ExoEngine

Table of contents

ExoEngine - WebEngine (Alpha 0.0.3 Doc)

PRESENTATION

The ExoEngine (EE) is a programming developed to facilitate web development through different modules. It usable from simple website to more complex WebApp or even video games.

This engine has two main sections:

- **ExoEngine (EE):** A system developed to make web site creation easier, the system works with independents modules. It work has a code extension.
- **ExoGameEngine (EGE):** System include in the EE and use to make **video games**, but it also works for **WebApp / WebSoftware**. Contrary to EE, the EGE is an all-in-one system, all the modules work together.

INSTALATION (EE)

ExoEngine is using the library **JQuery**, so before using the EE make sure you call it (See details below).

Once **JQuery** is set you can install EE on your page, EE is an external library system, so all you have to do is calling the main EE file: http://www.exodiastudio.com/ExoEngine/Alpha0.0.2/ExoEngine.js.

Example (you can copy this code to install):

```
1. <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
2. <script
   src="http://www.exodiastudio.com/ExoEngine/Alpha0.0.3/ExoEngine.js"></scrip
   t>
```

you now have access the EE class.

HOW TO USE (EE)

As said previously, to avoid a heavy loading the EE use a module system. Each modules are call separately based on your website needs.

To use an EE functionality / module follow those step:

1st: What you need to now

- EE is just a massive class where all the functionalities are stored. To access it, you just need to call the global variable EE or EE0_4 (See Multi-Version for more info).

2nd: Call a module

- To call a module simply use the function EE.Call(), with two arguments: the module name from the enumerator modulesList (EE.modulesList.MODULE_NAME) and a callback function (called when loaded -- the loading is asynchronous). For more details check the ExoEngine Class file.

3rd: Call the module functionality

- Now your module is ready, within the callback function or latter in an event (if you use the module just after the call it might not be loaded) use EE.Module.Functionality or EE.Group.Module.Functionality (some module are store in groups just enter the group name first). Groups and Modules names are the same as shown is this documentation.

```
1. EE.Call(EE.modulesList.SYSTEM_SCROLLING, function(){
2.     Scrolling = EE.System.Scrolling.AddScrolling();
3. }
```

MULTI-VERSION INSTALATION (EE)

At any time you can call another EE version (e.g. if you prefer how was working in a previous version), to implement it just add a second line:

```
1. <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
1. <script
   src="http://www.exodiastudio.com/ExoEngine/Alpha0.0.2/ExoEngine.js"></scrip
   t>
1. <script
```

```
src="http://www.exodiastudio.com/ExoEngine/Alpha0.0.3/ExoEngine.js"></scrip
t>
```

Important: When a `ExoEngine.js` file is called it create two global variable EE and EE0_3 (Change with versions called -- This is an example). When you use different version at the same time the EE variable will always be the last version entered. Functions called with EE will be automatically upgraded if you change the version.

We recommend to use EE0_3 format instead of EE. If you add a new version your code will not crash.

INSTALATION & HOW TO USE (EGE)

This section is detailed in another page: click here to reach it.

ExoEngine Class

Description

This page detail everything about the EE class.

We recommend to first check our installation and first use guide (Introduction) before using our systems.

Class Details

Variables (Members):

Main (Important information):

Name	Type	Default Value	Description
ENGINE_VERSION	Constant STRING	A0.0.3	Constant Variable defining the version the Engine used
EGE	EGE Class	null	The member that hold the EGE Class when the EGE is called.

Status (Modules loading status):

Name	Type	Default Value	Description
animationFunctionalStatus	Private Boolean	false	Status of the module: "**Animation: Functional**" (Check for loaded or not).
animationFlipStatus	Private Boolean	false	Status of the module: "**Animation: Flip**" (Check for loaded or not).
animationAnimatedGradientStatus	Private Boolean	false	Status of the module: "**Animation: Animated Gradient**" (Check for loaded or not).
arrayGrid2DStatus	Private Boolean	false	Status of the module: "**Array: 2D Grid**" (Check for loaded or not).
arrayMap2DStatus	Private Boolean	false	Status of the module: "**Array: 2D Map**" (Check for loaded or not).
GUI_ImageViewerStatus	Private Boolean	false	Status of the module: "**Images: Viewer**" from **GUI Group** (Check for loaded or not).
GUI_ImageImageListStatus	Private Boolean	false	Status of the module: "**Images: Image List**" from **GUI Group** (Check for loaded or not).
GUI_Tab_InteractiveTabStatus	Private Boolean	false	Status of the module: "**Tab: Interactive Tab**" from **GUI Group** (Check for loaded or not).
GUI_Tab_LegendStatus	Private Boolean	false	Status of the module: "**Tab: Legend**" from **GUI Group** (Check for loaded or not).
GUI_Tab_MultiTabStatus	Private Boolean	false	Status of the module: "**Tab: MultiTab**" from **GUI Group** (Check for loaded or not).
GUI_Tab_OverlappingTabStatus	Private Boolean	false	Status of the module: "**Tab: Overlapping Tab**" from **GUI Group** (Check for loaded or not).
GUI_TimeVerticalTimelineStatus	Private Boolean	false	Status of the module: "**Time: Vertical Timeline**" from **GUI Group** (Check for loaded or not).
GUI_Status_StateBarStatus	Private Boolean	false	Status of the module: "**Status: State Bar**" from **GUI Group** (Check for loaded or not).
GUI_Status_LoadingBarStatus	Private Boolean	false	Status of the module: "**Status: Loading Bar**" from **GUI Group** (Check for loaded or not).
GUI_ToolBox_InteractiveToolBoxStatus	Private Boolean	false	Status of the module: "**Tool Box: Interactive Tool Box**" from **GUI Group** (Check for loaded or not).
GUI_PopUp_SelectStatus	Private Boolean	false	Status of the module: "**Pop up: Select**" from **GUI Group** (Check for loaded or not).
DESIGN_Background_ImageBackgroundStatus	Private Boolean	false	Status of the module: "**Background: Image Background**" from **Design Group** (Check for loaded or not).
DESIGN_Hover_HoverGradientStatus	Private Boolean	false	Status of the module: "**Hover: Hover Gradient**" from **Design Group** (Check for loaded or not).
DESIGN_Hover_HoverBlurStatus	Private Boolean	false	Status of the module: "**Hover: Hover Blur**" from **Design Group** (Check for loaded or not).
mathTrigoStatus	Private Boolean	false	Status of the module: "**Math: Trigonometry**" (Check for loaded or not).
mathRandomStatus	Private Boolean	false	Status of the module: "**Math: Random**" (Check for loaded or not).

SYSTEM_LangStatus	Private Boolean	false	Status of the module: "**Language: Lang**" from **System Group** (Check for loaded or not).
SYSTEM_MouseStatus	Private Boolean	false	Status of the module: "**Mouse: Mouse**" from **System Group** (Check for loaded or not).
SYSTEM_ScrollStatus	Private Boolean	false	Status of the module: "**Scrolling: Scrolling**" from **System Group** (Check for loaded or not).
SYSTEM_FILE_DeviceFileStatus	Private Boolean	false	Status of the module: "**File: Device File**" from **System Group** (Check for loaded or not).
SYSTEM_CookiesStatus	Private Boolean	false	Status of the module: "**Cookies**" from **System Group** (Check for loaded or not).
SYSTEM_KeyboardStatus	Private Boolean	false	Status of the module: "**Keyboard: Keyboard**" from **System Group** (Check for loaded or not).
UTIL_LoadCheckStatus	Private Boolean	false	Status of the module: "**Load Check**" from **Util group** (Check for loaded or not).
UTIL_RedirectStatus	Private Boolean	false	Status of the module: "**Redirect**" from **Util group** (Check for loaded or not).
UTIL_ConverterStatus	Private Boolean	false	Status of the module: "**Converter**" from **Util group** (Check for loaded or not).
GameStatus	Private Boolean	false	Status of the EGE (Check for loaded or not).

Enumerations:

Name	List	Description
modulesList	• ANIMATIONS • ANIMATIONS_FUNCTIONAL • ANIMATIONS_FLIP • ANIMATIONS_ANIMATED_GRADIENT • • • ARRAY • ARRAY_GRID2D • ARRAY_MAP2D • • • GUI_IMAGES • GUI_IMAGES_VIEWER • GUI_IMAGES_IMAGELIST • • GUI_TAB • GUI_TAB_INTERACTIVETAB • GUI_TAB_LEGEND • GUI_TAB_MULTITAB	This list is used by the **call()** function to define which module has to be called

	• GUI_TAB_OVERLAPPINGTAB • • GUI_TIME • • GUI_STATUS • GUI_STATUS_STATEBAR • GUI_STATUS_LOADINGBAR • • GUI_TOOLBOX • GUI_TOOLBOX_INTERACTIVETOOLBOX • • GUI_POPUP • GUI_POPUP_SELECT • • • DESIGN_BACKGROUND • DESIGN_BACKGROUND_IMAGEBACKGROUND • • DESIGN_HOVER • DESIGN_HOVER_HOVERGRADIENT • DESIGN_HOVER_HOVERBLUR • • • MATHS • MATHS_RANDOM • MATHS_TRIGONOMETRY • • • SYSTEM_LANG • SYSTEM_MOUSE • SYSTEM_SCROLLING • SYSTEM_FILE_DEVICEFILE • SYSTEM_COOKIES • SYSTEM_KEYBOARD • • • UTIL_CONVERTER • UTIL_REDIRECT	

	• UTIL_LOADCHECK	
EGEModulesList	• LOADING_LOADING • • • GUI_OVERLAY • GUI_LINKER • GUI_WIDGET • GUI_WIDGET_BUTTON • GUI_WIDGET_TEXTINPUT • GUI_WIDGET_TEXT • GUI_WIDGET_SELECT • GUI_WIDGET_LAYOUT • • GUI_POPUP • • • WORLD_LAYEREDWORLD	Since EGE modules are called by EE they have to be in the main class. Those are used by **LaunchGE()** Function

Inner Classes:

All the inner classes are the modules or the modules groups.

Name	Status	Description
Animation	Module	Module used for all animation purpose.
Array	Module	Module within different array class.
GUI	Group of modules	This group contain all graphical modules
Design	Group of modules	This group contain website design enhancer
Math	Module	Module used for different math calculation
System	Group of modules	This group contain all classes used for back-ends operations
Util	Group of modules	This group contain all other modules that doesn't fit with other groups (mostly help modules). NOTE: From this version most of the Util content are not anymore modules and are directly loaded.

Functions:

Name	Arguments	Return	Description
Call	• **modulesList: Module** - the modulesList item	**VOID**	This function is called when you need to setup a module. It use a callback function to continue the process after the modules has

	you want to call (open) • **Function: Callback**: the call function which is call went the files are loaded		been loaded (because the loading in asynchronous).
ModuleCheck	• **modulesList: Module** - the modulesList item you want to test	**Boolean**	This function is used to test if the selected modules has already been called, it return False if the module has already been called. (This system is used by **Call()** to avoid loading modules two times).
LaunchGE	• **String: LocationID** - where in DOM the view-port and the EGE Structure has to be created. • **EGEModulesList []: ModulesList** - List of all the EGE modules you want to load. • **Function: Callback** - the call function which is call went the files are loaded.	**VOID**	This function is used to launch the EGE system. (See more details in the EGE Installation tutorial) Note: Only one EGE can be load at the time.

Groups

Groups

Some of the modules are separated in bigger groups (Usually because there is all lot of data in a modules. So we need to separate it in smaller parts otherwise the loading would be to expensive).

For now there is 3 groups:

- **GUI**: Group of graphical classes
- **DESIGN**: Group of graphical enhancer classes
- **SYSTEM**: Group of classes use in background process
- **UTIL**: Other king of useful functions

Note: in the future it will be possible to call a full group.

GUI

This group contain graphical modules used for simplify the creation of dynamics and nice looking website:

- **Image**: Classes used to handle properly images/pictures
- **Status**: Used to create to status info like progress / loading bar.
- **Tab**: This group offer different modules to generate interactive tab and boards
- **Time**: Timeline and other classes for time and timing.
- **ToolBox**: This module offer a selection of toolbox system.
- **PopUp**: This module offer a selection of pop up classes

Images

The images module is used to handle properly images.

Module Details

Variables (Members):

Name	Type	Default Value	Description
ImageClasses	• Viewer • ImageList	• null • null	Array holding the main class of this modules

Functions:

Name	Arguments	Return	Description
CreateViewer	• **None**	Viewer	This function is called to create a new **Viewer Class** (the function does not initiate it).
CreateImageList	• **None**	ImageList	This function is called to create a new **ImageList Class** (the function does not initiate it).

The images viewer will generate all HTML and Script code to add an interactive image viewer to your website.

How to Initialize the viewer ?

The initialization of the viewer is an one step process, just call the function **Init** of the class with his three arguments (**see above for more details**).*

Class Details

Functions:

Name	Arguments	Return	Description
Init	• **String: NewID** - HMTL ID defining the position in the DOM of the viewer • **String: Name** - Define the name of the viewer all viewer component will be defined by this name (this is to avoid mixing if you use different viewer so be sure to use different name for each viewer). • **JSON : data** - JSON data which define how the viewer will be build.	**VOID**	This function Initialized every part of the viewer: HMTL, CSS, Data, Events, etc... It will be generated in the defined ID as NewID and all HTML IDs used for this viewer will start by the name defined in arguments Also to define the data and the design you must you the data arguments, that work thanks to a JSON files **(for more details about the JSON check the section below).**

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
basedId	String	null	HTML Id of the viewer parent.
data	JSON	null	Data used to initialize the viewer.
id	String	null	Global HMTL Id of the viewer (# + name).
listPosition	Small Class	• **Integer: pos** - define the position of the images list • **Integer: on** - define number of movement of the image list	Hold the details need for the image list movements.

name	String	null	Name of the viewer used by the class for all HTML IDs
selectedImgPos	Integer	**0**	ID of the selected (active image).
topButtons	• String • String • String	• Code/GUI/Image/Data/Normal_TopButton.png • Code/GUI/Image/Data/Normal_TopButton.png • Code/GUI/Image/Data/Normal_TopButton.png	This member keep the default direction of the top buttons of the viewer.
topButtons	• String • String • String	• Code/GUI/Image/Data/Normal_BottomButton.png • Code/GUI/Image/Data/Hovered_BottomButton.png • Code/GUI/Image/Data/Pressed_BottomButton.png"	This member keep the default direction of the top buttons of the viewer.

