OS3Grid

Fabio Rotondo - fsoft (@) sourceforge (dot) net

September 25, 2005

Chapter 1

Introduction

1.1 Introduction

OS3Grid is a JavaScript component allowing you to use spreadsheet-like grids inside web pages. It is both flexible and powerful and comes with some good APIs to work with.

OS3Grid has to be considered like an advanced *presentation layer*, you can manipulate and interact with using JavaScript.

Chapter 2

What's New

2.1 What's New

Here there is a brief list of new features and enhancements:

- **0.6**

  - New methods `get_row_attrs()` and `set_row_attr()` enable the developer to hide application specific row attributes inside the row structure. These attributes are always available and tightly binded to the row, allowing the developer to implement special features. See example **Using OS3Grid to Edit a Dataset** for an advenced demo.
  - New method `get_col_attrs()` enables the developer to inspect cell information.
  - New method `length()` returns the number of rows inside the *OS3Grid* .
  - Added some new *public functions*: `os3grid_get_grid()` returns a valid pointer to a *OS3Grid* by its unique `id` name and `os3grid_set_cell_value()` allows the developer to set a cell value by providing the `full_id` and the cell value.

    NOTE: public functions are always prefixed `os3grid_`.
  - Column renderer functions (those set using the `set_col_render()` method) now can have a new, optional, parameter `full_id`, that identify the *OS3Grid* cell uniquely. See example **Advanced Cell Value Manipulation** for a demo.
  - A new **Advenced Examples** section has been added. For real life uses of *OS3Grid* , you should have a look to these examples.
  - Documentation has been improved with this *What's New* section, allowing the "old-time" *OS3Grid* developer to know what has changed at a glance. Also, the new section *Public Functions* has been added to document *OS3Grid* public functions.
  - From this version, *OS3Grid* will start making extensive use of the `full_id`, an attribute that was always present but never used outside *OS3Grid* internal functions.

Chapter 3

Getting Started

3.1 Getting Started

In this section, we'll give you all the advice you need to getting started with *OS3Grid* .
First of all, you have to include the OS3Grid files inside your page, this is as simply as:

```
<link type="text/css" rel="stylesheet" href="os3grid.css" />
<script src="os3grid.js" type="text/javascript"></script>
```

Please, remember to set the right paths of `os3grid.css` and `os3grid.js`.

Then, for the real stuff, you have to create an HTML object as a placeholder where the grid will be rendered once created. The HTML object **MUST** have an unique id (for the `getElementById()` call by *OS3Grid* itself).
Usually, I create a `<div>` block, like the following:

```
<div id="grid" ></div>
```

Then, you have to create your grid using JavaScript. Grid attributes and settings are explained in the following chapters, but here we'll show you a simple example:

```
<script type="text/javascript">
// Create an OS3Grid instance
var g = new OS3Grid ();

// Grid Headers are the grid column names
g.set_headers ( 'nick', 'Name', 'Surname', 'E-Mail Address' );

// If contents is bigger than container, Grid will automatically show scrollbars
g.set_scrollbars ( true );

// The grid will have a solid border (these are CSS attributes)
g.set_border ( 1, "solid", "#cccccc" );

// Now, we add some rows to the grid
g.add_row ( 'fsoft', 'Fabio', 'Rotondo', 'fsoft (@) sourceforge (dot) net'
);
g.add_row ( 'john', 'John', 'Bustone', 'jbust (@) somewhere (dot) net' );
g.add_row ( 'mkey', 'Mark', 'Key', 'mkey (@) okay (dot) net' );
g.add_row ( 'jdoe', 'John', 'Doe', 'redbull (@) batman (dot) net' );

// Show the grid replacing the original HTML object with the "grid" ID.
g.render ( 'grid' );
</script>
```

Now that you have learned all the basics, it is just time to give our grid a bit of life. One of the advan-

tages of having a grid on the webpage (from a user point of view) is that you can sort the rows by clicking on the column names. Let's see how to achieve this with *OS3Grid* .
Add the following line of code in example above before the `g.render()` instruction:

```
g.set_sortable ( true );
```

Just a single line of code and you have sortable rows!
But let's get a bit further... What I really like on grids is the ability to have a single row highlighted when I move the mouse, so I can immediately have a glimpse of all row data at the same time. This is another one-liner for *OS3Grid* . Just put this right after (or before :-) the `g.set_sortable()`

```
g.set_highlight ( true );
```

Chapter 4

Public Attributes

4.1 Public Attributes

There are some public attributes inside the *OS3Grid* . These attributes may be read or set at any time inside your JS code and they'll influence the behaviour *OS3Grid* will have on future actions.
Those attributes are:

- **id (read only) - string**
  This is the DOM "id" field of the grid. You should *never* change this attribute, or the script may quit working.

