

---

# Remi Documentation

*Release 1.0.0*

**Davide Rosa**

**Dec 15, 2018**



---

## Contents

---

<b>1</b>	<b>remi package</b>	<b>3</b>
1.1	Submodules . . . . .	3
1.2	remi.gui module . . . . .	3
1.3	remi.server module . . . . .	25
1.4	Module contents . . . . .	28
<b>2</b>	<b>remi</b>	<b>29</b>
<b>3</b>	<b>Indices and tables</b>	<b>31</b>
	<b>Python Module Index</b>	<b>33</b>



Contents:



## 1.1 Submodules

## 1.2 remi.gui module

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class remi.gui.BODY(*args, **kwargs)
    Bases: remi.gui.Widget

    EVENT_ONERROR = 'onerror'
    EVENT_ONLOAD = 'onload'
    EVENT_ONONLINE = 'ononline'
    EVENT_ONPAGEHIDE = 'onpagehide'
    EVENT_ONPAGESHOW = 'onpageshow'
    EVENT_ONRESIZE = 'onresize'

    onerror(message, source, lineno, colno)
        Called when an error occurs.

    onload()
        Called when page gets loaded.

    ononline()

    onpagehide()
```

**onpageshow** ()

**onresize** (*width, height*)

**class** `remi.gui.Button` (*text=*”, *\*args, \*\*kwargs*)

Bases: `remi.gui.Widget`, `remi.gui._MixinTextualWidget`

The Button widget. Have to be used in conjunction with its event onclick. Use `Widget.onclick.connect` in order to register the listener.

**class** `remi.gui.CheckBox` (*checked=False, user\_data=*”, *\*\*kwargs*)

Bases: `remi.gui.Input`

check box widget useful as numeric input field implements the onchange event.

**get\_value** ()

**Returns**

**Return type** bool

**onchange** (*value*)

**set\_value** (*checked, update\_ui=1*)

**class** `remi.gui.CheckBoxLabel` (*label=*”, *checked=False, user\_data=*”, *\*\*kwargs*)

Bases: `remi.gui.Widget`

**onchange** (*widget, value*)

**set\_on\_change\_listener** (*callback, \*userdata*)

Registers the listener For backward compatibility Suggested new dialect `event.connect(callback, *userdata)`

**class** `remi.gui.ClassEventConnector` (*event\_source\_instance,* *event\_name,*  
*event\_method\_bound*)

Bases: `object`

This class allows to manage the events. Decorating a method with `decorate_event` decorator The method gets the `__is_event` flag. At runtime, the methods that has this flag gets replaced by a `ClassEventConnector`. This class overloads the `__call__` method, where the event method is called, and after that the listener method is called too.

**connect** (*callback, \*userdata*)

The callback and userdata gets stored, and if there is some javascript to add the js code is appended as attribute for the event source

**class** `remi.gui.ColorPicker` (*default\_value='#995500', \*\*kwargs*)

Bases: `remi.gui.Input`

**class** `remi.gui.Date` (*default\_value='2015-04-13', \*\*kwargs*)

Bases: `remi.gui.Input`

**class** `remi.gui.DropDown` (*\*args, \*\*kwargs*)

Bases: `remi.gui.Widget`

Drop down selection widget. Implements the `onchange(value)` event. Register a listener for its selection change by means of the function `DropDown.onchange.connect`.

**append** (*value, key=*”)

Adds a child widget, generating and returning a key if not provided

In order to access to the specific child in this way `widget.children[key]`.

**Parameters**



- **value** (*Widget*, or *iterable of Widgets*) – The child to be appended. In case of a dictionary, each item’s key is used as ‘key’ param for the single append.
- **key** (*str*) – The unique string identifier for the child. Ignored in case of iterable ‘value’ param.

**Returns**

a key used to refer to the child for all future interaction, or a list of keys in case of an iterable ‘value’ param

**Return type** str

**empty()**

remove all children from the widget

**get\_item()**

**Returns** The selected item or None.

**Return type** *DropDownItem*

**get\_key()**

**Returns** The unique string identifier of the selected item or None.

**Return type** str

**get\_value()**

**Returns** The value of the selected item or None.

**Return type** str

**classmethod new\_from\_list** (*items*, *\*\*kwargs*)

**onchange** (*value*)

Called when a new DropDownItem gets selected.

**select\_by\_key** (*key*)

Selects an item by its unique string identifier.

**Parameters** **key** (*str*) – Unique string identifier of the DropDownItem that have to be selected.

**select\_by\_value** (*value*)

Selects a DropDownItem by means of the contained text-

**Parameters** **value** (*str*) – Textual content of the DropDownItem that have to be selected.

**set\_on\_change\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect `event.connect(callback, *userdata)`

**set\_value** (*value*)

**class** `remi.gui.DropDownItem` (*text*, *\*args*, *\*\*kwargs*)

Bases: `remi.gui.Widget`, `remi.gui._MixinTextualWidget`

item widget for the DropDown

**get\_value** ()

**set\_value** (*text*)

**class** `remi.gui.EventSource` (*\*args*, *\*\*kwargs*)

Bases: `object`

**setup\_event\_methods** ()

```
class remi.gui.FileDownloader(text, filename, path_separator='/', *args, **kwargs)
    Bases: remi.gui.Widget, remi.gui._MixinTextualWidget

    FileDownloader widget. Allows to start a file download.

    download()

class remi.gui.FileFolderItem(text, is_folder=False, **kwargs)
    Bases: remi.gui.Widget

    FileFolderItem widget for the FileFolderNavigator

    get_text()

    onclick(widget)
        Called when the Widget gets clicked by the user with the left mouse button.

    onselection(widget)

    set_on_click_listener(callback, *userdata)
        Registers the listener For backward compatibility Suggested new dialect event.connect(callback, *user-
        data)

    set_on_selection_listener(callback, *userdata)
        Registers the listener For backward compatibility Suggested new dialect event.connect(callback, *user-
        data)

    set_selected(selected)

    set_text(t)

class remi.gui.FileFolderNavigator(multiple_selection, selection_folder, allow_file_selection,
                                   allow_folder_selection, **kwargs)
    Bases: remi.gui.Widget

    FileFolderNavigator widget.

    chdir(directory)

    dir_go(widget)

    dir_go_back(widget)

    get_selected_filefolders()

    get_selection_list()

    on_folder_item_click(folderitem)

    on_folder_item_selected(folderitem)

    populate_folder_items(directory)

class remi.gui.FileSelectionDialog(title='File dialog', message='Select files and fold-
                                   ers', multiple_selection=True, selection_folder='.',
                                   allow_file_selection=True, allow_folder_selection=True,
                                   **kwargs)
    Bases: remi.gui.GenericDialog

    file selection dialog, it opens a new webpage allows the OK/CANCEL functionality implementing the “con-
    firm_value” and “cancel_dialog” events.