Functions:

Name	Arguments	Return	Description
GetImageId	• **Integer: id** - Number of the image	**String** - HTML ID of the button	Function used to get the ID of the images (used to short the code)
GetImgNumFromId	• **String: ID** - HTML ID of the image you want to get the number from	**Integer** - ID of the button	Convert an HTML ID into an Integer ID of a button
ImagesMouseEnter (Event Function)	• **JS Object: obj** - Reference to the calling obj	**VOID**	Event function called when mouse enter on one of the image in the list. It will change his design (css).
ImagesMouseLeave (Event Function)	• **JS Object: obj** - Reference to the	**VOID**	Event function called when mouse leave on one of the image in the list. It will reset his design (css).

	calling obj		
ImagesMouseUp (Event Function)	• **JS Object: obj** - Reference to the calling obj	**VOID**	Event function called when the mouse is released on one of the image in the list. It will remove to the old active image the borders, and set active the selected image.
InitButtonEvents	• **String: id -** HTML ID of the button that we want to Initiate the events • **Boolean: IsUp** - Define if the button belong to the top buttons or not • **Boolean: IsLeft** - Define if the button belong to the left buttons or not	**VOID**	Initiate buttons event and define the callback function
isOK	• **JS Object: obj** - Reference to the obj	**Boolean**	Check if the selected OBJ is valid (Defined // Non-Null)
MoveImageList (Event Funciton)	• **Boolean: toRight** - Define if the movement is to the right	**VOID**	Function that handle the moving and all calculation for the animation.
MovingList	• **Integer: Percent** - Advancement of the animation • **Integer: X** - X based on the advancement • **Class: data** - Data that the function need	**VOID**	Function used by the **animation process**, to move the list. (EaseInOut Animation)
SwitchImg (Event Function)	• **Boolean: isNext** - Define if the movement is to the next image or not	**VOID**	Function called when top buttons are pressed. It will change the selected image as a click
TopImageClick (Event Function)	• **Boolean: zoomIn** - Define if it's for a Zoom in or a Zoom Out	**VOID**	Function called when the user pressed the main image. It will zoom in.

Compatibility

LANGUAGE CLASS: Partially.

Data Structure

Main Structure Details

"ENTER_NAME": { //Class that hold the whole data

```
	"Global": {		//Section Where we define the design
		"CSSGroup": {	//CSS of the viewer
			"CSS": {		//CSS

			},
			"Children": [		//To add extra CSS for some children (see below how it work)

			]
		},
		"Buttons": {		//Modify buttons
			"TopButtons":  { //Modify top buttons
				"CSS": {		//CSS Details

				},
				"Normal": null,	//Link to the button design for customization - Normal Version (null will take the ones by default)
				"Hovered": null,	//Link to the button design for customization - Hover Version (null will take the ones by default)
				"Pressed": null	//Link to the button design for customization - Pressed Version (null will take the ones by default)
			},
			"BottomButtons": {		//Modify Bottom buttons
				"CSS": {		//CSS Details

				},
				"Normal": null,	//Link to the button design for customization - Normal Version (null will take the ones by default)
				"Hovered": null,	//Link to the button design for customization - Hover Version (null will take the ones by default)
				"Pressed": null	//Link to the button design for customization - Pressed Version (null will take the ones by default)
			}
		}
	},
	"Images": [		//List of images you want to add into the viewer

	]
}
```

Note:

- **CSS Children**: Children is a list of class composed by two data (Name are important):
  - **String: ID** - HTML ID of the target (it start by adding the ID of the target object then the children)
  - **Class: CSS** - CSS details to add

Download Sample:

SAMPLE: Here

ImageList

The images list will generate all HTML and Script code to add an interactive list of image.

You can add event on image selection, and add an "**Add Image**" button to more interaction

How to Initialize the Image List ?

The initialization of the Image List is an one step process, just call the function **Init** of the class with its three arguments (**see bellow for more details**).

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: ID** - HMTL ID defining the position in the DOM of the viewer • **String: Name** - Define the name of the ImageList all ImageList component will be defined by this name (this is to avoid mixing if you use different ImageList, so be sure to use different name for each viewer). • **Function: Callback** - Callback function.	**VOID**	This function Initialized every part of the viewer: HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this viewer will start by the name defined in arguments This system use the **Util: LoadCheck** module and to be Called, Use callback arguments to compensate the asynchronous loading.
Open	• **None**	**VOID**	Use this function to open the list.
AddImages	• **String or String[]: Images** - URL of image(s) to add. • **Function: AICallback** - Function called when all images are loaded	**VOID**	This function add images to the Image List and wait for them to load, use **AICallback** to add some action only after images are all loaded.

Setters Functions

Name	Arguments	Return	Description
SetBorder	• **String: Type** - CSS border type. • **Integer: Size** - Border size (only in px). • **String: Color** - String that define the color.	**VOID**	This function will add a border to the image list window.

SetBackgroundColor	• **String: Color** - String that define the color.	**VOID**	Use this function to change the background color.
AddClickCallback	• **Function: Func** - Function called. • **Class: Args** - Arguments to pass with function	**VOID**	This function will add a **callback function** for when the user click on an image.
AddNewCallback	• **Function: Func** - Function called. • **Class: Args** - Arguments to pass with function	**VOID**	This function will add a **callback function** for when the user click on the new image button. If this function is not used it will call the HTML Open file
SetCloseButton	• **Function: SCBCallback** - Function called when button Images are loaded.	**VOID**	Call this function to add the close button
SetNewButton	• **Function: SNBCallback** - Function called when some required modules are Called.	**VOID**	Call this function to add the new button

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
basedId	String	undefined	HTML Id of the viewer parent.
name	String	undefined	Name of this Image list.
id	String	undefined	Global HMTL Id of the viewer (# + name).
newDirFunc	Function	null	It hold the callback function when user press the new button
newDirArgs	Class	null	It hold the arguments used by newDirFunc()
selectedFunc	Function	null	It hold the callback function when user press an image.
selectedArgs	Class	null	It hold the arguments used by selectedFunc()

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
UpdateImages	• **None**	**VOID**	Reset all images and place them properly.

Util Functions

Name	Arguments	Return	Description
GetClose	• **None**	**JQuery DOM Object**	Get close button JQuery DOM Object.
GetNew	• **None**	**JQuery DOM Object**	Get new button JQuery DOM Object.
GetList	• **None**	**JQuery DOM Object**	Get list JQuery DOM Object.
GetImageDiv	• **String: Name** - Name of the image you want to select.	**JQuery DOM Object**	Get get main Div of the selected image.
GetImageImg	• **String: Name** - Name of the image you want to select.	**JQuery DOM Object**	Get get main img of the selected image.

Compatibility

LANGUAGE CLASS: Not compatible.

Tab

This group offer different modules to generate interactive tab and boards

Module Details

Variables (Members):

Name	Type	Default Value	Description
TabClasses	• **InteractiveTab** • **InteractiveTab**	• null • null • null • null	Array holding the main class of this module

	• **Initializer** • **Legend** • **MultiTab** • **OverlappingTab**	• null	

Functions:

Name	Arguments	Return	Description
GetInteractiveTabInitializer	• **None**	**InteractiveTabInitializer**	This function is called to get access to a **InteractiveTabInitializer** (But the initialize is not mandatory - see **InteractiveTabInitializer** to get a sample).
CreateInteractiveTab	• **None**	**InteractiveTab**	This function will create a new **InteractiveTab.** Not Initialized.
CreateLegend	• **None**	**Legend**	This function will create a new **Legend.** Not Initialized.
CreateMultiTab	• **None**	**MultiTab**	This function will create a new **MultiTab.** Not Initialized.
CreateOverlappingTab	• **String: To** - Where the Overlapping Tab has be created • **String: Name** - The name of **THIS** Overlapping Tab.	**OverlappingTab**	This function will create a new **OverlappingTab.**

InteractiveTab

InteractiveTab

The class will generate automatically an interactive tab interactive and entirely custom.

How to Initialize the tab ?

The initialization of the tab is a two step process:

- First call the **Init** function: It will initiate the tab HTML and some basic CSS
- Then call the **InitData** function: It will initiate all data in tab including CSS.

Class Details

Functions:

Name	Arguments	Return	Description
Init	• **TabInitializer: TabInitializer** - Needed data to Initiate the array • **String: TabName** - Define the name of the tab all tab component will be defined by this name (this is to avoid mixing if you use different tab so be sure to use different name for each tab). • **String: NewID** - HMTL ID defining the position in the DOM of the tab	**VOID**	This function will generate the whole HTML code of the tab, it will also apply some basic CSS. The function need a tab **TabInitializer** to setup the tab (**See TabInitializer for more details**)
InitData	• **JSON : data** - JSON data which define how the viewer will be build.	**VOID**	This function use a JSON format to Initiate all data and the CSS that you want to apply.
addLegend	**NONE**	**Legend**	Create a new Legend assign to the tab.

Structures:

Name	Data	Default Values	Description
Tab	• **Integer: nbColomn** - Define the number of column • **Integer: nbRow** - Define the number of row • **Integer: boxHeight** - Define the height of each boxes • **Integer: width** - Define the width of the tab in % • **Integer: strokeSize** - Define the size of each border.	• 0 • 0 • 0 • 100 • 0	Tab structure is used to initialize to tab.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
id	**String**	null	HTML ID where the tab is located
name	**String**	null	Name of table use to define all other IDs
tabDetails	**TabInitializer**	null	Hold the **TabInitializer** used to Initiate the tab.
timeoutHolder	**Timeout**	null	This member is used for hold the SetInterval event to be able to cancel it if the mouse leave the selected case.

Functions:

Name	Arguments	Return	Description
CheckAndReplaceKey (Util Function)	• **String**:**Str** - String to check • **Class**: **Data** - Access to a tab data JSON type	**String**	The tab class offer a way to define constant word (mostly for using with language). This function check the all text for a **%%WORD%%** to replace it by getting the reference trough the data (The **Mess**).
difinedBackPosition	• **Integer**: **line** - Line of the back you want to mouse • **Integer**: **row** - Row of the back you want to mouse • **Integer**: **spaceBtwMouseAndDiv** - Define the space you want between the mouse and the back	**VOID**	Move the selected back on the mouse position with a small gap(This is to avoid by moving the mouse to fast to move over the back and making it disappear).
EventInit	• **Integer**: **Line** - Y potion of the cell that you want to initialize the events • **Integer**: **Row** - X potion of the cell that you want to initialize the events • **Class**: **ref** - Reference the calling class (because it has to be called outside for technical reason). • **Boolean**: **enter** - Define if it's mouse entering or mouse mouse leaving.	**VOID**	Initialize the events for the cells of the tab: • Hover: that change the design of the cell the other on the same row and line • Hold: when you hold the mouse it make information appear.
GetBackId (Util Function)	• **Integer**: **line** - Line of the back you want to	**String**	Quick way to get the HTML ID of a back section.

	get the ID • **Integer: row** - Row of the back you want to get the ID		
GetBoxId (Util Function)	• **Integer: line** - Line of the box you want to get the ID • **Integer: row** - Row of the box you want to get the ID	**String**	Quick way to get the HTML ID of a box.
GetLineId (Util Function)	• **Integer: id** - Numerical ID of the line you want to get the ID	**String**	Quick way to get the HTML ID of a Line.
isOK	• **JS Object: obj** - Reference to the obj	**Boolean**	Check if the selected OBJ is valid (Defined // Non-Null)

Compatibility

LANGUAGE CLASS: Partially.

Data Structure

Main Structure Details

Note:
black -> Needed
Blue -> Optional

```
{
	"data": { //Main data class
		"HorizontalTitleLine": [], //String list for the title on the horizontal part

		"VerticalTitleLine": [], //String list for the title on the vertical part

		"Section": { //Define the sections (Section are groups for titles).
			"HorizontalSection": [ //Define the horizontal sections (Section are groups for titles).
				{
					"Value": "", //String for the value of the section
					"Size": 1 //How many title (lines) the section take (note that the next section size (count) will start after the one)
				}
			],

			"VerticalSection": [ //Define the vertical sections (Section are groups for titles).
				{
					"Value": "", //String for the value of the section
					"Size": 1 //How many title (rows) the section take (note that the next section size (count) will start after the one)
				}
```

```
                ],

                "Angle": "" //String the define the data in the angle.
        },

        "Mess": [  //The mess is the constant section, here you can define common data that can change just here and not everywhere.
                {
                        "Key": "" //The key the program will look for when a replacement is needed.
                        "Value": "" //Value to replace.
                }
        ],

        "Line": [ //Groups of lines you need to add the below class for each line of your array
                { //Line class brackets
                        "Box": [ //Groups of boxes on the line you need to add the below class for each line of your array
                                { //Box class brackets
                                        "Constant": { //Constant part of the boxes (this is just to help when writing different language files)
                                                "FrontCSS": {  //CSS Group for the box
                                                        "CSS":{  //CSS details

                                                        },
                                                        "Children":[ //To add extra CSS for some children (see below how it work)

                                                        ]
                                                },
                                                "BackCSS": { //CSS Group for the back part
                                                        "CSS":{ //CSS details

                                                        },
                                                        "Children":[ //To add extra CSS for some children (see below how it work)

                                                        ]
                                                }
                                        },
                                        "Value": { //Value group for the boxes
                                                "Front": [ //Front value: each String is a line.
                                                ],
                                                "Back": [ //Back value: each String is a line.
                                                ]
                                        }
                                }
                        ]
                }
        ],

        "BaseCSS": { //Group to define the basic CSS (This CSS is override by the local CSS)
                "Front": { //Group to define the basic CSS on the front part
                        "CSS":{
                        },
                        "Children":[
                        ]
                },
                "Back": { //Group to define the basic CSS on the back part front part
                        "CSS":{
                        },
                        "Children":[
                        ]
```

```
                },
                "Title":{ //Group to define the CSS of the titles
                        "Horizontal":{ //Group to define the CSS of the horizontal title
                                "CSS":{
                                },
                                "Children":[
                                ]
                        },
                        "Vertical":{ //Group to define the CSS of the vertical title
                                "CSS":{
                                },
                                "Children":[
                                ]
                        },
                        "Angle":{ //Group to define the CSS of the angle title
                                "CSS":{

                                },
                                "Children":[
                                ]
                        }
                },
                "Section":{
                        "Horizontal":{ //Group to define the CSS of the horizontal sections
                                "CSS":{
                                },
                                "Children":[
                                ]
                        },
                        "Vertical":{ //Group to define the CSS of the Vertical sections
                                "CSS":{
                                },
                                "Children":[
                                ]
                        },
                        "Void":{ //Group to define the CSS of the void sections (angle between the
two sections)
                                ":{
                                "CSS":{
                                },
                                "Children":[
                                ]
                        }
                }
        }
},
```

Note:

- **CSS Children**: Children is a list of class composed by two data (Name are important):
  - **String: ID** - HTML ID of the target (it start by adding the ID of the target object then the children)
  - **Class: CSS** - CSS details to add
- **Keys**: in the values you can use Key. To use one just add the key word between %% (e.g. %%Key%%) then in the **mess** you can define whet the key mean. It's used to avoid changing things that are the same between boxes.