- **autorender (read/write) - boolean**
  Some operations require the *OS3Grid* to be redrawn to take effect (for example, if you change a column alignment). If this flag is activated, such operations will automatically fire a render() request for you, forcing your *OS3Grid* to be redrawn istantly.
  Since grid redrawing is the most time consuming task of *OS3Grid* , this may or may not be what you want. Consider, as an example the following lines of code:
  ```
  g.autorender = true;

  g.set_highlight  ( true ); // Redraws the grid
  g.set_col_align  ( 0, "center" ); // Redraws the grid
  g.set_col_align  ( 3, "right" ); // Redraws the grid
  ```
  The grid is redrawn ¡b¿three¡/b¿ times. Now: this is a waste of resources.
  If you are planning to change quite often grid attributes consider the following solution:
  ```
  g.autorender = false;

  g.set_highlight  ( true );
  g.set_col_align  ( 0, "center" );
  g.set_col_align  ( 3, "right" )

  g.redraw (); // Redraws the grid
  ```
  OS3Grid redraw is done only once.

  > **NOTE**
  > The `autorender` flag is active after the *first actual drawn* of the grid. Before the first render, the `autorender` has **no effect**. So you can safetely set ot to `true` if you are planning to set all the grid attributes before the first render.

- **resize_cols (read/write) - boolean**
  Sometimes you would like the user to be able to resize *OS3Grid* columns. To do so, simply set this attribute to `true`. From the next `render()`, the user will be able to resize *OS3Grid* columns by simply placing the mouse pointer between two columns and dragging the pointer left or right while pressing the left mouse button.

NOTE
Remember that this feature will start to work **after** the next `render()` call.

- **start_counter (read/write) - integer**
  If you have set the `show_row_num()` to `true`, the *OS3Grid* will show row numbers as the first column of each row. By default, row numbers start from zero, but if you want, by setting the `start_counter` attribute, you can define the first value of the *OS3Grid* row numbers.

NOTE
Remember that this feature will start to work **after** the next `render()` call.

Chapter 5

Events and Callbacks

5.1 Events and Callbacks

OS3Grid is designed to smoothly integrate inside your own web page and to nicely interact with your environment. *OS3Grid* fires a lot of events you can intercept to make *OS3Grid* behave the way you like it.
Some of the *OS3Grid* behaviours can be set through some *OS3Grid* methods (for eg. `set_cell_click()`) while others are set by simply change an attribute value.
In this chapter, I'll tell you all the way you can control your *OS3Grid* once on a web page.

5.2 Public Callbacks

As the name suggests, *public callbacks* are functions that will be called when an event occurs, and they are implemented as public *OS3Grid* attributes. At the moment, the following callbacks are defined:

- **`onchange`**
  This function is called every time the user changes some contents inside the *OS3Grid* .
  So, in order to this callback to take effect, the *OS3Grid* has to be editable.
  The called function should have the following prototype:

  onchange callback

  NAME: onchange_callback (grid, x, y, new_val)

  SYNOPSIS: `(void) onchange_callback ( grid, x, y, new_val )`

  INPUT:
  - `grid`: The grid object which has fired the event.
  - `x`: X coordinate of the cell having its values changing.
  - `y`: Y coordinate of the cell having its values changing.
  - `new_val`: The new value contained inside the cell.

  Here there is a code snipped to show you how to use the `onchange` attribute.

```
function val_changed ( grid, x, y, new_val )
{
alert ( "On grid:  " + grid.id + " x:  " + x + " y:  " + y + " - value
changed to:  " + new_val );
}

g.onchange = val_changed;
```

- **`onrender`**
  This function is called every time the *OS3Grid* is rendered on the webpage. It can be used, for example, to update something else on your webpage. The called function should have the following

prototype:

onrender callback

NAME: onrender_callback (grid)

SYNOPSIS: `(void) onrender_callback ( grid )`

INPUT:

- `grid`: The grid object which has fired the event.

