JAVASCRIPT DOCUMENTATION

# External Libraries

I’ve used 2 external libraries to prepare all the code:

## Jquery (actually I use the 1.10.2 version)

## Base2 (OOP library)

(Reference: <http://base2.googlecode.com/svn/doc/base2.html#/doc/!base2>)

With this functionality:

### Extend

(inheritance programming) – could extend an object from another one.

Ex.

**var** class = base2.Base.extend({

//Instantiate

constructor: **function** () { },

// a property

property: **false**,

// a method

method: **function** () {

}

)};

**var** secondClass = class.extend({

//Instantiate

constructor: **function** () { },

// a property

property2: **false**,

// a method

method: **function** () {

// call the super class parent method

**this**.base();

}

)};

### Collection

an array like object with extended functionality. “Keys” cannot be duplicated

### Module

a collections of methods that act as static methods. This **global** methods can be implemented in a base2 class.

Ex.

**var** Ivalidator = base2.Module.extend({

validate: **function** (implementer) {

//Implementer is the object class that calls this method.

}

});

**var** class = base2.Base.extend({

//Instantiate

constructor: **function** () { }

)};

class.implement(Ivalidator);

//Now class has the validate global method

### Package

a mechanisms to build up classes, constants ans functions within a closure:

Ex.

// PACKAGE

**new** **function** (\_) { // create the closure

// create the package object

**var** packageName = **new** base2.Package(**this**, {

name: "packageName",

version: "1.0",

imports: "", // imports some packages

exports: "" // exports some functions,vars from the closure to be public (functions names separated by comma ex. “func1,prop1” )

});

// evaluate the imported namespace

eval(**this**.imports);

**var** func1 = **function** () {}

**var** prop1 = **function** () {}

**var** func2 = **function** () {}

**var** prop2 = **function** () {}

// evaluate the exported namespace (this initialises the Package)

eval(**this**.exports);

};

// now:

Base2.packageName.func1() //OK

Base2.packageName.func2() //Error – not exported

# Code libraries

I try to unify all the javascript code in some few files:

1. core.js
2. jquery.base.forms-min.js – related to forms.css stylesheet
3. gameLottery.js

## core.js

about this file:

1. include this file after the "base2.js" && "jQuery" library
2. include this file before the "jquery.base.forms-min.js" library
3. have some sections:

### Prototypes

### Global variables (browser, core)

### jQuery extensions

### base2 extensions (Pgraphics, Paudio, Pcanvas)

## jquery.base.forms-min.js

This library is related to a stylesheet (forms.css). It is composed of base2 objects with inherited properties for having a standard way of behaving of all the objects. The library creates custom controls with graphic extensions and interactions for a better user experience. The controls are then extended in jQuery library like new functionalities. Here is the code example that add the objects to jquery library:

// The base2 object control

**var** control = base2.Base.extend({

constructor: **function** (options, elem) {

**this**.options = $.extend({}, **this**.options, {// custom options});

//call the parent constructor (built in the same way)

**this**.base(options, elem);

}

});

$(**function** () {

$.fn.control = **function** (/\* (object) properties will be merge into jQuery component settings \*/ options) {

**if** (**this**.length) {

**return** **this**.each(**function** () {

**var** myObj = **new** control(options, **this**);

$.data(**this**, 'control', myObj);

});

}

};

});

The original base2 object is stored in the html jquery object inside the “data” and can be retrieved in this way:

$("#div").control({// custom options });

**var** myControl = $("#div").data('control');

### Components Hierarchy:

The library is created upon an hierarchy. This is an explanation table:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Objects** | logger | jqueryBase | wrapper | icon | placeHolder | Group | linker | Overlay | balloon | Validator |
|  |  |  |  |  |  |  |  |  |  |  |
| jqueryBase | X |  |  |  |  |  |  |  |  |  |
| wrapper | X | X |  |  |  |  |  |  |  |  |
| icon | X | X | X |  |  |  |  |  |  |  |
| placeholder | X | X | X | X |  |  |  |  |  |  |
| group\* | X | X | X | X | X |  |  |  |  |  |
| linker | X | X | X | X | X | X |  |  |  |  |
| menu | X | X | X | X | X | X | X |  |  |  |
| ticker | X | X | X | X | X | X |  |  |  |  |
| scroller | X | X | X | X | X | X |  |  |  |  |
| slider | X | X | X | X | X | X |  |  |  |  |
| overlay\*\* | X | X | X | X | X | X | X |  |  |  |
| balloon | X | X | X | X | X | X | X | X |  |  |
| progressBar | X | X | X |  |  |  |  |  |  |  |
| validator\*\*\* | X | X | X | X | X | X | X | X | X |  |
| slideInput | X | X | X | X | X | X | X | X | X | X |
| date | X | X | X | X | X | X | X | X | X | X |
| time | X | X | X | X | X | X | X | X | X | X |
| textbox | X | X | X | X | X | X | X | X | X | X |
| checkbox | X | X | X | X | X | X | X | X | X | X |
| select | X | X | X | X | X | X | X | X | X | X |
| dropdown | X | X | X | X | X | X | X | X | X | X |
| autocomplete | X | X | X | X | X | X |  |  |  |  |
| colorPicker | X | X | X | X | X | X | X | X | X | X |
| fileUpload | X | X | X | X | X | X | X | X | X | X |
| countdown\*\*\*\* | X | X | X | X | X | X |  |  |  |  |
| graphs | X | X | X | X | X | X | X | X |  |  |
| slideOpen | X | X | X |  |  |  |  |  |  |  |
| table | X | X | X |  |  |  |  |  |  |  |