Legend

Legend

The legend is an extension that can be added to an interactive tab

Note: that the legend will work with some other systems

How to Initialize the legend ?

The initialization of the legend is a two step process:

- First call the **Init** function: It will initiate the the HTML part and the CSS
- Then use **Add** function to add the data.

Class Details

Functions:

Name	Arguments	Return	Description
Init	• **String: NewID** - HMTL ID defining the position in the DOM of the tab • **String: Name** - Define the name of the legend all legend component will be defined by this name (this is to avoid mixing if you use different legend so be sure to use different name for each tab). • **CSS Class: LegendCSS** - CSS off the legend (some CSS are by default)	**VOID**	This function will generate the whole HTML code of the legend, it will also apply the CSS that you decide
add	• **String: Symbol** - This is the symbol (or text) that needed to be showed in the box. • **String: Definition** - Description of the symbol. • **CSS Class: CSS** - CSS of the box • **CSS Class: InnerCSS** - CSS that apply to everything inside the box	**VOID**	This function add a new definition to the legend.

Compatibility

LANGUAGE CLASS: Partially.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
id	**String**	null	HTML ID where the legend is located
idCount	**Integer**	0	This hold the number of definition to make different ID for each definition.
name	**String**	null	Name of table use to define all other IDs

Functions:

Name	Arguments	Return	Description
GetLegendId (Util function)	• **Integer: num** - Numerical ID of the definition	**String**	This function is called to convert a numerical ID of a definition into a string.

MultiTab

MultiTab

MultiTab is used to create a automatic tabulation system.

How to Initialize the MultiTab ?

The initialization of the MultiTab animation is an one step process, just call the function **Init** of the class with his three arguments (**see above for more details**)

Class Details

Functions:

Name	Arguments	Return	Description
Init	• **String: Id** - HMTL ID	**VOID**	This function Initialized every part of the flip: HMTL and the base CSS. It will generate in

	defining the position in the DOM of the viewer • **String: Name** - Define the name of the flip all flip component will be defined by this name (this is to avoid mix if you use different flip so be sure to use different name for each flip). • **CSSClass: MainCSS** - CSS of the main div		the defined ID as ID and all HTML IDs used for this MultiTab will start by the name defined in arguments
AddTab	• **String: Title** - Showed title of the tab • **String: Content** - HTML content you want to add to your tab	**VOID**	Add a new tab with it's content.
ClearTab	• **Integer: TabID** - Number of the tab in order starting by 0.	**VOID**	Remove all the content of a tab.
RemoveTab	• **Integer: TabID** - Number of the tab in order starting by 0.	**VOID**	Completely remove a tab and all it's data.
UpdateData	• **Integer: TabID** - Number of the tab in order starting by 0. • **String: Content** - HTML content you want to add to your tab	**VOID**	Completely delete the content of the tab and replace it by a new content.
AddData	• **Integer: TabID** - Number of the tab in order starting by 0. • **String: Content** - HTML content you want to add to your tab	**VOID**	Add data after the present data, using *append* method.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description

basedId	String	null	HTML Id of the viewer parent.
id	String	null	Global HMTL Id of the viewer (# + name).
name	String	null	Name of the viewer used by the class for all HTML IDs
topID	String	null	Define the ID of the title part
bodyID	String	null	Define the ID of the content part
tabList	• Tab	null	Hold all the tabs.
lastID	Integer	0	Keep the last ID used for the table (all table have a different ID that not based on the position in **TabList**).

Functions:

Name	Arguments	Return	Description
FlipAnimation (Animation Function)	• **Integer: Percent** - Advancement of the animation • **Integer: X** - X based on the advancement • **Class: data** - Data that the function need	**None**	This function is called to update the flipping animation.
GetContentDivFromID	• **Integer: ID** - ID of the tab you want the ID	**String** - HTML ID of the Content of the selected tab.	Call this function to get the ID of a tab content based on the tab ID.
GetContentDivFromArray	• **Integer: ID** - Position of the tab you want the ID	**String** - HTML ID of the Content of the selected tab.	Call this function to get the ID of a tab content based on the tab position.
GetTitleDivFromID	• **Integer: ID** - ID of the tab you want the ID	**String** - HTML ID of the Content of the selected tab.	Call this function to get the ID of a tab title based on the tab ID.
GetTitleDivFromArray	• **Integer: ID** - Position of the tab you want the ID	**String** - HTML ID of the Content of the selected tab.	Call this function to get the ID of a tab title based on the tab position.
GetTabClassFromID	• **Integer: ID** - ID of the tab	**Tab** - Tab class from selected ID	Call this function to get a class based on a ID.
UpdateStyle	• **None**	**None**	Function called to update the design of the tabs, especially the borders around the status.
InitTabClickEvent	• **Integer: TabID** - Number of the tab in order starting by 0	**None**	Function used initialize the events on a new tab.

Structures:

Name	Data	Default Values	Description
Tab	• **Integer: tabID** - ID of the tab based on last id to avoid two tab having the same name. • **Integer: currentArrayID** - Position the holding array of the tab. • **String: tabTitleRef** - HTML String containing the title of the tab. • **String: tabContentRef** - HTML String containing the content of the tab.	• 0 • 0 • null • null	Tab structure is used to hold needed data for the system.

Compatibility

LANGUAGE CLASS: Not Compatible.

OverlappingTab

The overlapping tab in an interactive slide tab system. This class will generate everything and will link all events.

How to Initialize the Overlapping Tab?

The initialization of the Overlapping Tab is an one step process, just call the function **Init** of the class with its two arguments (**see bellow for more details**).
Note: the Initialization is usually done by EE

Once called all setters and data functions, call **Launch** function to generate all the code.

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description

Init	• **String: ID** - HMTL ID defining the position in the DOM of the tab • **String: Name** - Define the name of the tab all tab components will be defined by this name (this is to avoid mixing if you use different ImageList, so be sure to use different name for each tab).	**VOID**	This function Initialized every part of the viewer: HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this tab will start by the name defined in arguments
Launch	• **None**	**VOID**	This function is doing the main setup but it mostly relay on the bigger function **CreateInteraction()** for all the interaction part.

Tab Functions

Name	Arguments	Return	Description
AddTab	• **String: DefaultText** - Default text shown if the language can't be found or no lang has been specified. • **@nullable String: Side** - not used, insert null • **@nullable Lang: Lang** - lang class used to set up the vertical title. • **String: LangID** - if you use Language enter the LangID **(WILL BE USED AS TAB ID)** • **@nullable String: LangSection** - if you use Language enter the LangSection	**VOID**	This function will create a new section, (all interaction will be created during the launch function). The lang information are only for the title of this section (When the section is not hovered, closed). To add the block content (When the section is opened) call **AddTabBlock** to add the large content.
AddTabBlock	• **String: TabId** - ID of the target type **(IDs Are defined by LangID in AddTab).** • **String: DefaultText** - Default text shown if the Language can't be found or no lang has been specified. • **@nullable Lang: Lang** - lang class used to set up the	**VOID**	With a **DefaultText** or the **language class** this function is used to add a block to the selected section. The block is the part shown when the section is open (When the section is hovered).

	vertical title. • **@nullable String: LangID** - if you use Language enter the LangID • **@nullable String: LangSection** - if you use Language enter the LangSection		
AddTitle	• **String: DefaultText** - Default text shown if the Language can't be found or no lang has been specified. • **@nullable String: Side** - not used, insert null • **@nullable Lang: Lang** - lang class used to set up the vertical title. • **@nullable String: LangID** - if you use Language enter the LangID • **@nullable String: LangSection** - if you use Language enter the LangSection	**VOID**	This function will add the section visible when nothing is hovered (**The Tab Title**), the languages content asked is only for the text when another section is hovered (The vertical Text). To add the block content (When the section is opened) call **AddTitleBlock** to add the large content.
AddTitleBlock	• **String: DefaultText** - Default text shown if the Language can't be found or no lang has been specified. • **@nullable Lang: Lang** - lang class used to set up the vertical title. • **@nullable String: LangID** - if you use Language enter the LangID • **@nullable String: LangSection** - if you use Language enter the LangSection	**VOID**	With a **DefaultText** or the **language class** this function is used to add a block to the title. The block is the part shown when the section is open (Here when nothing is hovered).

Setters Functions

Name	Arguments	Return	Description
SetBackgroundColor	• **String: Which** - Selected section ID (Use "Title" for the title div). • **Color: Color** - Color	**VOID**	Change the background color of the selected section.

	to add		
SetBackgroundImage	• **String: Which** - Selected section ID (Use "Title" for the title div). • **String: URL** - Image URL • **String: Size** - Background size • **@nullable String: Filtre** - Use to apply CSS filter • **@nullable Class: Offset** - Use to apply a background offset o **String: x** - X Offset o **String: y** - Y Offset	**VOID**	Use this function to add a background image to a specific section.
SetHeight	• **String or Integer: Height** - Tab height	**VOID**	Set the tab height (**Launch** need to be to apply the setting).
SetSmallWidth	• **Integer: Width** - Tab Width (in %)	**VOID**	Setting up the width when a section is closed.
SetVerticalFont	• **String: Which** - Selected section ID (Use "Title" for the title div). • **String: Name** - Name of the font.	**VOID**	Call this function to change the font of the vertical text of the selected section.
SetVerticalFontColor	• **String: Which** - Selected section ID (Use "Title" for the title div). • **Color: Color** - Color to add	**VOID**	Call this function to change the font color of the vertical text of the selected section.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default	Description

		Value	
allTabs	JQuery DOM Object []	[]	Hold the the all created tabs.
basedId	String	undefined	HTML Id of the viewer parent.
height	String or Integer	0	Hold the height of the tab.
id	String	undefined	Global HMTL Id of the viewer (# + name).
name	String	undefined	Name of this Image list.
smallWidth	Integer	10	Hold the width of the sections when they're closed.
titleTab	JQuery DOM Object	null	Hold the the title tab when created.

Animation Member

Name	Type	Default Value	Description
animations	Integer []	[]	Hold all animation IDs when launch (Match with allTabs ids).
animationsSize	Float []	[]	Hold all the actual width of the sections (Match with allTabs ids).
verticalOpacity	Float []	[]	Hold all the actual opacity of the sections' vertical title (Match with allTabs ids).

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
CreateInteraction	• **None**	**VOID**	This function Initialize all the events, and create the animations This is an initialization step function and is called by the **Launch** function.

Util Functions

Name	Arguments	Return	Description
GetTab	• **String: Name** - ID of the section	**JQuery DOM Object**	Get the selected section (the section) div object.
GetTab_Block	• **String: Name** - ID of the section	**JQuery DOM Object**	Get the selected section title block.
GetTab_Vertical	• **String: Name** - ID of the section	**JQuery DOM Object**	Get the selected section vertical text.

GetTitle	• **None**	**JQuery DOM Object**	Get the title (the section) div object.
GetTitle_Block	• **None**	**JQuery DOM Object**	Get the title block.
GetTitle_Vertical	• **None**	**JQuery DOM Object**	Get the title vertical text.

Compatibility

LANGUAGE CLASS: Entirely Compatible

Status

Status

The status module offer a nice way to show the progress (or the status) of your work, the loading etc...

Module Details

Variables (Members):

Name	Type	Default Value	Description
StatusClasses	• **StateBar** • **LoadingBar**	• null • null	Array holding the main classes of the modules

Functions:

Name	Arguments	Return	Description
CreateStateBar	• **None**	**StateBar**	Function used to create a new **StateBar class**. Does not initialize it.
CreateLoadingBar	• **None**	**LoadingBar**	Function used to create a new **LoadingBar class**. Does not initialize it.

StateBar

StateBar

The StateBar is class used to show the current state of something (Work, Project, Bug resolution, etc...).

How to Initialize the Flip ?

The initialization of the flip animation is an one step process, just call the function **Init** of the class with his three arguments (**see above for more details**)

Class Details

Functions:

Name	Arguments	Return	Description
Init	• **String: ID** - HMTL ID defining the position in the DOM of the viewer • **String: Name** - Define the name of the StateBar all StateBar component will be defined by this name (this is to avoid mix if you use different StateBar so be sure to use different name for each StateBar). • **JSON : Data** - Data used by the StateBar	**VOID**	This function Initialized every part of the StateBar: HMTL, CSS, DATA. It will generate in the defined ID as ID and all HTML IDs used for this timeline will start by the name defined in arguments. Also to define the data and the design you must fill the data arguments, that work thanks to a JSON files **(for more details about the JSON check the section below).**
ChangeState	• **Integer: StateID** - New State to go	**VOID**	Call this function to change the current state of the bar.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
basedId	String	null	HTML Id of the viewer parent.
id	String	null	Global HMTL Id of the viewer (# + name).
name	String	null	Name of the viewer used by the class for all HTML IDs
data	JSON	null	Data of the state bar

Functions:

Name	Arguments	Return	Description
GetStateNameID	• **None**	**String** - HTML ID of the State text	Function used to get the HTML ID of the State Text (used to short the code).
GetBarID	• **None**	**String** - HTML ID of the Bar div	Function used to get the HTML ID of the Bar div (used to short the code).
GetBarFrontID	• **None**	**String** - HTML ID of the Front Bar div	Function used to get the HTML ID of the Front part of the bar (used to short the code).
GetBarBackID	• **None**	**String** - HTML ID of the Back Bar div	Function used to get the HTML ID of the Back part of the bar (used to short the code).

Compatibility

LANGUAGE CLASS: Partially.