- **`onrowselect`**
  This function is called every time user selects or deselects a row. The called function should have the following prototype:

onrowselect callback

NAME: onrowselect_callback (grid, rownum, selected)

SYNOPSIS: `(void) onrowselect_callback ( grid, rownum, selected )`

INPUT:

- `grid`: The grid object which has fired the event.
- `rownum`: Number of the row the user has clicked on.
- `selected`: Flag true/false telling you if the row is now selected or deselected.

5.3 Setting Events

In this section, I'll list all methods you can use to set *OS3Grid* callbacks on special items.

set_cell_click

NAME: set_cell_click (func) - Sets cell click callback

SYNOPSIS: `( void ) set_cell_click ( func )`

DESCRIPTION: When the user clicks on a cell, the *OS3Grid* can be instructed to set the cell as selected or deselected (or simply do nothing). If you want to use this advanced feature, you have to provide a JavaScript function that should return `true` if the cell has to be selected and `false` if not.
Function callback prototype is the following:

cell_clicked callback

NAME: cell_clicked_callback (grid, cell, x, y, val)

SYNOPSIS: `(void) cell_clicked_callback ( grid, cell, x, y, val )`

INPUT:

- `grid`: The grid object which has fired the event.
- `cell`: The grid object which has fired the event.
- `x`: X coordinate of the cell in the *OS3Grid* .
- `y`: Y coordinate of the cell in the *OS3Grid* .
- `val`: Current cell value.

INPUT:

- `func`: Function callback

OUTPUT: This method returns nothing.

set_click_cb

NAME: set_click_cb (col, func) - Sets column click callback

SYNOPSIS: `( void ) set_click_cb ( colnum, func )`

DESCRIPTION: When the *OS3Grid* is set to be `sortable` (using `set_sortable`), all the headers columns become user-clickable and fire the `sort()` event.
You may wish to change this behaviour in two different ways:

1. by blocking a particular column from firing the `sort()` event
2. by providing a different, custom, callback to be fired in place of the `sort()` one.

To block a column from firing the `sort()` event, simply provide `-1` as function callback.
To provide a different, custom callback, pass the name of the function as the `func` element. Function callback prototype is the following:

header_clicked callback

NAME: header_clicked_callback (grid, colnum)

SYNOPSIS: `(void) header_clicked_callback ( grid, colnum )`

INPUT:

- `grid`: The grid object which has fired the event.
- `colnum`: Column clicked by the user.

INPUT:

- `colnum`: Header column number to set the function callback to.
- `func`: Function callback. Pass `-1` to block the provided column to stop firing any event.

OUTPUT: This method returns nothing.

Chapter 6

Methods

6.1 Methods

OS3Grid comes with lots of methods to interact with your environment.
This section covers them all.

add_row

NAME: add_row (col1 [, col2 [, col3 ...]]) - Adds a row of data to the grid

SYNOPSIS: `(void) add_row ( col1 [, col2 [, col3 ... ] ] )`

DESCRIPTION: This method adds a new row of data to the *OS3Grid* . Every grid row is created at once, so you have to pass all columns values for your grid, comma separated.
Each `col`n is a valid JavaScript object like `string` or `int`.

INPUT:

- `col1`: First column value.
- `[col2]`: (Optional) second column value. And so on.

OUTPUT: A new row will be added to the *OS3Grid* . This method returns nothing.

NOTES:

- Number of columns should match *OS3Grid* headers count.

SEE ALSO:

- `get_row()`
- `getv()`

get_col_attrs

NAME: get_col_attrs (colnum) - Returns a column attributes

SYNOPSIS: `(array) get_col_attrs ( colnum )`

DESCRIPTION: Use this method to retrieve all attributes set to a column.
Data will be returned as a standard JavaScript array you can read as you like.

INPUT:

- `colnum`: Column number you want the attributes from.

OUTPUT: A JavaScript standard array containing all the column values.

SEE ALSO:

- `get_row_attrs()`
- `getv()`

get_row

NAME: get_row (rownum) - Returns an array of values in a row

SYNOPSIS: `(array) get_row ( rownum )`

DESCRIPTION: Use this method to retrieve all data contained inside a row.
Data will be returned as a standard JavaScript array you can read as you like.