    confirm_value(widget)
        event called pressing on OK button. propagates the string content of the input field
```

**set\_on\_confirm\_value\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**class** `remi.gui.FileUploader` (*savepath*='.', *multiple\_selection\_allowed*=False, *\*args*, *\*\*kwargs*)

Bases: `remi.gui.Widget`

**FileUploader widget:** allows to upload multiple files to a specified folder. implements the onsuccess and onfailed events.

**ondata** (*filedata*, *filename*)

**onfailed** (*filename*)

**onsuccess** (*filename*)

**set\_on\_data\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_failed\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_success\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**class** `remi.gui.GenericDialog` (*title*="", *message*="", *\*args*, *\*\*kwargs*)

Bases: `remi.gui.Widget`

Generic Dialog widget. It can be customized to create personalized dialog windows. You can setup the content adding content widgets with the functions `add_field` or `add_field_with_label`. The user can confirm or dismiss the dialog with the common buttons Cancel/Ok. Each field added to the dialog can be retrieved by its key, in order to get back the edited value. Use the function

`get_field(key)` to retrieve the field.

The Ok button emits the 'confirm\_dialog' event. Register the listener to it with `set_on_confirm_dialog_listener`. The Cancel button emits the 'cancel\_dialog' event. Register the listener to it with `set_on_cancel_dialog_listener`.

**add\_field** (*key*, *field*)

Adds a field to the dialog with a unique identifier.

Note: You can access to the fields content calling the function `GenericDialog.get_field(key)`.

#### Parameters

- **key** (*str*) – The unique identifier for the field.
- **field** (*Widget*) – The widget to be added to the dialog, `TextInput` or any `Widget` for example.

**add\_field\_with\_label** (*key*, *label\_description*, *field*)

Adds a field to the dialog together with a descriptive label and a unique identifier.

Note: You can access to the fields content calling the function `GenericDialog.get_field(key)`.

#### Parameters

- **key** (*str*) – The unique identifier for the field.
- **label\_description** (*str*) – The string content of the description label.

- **field**(*Widget*) – The instance of the field Widget. It can be for example a TextInput or maybe
- **custom\_widget.**(*a*) –

**cancel\_dialog**(*emitter*)

Event generated by the Cancel button click.

**confirm\_dialog**(*emitter*)

Event generated by the OK button click.

**get\_field**(*key*)

**Parameters** **key** (*str*) – The unique string identifier of the required field.

**Returns** Widget field instance added previously with methods `GenericDialog.add_field` or `GenericDialog.add_field_with_label`.

**hide**()

**set\_on\_cancel\_dialog\_listener**(*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect `event.connect(callback, *userdata)`

**set\_on\_confirm\_dialog\_listener**(*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect `event.connect(callback, *userdata)`

**show**(*base\_app\_instance*)

**class** `remi.gui.GenericObject` (*filename*, *\*\*kwargs*)

Bases: `remi.gui.Widget`

GenericObject widget - allows to show embedded object like pdf,swf..

**class** `remi.gui.GridBox` (*\*args*, *\*\*kwargs*)

Bases: `remi.gui.Widget`

It contains widgets automatically aligning them to the grid. Does not permit children absolute positioning.

In order to add children to this container, use the `append(child, key)` function. The key have to be string and determines the children positioning in the layout.

Note: If you would absolute positioning, use the `Widget` container instead.

**append**(*value*, *key=""*)

Adds a child widget, generating and returning a key if not provided

In order to access to the specific child in this way `widget.children[key]`.

#### Parameters

- **value** (*Widget*, or *iterable of Widgets*) – The child to be appended. In case of a dictionary, each item's key is used as 'key' param for the single append.
- **key** (*str*) – The unique string identifier for the child. Ignored in case of iterable 'value' param. The key have to correspond to a an element provided in the 'define\_grid' method param.

#### Returns

a key used to refer to the child for all future interaction, or a list of keys in case of an iterable 'value' param

**Return type** `str`

**define\_grid** (*matrix*)

Populates the Table with a list of tuples of strings.

**Parameters** **matrix** (*list*) – list of iterables of strings (lists or something else). Items in the matrix have to correspond to a key for the children.

**remove\_child** (*child*)

Removes a child instance from the Tag's children.

**Parameters** **child** (*Tag*) – The child to be removed.

**set\_column\_gap** (*value*)

Sets the gap value between columns

**Parameters** **value** (*int or str*) – gap value (i.e. 10 or "10px")

**set\_column\_sizes** (*values*)

Sets the size value for each column

**Parameters** **values** (*iterable of int or str*) – values are treated as percentage.

**set\_row\_gap** (*value*)

Sets the gap value between rows

**Parameters** **value** (*int or str*) – gap value (i.e. 10 or "10px")

**set\_row\_sizes** (*values*)

Sets the size value for each row

**Parameters** **values** (*iterable of int or str*) – values are treated as percentage.

**class** remi.gui.HBox (\*args, \*\*kwargs)

Bases: *remi.gui.Widget*

**The purpose of this widget is to automatically horizontally aligning** the widgets that are appended to it.

Does not permit children absolute positioning.

In order to add children to this container, use the append(child, key) function. The key have to be numeric and determines the children order in the layout.

Note: If you would absolute positioning, use the Widget container instead.

**append** (*value, key=""*)

It allows to add child widgets to this. The key allows to access the specific child in this way widget.children[key]. The key have to be numeric and determines the children order in the layout.

**Parameters**

- **value** (*Widget*) – Child instance to be appended.
- **key** (*str*) – Unique identifier for the child. If key.isdigit()==True '0' '1'.. the value determines the order
- **the layout** (*in*) –

**class** remi.gui.HEAD (*title, \*args, \*\*kwargs*)

Bases: *remi.gui.Tag*

**repr** (*changed\_widgets=None*)

It is used to automatically represent the object to HTML format packs all the attributes, children and so on.

**Parameters** **changed\_widgets** (*dict*) – A dictionary containing a collection of tags that have to be updated. The tag that have to be updated is the key, and the value is its textual repr.

```
set_internal_js (net_interface_ip, pending_messages_queue_length, web-socket_timeout_timer_ms)  
set_title (title)
```

```
class remi.gui.HTML (*args, **kwargs)
```

Bases: `remi.gui.Tag`

```
repr (changed_widgets=None)
```

It is used to automatically represent the object to HTML format packs all the attributes, children and so on.

**Parameters** **changed\_widgets** (*dict*) – A dictionary containing a collection of tags that have to be updated. The tag that have to be updated is the key, and the value is its textual repr.

```
class remi.gui.Image (filename, *args, **kwargs)
```

Bases: `remi.gui.Widget`

image widget.

```
set_image (filename)
```

**Parameters** **filename** (*str*) – an url to an image

```
class remi.gui.Input (input_type="", default_value="", *args, **kwargs)
```

Bases: `remi.gui.Widget`

```
get_value ()
```

returns the new text value.

```
onchange (value)
```

```
set_on_change_listener (callback, *userdata)
```

Registers the listener For backward compatibility Suggested new dialect `event.connect(callback, *userdata)`

```
set_read_only (readonly)
```

```
set_value (value)
```

```
class remi.gui.InputDialog (title='Title', message='Message', initial_value="", *args, **kwargs)
```

Bases: `remi.gui.GenericDialog`

Input Dialog widget. It can be used to query simple and short textual input to the user. The user can confirm or dismiss the dialog with the common buttons Cancel/Ok. The Ok button click or the ENTER key presson emits the 'confirm\_dialog' event. Register the listener to it with `set_on_confirm_dialog_listener`. The Cancel button emits the 'cancel\_dialog' event. Register the listener to it with `set_on_cancel_dialog_listener`.

```
confirm_value (widget)
```

Event called pressing on OK button.

```
on_keydown_listener (widget, value, keycode)
```

event called pressing on ENTER key.

propagates the string content of the input field

```
set_on_confirm_value_listener (callback, *userdata)
```

Registers the listener For backward compatibility Suggested new dialect `event.connect(callback, *userdata)`

```
class remi.gui.Label (text, *args, **kwargs)
```

Bases: `remi.gui.Widget`, `remi.gui._MixinTextualWidget`

Non editable text label widget. Set its content by means of `set_text` function, and retrieve its content with the function `get_text`.

```
class remi.gui.Link(url, text, open_new_window=True, *args, **kwargs)
    Bases: remi.gui.Widget, remi.gui._MixinTextualWidget

    get_url()
```

```
class remi.gui.ListItem(text, *args, **kwargs)
    Bases: remi.gui.Widget, remi.gui._MixinTextualWidget

    List item widget for the ListView.
```

ListItems are characterized by a textual content. They can be selected from the ListView. Do NOT manage directly its selection by registering `set_on_click_listener`, use instead the events of the ListView.

```
get_value()

Returns The text content of the ListItem

Return type str
```

```
class remi.gui.ListView(selectable=True, *args, **kwargs)
    Bases: remi.gui.Widget
```

List widget it can contain ListItems. Add items to it by using the standard `append(item, key)` function or generate a filled list from a string list by means of the function `new_from_list`. Use the list in conjunction of its onselection event. Register a listener with `ListView.onselection.connect`.

```
append(value, key="")
    Appends child items to the ListView. The items are accessible by list.children[key].
```

#### Parameters

- **value** (*ListItem*, or *iterable of ListItems*) – The child to be appended. In case of a dictionary, each item's key is used as 'key' param for the single append.
- **key** (*str*) – The unique string identifier for the child. Ignored in case of iterable 'value' param.

```
empty()
    Removes all children from the list
```

```
get_item()

Returns The selected item or None
```

**Return type** *ListItem*

```
get_key()

Returns The key of the selected item or None if no item is selected.