Data Structure

Main Structure Details

Note:
black -> Needed
Blue -> Optional

```
{
	"Info": { //Main Info for the StateBar
		"BackColor": "#a6a6a6" //Define bar back color
	},

	"States": [
		{
			"Name": "None" //Display name of the state (The index 0 doesn't need color because there is no bar).
		},
		{
			"Name": "STATE",  //Display name of the state
			"Color": "#000000" //Color of the bar at this state
		}
	]
}
```

LoadingBar

This is a classic loading bar, fully generated.

How to Initialize the Loading Bar?

The initialization of the Overlapping Tab is an one step process, just call the function **Init** of the class with its two arguments (**see bellow for more details**).

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: BaseID** - HMTL ID defining the position in the DOM of the tab • **String: RegistryName** - Define the name of the Bar all Bar components will be defined by this name (this is to avoid mixing if you use different Bar, so be sure to use different name for each Bar).	**VOID**	This function Initialized every part of the viewer: HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this tab will start by the RegistryName defined in arguments
SetPercent	• **Integer: SetPercent** - %	**VOID**	Set the bar percent status (It will update CSS).

Setters Functions

Name	Arguments	Return	Description
SetPositionAbsolute	• **None**	**VOID**	Change CSS: Position to Absolute. Cancel **SetPositionFixed()**.
SetPositionFixed	• **None**	**VOID**	Change CSS: Position to Fixed. Cancel **SetPositionAbsolute()**.
SetTop	• **@CSS String: CssSize** - String for the size, using CSS	**VOID**	Set the top distance of the loading bar. It remove the Bottom property.

	size type		
SetTopMiddle	• **None**	**VOID**	Set the the loading bar position centered with the top property. It remove the Bottom property.
SetBottom	• @CSS String: **CssSize** - String for the size, using CSS size type	**VOID**	Set the bottom distance of the loading bar. It remove the Top property.
SetBottomMiddle	• **None**	**VOID**	Set the the loading bar position centered with the bottom property. It remove the top property.
SetLeft	• @CSS String: **CssSize** - String for the size, using CSS size type	**VOID**	Set the left distance of the loading bar. It remove the Right property.
SetLeftMiddle	• **None**	**VOID**	Set the the loading bar position centered with the left property. It remove the Right property.
SetRight	• @CSS String: **CssSize** - String for the size, using CSS size type	**VOID**	Set the left distance of the loading bar. It remove the Left property.
SetRightMiddle	• **None**	**VOID**	Set the the loading bar position centered with the Right property. It remove the Left property.
SetHeight	• Integer: **Size** - The height size • String: **Scale** - The CSS unit you want to use.	**VOID**	Change the loading bar height.
SetWidth	• Integer: **Size** - The height size • String: **Scale** - The CSS unit you want to use.	**VOID**	Change the loading bar width.
SetRounded	• Boolean: **IsRounded** - If the loading bar have to be rounded on the edges.	**VOID**	Round up (or remove) the loading bar edges.
SetBackColor	• @CSS String: **Color** - CSS Color	**VOID**	Change the color of the inactive part of the loading bar.
SetFontColor	• @CSS String: **Color** - CSS Color	**VOID**	Change the color of the active part of the loading bar.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default Value	Description
height	Integer	20	The value in Integer of the height.
heightType	String	"px"	The CSS Type of the height size.
width	Integer	200	The value in Integer of the width.
widthType	String	"px"	The CSS Type of the width size.
id	String	undefined	Global HMTL Id of the viewer (# + name).
registryName	String	undefined	Name of this Loading Bar.

Functions:

Util Functions

Name	Arguments	Return	Description
GetBackDivID	• **None**	**String**	return the ID of the back div (the inactive bar).
GetFrontDivID	• **None**	**String**	return the ID of the front div (the active bar).

Compatibility

LANGUAGE CLASS: Not compatible.

Time

Time

The time module is used to generate dynamically some timing interface like timelines

Module Details

Variables (Members):

Name	Type	Default	Description

		Value	
TimeClasses	• **VerticalTimeline**	• null	Array holding the main class of this modules

Functions:

Name	Arguments	Return	Description
CreateVerticalTimeline	• **None**	• **VerticalTimeline**	This function is called to create a new **VerticalTimeline** Class (the function does not initiate it).

VerticalTimeline

The vertical timeline is a dynamic way to create a timeline on the vertical way.

How to Initialize the Timeline ?

The initialization of the viewer is an one step process, just call the function **Init** of the class with his three arguments (**see above for more details**)

Class Details

Functions:

Name	Arguments	Return	Description
Init	• **String: NewID** - HMTL ID defining the position in the DOM of the viewer • **String: Name** - Define the name of the timeline all timeline component will be defined by this name (this is to avoid mix if you use different timeline so be sure to use different name for each timeline). • **JSON : data** - JSON data which define how the viewer will be build.	**VOID**	This function Initialized every part of the timeline: HMTL, CSS, Data, etc... It will generate in the defined ID as NewID and all HTML IDs used for this timeline will start by the name defined in arguments Also to define the data and the design you must fill the data arguments, that work thanks to a JSON files **(for more details about the JSON check the section below).**

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
basedId	String	null	HTML Id of the viewer parent.
id	String	null	Global HMTL Id of the viewer (# + name).
name	String	null	Name of the viewer used by the class for all HTML IDs

Functions:

Name	Arguments	Return	Description
GetDateDivId	• **Integer: id** - Number of the of the Div	**String** - HTML ID of the Date div	Function used to get the HTML ID of the a date div (used to short the code).
GetDateTextId	• **Integer: id** - Number of the of the Div	**String** - HTML ID of the text div	Function used to get the HTML ID of the a text div (used to short the code).
GetLineId	• **Integer: date** - date of the line you want the ID • **Integer: line** - Numerical ID of the line you want	**String** - HTML ID of the line	Function used to get the HTML ID of the a line (used to short the code).
isOK	• **JS Object: obj** - Reference to the obj	**Boolean**	Check if the selected OBJ is valid (Defined // Non-Null)

Compatibility

LANGUAGE CLASS: Partially.

Data Structure

Main Structure Details

Note:
black -> Needed
Blue -> Optional

```
"History": {   //Data class holder
        "Global": { //Define the main settings of the timeline
                "CSSGroup": { //Define the CSS of the time
                        "CSS": { //CSS Class
                        },
                        "Children": [ //To add extra CSS for some children (see below how it work)

                        ]
                },
                "bgURL": null, //To change the line of the Timeline (it act as a Background), specify null  for using the default line
                "arrowURL": null, //To change the arrow of the Timeline (it act as a Background), specify null  for using the default arrow
                "Info": { //Define some extra setting
                        "SpaceTitle": "50px", //like the margin for the titles (dates)
                        "SpaceLine": "25px" //Define some extra setting
                }
        },


        "Dates": [ //Groups of the dates and events. Each date is group of events.
                {
                        "Title": { //Group to define the title of the date (Which basically the date for the user)
                                "Text": "2018", //Text of the date
                                "CSSGroup": { //CSS groups for the date
                                        "CSS": {

                                        },
                                        "Children": [

                                        ]
                                }
                        },
                        "EventsList": [  //List of event in chronological order that belong to this date
                                {
                                        "Text": " - Test line", //Text of the event
                                        "CSSGroup": { //CSS groups for the event
                                                "CSS": {
                                                },
                                                "Children": [

                                                ]
                                        }
                                }
                        ]
                }
        ]
},
```

Note:

- **CSS Children**: Children is a list of class composed by two data (Name are important):
  - **String: ID** - HTML ID of the target (it start by adding the ID of the target object then the children)
  - **Class: CSS** - CSS details to add

ToolBox

The tool box module offer a selection of tool class.

Module Details

Variables (Members):

Name	Type	Default Value	Description
ToolBoxClasses	• InteractiveToolBox	• null	Array holding the main classes of the modules

Functions:

Name	Arguments	Return	Description
CreateInteractiveToolBox	• **None**	InteractiveToolBox	Function used to create a new **InteractiveToolBox class**. Does not initialize it.

InteractiveToolBox

The InteractiveToolBox class offer a way to replace the default **context menu** and create yours.

How to Initialize the Interactive Tool Box?

The initialization of the Interactive Tool Box is an one step process, just call the function **Init** of the class with its two arguments (**see bellow for more details**).

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: ID** - HMTL ID defining the position in the DOM of the tab (Use body generic use). • **String: Name** - Define the name of the ToolBox all ToolBox components will be defined by this name (this is to avoid mixing if you use different ToolBox, so be sure to use different name for each ToolBox).	**VOID**	This function Initialized every part of the viewer: HMTL, CSS, Data, Events, etc… It will be generated in the defined ID and all HTML IDs used for this tab will start by the RegistryName defined in arguments
Open	• **Class: Pos** - tool box position in o **int: X** - Left position o **int: Y** - Top position	**VOID**	Open the InteractiveToolBox at defined position
Close	• **None**	**VOID**	Close the box.
AddOption	• **String: Name** - Define the option name (make sure it's unique. • **String: Text** - Text to add in the option. • **Function: Event** - Function called when this option is clicked. • **Class: EventsArgs** - Arguments that need to be passed for the Event function	**VOID**	Add an option to the tool box.
ClearOptions	• **None**	**VOID**	Remove all options (tool is not opening if there is no option).

Setters Functions

Name	Arguments	Return	Description
SetWidth	• **Integer - @CSS String: Width** - Width	**VOID**	Change the tool box width.
SetBorder	• **String: Where** -	**VOID**	The a border to the tool box

	Where the border should be created (for the the tool box): "top", "bottom", "left", "right", "all". You can add multiple location at the same time • @CSS String: **Type** - Set the border Type. • **Integer** - @CSS String: **Size** - The border size. • Color: **Color** - set the border color		
SetOptionHeight	• **Integer** - @CSS String: **Height** - The height of each options • @Nullable String: **On** - Define if the this height is applying for when the option is "Normal" or 'Hovered" or "Pressed". You can use "All" to apply at any condition.	**VOID**	Set the option height size. (Same for all).
SetOptionPadding	• String: **Where** - Where the padding should be set (for the the tool box): "top", "bottom", "left", "right", "all". You can add multiple location at the same time • **Integer** - @CSS String: **size** - The CSS padding size • @Nullable String: **On** - Define if the this height is applying for when the option is "Normal" or 'Hovered" or "Pressed". You can use "All" to apply at any condition.	**VOID**	Set the padding inside all options. (Same for all).
SetOptionBorder	• String: **Where** - Where the padding should be set (for the the tool box): "top", "bottom", "left", "right", "all". You can add multiple location at the same time • @CSS String: **type** - CSS border type. • **Integer** - @CSS String: **size** - The CSS border size	**VOID**	Set the border of each options.

	• **Integer []: Color** - Array defining the RGB. • @Nullable String: **On** - Define if the this height is applying for when the option is "Normal" or 'Hovered" or "Pressed". You can use "All" to apply at any condition.		
SetOptionBackgroundColor	• **Integer [] -** @CSS String**: Color** - Array or string defining the RGB. • @Nullable String: **On** - Define if the this height is applying for when the option is "Normal" or 'Hovered" or "Pressed". You can use "All" to apply at any condition.	**VOID**	Set the background color for each options (Same for all).
SetOptionGradientBackgroundColor	• **Integer -** @CSS String**: to** - Degrees of the gradient • **Integer [][] -** @CSS String[]**: Colors** - Array that contain arrays of Integer for the RGB or Strings of the colors • @Nullable String: **On** - Define if the this height is applying for when the option is "Normal" or 'Hovered" or "Pressed". You can use "All" to apply at any condition.	**VOID**	Set the background color for each options with a gradient effect. **/!\ /!\ This function is not done yet and might not work properly /!\ /!**

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default	Description

		Value	
options	Class[] : • String: name - name of the option • String: text - text of the option • Event: event - callback when option is pressed.	[]	Hold a list of the options details.
isOpen	Boolean	false	Used to check if the toolbox is or not open.

style	Class:
	• CSS Class: normal - CSS when option is normal
	• CSS Class: hovered - CSS when option is hovered
	• CSS Class: pressed - CSS when option is pressed
		Each CSS Class has this default setting:
		
		//position
		
		height: "100px",
		
		padding: "none", padding-top: "none", padding-right: "none", padding-bottom: "none", padding-left: "none",
		
		//Design
		
		border: "none", border-top: "none", border-right:
			Hold the 3 CSS Class of the option, it is used by the options event when hovered or pressed

		"none", border-bottom: "none", border-left: "none", background-color: "white", background-image: "none"	
id	String	undefined	Global HMTL Id of the viewer (# + name).
name	String	undefined	Name of this Tool box.
basedId	String	undefined	Where it is located is the existing DOM (its parent).

Functions:

Updates functions

Name	Arguments	Return	Description
UpdateStyle	• **None**	**None**	Set all option to default style.

Util Functions

Name	Arguments	Return	Description
GetOptionFromName	• **String: Name** - name of the selected option	**JQuery DOM Object**	Get the option div.

Compatibility

LANGUAGE CLASS: Not compatible.

PopUp

The pop up module offer a selection pop up windows that can be used at any time.

Module Details

Variables (Members):

Name	Type	Default Value	Description
PopUpFunctions	• **PopUp Select**	• null	Array holding the main classes of the modules **Note: there is a mistake in the same and will corrected.**

Functions:

Name	Arguments	Return	Description
AddPopUpSelect	• **String: ID** - Where the popup should be located in the DOM. • **String: Name** - Name of this popup (should be unique	**PopUpSelect**	Function used to create and Initialize a **PopUpSelect class**.

PopUpSelect

The PopUp Select class offer a way to open a pop up window where there is a list of different options. where for each you can add events.

How to Initialize the PopUp Select?

The initialization of the PopUp Select is an one step process, just call the function **Init** of the class with its two arguments (**see bellow for more details**).

Also to be able to open the PopUp the function **SetOpening()** has to be called

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description

Init	• **String: ID** - HMTL ID defining the position in the DOM of the tab (Use body generic use). • **String: Name** - Define the name of the PopUp all PopUp components will be defined by this name (this is to avoid mixing if you use different PopUp, so be sure to use different name for each PopUp).	**VOID**	This function Initialized every part of the viewer: HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this PopUp will start by the RegistryName defined in arguments
SetOpening	• **Curve: CurveType** - Animation curve type for opening. • **Integer: Duration** - Duration of the opening • **Integer: FPS** - How many frame per second • **Class: CurveDetails** - Any curve details require by the Animation	**VOID**	Setting up all information about the opening animation of the pop up.
Open	• **None**	**VOID**	Open the popup **Note: SetOpening() has to be called first.**
Close	• **None**	**VOID**	Close the popup
Add	• **String: Text** - Text added to the option (it also represent default text if you use the Lang Class). • **String: LangID** - LangID of the text (it also used as option name). • **Function: Func** - Function called when this option is clicked. • **@Nullable Lang: Lang** - Lang Class used for this option. • **@Nullable String: LangSection** - If you using the Lang Class then add the Lang Section here.	**VOID**	Add an option to the select popup
ClearOptions	• **None**	**VOID**	Remove all options (tool is not opening if there is no option).

