120

unserialize() (flare.viur.bones.date.DateEditWidget method), 43
 unserialize() (flare.viur.bones.date.DateEditWidget method), 120
 unserialize() (flare.viur.bones.date.DateViewWidget method), 121
 unserialize() (flare.viur.bones.email.EmailViewWidget method), 121
 unserialize() (flare.viur.bones.numeric.NumericEditWidget method), 122
 unserialize() (flare.viur.bones.numeric.NumericViewWidget method), 123
 unserialize() (flare.viur.bones.raw.RawViewWidget method), 124
 unserialize() (flare.viur.bones.record.RecordEditWidget method), 125
 unserialize() (flare.viur.bones.record.RecordViewWidget method), 125
 unserialize() (flare.viur.bones.relational.FileEditDirectWidget method), 128
 unserialize() (flare.viur.bones.relational.FileEditWidget method), 129
 unserialize() (flare.viur.bones.relational.FileMultiEditDirectWidget method), 129
 unserialize() (flare.viur.bones.relational.FileViewWidget method), 128
 unserialize() (flare.viur.bones.relational.RelationalEditWidget method), 127
 unserialize() (flare.viur.bones.relational.RelationalViewWidget method), 127
 unserialize() (flare.viur.bones.select.SelectMultipleEditWidget method), 130
 unserialize() (flare.viur.bones.select.SelectSingleEditWidget method), 130
 unserialize() (flare.viur.bones.select.SelectViewWidget method), 131
 unserialize() (flare.viur.bones.spatial.SpatialEditWidget method), 132
 unserialize() (flare.viur.bones.string.StringEditWidget method), 132
 unserialize() (flare.viur.bones.string.StringViewWidget method), 133
 unserialize() (flare.viur.bones.text.TextViewWidget method), 134
 unserialize() (flare.viur.forms.ViurForm method), 145
 unserialize() (flare.viur.forms.ViurFormBone method), 145
 unsinkEvent() (flare.html5.core.Widget method), 44
 unsinkEvent() (flare.html5.Widget method), 85
 update() (flare.button.Button method), 148
 update() (flare.Cache method), 169
 update() (flare.cache.Cache method), 149
 update() (flare.html5._WidgetDataWrapper method), 84
 update() (flare.html5._WidgetStyleWrapper method), 85
 update() (flare.html5.core._WidgetDataWrapper method), 43
 update() (flare.html5.core._WidgetStyleWrapper method), 44
 update() (flare.viur.forms.ViurForm method), 145
 updateConf() (in module flare), 169
 updateConf() (in module flare.config), 150
 updateDefaultView() (in module flare.views.helpers), 113
 updateLength() (flare.viur.bones.string.StringEditWidget method), 132
 updateState() (flare.observable.StateHandler method), 162
 updateState() (flare.views.StateHandler method), 115
 updateString() (flare.viur.bones.relational.RelationalEditWidget method), 127
 updateStructure() (flare.Cache method), 169
 updateStructure() (flare.cache.Cache method), 149
 updateWidget() (flare.viur.bones.base.BaseEditWidget method), 116
 updateWidget() (flare.viur.bones.boolean.BooleanEditWidget method), 119
 updateWidget() (flare.viur.bones.color.ColorEditWidget method), 119
 updateWidget() (flare.viur.bones.date.DateEditWidget method), 120
 updateWidget() (flare.viur.bones.email.EmailEditWidget method), 121
 updateWidget() (flare.viur.bones.numeric.NumericEditWidget method), 122
 updateWidget() (flare.viur.bones.password.PasswordEditWidget method), 123
 updateWidget() (flare.viur.bones.raw.RawEditWidget method), 124
 updateWidget() (flare.viur.bones.record.RecordEditWidget method), 125
 updateWidget() (flare.viur.bones.relational.FileEditDirectWidget method), 128
 updateWidget() (flare.viur.bones.relational.RelationalEditWidget method), 127
 updateWidget() (flare.viur.bones.select.SelectMultipleEditWidget method), 130
 updateWidget() (flare.viur.bones.select.SelectSingleEditWidget method), 130
 updateWidget() (flare.viur.bones.spatial.SpatialEditWidget method), 131
 updateWidget() (flare.viur.bones.string.StringEditWidget method), 132
 updateWidget() (flare.viur.bones.text.TextEditWidget method), 133
 Uploader (class in flare.viur.widgets.file), 135
 urlForArgs() (flare.network.NetworkService static method), 161

V

`value` (*flare.observable.ObservableValue* attribute), 162
`Var` (class in *flare.html5*), 103
`Var` (class in *flare.html5.core*), 61
`Video` (class in *flare.html5*), 110
`Video` (class in *flare.html5.core*), 69
`View` (class in *flare.views.view*), 114
`ViewWidget` (class in *flare.views.view*), 114
`viewWidget()` (*flare.viur.bones.base.BaseBone* method), 118
`viewWidgetFactory` (*flare.viur.bones.base.BaseBone* attribute), 118
`viewWidgetFactory` (*flare.viur.bones.boolean.BooleanBone* attribute), 119
`viewWidgetFactory` (*flare.viur.bones.color.ColorBone* attribute), 120
`viewWidgetFactory` (*flare.viur.bones.date.DateBone* attribute), 121
`viewWidgetFactory` (*flare.viur.bones.email.EmailBone* attribute), 122
`viewWidgetFactory` (*flare.viur.bones.numeric.NumericBone* attribute), 123
`viewWidgetFactory` (*flare.viur.bones.raw.RawBone* attribute), 124
`viewWidgetFactory` (*flare.viur.bones.record.RecordBone* attribute), 125
`viewWidgetFactory` (*flare.viur.bones.relational.FileBone* attribute), 129
`viewWidgetFactory` (*flare.viur.bones.relational.FileDirectBone* attribute), 129
`viewWidgetFactory` (*flare.viur.bones.relational.RelationalBone* attribute), 127
`viewWidgetFactory` (*flare.viur.bones.select.SelectMultipleBone* attribute), 131
`viewWidgetFactory` (*flare.viur.bones.string.StringBone* attribute), 133
`viewWidgetFactory` (*flare.viur.bones.text.TextBone* attribute), 134
`ViurForm` (class in *flare.viur.forms*), 144
`ViurFormBone` (class in *flare.viur.forms*), 145
`ViurFormSubmit` (class in *flare.viur.forms*), 145

W

`warn()` (*webworker_scripts.weblog* static method), 170
`Wbr` (class in *flare.html5*), 103
`Wbr` (class in *flare.html5.core*), 62
`weblog` (class in *webworker_scripts*), 170
`webworker_scripts` module, 170
`Widget` (class in *flare.html5*), 85
`Widget` (class in *flare.html5.core*), 44

Z

`zip_listdir()` (in module *flare.views.helpers*), 114