Return type str
```

```
get_value()

Returns The value of the selected item or None

Return type str
```

```
classmethod new_from_list(items, **kwargs)
    Populates the ListView with a string list.
```

**Parameters** **items** (*list*) – list of strings to fill the widget with.

**onselection** (*widget*)

Called when a new item gets selected in the list.

**select\_by\_key** (*key*)

Selects an item by its key.

**Parameters** **key** (*str*) – The unique string identifier of the item that have to be selected.

**select\_by\_value** (*value*)

Selects an item by the text content of the child.

**Parameters** **value** (*str*) – Text content of the item that have to be selected.

**set\_on\_selection\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect `event.connect(callback, *userdata)`

**set\_value** (*value*)

**class** `remi.gui.Menu` (*\*args*, *\*\*kwargs*)

Bases: `remi.gui.Widget`

Menu widget can contain MenuItem.

**class** `remi.gui.MenuBar` (*\*args*, *\*\*kwargs*)

Bases: `remi.gui.Widget`

**class** `remi.gui.MenuItem` (*text*, *\*args*, *\*\*kwargs*)

Bases: `remi.gui.Widget`, `remi.gui._MixinTextualWidget`

MenuItem widget can contain other MenuItem.

**append** (*value*, *key=""*)

Adds a child widget, generating and returning a key if not provided

In order to access to the specific child in this way `widget.children[key]`.

#### Parameters

- **value** (*Widget*, or *iterable of Widgets*) – The child to be appended. In case of a dictionary, each item's key is used as 'key' param for the single append.
- **key** (*str*) – The unique string identifier for the child. Ignored in case of iterable 'value' param.

#### Returns

a key used to refer to the child for all future interaction, or a list of keys in case of an iterable 'value' param

**Return type** `str`

**class** `remi.gui.Progress` (*value=0*, *\_max=100*, *\*args*, *\*\*kwargs*)

Bases: `remi.gui.Widget`

Progress bar widget.

**set\_max** (*\_max*)

**Parameters** **max** (*int*) – The maximum progress value.

**set\_value** (*value*)

**Parameters** **value** (*int*) – The actual progress value.

**class** `remi.gui.Slider` (*default\_value=""*, *min=0*, *max=10000*, *step=1*, *\*\*kwargs*)

Bases: `remi.gui.Input`



**oninput** (*value*)

**set\_oninput\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect `event.connect(callback, *userdata)`

**class** `remi.gui.SpinBox` (*default\_value=100*, *min\_value=100*, *max\_value=5000*, *step=1*, *allow\_editing=True*, *\*\*kwargs*)

Bases: `remi.gui.Input`

spin box widget useful as numeric input field implements the onchange event.

**class** `remi.gui.Svg` (*width*, *height*, *\*args*, *\*\*kwargs*)

Bases: `remi.gui.Widget`

svg widget - is a container for graphic widgets such as `SvgCircle`, `SvgLine` and so on.

**set\_viewbox** (*x*, *y*, *w*, *h*)

Sets the origin and size of the viewbox, describing a virtual view area.

#### Parameters

- **x** (*int*) – x coordinate of the viewbox origin
- **y** (*int*) – y coordinate of the viewbox origin
- **w** (*int*) – width of the viewbox
- **h** (*int*) – height of the viewbox

**class** `remi.gui.SvgCircle` (*x*, *y*, *radius*, *\*args*, *\*\*kwargs*)

Bases: `remi.gui.SvgShape`

svg circle - a circle represented filled and with a stroke.

**set\_position** (*x*, *y*)

Sets the circle position.

#### Parameters

- **x** (*int*) – the x coordinate
- **y** (*int*) – the y coordinate

**set\_radius** (*radius*)

Sets the circle radius.

**Parameters** **radius** (*int*) – the circle radius

**class** `remi.gui.SvgGroup` (*x*, *y*, *\*args*, *\*\*kwargs*)

Bases: `remi.gui.SvgShape`

svg group widget.

**class** `remi.gui.SvgLine` (*x1*, *y1*, *x2*, *y2*, *\*args*, *\*\*kwargs*)

Bases: `remi.gui.Widget`

**set\_coords** (*x1*, *y1*, *x2*, *y2*)

**set\_p1** (*x1*, *y1*)

**set\_p2** (*x2*, *y2*)

**set\_stroke** (*width=1*, *color='black'*)

**class** `remi.gui.SvgPath` (*path\_value*, *\*args*, *\*\*kwargs*)

Bases: `remi.gui.Widget`

**add\_arc** (*x*, *y*, *rx*, *ry*, *x\_axis\_rotation*, *large\_arc\_flag*, *sweep\_flag*)

**add\_position** (*x*, *y*)

**set\_fill** (*color*=*'black'*)

Sets the fill color.

**Parameters** **color** (*str*) – stroke color

**set\_stroke** (*width*=*1*, *color*=*'black'*)

Sets the stroke properties.

**Parameters**

- **width** (*int*) – stroke width
- **color** (*str*) – stroke color

**class** remi.gui.SvgPolyline (*\_maxlen*=*None*, *\*args*, *\*\*kwargs*)

Bases: *remi.gui.Widget*

**add\_coord** (*x*, *y*)

**set\_stroke** (*width*=*1*, *color*=*'black'*)

**class** remi.gui.SvgRectangle (*x*, *y*, *w*, *h*, *\*args*, *\*\*kwargs*)

Bases: *remi.gui.SvgShape*

svg rectangle - a rectangle represented filled and with a stroke.

**set\_size** (*w*, *h*)

Sets the rectangle size.

**Parameters**

- **w** (*int*) – width of the rectangle
- **h** (*int*) – height of the rectangle

**class** remi.gui.SvgShape (*x*, *y*, *\*args*, *\*\*kwargs*)

Bases: *remi.gui.Widget*

svg shape generic widget. Consists of a position, a fill color and a stroke.

**set\_fill** (*color*=*'black'*)

Sets the fill color.

**Parameters** **color** (*str*) – stroke color

**set\_position** (*x*, *y*)

Sets the shape position.

**Parameters**

- **x** (*int*) – the x coordinate
- **y** (*int*) – the y coordinate

**set\_stroke** (*width*=*1*, *color*=*'black'*)

Sets the stroke properties.

**Parameters**

- **width** (*int*) – stroke width
- **color** (*str*) – stroke color

```
class remi.gui.SvgText (x, y, text, *args, **kwargs)
    Bases: remi.gui.SvgShape, remi.gui._MixinTextualWidget
```

```
class remi.gui.TabBox (*args, **kwargs)
    Bases: remi.gui.Widget
```

```
    add_tab (widget, name, tab_cb)
```

```
    select_by_index (index)
        shows a tab identified by its index
```

```
    select_by_name (name)
        shows a tab identified by the name
```

```
    select_by_widget (widget)
        shows a tab identified by the contained widget
```

```
class remi.gui.Table (*args, **kwargs)
    Bases: remi.gui.Widget
```

table widget - it will contains TableRow

```
append (value, key="")
    Adds a child widget, generating and returning a key if not provided

    In order to access to the specific child in this way widget.children[key].
```

#### Parameters

- **value** (*Widget*, or *iterable of Widgets*) – The child to be appended. In case of a dictionary, each item's key is used as 'key' param for the single append.
- **key** (*str*) – The unique string identifier for the child. Ignored in case of iterable 'value' param.

#### Returns

a key used to refer to the child for all future interaction, or a list of keys in case of an iterable 'value' param

#### Return type

```
append_from_list (content, fill_title=False)
```

Appends rows created from the data contained in the provided list of tuples of strings. The first tuple of the list can be set as table title.

#### Parameters

- **content** (*list*) – list of tuples of strings. Each tuple is a row.
- **fill\_title** (*bool*) – if true, the first tuple in the list will be set as title.

```
classmethod new_from_list (content, fill_title=True, **kwargs)
    Populates the Table with a list of tuples of strings.
```

#### Parameters

- **content** (*list*) – list of tuples of strings. Each tuple is a row.
- **fill\_title** (*bool*) – if true, the first tuple in the list will be set as title

```
on_table_row_click (row, item)
```

```
set_on_table_row_click_listener (callback, *userdata)
```

Registers the listener For backward compatibility Suggested new dialect event.connect(callback, \*userdata)

```
class remi.gui.TableEditableItem(text="", *args, **kwargs)
    Bases: remi.gui.Widget, remi.gui._MixinTextualWidget
    item widget for the TableRow.
    onchange (emitter, new_value)
    set_on_change_listener (callback, *userdata)
        Registers the listener For backward compatibility Suggested new dialect event.connect(callback, *user-
        data)
```

```
class remi.gui.TableItem(text="", *args, **kwargs)
    Bases: remi.gui.Widget, remi.gui._MixinTextualWidget
    item widget for the TableRow.
```

```
class remi.gui.TableRow(*args, **kwargs)
    Bases: remi.gui.Widget
    row widget for the Table - it will contains TableItem
    append (value, key="")
        Adds a child widget, generating and returning a key if not provided
        In order to access to the specific child in this way widget.children[key].
```

#### Parameters

- **value** (*Widget*, or *iterable of Widgets*) – The child to be appended. In case of a dictionary, each item's key is used as 'key' param for the single append.
- **key** (*str*) – The unique string identifier for the child. Ignored in case of iterable 'value' param.

#### Returns

a key used to refer to the child for all future interaction, or a list of keys in case of an iterable 'value' param

#### Return type

str

```
on_row_item_click (item)
    Event on item click.
```

**Note:** This is internally used by the Table widget in order to generate the Table.on\_table\_row\_click event. Use Table.on\_table\_row\_click instead.

#### Parameters

- **emitter** (*TableRow*) – The emitter of the event.
- **item** (*TableItem*) – The clicked TableItem.