Setters Functions

Name	Arguments	Return	Description
SetHeight	• **Integer** - @CSS String: **Height** - Height	**VOID**	Change the popup height but also reset position to center the window.
SetHeight	• **Integer** - @CSS String: **Width** - Width	**VOID**	Change the popup width but also reset position to center the window.
SetIndex	• **Integer: Index** - z-index of the popup	**VOID**	Change the z-index of the pop up.
SetTitleText	• **String: Text** - Text added to the option (it also represent default text if you use the Lang Class). • **@Nullable Lang: Lang** - Lang Class used for this option. • **String: LangID** - LangID of the text • **@Nullable String: LangSection** - If you using the Lang Class then add the Lang Section here.	**VOID**	Set the title text of this popup.
SetTitleSize	• **Integer** - @CSS String: **Size** - Set the title section size	**VOID**	Resize the title section, the selection part will be automatically resized and moved too.
SetTitleBackgroundColor	• **Color - : Color** - Color of the title section background	**VOID**	Setup the background color of the title section.
SetSelectBackgroundColor	• **Color - : Color** - Default background color of the selection. • **Color - : HoveredColor** - Background color of the selection when Hovered. • **Color - : PressedColor** - Background color of the selection when Pressed.	**VOID**	Setup background color of the selection sections. apply for each
SetBorderStyle	• @CSS String: **Type** - Border type • **Integer** - @CSS String: **Thickness** - Border Size • **Color - : Color** - Color of the border	**VOID**	Set the border that will be used as separation between two options.

SetHorizontalAlign	• **Integer** - @CSS **String** - **String**: **Position** - Set the text margin value (use "centered" to center the text).	**VOID**	Set the options horizontal alignment.
SetBackColor	• **Color** - : **Color** - background color	**VOID**	Set the popup background color.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default Value	Description
id	String	undefined	Global HMTL Id of the viewer (# + name).
name	String	undefined	Name of this PopUp.
basedId	String	undefined	Where it is located is the existing DOM (its parent).
select	Class[]: • JQuery DOM Object: div - option main div • JQuery DOM Object: p - option p • String: langId • String: id	[]	Where all options details are stored.
scale	Integer	0	Store the actual size of the PopUp. (Used by animation).
animation	Integer	null	Store the actual animation (used with ClearInterval).
curve	Class: • Curve: CurveType	{}	Hold animation details, this class is set with **SetOpening()** function.

	• Integer: duration • Integer: FPS • Class: details		
selectMargin	Class: • String: margin-left	0px	Hold the text margin for the options
selectBorder	String	none 0px black	Hold the text border for the options
NormalBG	String	white	Hold the background color for the options
HoveredBG	String	none	Hold the background color for the options when hovered
PressedBG	String	none	Hold the background color for the options when pressed

Functions:

Updates functions

Name	Arguments	Return	Description
UpdateTitle	• **None**	**None**	Reset text centering of the title.
UpdateSelects	• **None**	**None**	Reset text centering of the options.
UpdateStyle	• **None**	**None**	Set all design of the popup

Util Functions

Name	Arguments	Return	Description
GetTitle	• **None**	**JQuery DOM Object**	Get the title div.
GetTitleP	• **None**	**JQuery DOM Object**	Get the title p (text).
GetSelect	• **None**	**JQuery DOM Object**	Get select section div.
GetBG	• **None**	**JQuery DOM Object**	Get background section div.

Compatibility

LANGUAGE CLASS: Entirely Compatible

Design

This group is composed by module used to enhance existing structure:

- **Background**: Classes used to enhance some blocks background.
- **Hover**: Classes to add special design when a div is hovered.

Background

This module will offer a way to add new type of background, more dynamics and animated.

Module Details

Variables (Members):

Name	Type	Default Value	Description
BackgroundClasses	• **Image Background**	• null	Array holding the main classes of the modules

Functions:

Name	Arguments	Return	Description
AddImageBackground	• **String: To** - Which block the background should be set with. • **String: Name** - Name of this background (should be unique).	**ImageBackground**	Function used to create and Initialize a **ImageBackground class**.

ImageBackground

The Image Background class offer a way to add images in background and adding effect to it (e.g. scrolling movement), also you can use this class to do a image slide in the back.

How to Initialize the Image Background?

1: The initialization of the Image Background is an one step process, just call the function **Init** of the class with its two arguments (**see bellow for more details**).

2: Use **setter functions** to customize your background.

3: Then use **Launch()** with its 3 arguments.

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: ID** - HMTL ID defining the position in the DOM of the tab (Use body generic use). • **String: Name** - Define the name of the Background all components will be defined by this name (this is to avoid mixing if you use different Background, so be sure to use different name for each Background).	**VOID**	This function Initialized every part of the viewer: HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this PopUp will start by the RegistryName defined in arguments. Usually Initialize by EE.
Launch	• **Integer: Time** - How many time an image stay active (In MS). • **Integer: TransitionTime** - How many time it take to switch between images. • **@Nullable @CSS String: positionType** - Used to set what is the background position type.	**VOID**	It is used to launch the image slide of the background.
AddImage	• **String: URL** - URL of this image you want to add.	**VOID**	Add an image to the background
AddImageMovement	• **Scrolling: Scroll** - Required scrolling class	**VOID**	Add add a movement when the page is scrolling down.

Setters Functions

Name	Arguments	Return	Description

MatchHorizontal	• **None**	**VOID**	Change if the background image has to match (100%) horizontally.
AddBlur	• **Integer: Intensity**- intensity	**VOID**	Setup background blur intensity (default: none).
Addbrightness	• **Integer: Intensity** - intensity	**VOID**	Setup background brightness intensity (default: none).

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default Value	Description
id	String	undefined	Global HMTL Id of the viewer (# + name).
name	String	undefined	Name of this PopUp.
basedId	String	undefined	Where it is located is the existing DOM (its parent).
Images	String[]	[]	Store all images URL.
activeImage	Integer	0	ID of the active image.
nextImage	Integer	1	ID of the next image.
matchHorizontal	Boolean	false	Hold the information about the need to make sure image are a 100% horizontally
blurIntensity	Float	0	Hold blur intensity
brightness	Float	0	Hold brightness intensity

Functions:

Updates functions

Name	Arguments	Return	Description
UpdateTitle	• **None**	**None**	Reset text centering of the title.
UpdateSelects	• **None**	**None**	Reset text centering of the options.
UpdateStyle	• **None**	**None**	Set all design of the popup

Util Functions

Name	Arguments	Return	Description
GetBG	• **None**	**JQuery DOM Object**	Get the background image div.
GetFG	• **None**	**JQuery DOM Object**	Get the foreground image div.

Compatibility

LANGUAGE CLASS: Not compatible.

Hover

This module will offer a way to add new type of background, more dynamics and animated.

Module Details

Variables (Members):

Name	Type	Default Value	Description
HoverClasses	• **Hover Gradient** • **Hover Blur**	• null • null	Array holding the main class of this modules

Functions:

Name	Arguments	Return	Description
AddHoverGradient	• **String: ID** - Which block the hover should be set with.	**HoverGradient**	This function is called to create and initialize a new **HoverGradient Class.**
AddHoverBlur	• **String: ID** - Which block the hover should be set with. • **String: Name** - Name of this hover (should	**HoverBlur**	This function is called to create and initialize a new **HoverBlur Class.**

	be unique).		

HoverBlur

THE HOVER_BLUR CLASSES IS NOT WORKING YET, COME BACK LATER

How to Initialize the HoverBlur ?

THE HOVER_BLUR CLASSES IS NOT WORKING YET, COME BACK LATER

HoverGradient

The HoverGradient classes will create an gradient animation when you hover the selected div.

How to Initialize the Hover Gradient?

The initialization of the Hover Gradient is an one step process, just call the function **Init** of the class with its only arguments (**see bellow for more details**).

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: ID** - HMTL ID defining the position in the DOM of the viewer	**VOID**	This function will set the main members and then relay the initialization to **InitEvent**.
InitEvent	• **None**	**VOID**	This function will add the Events to the selected div. This function also call the Animation module.

Setters Functions

Name	Arguments	Return	Description
SetDuration	• **Integer: TimeInMS** - Duration time of the full animation.	**VOID**	Setup the animation duration.
SetFirstColor	• **Color: Color** - Left	**VOID**	Setup the gradient left color

	Color		
SetSecondColor	• **Color: Color** - Right Color	**VOID**	Setup the gradient right color
SetSize	• **Integer: Size** - Size in % of 50%.	**VOID**	This function change the transition size on both side (e.g. Size = 10% will be 20%).

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
id	String	undefined	Global HMTL Id of the viewer (# + name).

animationInfo	Class:
	• Integer: time
	• String: firstColor
	• String: secondColor
	• size: time
	• String: direction - if animation goes "forward" or "backward".
		{
		
		//Animation
		time: 500,
		
		//Color
		firstColor: "#000000",
		
		secondColor: "#ffffff",
		
		//Size
		size: 10,
		
		//Direction
		direction: "Forward"
		}
			Hold all information required by the animation function **Animate()**.

animation	Integer	null	Hold the **SetInterval()** ID, that can be use to stop the interval.
animationStoppedAt	Integer	null	Where the last animation stopped (Percent not X).

Functions:

Animation

Name	Arguments	Return	Description
Animate	• **Integer: Per** - percent of the animation. • **Integer: X** - X relative to **Per**. • **Class: Ref** - Reference to this class	**VOID**	Function called by Animation module.

Compatibility

LANGUAGE CLASS: Not compatible.

System

Systems

This group contain system to add some functionality on your website and also help for complex task.
The classes inside those modules can be create once (one per program - no need for many).

- **Language**: languages handling classes.
- **Mouse**: Mouse help functions.
- **Scrolling**: Easier scrolling event system.
- **File**: Help with system file.
- **Cookies**: Handle cookies.
- **Keyboard**: Keyboard help functions.

Language

Language

The language module is used to handle on a dynamic way languages, for example with the lang class you can change the language without reloading the page and use a JSON file to load languages

Module Details

Variables (Members):

Name	Type	Default Value	Description
LanguageClasses	**Lang**	null	Array holding the main classes of this modules

Enumeration:

Name	List	Description
Languages	• ENGLISH: 0 • FRENCH: 1	Enumerator used to defined which the language you want.

Functions:

Name	Arguments	Return	Description
GetLangClass	• **None**	**Lang**	Return the active **Lang** class (You can use only one **Lang** at the time - this function is not creating a new one).

Lang

The Lang classes will be used to handle language through your website or web app. It will use a single JSON file, read all and update all yours texts so can easily translate your whole website.

How to Initialize the Lang?

The initialization of the Lang is an one step process, just call the function **Init** of the class with its two arguments (**see bellow for more details**).

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: FileLocation** - Where all **JSON files** are stored • **Function: Callback** -	**VOID**	This function will try to get where the user is located to set a default languages (Cookies has to be handle the external code). Once done it will call the **InitText** function.

	function called when everything has been set.		
InitText	• **String: FileLocation** - Where all **JSON files** are stored • **Function: Callback** - function called when everything has been set.	**VOID**	Function called by the **Init** function it will load all languages files.
ChangeLanguage	• **Languages: Language** - Define which language you want to set.	**VOID**	Will change the whole website language (It will update automatically, no page reload is required).
ChangeText	• **String: htmlID** - Which block the text should be add. • **String: langID** - ID in the **JSON files** of your text. • **String: langSection** - Group in the **JSON files** where the langID is located. • **String: defaultText** - default text if Lang cannot find the specified langID	**VOID**	This is the main function of this class, with this you will be able to add your text into your website. This function will look for the text in selected language file : FILE->GROUP->LANG_ID If nothing was found it will look in the default text file (English). And if not text was it will use **defaultText** as text. All the data will be stored into binders class, will updated at each **Update()**. **Note: everything that is in the defined htmlID is deleted at each update**
ChangeTextHolder	• **String: htmlID** - Which block the text should be add. • **String: langID** - ID in the **JSON files** of your text. • **String: langSection** - Group in the **JSON files** where the langID is located. • **String: defaultText** - default text if Lang cannot find the specified langID	**VOID**	Same as **ChangeText** but it will apply to text inputs text-holder.
DeleteText	• **String: htmlID** - Define where in the DOM the Lang should be unbind.		Unbind language from a specified HTML ID. **Note: Text will not be removed but will be updated.**

Setters Functions

Name	Arguments	Return	Description
AddCallback	• **Function: CallbackFunc** -	**VOID**	Add another callback function (can add infinite amount). The function are called

	Function to add		when texts are updated. **Note**: this functionality is often use when an update modify something that JS need correct each change.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
language	Languages	Languages .ENGLISH	Hold the active language.
data	JSON Class[]	[]	Hold all languages data structure. The array IDs match with the Language Enumerator.
defaultData	JSON Class	.	Hold the default languages data structure (English). Used for backup.
bindedList	Binder[]	[]	List of all the binders.
callbackList	Function[]	[]	Hold the list of all callback that **UpdateText** need to call

Functions:

Update

Name	Arguments	Return	Description
UpdateText	• **None**	**VOID**	Will go trough all binders and update all texted (it delete and rewrite content). At the function end it will call all callbacks.

Accessor

Name	Arguments	Return	Description
GetData	• **Boolean**: **NeedDefault** - Define if it was the default data you require.	**VOID**	Send back the data or defaultData.

JSON Data

The JSON Files are define by they name

EN.json = ENGLISH
FR.json = FRENCH

Here is a sum/example of the JSON Structure:

```
{
	"Section1": [

		{
			"LineID" : "LangID1",
			"Line" : "Text1"
		},

		{
			"LineID" : "LangID2",
			"Line" : "Text"
		}
	],

	"Example": [

		{
			"LineID" : "ExampleLine",
			"Line" : "Example Text"
		},

		{
			"LineID" : "Test_Line",
			"Line" : "This is a test line."
		}
	]
}
```

Compatibility

LANGUAGE CLASS: N/A

Binder

Binder is a private class used by EE Lang class to bind all the data required for each text

Private Details

/!\/!\/!\/!\ All private details are usable but it is highly recommended to do not use any of those. It might make the class unstable. /!\/!\/!\/!\

Variables (Members):

Name	Type	Default Value	Description
htmlID	String	null	Where the text is located
langID	String	null	What is the text ID in **JSON files**
langSection	String	null	What is the text Group/Section in **JSON files**
defaultText	String	null	Text to show the Lang class couldn't find any text.
holder	Boolean	false	Define if it is for a text-holder.