INPUT:

- `rownum`: Row to be returned.

OUTPUT: A JavaScript standard array containing all the row values.

SEE ALSO:

`- add_row()`

`- getv()`

get_row_attrs

NAME: get_row_attrs (rownum) - Returns a row attributes

SYNOPSIS: `(array) get_row_attrs ( rownum )`

DESCRIPTION: Use this method to retrieve all attributes set to a row.
Data will be returned as a standard JavaScript array you can read as you like.

INPUT:

- `rownum`: Row number you want the attributes from.

OUTPUT: A JavaScript standard array containing all the row values.

SEE ALSO:

`- set_row_attr()`

`- get_col_attrs()`

get_str

NAME: get_str () - Returns the grid HTML code

SYNOPSIS: `(string) get_str ()`

DESCRIPTION: This method build the HTML code needed by the `render()` method to actually produce the *OS3Grid* .
This is almost an intermediate method actually not very interesting in everyday coding, but you could use this, for example, to see how the grid is constructed for debugging purposes.

INPUT:

- `NONE`: This method gets no input.

OUTPUT: A `string` containing all the HTML code needed to build the *OS3Grid* .

getv

NAME: getv (x, y) - Returns cell contents

SYNOPSIS: `(object) getv ( x, y )`

DESCRIPTION: This method returs the contents of the cell pointed by `x` and `y`.

INPUT:
- `x`: Grid x coordinate.
- `y`: Grid y coordinate.

OUTPUT: The object held by the *OS3Grid* cell.

NOTES:
- Remember that *OS3Grid* coordinates starts at $0, 0$ and that headers and row num do not count.

SEE ALSO:
- `get_row()`

length

NAME: length () - Returns grid length

SYNOPSIS: `(int) length ()`

DESCRIPTION: This method returs the length of the Grid, that is: how many rows the Grid contains.

INPUT:
- `NONE`: No input fields required.

OUTPUT: An integer containing how many rows are inside the Grid.

render

NAME: render ([objId]) - Renders the grid on the page

SYNOPSIS: `( void ) render ( [ objId ] )`

DESCRIPTION: This is the method that actually renders the *OS3Grid* on your web page.
The `objId` is the original DOM element ID that will be replaced by the `render()` method. You **must** specify this ID the first time you call the `render()` method, but it becomes an optional parameter on all subsequent calls.

INPUT:
- `objId`: Original DOM object ID element to be replaced by the *OS3Grid* .

OUTPUT: The *OS3Grid* is rendered on the page.

SEE ALSO:
- `get_str()`
.

set_border

NAME: set_border (bsize, style, color) - Sets grid border

SYNOPSIS: `( void ) set_border ( bsize, style, color )`

DESCRIPTION: Using this method you can specify (by using the same *CSS* syntax of stylesheets) the *OS3Grid* border.

INPUT:

- `bsize`: Border size. Valid values are, for example: `2px` or `0.2em`
- `style`: Border style. Valid values are, for eg. `dashed`, `dotted` or `solid`
- `color`: Border color. Valid values are, for eq. `#ffaa12` or `red`

OUTPUT: This method returns nothing.

set_col_align

NAME: set_col_align (col, align) - Sets the text alignment for a column

SYNOPSIS: `( void ) set_col_align ( col, align )`

DESCRIPTION: This method sets the text justification alignment for a specific column. `col` is the column you are setting align to and `align` is the new alignment.

INPUT:

- `col`: Column to set the alignment.
- `align`: Alignment to set for the column. Valid values for `align` are: `left`, `right`, `center` and `justify`.

OUTPUT: This method returns nothing.

NOTES:

- Remember that `col` starts from 0.

set_col_editable

NAME: set_col_editable (col, editable) - Defines if a column is editable or not.

SYNOPSIS: `( void ) set_col_editable ( col, editable )`

DESCRIPTION: You can make cell contents editable by column selection. By setting the `editable` flag to `true`, the user will be able to click inside a cell and change its contents.
As usual `col` is the column ordinal number you want to set the attribute to.

INPUT:

- `col`: Column to set the alignment.
- `editable`: Flag true/false. If `true`, the column will be editable.

OUTPUT: This method returns nothing.