```
set_on_row_item_click_listener (callback, *userdata)
    Registers the listener For backward compatibility Suggested new dialect event.connect(callback, *user-
    data)
```

```
class remi.gui.TableTitle(text="", *args, **kwargs)
    Bases: remi.gui.TableItem, remi.gui._MixinTextualWidget
    title widget for the table.
```

```
class remi.gui.TableWidget(n_rows, n_columns, use_title=True, editable=False, *args, **kwargs)
    Bases: remi.gui.Table
```

Basic table model widget. Each item is addressed by stringified integer key in the children dictionary.

**column\_count** ()

Returns table's columns count.

**item\_at** (*row*, *column*)

Returns the TableItem instance at row, column coordinates

#### Parameters

- **row** (*int*) – zero based index
- **column** (*int*) – zero based index

**item\_coords** (*table\_item*)

Returns table\_item's (row, column) coordinates. Returns None in case of item not found.

**Parameters** **table\_item** (*TableItem*) – an item instance

**on\_item\_changed** (*item*, *new\_value*, *row*, *column*)

Event for the item change.

#### Parameters

- **emitter** (*TableWidget*) – The emitter of the event.
- **item** (*TableItem*) – The TableItem instance.
- **new\_value** (*str*) – New text content.
- **row** (*int*) – row index.
- **column** (*int*) – column index.

**row\_count** ()

Returns table's rows count (the title is considered as a row).

**set\_column\_count** (*count*)

Sets the table column count.

**Parameters** **count** (*int*) – column of rows

**set\_on\_item\_changed\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect event.connect(callback, *\*userdata*)

**set\_row\_count** (*count*)

Sets the table row count.

**Parameters** **count** (*int*) – number of rows

**set\_use\_title** (*use\_title*)

Returns the TableItem instance at row, column coordinates

**Parameters** **use\_title** (*bool*) – enable title bar.

**class** remi.gui.Tag (*attributes=None*, *\_type=""*, *\_class=None*, *\*\*kwargs*)

Bases: object

Tag is the base class of the framework. It represents an element that can be added to the GUI, but it is not necessarily graphically representable. You can use this class for sending javascript code to the clients.

**add\_child** (*key*, *value*)

Adds a child to the Tag

To retrieve the child call get\_child or access to the Tag.children[key] dictionary.

**Parameters**

- **key** (*str*) – Unique child’s identifier, or iterable of keys
- **value** (*Tag*, *str*) – can be a Tag, an iterable of Tag or a str. In case of iterable of Tag is a dict, each item’s key is set as ‘key’ param

**add\_class** (*cls*)**disable\_refresh** ()**empty** ()

remove all children from the widget

**enable\_refresh** ()**get\_child** (*key*)

Returns the child identified by ‘key’

**Parameters** **key** (*str*) – Unique identifier of the child.**get\_parent** ()

Returns the parent tag instance or None where not applicable

**identifier****innerHTML** (*local\_changed\_widgets*)**remove\_child** (*child*)

Removes a child instance from the Tag’s children.

**Parameters** **child** (*Tag*) – The child to be removed.**remove\_class** (*cls*)**repr** (*changed\_widgets=None*)

It is used to automatically represent the object to HTML format packs all the attributes, children and so on.

**Parameters** **changed\_widgets** (*dict*) – A dictionary containing a collection of tags that have to be updated. The tag that have to be updated is the key, and the value is its textual repr.**set\_identifier** (*new\_identifier*)

Allows to set a unique id for the Tag.

**Parameters** **new\_identifier** (*str*) – a unique id for the tag**class** `remi.gui.TextInput` (*single\_line=True*, *hint=""*, *\*args*, *\*\*kwargs*)Bases: `remi.gui.Widget`, `remi.gui._MixinTextualWidget`Editable multiline/single\_line text area widget. You can set the content by means of the function `set_text` or retrieve its content with `get_text`.**get\_value** ()**Returns** The text content of the TextInput. You can set the text content with `set_text(text)`.**Return type** `str`**onchange** (*new\_value*)Called when the user changes the TextInput content. With `single_line=True` it fires in case of focus lost and Enter key pressed. With `single_line=False` it fires at each key released.**Parameters** **new\_value** (*str*) – the new string content of the TextInput.

**onkeydown** (*new\_value*, *keycode*)

Called when the user types a key into the TextInput.

Note: This event can't be registered together with Widget.onChange.

#### Parameters

- **new\_value** (*str*) – the new string content of the TextInput.
- **keycode** (*str*) – the numeric char code

**onkeyup** (*new\_value*, *keycode*)

Called when user types and releases a key into the TextInput

Note: This event can't be registered together with Widget.onChange.

#### Parameters

- **new\_value** (*str*) – the new string content of the TextInput
- **keycode** (*str*) – the numeric char code

**set\_on\_change\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_key\_down\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_key\_up\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_value** (*text*)

Sets the text content.

**Parameters** **text** (*str*) – The string content that have to be appended as standard child identified by the key 'text'

**class** `remi.gui.TreeItem` (*text*, *\*args*, *\*\*kwargs*)

Bases: `remi.gui.Widget`, `remi.gui._MixinTextualWidget`

TreeItem widget can contain other TreeItem.

**append** (*value*, *key=""*)

Adds a child widget, generating and returning a key if not provided

In order to access to the specific child in this way `widget.children[key]`.

#### Parameters

- **value** (*Widget*, or *iterable of Widgets*) – The child to be appended. In case of a dictionary, each item's key is used as 'key' param for the single append.
- **key** (*str*) – The unique string identifier for the child. Ignored in case of iterable 'value' param.

#### Returns

a key used to refer to the child for all future interaction, or a list of keys in case of an iterable 'value' param

**Return type** `str`

**onclick()**

Called when the Widget gets clicked by the user with the left mouse button.

**class** `remi.gui.TreeView(*args, **kwargs)`

Bases: `remi.gui.Widget`

TreeView widget can contain TreeItem.

**class** `remi.gui.VBox(*args, **kwargs)`

Bases: `remi.gui.HBox`

**The purpose of this widget is to automatically vertically aligning** the widgets that are appended to it.

Does not permit children absolute positioning.

In order to add children to this container, use the `append(child, key)` function. The key have to be numeric and determines the children order in the layout.

Note: If you would absolute positioning, use the Widget container instead.

**class** `remi.gui.VideoPlayer(video, poster=None, autoplay=False, loop=False, *args, **kwargs)`

Bases: `remi.gui.Widget`

**onended()**

Called when the media has been played and reached the end.

**set\_autoplay** (*autoplay*)

**set\_loop** (*loop*)

Sets the VideoPlayer to restart video when finished.

Note: If set as True the event onended will not fire.

**set\_on\_ended\_listener** (*callback, \*userdata*)

Registers the listener For backward compatibility Suggested new dialect `event.connect(callback, *userdata)`

**class** `remi.gui.Widget(children=None, style=None, *args, **kwargs)`

Bases: `remi.gui.Tag`, `remi.gui.EventSource`

Base class for gui widgets.

Widget can be used as generic container. You can add children by the `append(value, key)` function. Widget can be arranged in absolute positioning (assigning `style['top']` and `style['left']` attributes to the children or in a simple auto-alignment. You can decide the horizontal or vertical arrangement by the function `set_layout_orientation(layout_orientation)` passing as parameter `Widget.LAYOUT_HORIZONTAL` or `Widget.LAYOUT_VERTICAL`.

Tips: In html, it is a DIV tag The `self.type` attribute specifies the HTML tag representation The `self.attributes[]` attribute specifies the HTML attributes like 'style' 'class' 'id' The `self.style[]` attribute specifies the CSS style content like 'font' 'color'. It will be packed together with 'self.attributes'.

**EVENT\_ONBLUR** = 'onblur'

**EVENT\_ONCHANGE** = 'onchange'

**EVENT\_ONCLICK** = 'onclick'

**EVENT\_ONCONTEXTMENU** = 'oncontextmenu'

**EVENT\_ONDBLCLICK** = 'ondblclick'

**EVENT\_ONFOCUS** = 'onfocus'

**EVENT\_ONINPUT** = 'oninput'