Mouse

This modules is here to create API with user mouse by providing some functionality that JS or JQuery don't bring to you.

like all other System modules the classes here can be only ones.

Note: this module is still light and might not be interesting for now.

Module Details

Variables (Members):

Name	Type	Default Value	Description
mouse	**Mouse**	• null	Variable that contain the **Mouse** class that you load (only one mouse can be load, if you want to launch it call the **AddMouse** function to Initialized the mouse).
initialized	**Boolean**	false	Boolean used to check if the mouse class has already been Initialized.

Functions:

Name	Arguments	Return	Description
AddMouse	• **None**	**None**	This function check if the module has been loaded and if it has never been called. If all the condition has been fulfill then it Initialized the mouse class (note the class has been instanced during the **Call** function of the **EE**.

Mouse

This class provide some data info on the mouse to help handling it

Class Details

Functions:

Name	Arguments	Return	Description
AddCallback	• **Function**: **Func** - Callback function	**None**	Add callback function for when the mouse mouse. **Note: it will sent e arguments to the function.**

Variables (Members):

Name	Type	Default Value	Description
events	**Class**	• **Event Class**: **windowEvent** - Class that come from MouseMove event on the "Window"	Hold event from the mouse
winPos	**Class**	• **Integer**: **height** - Define the height position	Hold the mouse position based on the window

		based on the window • **Integer: Width** - Define the width position based on the window	

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Functions:

Name	Arguments	Return	Description
Init	• **None**	**None**	Setup a move mouse event over the window that will update the details of the mouse class

Compatibility

LANGUAGE CLASS: Not compatible.

Scrolling

This module provide a good way to control scrolling events.

like all other System modules the classes here can be only ones.

Module Details

Variables (Members):

Name	Type	Default Value	Description

scrolling	**Scrolling**	• null	Variable that contain the **Scrolling** class that you load (only one scrolling can be load, if you want to launch it call the **AddScrolling** function to Initialized the mouse).
initialized	**Boolean**	false	Boolean used to check if the scrolling class has already been Initialized.

Functions:

Name	Arguments	Return	Description
AddScrolling	• **None**	**None**	This function check if the module has been loaded and if it has never been called. If all the condition has been fulfill then it Initialized the scrolling class (note the class has been instanced during the **Call** function of the **EE**.

Scrolling

Scrolling

As the module description say: this class provide a good way to control scrolling events.

Class Details

Variables (Members):

Name	Type	Default Value	Description
percent	**Integer**	0	Percent of the position of the scroll
maximum	**Integer**	0	Height of the body that define the maximum scrolling

Functions:

Name	Arguments	Return	Description
AddCallback	• **Function**: **Func** - Callback function to add.	**None**	Add a callback function to the **callbackList**. The Callback **function** will received as arguments a class with: **String**: **scrollWay** - Inform the direction of the scrolling. Can be "up" and "Down". **Float**: **percent** - send the actual percent

			of the scrolling. **Integer: scrollPos** - send the page position in px
ScrollTo	• **Integer: pos** - Define the position on the window you want to go.	**None**	Move the page with an **EaseInOut animation** to the designated position.
Reset	• **None**	**None**	Will reset all data if there is re-dimension and/or if a moving is not coming from the class

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
callbackList	• **Function**	• None	Define a list of called back function that are called when the user scroll.

Functions:

Name	Arguments	Return	Description
Init	• **None**	**None**	Setup the scrolling class by Initiating the main event which is basically a scrolling event on windows. The callback of this event will modify all data and information of the class and also it will call the callback functions of the **callbackList**.
UpdateMaximum	• **None**	**None**	Update the maximum value called by the **Reset** function.
callbackFunctions	• **String: direction** - Define which way the scrolling was. Can be "up" or "Down"	**None**	Call all the callback function from the **callbackList.** This function is called the event function initialized in the **Init** function.
Scrolling (Animation Function)	• **Integer: Percent** - Advancement of the animation • **Integer: X** - X based on the advancement • **Class: data** - Data that the function need	**None**	Move the page.

Compatibility

LANGUAGE CLASS: Not compatible.

File

The file module has been create to help with handling files from the system or to the system.

Module Details

Variables (Members):

Name	Type	Default Value	Description
DeviceFileFunctions	• MakeFileAndDownload • LoadFile	• null • null	Array holding the device file functions of the module

Functions:

Device File:

Name	Arguments	Return	Description
MakeFileAndDownload	• String: **Name** - File name (with extension). • String: **Text** - File content.	**VOID**	Write and download a file.
LoadFile	• Function: **Callback** - Function called when loaded.	**VOID**	**NOT WORKING YET.**

Cookies

The cookies module has been create to help with handling cookies.

Module Details

Variables (Members):

Name	Type	Default Value	Description
Functions	• **GetCookies**	• null	Array holding the device file functions of the module

Functions:

Cookies:

Name	Arguments	Return	Description
GetCookies	• **String: Name** - Cookie name	**VOID**	Return the data stored in the cookie

Keyboard

This modules is here to create API with user keyboard by providing some functionality that JS or JQuery don't bring to you.

like all other System modules the classes here can be only ones..

Module Details

Variables (Members):

Name	Type	Default Value	Description
mouse	**Keyboard**	• null	Variable that contain the **Keyboard** class that you load (only one keyboard can be load, if you want to launch it call the **AddKeyboard** function to Initialized the keyboard).
initialized	**Boolean**	false	Boolean used to check if the mouse class has already been Initialized.

Functions:

Name	Arguments	Return	Description
AddKeyboard	• **None**	**None**	This function check if the module has been loaded and if it has never been called. If all the condition has been fulfill then it Initialized the mouse class (note the class

			has been instanced during the **Call** function of the **EE**.

Keyboard

WE STILL WORKING ON IT.

Util

This group contain diverse module that can be useful (also the util group will now handle non-module functions for a quick access):

- **Converter**: Diverse conversion functions
- **Redirect**: Offer a good way to redirect in JS with different methods.
- **LoadCheck**: Making sure that everything is loaded before use.

Converter

The Converter module help to convert a lot of different things (for now the class is still light but it will be upgraded in the future).

Module Details

Variables (Members):

Name	Type	Default Value	Description
rgbToString	**Function**	null	function **rgbToString**.

Functions:

Name	Arguments	Return	Description
rgbToString	• **String: css** - CSS command for a RGB modification	**Integer Array:** • 1: Red • 2: Green • 3: Blue • 4 (Optional): Alpha	Transform a RGB CSS command to an array of integer defining the colors value. **Note: you can put RGBA CSS command.**

Non-Module Functions:

Name	Arguments	Return	Description
@Non-Module HexToArray	• **String: Hex** - Hexadecimal of a color.	**Integer[3]:** • 1: Red • 2: Green • 3: Blue	Transform an Hexadecimal color to an Integer Array.
@Non-Module ArrayToHex	• **Integer[3]:** o 1: Red o 2: Green o 3: Blue	**String**	Transform an Integer Array to a Hexadecimal color.

Redirect

The Redirect module help to send redirect easily on other page with different methods.

Module Details

Variables (Members):

Name	Type	Default Value	Description
RedirectFunctions	• **RedirectWithPost**	• null	Array holding the two main functions of the module (see the functions details below).

Functions:

Name	Arguments	Return	Description
Redirect	• **String: URL** - URL To call • **String: Method** - Method to call (only "POST" for now). • **Class: Args** - Java class: the names will the references to receive and the values will be the values.	**None**	Redirect to the select URL with data trough classics methods.

Functions Details

RedirectWithPost:

Arguments

- **String: URL** - URL To call
- **Class: Args** - Java class: the names will the references to receive and the values will be the values.

Description

This function while generate in the body of your a HMTL a form with all the data and will submit it.

LoadCheck

The LoadCheck module make your code waiting for an element to load.

Module Details

Variables (Members):

Name	Type	Default Value	Description
LoadCheck	**Function**	• null	Member hold a function.

Functions:

Name	Arguments	Return	Description
ImgLoaded	• **String: URL** - URL To call • **Function: OnLoadedCallback** - Function called when the image is loaded. • **Boolean: Debug** - Write the console when and which image is loaded.	**None**	Will use the web browser to check when the image is loaded to display it.

Animation

Animation

The Animation module has been create to simplify animation using javascript.

The module present a simple function **animate** to help calculating your animation position (More details bellow), moreover it include two other under modules: **Flip** and **Animated gradient.**

Module Details

Variables (Members):

Name	Type	Default Value	Description
animationClass	• **Flip** • **Animated Gradient**	• null • null	Class holding the main classes of the modules.
animationFunctions	• **Linear Animation** • **EaseInOutAnimation** • **BumpEaseInOutAnimation**	• null • null	Class holding the two main functions of the module (see the functions details below).

Enumeration:

Name	List	Description
Curves	• LINEAR: 0 • EASE_IN_OUT: 1 • BUMP_EASE_IN_OUT: 2	Enumerator used to defined which kind of animation you want (it will pick one of the functions above - see **functions description for the difference between curves**).

Functions:

Name	Arguments	Return	Description
Animate	• **Curves: Type** - The	**VOID**	Animate will choose one of the **AnimationFunctions** based on your

	type of animation curve you want. • **String: Name** - The name of the animation (for debug) can be null (no debug) • **Integer: Duration** - Total time for the animation in millisecond. • **Integer: FrameRate** - animation frame rate in FPSeconds. • **Function: AnimationFunction** - callback animation (see description for more details). • **Class: ExtraData** - Extra data need by the callback function, can be null or not specified. • **Class: CurveDetails** - Extra information curves need (See **curves details** for more information). • **Float: StartAt** - Used to force the of the animation at specific % (must be between 0-100) , can be null or not specified.		selection. The callback function has to take two arguments: ▸ **Integer: percent** status of the animation (0-100). ▸ **Integer: X** in function of percent above (0-100). If needed, the callback function can take a third argument which is extra data you need to transfer to the function (**ExtraData**). Note that the callback function no longer belong to the calling class when executed.
CreateFlip	• **None**	**Flip**	Function is used to create a new **Flip class.**
AddGradient	• **String: ID** - The DOM id of the targeted section.	**AnimatedGradient**	Create and Initialize a new **AnimatedGradient Class.**

Functions Details

LinearAnimation:

Arguments

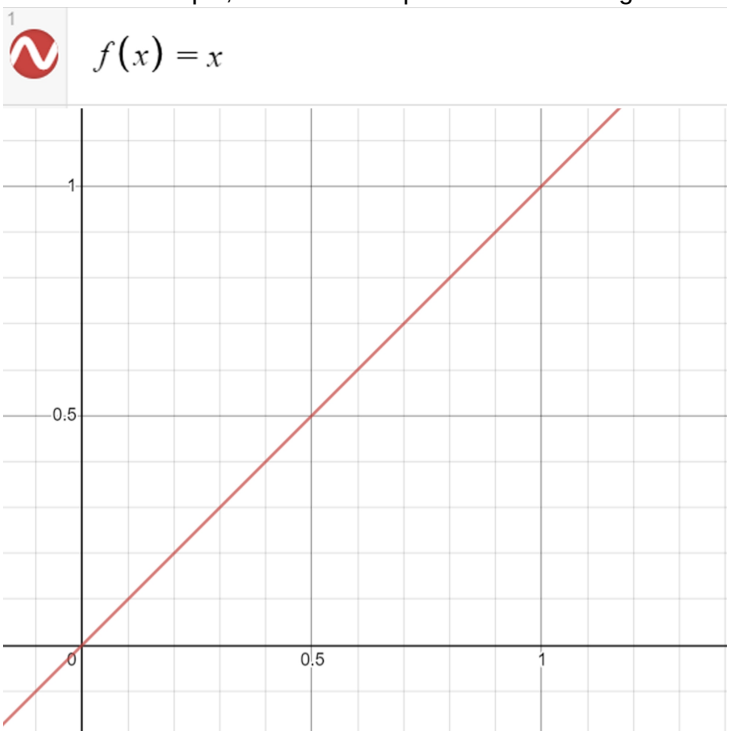
- **Integer: MaxTimeInMS** - Duration of the animation in milliseconds.
- **Integer: FrameRate** - Animation frame rate in FPSeconds.
- **Function: Function** - Call back animation with two or 3 arguments (Percents, X, ExtraData).
- **String: INFO_NAME** - Name of the animation (debug purpose).
- **Boolean: IsOtherArguments** - Define if callback function need extra data.
- **Class: FUNCTION_NEEDED_INFO** - Extra data needed by function.
- **Float: StartAt** - Used to force the of the animation at specific % (must be between 0-100) , can be null

or not specified.

Description

Like other animation function LinearAnimation will calculate x position based on a advancement and sent back the info trough a callback function with arguments (Percent, X). The callback function just need to place items on there position based on X value.

The linear is simple, it is the same percent as X during the whole animation (x=y):



You can see the linear function which stay the same all along.