NOTES:

- Remember that `col` starts from 0.

set_col_render

NAME: set_col_render (col, render_callback) - Sets a column renderer.

SYNOPSIS: `( void ) set_col_render ( col, render_callback )`

DESCRIPTION: *OS3Grid* cell data can be manipulated before being output to the browser. To do so, you simply have to define a *column renderer*, a JavaScript funtion that will be called by the *OS3Grid* itself each time it needs to render some data inside a given cell of a specific column. Column renderers are great, for example, if you want to create links on the fly without having to hardcode them in the data added with `add_row()` method.
Column renderer prototype is the following:

column_renderer

NAME: column_renderer (txt)

SYNOPSIS: `( string ) column_renderer ( txt, full_id )`

INPUT:

- `txt`: *"Plain"* cell content to be manipulated before output.
- `full_id`: The full id identifying the cell.

OUTPUT: the string to be output to the browser.

As an example, consider the following code snippet which defines a renderer that will make the given text **bold**.

```
function bold_render ( txt, full_id )
{
return '<strong>' + txt + '<\/strong>';
}

g.set_col_render = bold_render;
```

INPUT:

- `col`: Column to set the alignment.
- `render_callback`: Render function name to be called.

OUTPUT: This method returns nothing.

NOTES:

- Remember that `col` starts from 0.

set_col_type

NAME: set_col_type (colno, type) - Sets the column type for sort and edit

SYNOPSIS: `( void ) set_col_type ( colnum, type )`

DESCRIPTION: By default, all columns inside a *OS3Grid* are considered strings. Since you may want to insert numbers or dates inside your string, this method comes in help.

INPUT:

- `colnum`: The column number to set the type to
- `type`: The column type. It must be one of the following strings:
  - **str**
    The column is a string
  - **int**
    The column is an integer number
  - **date**
    The column is a date

OUTPUT: This method returns nothing.

set_col_valid_chars

NAME: set_col_valid_chars (col, chars) - Declares valid inputable chars for the column.

SYNOPSIS: `( void ) set_col_valid_chars ( col, chars )`

DESCRIPTION: When a cell is marked as editable, you may wish to filter the user input by allowing only a subset of valid chars. For example, in a number text field, you may wish to set the valid range of chars to `0123456789` avoiding the need of validating the user input. By using this method, you can enable the chars filtering by simply provide the chars the user will be allowed to type in.

INPUT:

- `col`: Column to set the alignment.
- `chars`: A string containing all the chars that the user can input inside the column edit box.

OUTPUT: This method returns nothing.

NOTES: - Remember that `col` starts from 0.

SEE ALSO: - `set_col_editable()`

- `set_col_validator()`

set_col_validation

NAME: set_col_validation (col, func) - Sets the column input data validator.

SYNOPSIS: `( void ) set_col_validation ( col, func )`

DESCRIPTION: If the user is able to input new data in the grid, you may wish to check if inputted data is actually valid. To filter user input you can use `set_col_valid_chars()` method, but to be sure that the provided data is definitely good, you may provide a JS function to completely validate the new data for you. The `func` prototype is the following:

```
function validate ( txt )
```

and it should return `true` if `txt` is valid, and `false` otherwise. If the validation function fails, the user will be prompted to change his/her data.

INPUT:

- `col`: Column to set the alignment.
- `func`: Function callback.

OUTPUT: This method returns nothing.

NOTES:

- Remember that `col` starts from 0.

SEE ALSO:

- `set_col_editable()`
- `set_col_valid_chars()`

set_headers

NAME: set_headers (header1 [, header2 [, header3 ...]]) - Sets grid headers.

SYNOPSIS: `( void ) set_headers ( header1 [, header2 [, header3 ... ] ] )`

DESCRIPTION: Use this method to set (or change) grid headers. Here there is an example:

```
// Grid Headers are the grid column names
g.set_headers ( 'feat.no', 'Name', 'Descr',
'Importance' );
```

INPUT:

- `header1`: First column header.
- `[header2]`: Second column header...

OUTPUT: This method returns nothing.

set_highlight

NAME: set_highlight (enabled) - Sets rows highlight.

SYNOPSIS: `( void ) set_highlight ( enabled ).`

DESCRIPTION: By using this method you can enable or disable *row highlighting*, a special visual effect that will highlight the row currently under the mouse. This is extremely useful for the users so they can see all row data in a glimpse.