```

EVENT_ONKEYDOWN = 'onkeydown'
EVENT_ONKEYPRESS = 'onkeypress'
EVENT_ONKEYUP = 'onkeyup'
EVENT_ONMOUSEDOWN = 'onmousedown'
EVENT_ONMOUSELEAVE = 'onmouseleave'
EVENT_ONMOUSEMOVE = 'onmousemove'
EVENT_ONMOUSEOUT = 'onmouseout'
EVENT_ONMOUSEOVER = 'onmouseover'
EVENT_ONMOUSEUP = 'onmouseup'
EVENT_ONTOUCHCANCEL = 'ontouchcancel'
EVENT_ONTOUCHEND = 'ontouchend'
EVENT_ONTOUCHENTER = 'ontouchenter'
EVENT_ONTOUCHLEAVE = 'ontouchleave'
EVENT_ONTOUCHMOVE = 'ontouchmove'
EVENT_ONTOUCHSTART = 'ontouchstart'
EVENT_ONUPDATE = 'onupdate'
LAYOUT_HORIZONTAL = True
LAYOUT_VERTICAL = False

```

**append** (*value*, *key*=*''*)

Adds a child widget, generating and returning a key if not provided

In order to access to the specific child in this way `widget.children[key]`.

#### Parameters

- **value** (*Widget*, or *iterable of Widgets*) – The child to be appended. In case of a dictionary, each item's key is used as 'key' param for the single append.
- **key** (*str*) – The unique string identifier for the child. Ignored in case of iterable 'value' param.

#### Returns

**a key used to refer to the child for all future interaction, or a list of keys in case of an iterable 'value' param**

**Return type** `str`

**onblur** ()

Called when the Widget loses focus

**onclick** ()

Called when the Widget gets clicked by the user with the left mouse button.

**oncontextmenu** ()

Called when the Widget gets clicked by the user with the right mouse button.

**ondblclick** ()

Called when the Widget gets double clicked by the user with the left mouse button.

**onfocus ()**

Called when the Widget gets focus.

**onkeydown** (*key, keycode, ctrl, shift, alt*)

Called when user types and releases a key. The widget should be able to receive the focus in order to emit the event. Assign a 'tabindex' attribute to make it focusable.

**Parameters**

- **key** (*str*) – the character value
- **keycode** (*str*) – the numeric char code

**onkeyup** (*key, keycode, ctrl, shift, alt*)

Called when user types and releases a key. The widget should be able to receive the focus in order to emit the event. Assign a 'tabindex' attribute to make it focusable.

**Parameters**

- **key** (*str*) – the character value
- **keycode** (*str*) – the numeric char code

**onmousedown** (*x, y*)

Called when the user presses left or right mouse button over a Widget.

**Parameters**

- **x** (*float*) – position of the mouse inside the widget
- **y** (*float*) – position of the mouse inside the widget

**onmouseleave** ()

Called when the mouse cursor moves outside a Widget.

**Note:** This event is often used together with the **Widget.onmouseenter** event, which occurs when the mouse pointer is moved onto a Widget.

**Note:** The **Widget.onmouseleave** event is similar to the **Widget.onmouseout** event. The only difference is that the **onmouseleave** event does not bubble (does not propagate up the Widgets tree).

**onmousemove** (*x, y*)

Called when the mouse cursor moves inside the Widget.

**Parameters**

- **x** (*float*) – position of the mouse inside the widget
- **y** (*float*) – position of the mouse inside the widget

**onmouseout** ()

Called when the mouse cursor moves outside a Widget.

**Note:** This event is often used together with the **Widget.onmouseover** event, which occurs when the pointer is moved onto a Widget, or onto one of its children.

**onmouseup** (*x, y*)

Called when the user releases left or right mouse button over a Widget.

**Parameters**

- **x** (*float*) – position of the mouse inside the widget
- **y** (*float*) – position of the mouse inside the widget

**ontouchcancel()**

Called when a touch point has been disrupted in an implementation-specific manner (for example, too many touch points are created).

**ontouchend**(*x*, *y*)

Called when a finger is released from the widget.

**Parameters**

- **x** (*float*) – position of the finger inside the widget
- **y** (*float*) – position of the finger inside the widget

**ontouchenter**(*x*, *y*)

Called when a finger touches from outside to inside the widget.

**Parameters**

- **x** (*float*) – position of the finger inside the widget
- **y** (*float*) – position of the finger inside the widget

**ontouchleave**()

Called when a finger touches from inside to outside the widget.

**ontouchmove**(*x*, *y*)

Called continuously while a finger is dragged across the screen, over a Widget.

**Parameters**

- **x** (*float*) – position of the finger inside the widget
- **y** (*float*) – position of the finger inside the widget

**ontouchstart**(*x*, *y*)

Called when a finger touches the widget.

**Parameters**

- **x** (*float*) – position of the finger inside the widget
- **y** (*float*) – position of the finger inside the widget

**redraw**()

Forces a graphic update of the widget

**repr**(*changed\_widgets=None*)

Represents the widget as HTML format, packs all the attributes, children and so on.

**Parameters**

- **client** (*App*) – Client instance.
- **changed\_widgets** (*dict*) – A dictionary containing a collection of widgets that have to be updated. The Widget that have to be updated is the key, and the value is its textual repr.

**set\_enabled**(*enabled*)**set\_layout\_orientation**(*layout\_orientation*)

For the generic Widget, this function allows to setup the children arrangement.

**Parameters** **layout\_orientation** (*Widget.LAYOUT\_HORIZONTAL* or *Widget.LAYOUT\_VERTICAL*) –

**set\_on\_blur\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_click\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_contextmenu\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_dbclick\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_focus\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_key\_down\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_key\_up\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_mousedown\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_mouseleave\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_mousemove\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_mouseout\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_mouseup\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_touchcancel\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_touchend\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_touchenter\_listener** (*callback*, *\*userdata*)  
Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_touchleave\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_touchmove\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_on\_touchstart\_listener** (*callback*, *\*userdata*)

Registers the listener For backward compatibility Suggested new dialect event.connect(*callback*, *\*userdata*)

**set\_size** (*width*, *height*)

Set the widget size.

#### Parameters

- **width** (*int* or *str*) – An optional width for the widget (es. width=10 or width='10px' or width='10%').
- **height** (*int* or *str*) – An optional height for the widget (es. height=10 or height='10px' or height='10%').

**set\_style** (*style*)

Allows to set style properties for the widget. :param style: The style property dictionary or json string.  
:type style: str or dict

**remi.gui.decorate\_constructor\_parameter\_types** (*type\_list*)

Private decorator for use in the editor. Allows Editor to instantiate widgets.

**Parameters** **params** (*str*) – The list of types for the widget constructor method (i.e. “(int, int, str)”).

**remi.gui.decorate\_event** (*method*)

setup a method as an event

**remi.gui.decorate\_event\_js** (*js\_code*)

setup a method as an event, adding also javascript code to generate

**Parameters** **js\_code** (*str*) – javascript code to generate the event client-side. js\_code is added to the widget html as widget.attributes['onclick'] = js\_code%{'emitter\_identifier':widget.identifier, 'event\_name': 'onclick'}

**remi.gui.decorate\_explicit\_alias\_for\_listener\_registration** (*method*)

**remi.gui.decorate\_set\_on\_listener** (*prototype*)

Private decorator for use in the editor. Allows the Editor to create listener methods.

**Parameters** **params** (*str*) – The list of parameters for the listener method (es. “(self, new\_value)”).

**remi.gui.from\_pix** (*x*)

**remi.gui.jsonize** (*d*)

**remi.gui.to\_pix** (*x*)

## 1.3 remi.server module

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

**class** `remi.server.App` (*request, client\_address, server, \*\*app\_args*)

Bases: `BaseHTTPServer.BaseHTTPRequestHandler`, `object`

This class will handles any incoming request from the browser The main application class can subclass this In the `do_POST` and `do_GET` methods it is expected to receive requests such as:

- function calls with parameters
- file requests

**close** ()

Command to initiate an App to close

**do\_AUTHHEAD** ()

**do\_GET** ()

**do\_HEAD** ()

**do\_POST** ()

**do\_gui\_update** ()

This method gets called also by Timer, a new thread, and so needs to lock the update

**execute\_javascript** (*code*)

**idle** ()

Idle function called every `UPDATE_INTERVAL` before the gui update. Useful to schedule tasks.

**main** (\*\_)

Subclasses of App class *must* declare a main function that will be the entry point of the application. Inside the main function you have to declare the GUI structure and return the root widget.

**notification\_message** (*title, content, icon=""*)

This function sends “javascript” message to the client, that executes its content. In this particular code, a notification message is shown

**on\_close** ()

Called by the server when the App have to be terminated

**onerror** (*emitter, message, source, lineno, colno*)

WebPage Event that occurs on webpage errors

**onload** (*emitter*)

WebPage Event that occurs on webpage loaded

**ononline** (*emitter*)

WebPage Event that occurs on webpage goes online after a disconnection

**onpagehide** (*emitter*)

WebPage Event that occurs on webpage when the user navigates away

**onpageshow** (*emitter*)

WebPage Event that occurs on webpage gets shown

**onresize** (*emitter, width, height*)

WebPage Event that occurs on webpage gets resized

**re\_attr\_call** = `<_sre.SRE_Pattern object at 0x22f1260>`