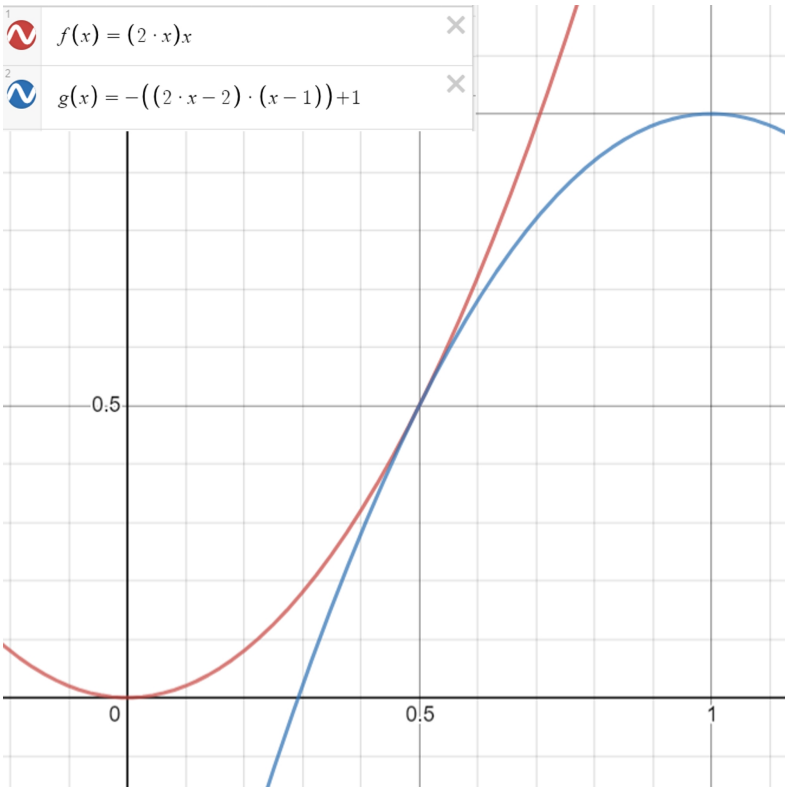
EaseInOutAnimation:

Arguments

- **Integer: MaxTimeInMS** - Duration of the animation in milliseconds.
- **Integer: FrameRate** - Animation frame rate in FPSeconds.
- **Function: Function** - Call back animation with two or 3 arguments (Percents, X, ExtraData).
- **String: INFO_NAME** - Name of the animation (debug purpose).
- **Boolean: IsOtherArguments** - Define if callback function need extra data.
- **Class: FUNCTION_NEEDED_INFO** - Extra data needed by function.
- **Float: StartAt** - Used to force the of the animation at specific % (must be between 0-100) , can be null or not specified.

Description

Like other animation function LinearAnimation animation will calculate x position based on a advancement and sent back the info trough a callback function with arguments (Percent, X). The callback function just need to place items on there position based on X value.
The ease_in_out curve is a bit complex it use two different curves cut at the middle, it first accelerate at the fastest speed then decelerate:



BumpEaseInOutAnimation:

Arguments

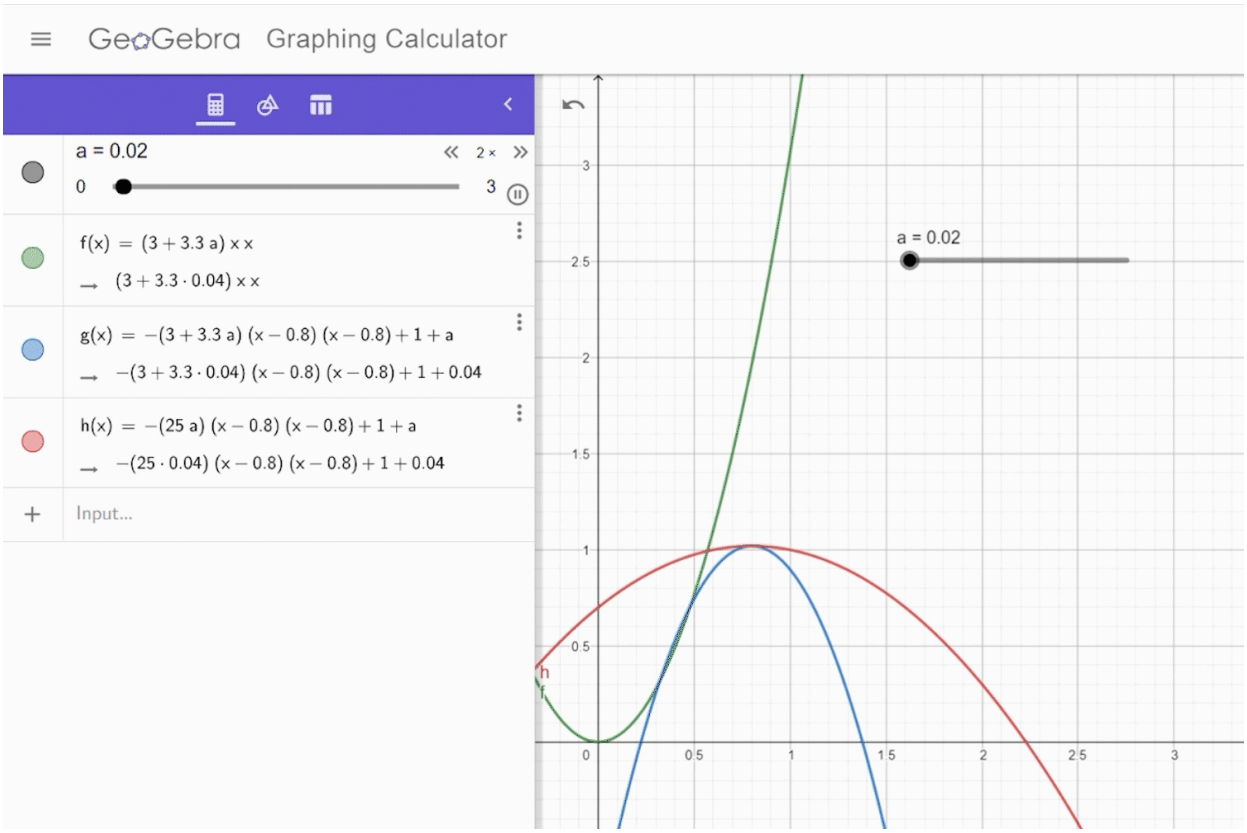
- **Integer: MaxTimeInMS** - Duration of the animation in milliseconds.
- **Integer: FrameRate** - Animation frame rate in FPSeconds.
- **Function: Function** - Call back animation with two or 3 arguments (Percents, X, ExtraData).
- **String: INFO_NAME** - Name of the animation (debug purpose).
- **Boolean: IsOtherArguments** - Define if callback function need extra data.
- **Class: FUNCTION_NEEDED_INFO** - Extra data needed by function.
- **Integer: BumpSize** - Size of the bump.(**CurveDetails** in **Animate()**)
- **Float: StartAt** - Used to force the of the animation at specific % (must be between 0-100) , can be null or not specified.

Description

Like other animation function LinearAnimation animation will calculate x position based on a advancement and sent back the info trough a callback function with arguments (Percent, X). The callback function just need to place items on there position based on X value.

The Bump Ease-In-Out is working in 3 steps:

1st f(x): This is the accelerating step (the curve is adjusting based on the bump size), it is used between X: 0-0.4.
2st g(x): This is the decelerating step (the curve is adjusting based on the bump size) but the curve it finishing at g(x) = Bump Size + 1, it is used between X: 0-0.8.
3st h(x): This is part is transition between h(x) = Bump Size +1 and 1.



Flip

The flip is simple way to flip an item, it create dynamically every block that is needed to work

Note: this class use animations functions: make sure you called the module part: Functional Animation before use it.

How to Initialize the Flip ?

The initialization of the flip animation is an one step process, just call the function **Init** of the class with his three arguments (**see bellow for more details**)

Class Details

Functions:

Name	Arguments	Return	Description
Init	• **String: Id** - HMTL ID defining the position in the DOM of the viewer • **String: Name** - Define the name of the flip all flip component will be defined by this name (this is to avoid mix if you use different flip so be sure to use different name for each flip). • **Integer : Speed** - Flipping animation time in MS.	**VOID**	This function Initialized every part of the flip: HMTL and the base CSS. It will generate in the defined ID as ID and all HTML IDs used for this timeline will start by the name defined in arguments
AddData	• **State: To** - Define which side you want to add data (None is not valid) • **String: Data** - New HTML Data to add to the side.	**VOID**	Call this function to add new data in one of the two sides.
ChangeCSS	• **State: To** - Define which side you want to add data (Call None to change the CSS of the main block). • **Class: CSS** - CSS to add or modify	**VOID**	Call this function to change the CSS of the flip.
ChangeChildrenCSS	• **State: To** - Define which side you want to add data (Call None to change the CSS of the main block). • **String: ChildrenID** - Id of the children to modify his CSS • **Class: CSS** - CSS to add or modify	**VOID**	Is functions is used to modify the CSS of children in the sides.
flip	• **State: To (Optional)** - Define which side you want to go	**VOID**	This function is used to process the flipping animation, you can either define the side to go or just call the function without arguments and it will switch state Note: that you can't go back to state: NONE.

Enumerations:

Name	List	Description
modulesList	• NONE • FRONT • BACK	This is the different flipping sides.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
basedId	String	null	HTML Id of the viewer parent.
id	String	null	Global HMTL Id of the viewer (# + name).
name	String	null	Name of the viewer used by the class for all HTML IDs
animationSpeed	Integer	null	Define the flipping animation speed
activeSide	State	NONE	Hold the active side of the flipping

Functions:

Name	Arguments	Return	Description
FlipAnimation (Animation Function)	• **Integer: Percent** - Advancement of the animation • **Integer: X** - X based on the advancement • **Class: data** - Data that the function need	**None**	This function is called to update the flipping animation.
GetFrontID	• **None**	**String** - HTML ID of the Front div	Function used to get the HTML ID of the front div (used to short the code).
GetBackID	• **None**	**String** - HTML ID of the Back div	Function used to get the HTML ID of the back div (used to short the code).

Compatibility

LANGUAGE CLASS: Not compatible.

Animated Gradient

Animated Gradient create a constantly moving background gradient.

How to Initialize the Animated Gradient ?

The initialization of the flip animation is an one step process, just call the function **Init** of the class with its one argument (**see bellow for more details**)

Class Details

Functions:

Initialization and main functions:

Name	Arguments	Return	Description
Init	• **String: Id** - Target DOM ID.	**VOID**	This function Initialized id member; Note: EE function: **AddGradient()** will always Initialize the class before return it.
Launch	• **Integer: Time** - Time for the animation to do a complete rotation. • **FPS: Time** - Frame per seconds	**VOID**	This function will create the Interval loop for the animation by calculating time ratio and shift for each frame, this function can stop prematurely if some condition are not filled. Important: Make sure you use **Setters Functions** first.
Stop	**NONE**	**VOID**	Call to stop the animation.

Setters functions:

Name	Arguments	Return	Description
AddColor	• **Color: Color** - Color to add at position • **Integer: Position** - position of the new color, in percent.	**VOID**	This function Initialized id member;
SetLastFirstDistance	• **Integer: Distance** - Distance in percent	**VOID**	Used this function to specify the distance between the last color (at 100%) and the first (at 0%).

SetCallback	• **Function**: **Distance** - Distance in percent	**VOID**	Add callback called at each frames.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
id	String	null	Target DOM ID.
colors	Class []: • **String**: col • **Integer** : pos	[]	List of colors and position used by the **Update** function.
shift	Float	0	Hold the gradient shift animation calculated by the **launch** function.
LastFirstDist	Integer	50	Distance it take to reach the first color from the last, in %.
Animation	Integer	null	Hold JQuery Interval id, to cancel it when stop function is called.
CallbackFunc	Function	null	Hold the callback function called each frame.

Functions:

Name	Arguments	Return	Description
Update	• **None**	**None**	This function will increment and update the gradient.

Compatibility

LANGUAGE CLASS: Not compatible.

Array

Array

The Array module has been made to handle different kind of array, it offer a main function that allow the creation of different type of array by guessing your need.

Module Details

Variables (Members):

Name	Type	Default Value	Description
ArrayClass	• Grid2D • Map2D	• null • null	Array holding all the classes use by this module.

Functions:

Name	Arguments	Return	Description
CreateArray	• int: **dimensionNumber** - The number of dimension needed for the array. • Boolean: **NeedNegatives** - Set as true if you need an array handling negatives coordinate.	**An ExoArray Class**	This function create a new Array based on some arguments it will choose the best of our arrays based on your need.

Grid2D

The Grid2D is a two dimensional dynamics array.
The class a one dimensional array to contain all data and act as a 2D array for its users.

Class Details

Variables (Members):

Name	Type	Default Value	Description

xLength	Integer	0	Define the actual x length of the array (horizontal).
yLength	Integer	0	Define the actual y length of the array (vertical).

Functions:

Name	Arguments	Return	Description
Add	• **Class: Data - data you want to store in the array** • **Integer: PosX - horizontal position of the new data** • **Integer: PosY - vertical position of the new data** • **Boolean: Override - define if the new data can override an existing data if there is already something at designated position**	**VOID**	This function is used to store a new data inside the dynamics array. By going out of scope the class will immediately "extends the array" (just update information). !! You can't use the [][] for this array !!
Clear	**None**	**VOID**	Empty the whole array and information
Get	• **Integer: X - X (horizontal) position of the data** • **Integer: Y - Y (vertical) position of the data**	**Class**	Get the data at designated position
GetAll	**None**	**Array (Array (Class))**	Get the entire array in form of a 2 dimensional array (javascript form), it add the null values.
GetColumn	• **Integer: X - Column number to extract** • **Boolean: WithNull - Define if the output array contain or not the empty cases**	**Array (class)**	Get an entire column of data with or without the empty cases
GetLine	• **Integer: Y - Line number to extract** • **Boolean: WithNull - Define if the output array contain or not the empty cases**	**Array (class)**	Get an entire line of data with or without the empty cases
Move	• **Integer: X - original X (horizontal) position of the data** • **Integer: Y - original Y (vertical) position of the data** • **Integer: ToX - new X**	**VOID**	Move a data from a position to another. Note, like add and remove the array will adjust it self.

	(horizontal) position of the data • **Integer: ToY - new Y (vertical) position of the data** • **Boolean: Override - define if the moved data can override an existing data if there is already something at designated position**		
Remove	• **Integer: PosX - horizontal position of the data to remove** • **Integer: PosY - vertical position of the data to remove**	**VOID**	Call this function to delete a data at designated position. If the removing let empty space and the array can reduce his size it will update length info automatically.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
Data	• *(Empty Array)*	• N/A	Array holding all the data. This is a one dimensional array. All the data are in a binding class: GridObject !! Is strongly advise for use to not touching it and use class property instead !!
lastID	*Integer*	0	Hold the last ID used for one of the data (see GridObject for more info about id).

Functions:

Name	Arguments	Return	Description
UpdateInfo	**None**	**VOID**	This function will go through the array **Data** to update all info about the Array.

GridObject

GridObject is a private binder used by EE array classes to bind the user data and data used by array to insure the proper working of the class.

Private Details

/!\/!\/!\/!\ All private details are usable but it is highly recommended to do not use any of those. It might make the class unstable. /!\/!\/!\/!\

Variables (Members):

Name	Type	Default Value	Description
ID	Integer	null	For now the ID is non pertinent data.
X	Integer	null	Define the X (horizontal) position of the data.
Y	Integer	null	Define the Y (vertical) position of the data.
objRef	Class	null	Object of the user

Map2D

The Map2D is a two dimensional dynamics array which use negative coordinate.
The class a one dimensional array to contain all data and act as a 2D array for its users.