INPUT:

- `enabled`: Flag `true`/`false`. It `true`, rows will be highlighted when the mouse passes on them.

OUTPUT: This method returns nothing.

set_row_attr

NAME: set_row_attr (rownum, attr, val) - Sets a new attribute to the row.

SYNOPSIS: `( void ) set_row_attr ( rownum, attr, val )`

DESCRIPTION: This method sets the background color of the `row_num` color. If you have just done a `add_row` operation and wish to change the last row background, simply pass `-1` as `row_num` and *OS3Grid* will set the provided background color to the last row.

INPUT:

- `rownum`: The row number you want to set the attribute to.
- `attr`: Attribute name
- `val`: Attribute value

NOTES: - Remember that *attribute names* are **case sensitive**!

SEE ALSO: - `get_row_attrs()`

set_row_color

NAME: set_row_color (col [, row_num]) - Sets a row color.

SYNOPSIS: `( void ) set_row_color ( col [, row_num] )`

DESCRIPTION: This method sets the background color of the `row_num` color. If you have just done a `add_row` operation and wish to change the last row background, simply pass `-1` as `row_num` and *OS3Grid* will set the provided background color to the last row.

INPUT:

- `col`: `col` is a color specification following the *CSS* rules, for eg. `#fc3285` or `grey`.
- `row_num`: Row number to set the color to. If set to `-1` or simply omitted, the last row inserted will be affected by this method.

NOTES: - Remember that `row_num` starts from 0.

SEE ALSO: - `set_row_style()`

- `set_style()`

- `add_row()`

set_row_select

NAME: set_row_select (select) - Determines if the *OS3Grid* rows are selectable

SYNOPSIS: `( void ) set_row_select ( select )`

DESCRIPTION: If you want the user to be able to multiple select and deselect entire rows, set this option to `true`.

INPUT:

- `select:` Flag `true`/`false`. It `true`, *OS3Grid* rows will be selectableby the user by clicking on row number.

OUTPUT: This method returns nothing.

NOTES:

- By default, rows are **not** selectablem, and row numbers are not shown.

- If set to `true`, the method also implies `show_row_num` to be `true`.

SEE ALSO:

- `onrowselect event`

- `show_row_num()`

set_row_style

NAME: set_row_style (style, rownum) - Sets the row style.

SYNOPSIS: `( void ) set_row_style ( style, rownum )`

DESCRIPTION: You can set the row style of a row by using this method.

INPUT:

- `style:` This is the name of the style. See `set_style()` for a list of default provided styles.
- `rownum:` This is the row the style will be applied to. If set to `-1` or simply undefined, the style will be applied to the last inserted row.

OUTPUT: This method returns nothing.

SEE ALSO:

- `set_style()`

set_scrollbars

NAME: set_scrollbars (enabled) - Shows grid scrollbars

SYNOPSIS: `( void ) set_scrollbars ( enabled )`

DESCRIPTION: If you want the *OS3Grid* to have scrollbars when needed, pass `true` to this method. By default, *OS3Grid* does **not** show scrollbars and tries to get all the space it needs.

INPUT:

- `enabled:` Flag `true`/`false`. It `true`, scrollbars will be displayed if needed.

OUTPUT: This method returns nothing.

set_size

NAME: set_size (width, height) - Sets grid size

SYNOPSIS: `( void ) set_size ( width, height )`

DESCRIPTION: Use this method to set the *OS3Grid* sizes. If you want the browser to set one of the sizes for you, pass `0` as value. Both `width` and `height` follow the *CSS* syntax, so you can specify, for example `220px` or `80%`.

INPUT:

- `width`: *OS3Grid* width.
- `height`: *OS3Grid* height.

OUTPUT: This method returns nothing.

NOTES:

- You **must** provide both values at once! If you set `width` or `height` to zero or `undefined`, this method will not have any effect.

set_sort_field

NAME: set_sort_field (col) - Sets the current sort field

SYNOPSIS: `( void ) set_size ( col )`

DESCRIPTION: Sets the new field used for `sort()` operations. If the field is the same as the preceeding one, sort order is switched from ascending to descending and vice-versa.
Please note that this method just sets the new sort field but actually does not perform any sort operation. If you want an immediate grid refresh, call the `sort()` method by yourself.