```

    re_static_file = <_sre.SRE_Pattern object>

    set_root_widget (widget)

    websocket_handshake_done (ws_instance_to_update)

class remi.server.Server (gui_class,      title="",      start=True,      address='127.0.0.1',
                          port=8081,      username=None, password=None,      multi-
                          ple_instance=False, enable_file_cache=True, update_interval=0.1,
                          start_browser=True, websocket_timeout_timer_ms=1000, pend-
                          ing_messages_queue_length=1000, certfile=None, keyfile=None,
                          ssl_version=None, userdata=())

    Bases: object

    address

    serve_forever ()

    start ()

    stop ()

    title

class remi.server.StandaloneServer (gui_class, title="", width=800, height=600, resiz-
                                     able=True, fullscreen=False, start=True, userdata=())

    Bases: remi.server.Server

    serve_forever ()

class remi.server.ThreadedHTTPServer (server_address,      RequestHandlerClass,      auth,
                                       multiple_instance,      enable_file_cache,      up-
                                       date_interval,      websocket_timeout_timer_ms,
                                       pending_messages_queue_length,      title,
                                       server_starter_instance, certfile, keyfile, ssl_version,
                                       *userdata)

    Bases: SocketServer.ThreadingMixIn, BaseHTTPServer.HTTPServer

    daemon_threads = False

class remi.server.WebSocketsHandler (headers, *args, **kwargs)

    Bases: SocketServer.StreamRequestHandler

    static bytetonum (b)

    close ()

    handle ()

    handshake ()

    magic = '258EAF5A5-E914-47DA-95CA-C5AB0DC85B11'

    on_message (message)

    read_next_message ()

    send_message (message)

    setup ()

remi.server.encode_text (data)

remi.server.from_websocket (data)

remi.server.get_method_by_id (_id)