(\*) implement the Igroup Module

(\*\*) implement the IOverlay Module.

You can use it separately in this way thought the base2 object class COverlay.

**var** overlay = **new** COverlay({// custom options});

IOverlay.open(overlay);

(\*\*\*) implement the IValidator Module

Uses a base2 object CValid for passing errors through the code.

(\*\*\*\*) implement the IFlipBox Module

### Components Description

|  |  |
| --- | --- |
| Objects | Description |
| logger | Has the log functionality:  **this**.base(); |
| jqueryBase | Store:  **this**.el // jQuery element  **this**.oEl // original html element |
| wrapper | Wrap the element into a Div container |
| icon | Add an Icon into the wrapper |
| placeholder | Add A label into the wrapper |
| group\* | Groups object thought the groupName property  Add the active class and the functionality linked to this class. |
| linker | Add possibility to link actions to other page elements through linkElems property |
| menu | Creates a menu from a html predifined structure |
| ticker | Creates a ticker |
| scroller | Creates an element with scrolling capabilities. Implements also touching crossbrowsing gestures. This object is used in other components:   1. slider 2. slideInput |
| slider | A component for sliding html elements |
| overlay\*\* | A graphical popup |
| balloon | A graphical Tooltip – with error message container |
| progressBar | A graphical progress bar |
| validator\*\*\* | A component to add validaition ti Html form elements – uses Tooltip balloon and icon to display errors |
| slideInput | A control for input a value through a graphical interface |
| date | A control that uses the HTML5 date input or fallback to a similar bult in calendar. |
| time | A control that uses the HTML5 date input or fallback to a similar bult in time display. |
| textbox | An extended textbox element |
| checkbox | An extended checkbox element |
| select | An extended select element |
| dropdown | An extended select element with dropdown redesigned |
| autocomplete | An extended textbox element with autocomplete functionality |
| colorPicker | An extended textbox element with color picking functionality |
| fileUpload | An extended file input element – upload one file to a given url through Ajax – iFrame method |
| countdown\*\*\*\* | A graphical countdown with flipboxes – **related to forms-clock.css** |
| graphs | A component to draw graphs in canvas: Area, Line, Points, Bars, Pie |
| slideOpen | A component to display a hidden menu behind a container. |
| table | An extended table component with paginating and ordering functionality |

## gameLottery.js

### eNumber

**var** eNumber – extends the logger base2 object

the object that contains a number:

* from-to are the bound of the extraction settable for each number
* value is the number stored
* group is a string containing the a group with wich the number is linked to. For exmple if you give a same group to to number they can be validated to don't be the same number
* extract fill the number with a random number between the from-to interval
* toString add a 0 before 1 digit numbers
* setValue set the value of the number and throw exception if it's not numeric, outofbound or duplicate (confronting it with an array "not" of eNumbers).

### Game

**var** Game – extends the logger base2 object

Manage the:

1. Form submit
2. Popup error messages
3. Refreshing the price

### Extraction

**var** Extraction – extends the Game base2 object

Extraction of the German lotto (Wednesday, Saturday)

Manage the:

1. Number of draws
2. Number of weeks

### Discount

**var** Discount – extends the Extraction base2 object

Manage the:

1. Discount product info

### Lotto

**var** Lotto – extends the Discount base2 object

Manage the:

1. Extraction numbers
2. Legends/stars numbers
3. All the form with numbers, buttons, showlines, delete lines, etc …

### Syndicate

**var** Syndicate – extends the Discount base2 object

Manage the:

1. Quite nothing

### Promo Syndicate

**var** Syndicate – extends the Discount base2 object

Manage the:

1. Quite nothing