INPUT:

- `col`: Column used as main key to sort the *OS3Grid* .

OUTPUT: This method returns nothing.

NOTES:

- Remember that `col` starts from 0.

SEE ALSO:

- `sort()`

set_sortable

NAME: set_sortable (sortable) - Determines if the *OS3Grid* is sortable

SYNOPSIS: `( void ) set_sortable ( sortable )`

DESCRIPTION: If you want the grid to behave like a real spreadsheed grid, you may wish the user to be able to sort rows by clicking on *OS3Grid* title columns. By setting this flag to `true` (and eventually calling the `render()` method) you can achieve this goal.

INPUT:

- `sortable`: Flag `true`/`false`. It `true`, *OS3Grid* will be sortable by clicking on its headers.

OUTPUT: This method returns nothing.

NOTES:

- By default, title columns are **not** clickable and *OS3Grid* is not sortable.

set_style

NAME: set_style (stylename) - Sets the current *OS3Grid* style.

SYNOPSIS: `( void ) set_style ( stylename )`

DESCRIPTION: Use this method to globally set *OS3Grid* style to be used.
After this method is called, every row added to the *OS3Grid* will use this style (unless rederined using `set_row_style()` method).

INPUT:

- `stylename`: One of the described style names Default provided styles are the following:
  - **normal**
    The default style. Defines a ”normal” row.
  - **error**
    This style should be used to show the user very critical rows.
  - **warn**
    This style should be used to show the user important rows.
  - **note**
    This style should be used to highlight to the user some rows.

OUTPUT: This method returns nothing.

SEE ALSO:

- `set_row_style()`

show_row_num

NAME: show_row_num (show) - Determines if row numbers should be shown or not.

SYNOPSIS: `( void ) show_row_num ( show )`

DESCRIPTION: If you want *OS3Grid* to automatically created and show row numbers, simply pass `true` to this method.

INPUT:

- `show`: Flag `true`/`false`. It `true`, *OS3Grid* row numbers will be shown.

OUTPUT: This method returns nothing.

NOTES:

- By default, row numbers are not shown.

SEE ALSO:

- `set_row_select()`

sort

NAME: sort() - Sorts the *OS3Grid*

SYNOPSIS: `( void ) sort()`

DESCRIPTION: This method sorts the *OS3Grid* accordingly to the current sort field. Sort field may be set with the `set_sort_field` method or by the user clicking on a column title.

OUTPUT: This method returns nothing.

SEE ALSO:

- `set_sort_field`

Chapter 7

Public Functions

7.1 Public Functions

OS3Grid comes with some *public functions* to help developer. A *public function* is a JavaScript function which is not related to a specific *OS3Grid* (ie. it is not a *OS3Grid* method), but it can be called by any JavaScript code snippet to interact with an existing *OS3Grid* .

These are the *public functions* the developer can count on:

os3grid_get_grid

NAME:	os3grid_get_grid (id) - Gets an OS3Grid instance by its id
SYNOPSIS:	`(`*`OS3Grid`*` ) os3grid_get_grid ( id )`
DESCRIPTION:	This function will return the desired *OS3Grid* instance by obtaining it from the hidden structures in the web page. This function is usefull when you don't have a *OS3Grid* instance saved in some public variable, but need to interact with a specific *OS3Grid* you know the id.
INPUT:	• `id:` *OS3Grid* id
OUTPUT:	A valid instance of *OS3Grid* , or `undefined` if the `id` does not exists.

os3grid_set_cell_value

NAME:	os3grid_set_cell_value (full_id, val) - Sets a cell value
SYNOPSIS:	`(`*`OS3Grid`*` ) os3grid_set_cell_value ( full_id, val )`
DESCRIPTION:	This function allows the user to modify a specific cell value by simply providing the `full_id` and the new cell value.
INPUT:	• `full_id:` Fully qualified *OS3Grid* cell id. • `val:` *OS3Grid* cell value to be set.
OUTPUT:	A valid instance of the *OS3Grid* where the cell value has been changed or `undefined` if the *OS3Grid* cound not be found.
NOTES:	- The *OS3Grid* will **not** be rendered back to reflect *OS3Grid* changes. If you need it, remember to call the `render()` method.

Contents