remi.server.get_method_by_name (root_node, name)

```

`remi.server.parse_parameters(p)`

Parses the parameters given from POST or websocket reqs expecting the parameters as:  
“11|par1='asd'|6|par2=1” returns a dict like {par1:'asd',par2:1}

`remi.server.parse_session_cookie(cookie_to_cook)`

`cookie_to_cook = http_header['cookie']`

`remi.server.start(main_gui_class, **kwargs)`

This method starts the webserver with a specific App subclass.

`remi.server.to_websocket(data)`

## 1.4 Module contents



## CHAPTER 2

---

remi

---



## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### **r**

- `remi`, [28](#)
- `remi.gui`, [3](#)
- `remi.server`, [25](#)



## A

`add_arc()` (remi.gui.SvgPath method), 13  
`add_child()` (remi.gui.Tag method), 17  
`add_class()` (remi.gui.Tag method), 18  
`add_coord()` (remi.gui.SvgPolyline method), 14  
`add_field()` (remi.gui.GenericDialog method), 7  
`add_field_with_label()` (remi.gui.GenericDialog method), 7  
`add_position()` (remi.gui.SvgPath method), 14  
`add_tab()` (remi.gui.TabBox method), 15  
`address` (remi.server.Server attribute), 27  
`App` (class in remi.server), 26  
`append()` (remi.gui.DropDown method), 4  
`append()` (remi.gui.GridBox method), 8  
`append()` (remi.gui.HBox method), 9  
`append()` (remi.gui.ListView method), 11  
`append()` (remi.gui.MenuItem method), 12  
`append()` (remi.gui.Table method), 15  
`append()` (remi.gui.TableRow method), 16  
`append()` (remi.gui.TreeItem method), 19  
`append()` (remi.gui.Widget method), 21  
`append_from_list()` (remi.gui.Table method), 15

## B

`BODY` (class in remi.gui), 3  
`Button` (class in remi.gui), 4  
`bytetonum()` (remi.server.WebSocketsHandler static method), 27

## C

`cancel_dialog()` (remi.gui.GenericDialog method), 8  
`chdir()` (remi.gui.FileFolderNavigator method), 6  
`CheckBox` (class in remi.gui), 4  
`CheckBoxLabel` (class in remi.gui), 4  
`ClassEventConnector` (class in remi.gui), 4  
`close()` (remi.server.App method), 26  
`close()` (remi.server.WebSocketsHandler method), 27  
`ColorPicker` (class in remi.gui), 4  
`column_count()` (remi.gui.TableWidget method), 17

`confirm_dialog()` (remi.gui.GenericDialog method), 8  
`confirm_value()` (remi.gui.FileSelectionDialog method), 6  
`confirm_value()` (remi.gui.InputDialog method), 10  
`connect()` (remi.gui.ClassEventConnector method), 4

## D

`daemon_threads` (remi.server.ThreadedHTTPServer attribute), 27  
`Date` (class in remi.gui), 4  
`decorate_constructor_parameter_types()` (in module remi.gui), 25  
`decorate_event()` (in module remi.gui), 25  
`decorate_event_js()` (in module remi.gui), 25  
`decorate_explicit_alias_for_listener_registration()` (in module remi.gui), 25  
`decorate_set_on_listener()` (in module remi.gui), 25  
`define_grid()` (remi.gui.GridBox method), 8  
`dir_go()` (remi.gui.FileFolderNavigator method), 6  
`dir_go_back()` (remi.gui.FileFolderNavigator method), 6  
`disable_refresh()` (remi.gui.Tag method), 18  
`do_AUTHHEAD()` (remi.server.App method), 26  
`do_GET()` (remi.server.App method), 26  
`do_gui_update()` (remi.server.App method), 26  
`do_HEAD()` (remi.server.App method), 26  
`do_POST()` (remi.server.App method), 26  
`download()` (remi.gui.FileDownloader method), 6  
`DropDown` (class in remi.gui), 4  
`DropDownItem` (class in remi.gui), 5

## E

`empty()` (remi.gui.DropDown method), 5  
`empty()` (remi.gui.ListView method), 11  
`empty()` (remi.gui.Tag method), 18  
`enable_refresh()` (remi.gui.Tag method), 18  
`encode_text()` (in module remi.server), 27  
`EVENT_ONBLUR` (remi.gui.Widget attribute), 20  
`EVENT_ONCHANGE` (remi.gui.Widget attribute), 20  
`EVENT_ONCLICK` (remi.gui.Widget attribute), 20

EVENT\_ONCONTEXTMENU (remi.gui.Widget attribute), 20  
EVENT\_ONDBLCLICK (remi.gui.Widget attribute), 20  
EVENT\_ONERROR (remi.gui.BODY attribute), 3  
EVENT\_ONFOCUS (remi.gui.Widget attribute), 20  
EVENT\_ONINPUT (remi.gui.Widget attribute), 20  
EVENT\_ONKEYDOWN (remi.gui.Widget attribute), 20  
EVENT\_ONKEYPRESS (remi.gui.Widget attribute), 21  
EVENT\_ONKEYUP (remi.gui.Widget attribute), 21  
EVENT\_ONLOAD (remi.gui.BODY attribute), 3  
EVENT\_ONMOUSEDOWN (remi.gui.Widget attribute), 21  
EVENT\_ONMOUSELEAVE (remi.gui.Widget attribute), 21  
EVENT\_ONMOUSEMOVE (remi.gui.Widget attribute), 21  
EVENT\_ONMOUSEOUT (remi.gui.Widget attribute), 21  
EVENT\_ONMOUSEOVER (remi.gui.Widget attribute), 21  
EVENT\_ONMOUSEUP (remi.gui.Widget attribute), 21  
EVENT\_ONONLINE (remi.gui.BODY attribute), 3  
EVENT\_ONPAGEHIDE (remi.gui.BODY attribute), 3  
EVENT\_ONPAGESHOW (remi.gui.BODY attribute), 3  
EVENT\_ONRESIZE (remi.gui.BODY attribute), 3  
EVENT\_ONTOUCHCANCEL (remi.gui.Widget attribute), 21  
EVENT\_ONTOUCHEND (remi.gui.Widget attribute), 21  
EVENT\_ONTOUCHENTER (remi.gui.Widget attribute), 21  
EVENT\_ONTOUCHLEAVE (remi.gui.Widget attribute), 21  
EVENT\_ONTOUCHMOVE (remi.gui.Widget attribute), 21  
EVENT\_ONTOUCHSTART (remi.gui.Widget attribute), 21  
EVENT\_ONUPDATE (remi.gui.Widget attribute), 21  
EventSource (class in remi.gui), 5  
execute\_javascript() (remi.server.App method), 26

## F

FileDownloader (class in remi.gui), 5  
FileFolderItem (class in remi.gui), 6  
FileFolderNavigator (class in remi.gui), 6  
FileSelectionDialog (class in remi.gui), 6  
FileUploader (class in remi.gui), 7  
from\_pix() (in module remi.gui), 25  
from\_websocket() (in module remi.server), 27

## G

GenericDialog (class in remi.gui), 7  
GenericObject (class in remi.gui), 8  
get\_child() (remi.gui.Tag method), 18

get\_field() (remi.gui.GenericDialog method), 8  
get\_item() (remi.gui.DropDown method), 5  
get\_item() (remi.gui.ListView method), 11  
get\_key() (remi.gui.DropDown method), 5  
get\_key() (remi.gui.ListView method), 11  
get\_method\_by\_id() (in module remi.server), 27  
get\_method\_by\_name() (in module remi.server), 27  
get\_parent() (remi.gui.Tag method), 18  
get\_selected\_filefolders() (remi.gui.FileFolderNavigator method), 6  
get\_selection\_list() (remi.gui.FileFolderNavigator method), 6  
get\_text() (remi.gui.FileFolderItem method), 6  
get\_url() (remi.gui.Link method), 11  
get\_value() (remi.gui.CheckBox method), 4  
get\_value() (remi.gui.DropDown method), 5  
get\_value() (remi.gui.DropDownItem method), 5  
get\_value() (remi.gui.Input method), 10  
get\_value() (remi.gui.ListItem method), 11  
get\_value() (remi.gui.ListView method), 11  
get\_value() (remi.gui.TextInput method), 18  
GridBox (class in remi.gui), 8

## H

handle() (remi.server.WebSocketsHandler method), 27  
handshake() (remi.server.WebSocketsHandler method), 27  
HBox (class in remi.gui), 9  
HEAD (class in remi.gui), 9  
hide() (remi.gui.GenericDialog method), 8  
HTML (class in remi.gui), 10

## I

identifier (remi.gui.Tag attribute), 18  
idle() (remi.server.App method), 26  
Image (class in remi.gui), 10  
innerHTML() (remi.gui.Tag method), 18  
Input (class in remi.gui), 10  
InputDialog (class in remi.gui), 10  
item\_at() (remi.gui.TableWidget method), 17  
item\_coords() (remi.gui.TableWidget method), 17

## J

jsonize() (in module remi.gui), 25

## L

Label (class in remi.gui), 10  
LAYOUT\_HORIZONTAL (remi.gui.Widget attribute), 21  
LAYOUT\_VERTICAL (remi.gui.Widget attribute), 21  
Link (class in remi.gui), 11  
ListItem (class in remi.gui), 11  
ListView (class in remi.gui), 11



## M

magic (remi.server.WebSocketsHandler attribute), 27  
 main() (remi.server.App method), 26  
 Menu (class in remi.gui), 12  
 MenuBar (class in remi.gui), 12  
 MenuItem (class in remi.gui), 12

## N

new\_from\_list() (remi.gui.DropDown class method), 5  
 new\_from\_list() (remi.gui.ListView class method), 11  
 new\_from\_list() (remi.gui.Table class method), 15  
 notification\_message() (remi.server.App method), 26

## O

on\_close() (remi.server.App method), 26  
 on\_folder\_item\_click() (remi.gui.FileFolderNavigator method), 6  
 on\_folder\_item\_selected() (remi.gui.FileFolderNavigator method), 6  
 on\_item\_changed() (remi.gui.TableWidget method), 17  
 on\_keydown\_listener() (remi.gui.InputDialog method), 10  
 on\_message() (remi.server.WebSocketsHandler method), 27  
 on\_row\_item\_click() (remi.gui.TableRow method), 16  
 on\_table\_row\_click() (remi.gui.Table method), 15  
 onblur() (remi.gui.Widget method), 21  
 onchange() (remi.gui.CheckBox method), 4  
 onchange() (remi.gui.CheckBoxLabel method), 4  
 onchange() (remi.gui.DropDown method), 5  
 onchange() (remi.gui.Input method), 10  
 onchange() (remi.gui.TableEditableItem method), 16  
 onchange() (remi.gui.TextInput method), 18  
 onclick() (remi.gui.FileFolderItem method), 6  
 onclick() (remi.gui.TreeItem method), 19  
 onclick() (remi.gui.Widget method), 21  
 oncontextmenu() (remi.gui.Widget method), 21  
 ondata() (remi.gui.FileUploader method), 7  
 ondblclick() (remi.gui.Widget method), 21  
 onended() (remi.gui.VideoPlayer method), 20  
 onerror() (remi.gui.BODY method), 3  
 onerror() (remi.server.App method), 26  
 onfailed() (remi.gui.FileUploader method), 7  
 onfocus() (remi.gui.Widget method), 21  
 oninput() (remi.gui.Slider method), 12  
 onkeydown() (remi.gui.TextInput method), 18  
 onkeydown() (remi.gui.Widget method), 22  
 onkeyup() (remi.gui.TextInput method), 19  
 onkeyup() (remi.gui.Widget method), 22  
 onload() (remi.gui.BODY method), 3  
 onload() (remi.server.App method), 26  
 onmousedown() (remi.gui.Widget method), 22  
 onmouseleave() (remi.gui.Widget method), 22

onmousemove() (remi.gui.Widget method), 22  
 onmouseout() (remi.gui.Widget method), 22  
 onmouseup() (remi.gui.Widget method), 22  
 ononline() (remi.gui.BODY method), 3  
 ononline() (remi.server.App method), 26  
 onpagehide() (remi.gui.BODY method), 3  
 onpagehide() (remi.server.App method), 26  
 onpageshow() (remi.gui.BODY method), 3  
 onpageshow() (remi.server.App method), 26  
 onresize() (remi.gui.BODY method), 4  
 onresize() (remi.server.App method), 26  
 onselection() (remi.gui.FileFolderItem method), 6  
 onselection() (remi.gui.ListView method), 11  
 onsuccess() (remi.gui.FileUploader method), 7  
 ontouchcancel() (remi.gui.Widget method), 22  
 ontouchend() (remi.gui.Widget method), 23  
 ontouchenter() (remi.gui.Widget method), 23  
 ontouchleave() (remi.gui.Widget method), 23  
 ontouchmove() (remi.gui.Widget method), 23  
 ontouchstart() (remi.gui.Widget method), 23

## P

parse\_params() (in module remi.server), 28  
 parse\_session\_cookie() (in module remi.server), 28  
 populate\_folder\_items() (remi.gui.FileFolderNavigator method), 6  
 Progress (class in remi.gui), 12

## R

re\_attr\_call (remi.server.App attribute), 26  
 re\_static\_file (remi.server.App attribute), 26  
 read\_next\_message() (remi.server.WebSocketsHandler method), 27  
 redraw() (remi.gui.Widget method), 23  
 remi (module), 28  
 remi.gui (module), 3  
 remi.server (module), 25  
 remove\_child() (remi.gui.GridBox method), 9  
 remove\_child() (remi.gui.Tag method), 18  
 remove\_class() (remi.gui.Tag method), 18  
 repr() (remi.gui.HEAD method), 9  
 repr() (remi.gui.HTML method), 10  
 repr() (remi.gui.Tag method), 18  
 repr() (remi.gui.Widget method), 23  
 row\_count() (remi.gui.TableWidget method), 17

## S

select\_by\_index() (remi.gui.TabBox method), 15  
 select\_by\_key() (remi.gui.DropDown method), 5  
 select\_by\_key() (remi.gui.ListView method), 12  
 select\_by\_name() (remi.gui.TabBox method), 15  
 select\_by\_value() (remi.gui.DropDown method), 5  
 select\_by\_value() (remi.gui.ListView method), 12  
 select\_by\_widget() (remi.gui.TabBox method), 15

`send_message()` (remi.server.WebSocketsHandler method), 27

`serve_forever()` (remi.server.Server method), 27

`serve_forever()` (remi.server.StandaloneServer method), 27

`Server` (class in remi.server), 27

`set_autoplay()` (remi.gui.VideoPlayer method), 20

`set_column_count()` (remi.gui.TableWidget method), 17

`set_column_gap()` (remi.gui.GridBox method), 9

`set_column_sizes()` (remi.gui.GridBox method), 9

`set_coords()` (remi.gui.SvgLine method), 13

`set_enabled()` (remi.gui.Widget method), 23

`set_fill()` (remi.gui.SvgPath method), 14

`set_fill()` (remi.gui.SvgShape method), 14

`set_identifier()` (remi.gui.Tag method), 18

`set_image()` (remi.gui.Image method), 10

`set_internal_js()` (remi.gui.HEAD method), 9

`set_layout_orientation()` (remi.gui.Widget method), 23

`set_loop()` (remi.gui.VideoPlayer method), 20

`set_max()` (remi.gui.Progress method), 12

`set_on_blur_listener()` (remi.gui.Widget method), 23

`set_on_cancel_dialog_listener()` (remi.gui.GenericDialog method), 8

`set_on_change_listener()` (remi.gui.CheckBoxLabel method), 4

`set_on_change_listener()` (remi.gui.DropDown method), 5

`set_on_change_listener()` (remi.gui.Input method), 10

`set_on_change_listener()` (remi.gui.TableEditableItem method), 16

`set_on_change_listener()` (remi.gui.TextInput method), 19

`set_on_click_listener()` (remi.gui.FileFolderItem method), 6

`set_on_click_listener()` (remi.gui.Widget method), 24

`set_on_confirm_dialog_listener()` (remi.gui.GenericDialog method), 8

`set_on_confirm_value_listener()` (remi.gui.FileSelectionDialog method), 6

`set_on_confirm_value_listener()` (remi.gui.InputDialog method), 10

`set_on_contextmenu_listener()` (remi.gui.Widget method), 24

`set_on_data_listener()` (remi.gui.FileUploader method), 7

`set_on_dbclick_listener()` (remi.gui.Widget method), 24

`set_on_ended_listener()` (remi.gui.VideoPlayer method), 20

`set_on_failed_listener()` (remi.gui.FileUploader method), 7

`set_on_focus_listener()` (remi.gui.Widget method), 24

`set_on_item_changed_listener()` (remi.gui.TableWidget method), 17

`set_on_key_down_listener()` (remi.gui.TextInput method), 19

`set_on_key_down_listener()` (remi.gui.Widget method), 24

`set_on_key_up_listener()` (remi.gui.TextInput method), 19

`set_on_key_up_listener()` (remi.gui.Widget method), 24

`set_on_mousedown_listener()` (remi.gui.Widget method), 24

`set_on_mouseleave_listener()` (remi.gui.Widget method), 24

`set_on_mousemove_listener()` (remi.gui.Widget method), 24

`set_on_mouseout_listener()` (remi.gui.Widget method), 24

`set_on_mouseup_listener()` (remi.gui.Widget method), 24

`set_on_row_item_click_listener()` (remi.gui.TableRow method), 16

`set_on_selection_listener()` (remi.gui.FileFolderItem method), 6

`set_on_selection_listener()` (remi.gui.ListView method), 12

`set_on_success_listener()` (remi.gui.FileUploader method), 7

`set_on_table_row_click_listener()` (remi.gui.Table method), 15

`set_on_touchcancel_listener()` (remi.gui.Widget method), 24

`set_on_touchend_listener()` (remi.gui.Widget method), 24

`set_on_touchenter_listener()` (remi.gui.Widget method), 24

`set_on_touchleave_listener()` (remi.gui.Widget method), 24

`set_on_touchmove_listener()` (remi.gui.Widget method), 25

`set_on_touchstart_listener()` (remi.gui.Widget method), 25

`set_oninput_listener()` (remi.gui.Slider method), 13

`set_p1()` (remi.gui.SvgLine method), 13

`set_p2()` (remi.gui.SvgLine method), 13

`set_position()` (remi.gui.SvgCircle method), 13

`set_position()` (remi.gui.SvgShape method), 14

`set_radius()` (remi.gui.SvgCircle method), 13

`set_read_only()` (remi.gui.Input method), 10

`set_root_widget()` (remi.server.App method), 27

`set_row_count()` (remi.gui.TableWidget method), 17

`set_row_gap()` (remi.gui.GridBox method), 9

`set_row_sizes()` (remi.gui.GridBox method), 9

`set_selected()` (remi.gui.FileFolderItem method), 6

`set_size()` (remi.gui.SvgRectangle method), 14

`set_size()` (remi.gui.Widget method), 25

`set_stroke()` (remi.gui.SvgLine method), 13

`set_stroke()` (remi.gui.SvgPath method), 14

`set_stroke()` (remi.gui.SvgPolyline method), 14

`set_stroke()` (remi.gui.SvgShape method), 14

`set_style()` (remi.gui.Widget method), 25

[set\\_text\(\) \(remi.gui.FileFolderItem method\), 6](#)  
[set\\_title\(\) \(remi.gui.HEAD method\), 10](#)  
[set\\_use\\_title\(\) \(remi.gui.TableWidget method\), 17](#)  
[set\\_value\(\) \(remi.gui.CheckBox method\), 4](#)  
[set\\_value\(\) \(remi.gui.DropDown method\), 5](#)  
[set\\_value\(\) \(remi.gui.DropDownItem method\), 5](#)  
[set\\_value\(\) \(remi.gui.Input method\), 10](#)  
[set\\_value\(\) \(remi.gui.ListView method\), 12](#)  
[set\\_value\(\) \(remi.gui.Progress method\), 12](#)  
[set\\_value\(\) \(remi.gui.TextInput method\), 19](#)  
[set\\_viewbox\(\) \(remi.gui.Svg method\), 13](#)  
[setup\(\) \(remi.server.WebSocketsHandler method\), 27](#)  
[setup\\_event\\_methods\(\) \(remi.gui.EventSource method\), 5](#)  
[show\(\) \(remi.gui.GenericDialog method\), 8](#)  
[Slider \(class in remi.gui\), 12](#)  
[SpinBox \(class in remi.gui\), 13](#)  
[StandaloneServer \(class in remi.server\), 27](#)  
[start\(\) \(in module remi.server\), 28](#)  
[start\(\) \(remi.server.Server method\), 27](#)  
[stop\(\) \(remi.server.Server method\), 27](#)  
[Svg \(class in remi.gui\), 13](#)  
[SvgCircle \(class in remi.gui\), 13](#)  
[SvgGroup \(class in remi.gui\), 13](#)  
[SvgLine \(class in remi.gui\), 13](#)  
[SvgPath \(class in remi.gui\), 13](#)  
[SvgPolyline \(class in remi.gui\), 14](#)  
[SvgRectangle \(class in remi.gui\), 14](#)  
[SvgShape \(class in remi.gui\), 14](#)  
[SvgText \(class in remi.gui\), 14](#)

## T

[TabBox \(class in remi.gui\), 15](#)  
[Table \(class in remi.gui\), 15](#)  
[TableEditableItem \(class in remi.gui\), 15](#)  
[TableItem \(class in remi.gui\), 16](#)  
[TableRow \(class in remi.gui\), 16](#)  
[TableTitle \(class in remi.gui\), 16](#)  
[TableWidget \(class in remi.gui\), 16](#)  
[Tag \(class in remi.gui\), 17](#)  
[TextInput \(class in remi.gui\), 18](#)  
[ThreadedHTTPServer \(class in remi.server\), 27](#)  
[title \(remi.server.Server attribute\), 27](#)  
[to\\_pix\(\) \(in module remi.gui\), 25](#)  
[to\\_websocket\(\) \(in module remi.server\), 28](#)  
[TreeItem \(class in remi.gui\), 19](#)  
[TreeView \(class in remi.gui\), 20](#)

## V

[VBox \(class in remi.gui\), 20](#)  
[VideoPlayer \(class in remi.gui\), 20](#)

## W

[websocket\\_handshake\\_done\(\) \(remi.server.App method\),](#)