Class Details

Variables (Members):

Name	Type	Default Value	Description
xMin	Integer	0	Define the minimum x (Negative length).
xMax	Integer	0	Define the maximum x (Positive length).
yMin	Integer	0	Define the minimum y (Negative length).
yMax	Integer	0	Define the maximum y (Positive length).

Functions:

Name	Arguments	Return	Description
Add	• **Class: Data - data you want to store in**	**VOID**	This function is used to store a new data inside the dynamics array. By going out of

	the array • **Integer: PosX - horizontal position of the new data** • **Integer: PosY - vertical position of the new data** • **Boolean: Override - define if the new data can override an existing data if there is already something at designated position**		scope the class will immediately "extends the array" (just update information). !! You can't use the [][] for this array !!
Clear	**None**	**VOID**	Empty the whole array and information
Get	• **Integer: X - X (horizontal) position of the data** • **Integer: Y - Y (vertical) position of the data**	**Class**	Get the data at designated position
GetAll	**None**	**Array (Array (Class))**	Get the entire array in form of a 2 dimensional array (javascript form), it add the null values.
GetColumn	• **Integer: X - Column number to extract** • **Boolean: WithNull - Define if the output array contain or not the empty cases**	**Array (class)**	Get an entire column of data with or without the empty cases
GetLine	• **Integer: Y - Line number to extract** • **Boolean: WithNull - Define if the output array contain or not the empty cases**	**Array (class)**	Get an entire line of data with or without the empty cases
Move	• **Integer: X - original X (horizontal) position of the data** • **Integer: Y - original Y (vertical) position of the data** • **Integer: ToX - new X (horizontal) position of the data** • **Integer: ToY - new Y (vertical) position of the data** • **Boolean: Override - define if the moved data can override an existing data if there is already something at designated position**	**VOID**	Move a data from a position to another. Note, like add and remove the array will adjust it self.

Remove	• **Integer: PosX - horizontal position of the data to remove** • **Integer: PosY - vertical position of the data to remove**	**VOID**	Call this function to delete a data at designated position. If the removing let empty space and the array can reduce his size it will update length info automatically.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
Data	• *(Empty Array)*	• N/A	Array holding all the data. This is a one dimensional array. All the data are in a binding class: GridObject !! Is strongly advise for use to not touching it and use class property instead !!
lastID	*Integer*	0	Hold the last ID used for one of the data (see GridObject for more info about id).

Functions:

Name	Arguments	Return	Description
UpdateInfo	**None**	**VOID**	This function will go through the array **Data** to update all info about the Array.

GridObject

GridObject

GridObject is a private binder used by EE array classes to bind the user data and data used by array to insure the proper working of the class.

Private Details

/!\/!\/!\/!\ All private details are usable but it is highly recommended to do not use any of those. It might

make the class unstable. /!\/!\/!\/!\

Variables (Members):

Name	Type	Default Value	Description
ID	Integer	null	For now the ID is non pertinent data.
X	Integer	null	Define the X (horizontal) position of the data.
Y	Integer	null	Define the Y (vertical) position of the data.
objRef	Class	null	Object of the user

Math

Math

The math module has been create to help with common mathematical problems.

Module Details

Variables (Members):

Name	Type	Default Value	Description
Random	• **Randomizer**	• null	Array holding the Randoms functions of the module
Trigonometry	• **CoordsFromAngleRadius** • **AngleRadiusFromCoordsAndObject** • **AngleFromCoords**	• null • null • null	Array holding the Trigonometry functions of the module

Functions:

Random:

Name	Arguments	Return	Description
Randomizer	• **Integer: min** - Minimum number to generate • **Integer: max** - Maximum number to generate	**VOID**	Generate a Random between a range

Trigonometry:

Name	Arguments	Return	Description
AngleRadiusFromCoordsAndObject	• **Integer: min** - Minimum number to generate • **Integer: max** - Maximum number to generate • **JQuery DOM Object: Object** - Reference object to define position	**Integer**	Get the angle based on the co-ordnance ON a object.
CoordsFromAngleRadius	• **Integer: angle** - angles you want to use for your calculation • **Integer: radius** - Position from the center	**Class:** • **Integer: x** - X co-ordnance • **Integer: y** - Y co-ordnance	Simple way to do the cosinus (x) and sinus (y) of the angle.
AngleFromCoords	• **Float: X** - X Position [-1,1] (more if outside defined circle). • **Float: Y** - Y Position [-1,1] (more if outside defined circle). • **Float: Radius** - Radius reference	**Float**	Convert coordinate to an angle (it consider radius use 1 for default radius).

ExoGameEngine Class

Description

This page detail everything about the EGE class.

We recommend to first check our tutorial before using our systems.

Class Details

Variables (Members):

Main (Important information):

Name	Type	Default Value	Description
id	String	None	Store where in DOM the EGE is set.
overlayList	overlays[]	[]	The list of all the overlays in used
isFullscreenSet	Boolean	false	Is holding the information if the fullscreen button has been set or not.
isFullscreen	Boolean	false	True when the app is fullscreen.

Status (Modules loading status):

Name	Type	Default Value	Description
GUI_menuOverlayStatus	Private Boolean	false	Status of the module: "**MenuOverlay**" from **GUI Group** (Check for loaded or not).
GUI_linkerStatus	Private Boolean	false	Status of the module: "**Other: Linker**" from **GUI Group** (Check for loaded or not).
GUI_widgetStatus	Private Boolean	false	Status of the module: "**Widgets: Widget**" from **GUI Group** (Check for loaded or not).
GUI_widgetButtonStatus	Private Boolean	false	Status of the module: "**Widgets: Widget Button**" from **GUI Group** (Check for loaded or not).
GUI_widgetTextInputStatus	Private Boolean	false	Status of the module: "**Widgets: Widget Text Input**" from **GUI Group** (Check for loaded or not).
GUI_widgetTextStatus	Private Boolean	false	Status of the module: "**Widgets: Widget Text**" from **GUI Group** (Check for loaded or not).
GUI_widgetSelectStatus	Private Boolean	false	Status of the module: "**Widgets: Widget Select**" from **GUI Group** (Check for loaded or not).
GUI_widgetLayoutStatus	Private Boolean	false	Status of the module: "**Widgets-Layout: Layout**" from **GUI Group** (Check for loaded or not).
GUI_popUpStatus	Private Boolean	false	Status of the module: "**PopUp: PopUp**" from **GUI Group** (Check for loaded or not).

LOADING_loadingStatus	Private Boolean	false	Status of the module: "**Loading: Loading**" from **Loading Group** (Check for loaded or not).
WORLD_layeredWorldStatus	Private Boolean	false	Status of the module: "**LayeredWorld**" from **World Group** (Check for loaded or not).

Inner Classes:

Name	Status	Description
Classes	Holder	Unlike EE, classes are not hold in module but in this class.
GUI	Group of modules	This group contain all graphical modules.
World	Group of modules	This group contain all worlds modules.

Functions:

Main Methods:

Name	Arguments	Return	Description
Init	• **String: LocationID** - Where the viewport is located (See tutorial). • **EGEModulesList []: ModulesList** - Unlike the EE you cannot call module during the EGE use. This is where you call them, all the modules you need during with **EGE** as to be entered here. • **Function: Callback** - This is the function called when everything is set (The function can take EGE as first arguments).	**VOID**	This will initialize the **EGE** class by setting up basics structures and calling all asked modules. **Note: this function is only called by EE with LaunchGE**
Launch	• **None**	**VOID**	You must called this function after doing all initialization of everything you need to launch all the systems.
Call	• **EGEModulesList: Module** - the EGEModulesList item you want to call (open) • **Function: Callback**: the call function which is call went the files are loaded	**VOID**	This function is called when you need to setup a module. It use a callback function to continue the process after the modules has been loaded (because the loading in asynchronous). **Note: this function is only called by EE during EGE initialization**

Modules and Setters Methods:

Name	Arguments	Return	Description
AddOverAllLoading	• **None**	**Loading**	This function create a loading system that entirely cover the viewport (and it return it for modification and **Launch**).
SetFullscreen	• **Function: FullscreenCallback**: call when all loaded.	**VOID**	This function set the fullscreen button and it's images.

Util:

Name	Arguments	Return	Description
GetFullScreenButton	• **None**	String	Use this function to get the fullscreen button ID.

ExoGameEngine
(Initialization and information)

HOW TO INITIALIZE THE EGE.

Once you've installed the EE class, you can now initialize the ExoGameEngine.

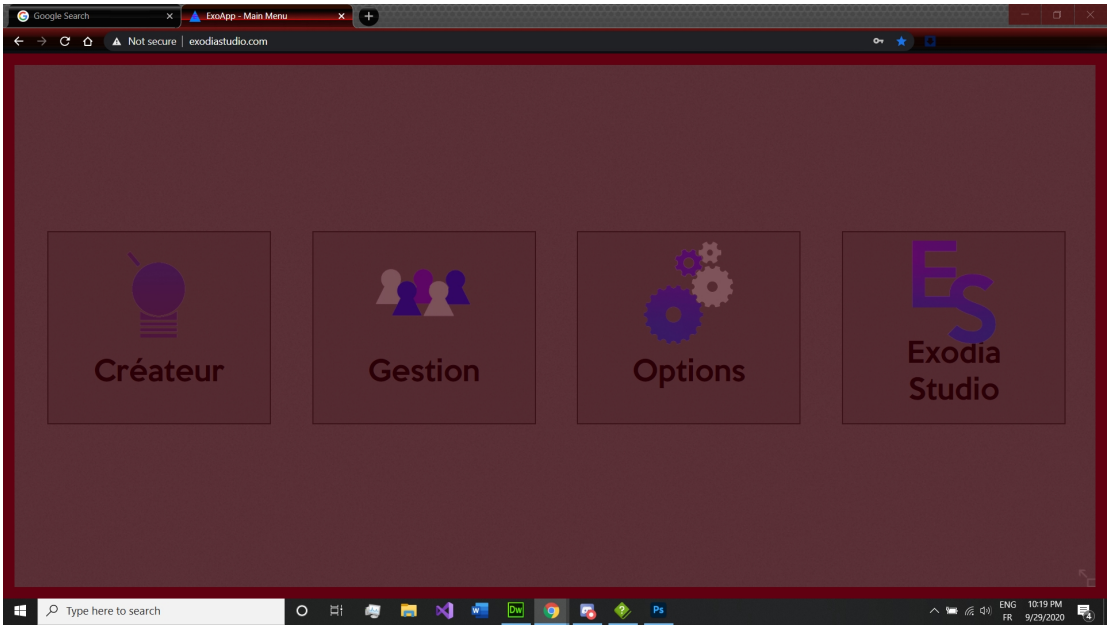
To do so, call the function **LaunchGE** from the EE with the 3 arguments:

- **String: LocationID** - Where the viewport is located (See details below).
- **EGEModulesList[]: ModulesList** - Unlike the EE you cannot call module during the EGE use. This is where you call them, all the modules you need during with **EGE** as to be entered here.
- **Function: Callback** - This is the function called when everything is set (The function can take EGE as first arguments).

Note:
- ❑ **If you using different EE versions make sure to remember on which version you've initialized it.**
- ❑ **Only one EGE can be initialize (per version).**

THE VIEWPORT

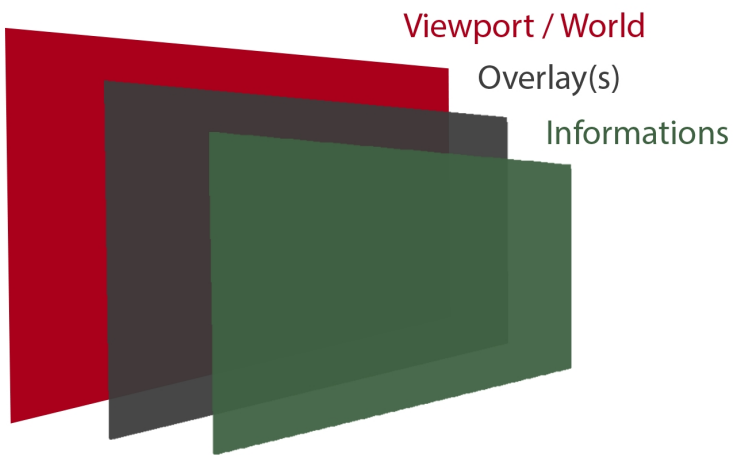
The EGE is used to handle games or apps, so unlike website the app will set in a predefined space (you can't scroll). This space is call the viewport, you can set as the size you want (the viewport will always take the size of the **LocationID**):



everything you will create with EGE will be set in this area with top left as 0 (x = → and y = ↓).

The viewport is also composed by different layers:

1. **The world**: Where all sprite are moving and living
2. **The overlays**: HUDs / GUIs or menu, its a flat 2D intractable graphical interface using widgets system.
3. **The information**: Extra information (Popup, Full screen button, etc..).



THE WORLD

The world is specifically created for Web-Video Game, if you want to create a Web-App let the world section empty and use only **Overlays**.

The world is where all the action going, as for everything else, just call the world module that you want to use during EGE initialization, then call a world creation function.

Note: only one world can be set.

Note: World classes are not finished yet.

THE OVERLAYS

The overlays can be used different way, it can can only cover some part of the viewport (like HUD, some menus, etc..) or it can cover the whole viewport. Use **IsFillingViewport** from the MenuOverlay class setup the type of your overlay (e.g. if you want HUD and you not set IsFillingViewport(false) then you will not be able to click on the world).

HOW TO USE

> The first is to add the OverlayModule in the list during EGE initialization, then call the overlay creation function.

Now can use its all functionality, note that EGE classes use double step initialization, the **Init()** function first thing to call (Usually done creating functions) then when you set

everything you call the **Launch()** function to make it working.

For the second step you have to add widgets. Widgets are the best way to structure your overlay (don't hesitate to use them a lot), it will basically help by using grid system: each widgets depend of it's parent. They also offer a lot of different functionalities.

To add widgets call the function **AddWidget()** from the MenuOverlay or directly from the widgets widgets.

- For the final step all you have to do is call the **Launch()** function of the overlay to make it appear and working **(by calling this function it will automatically call all Launch() functions of its children and etc...)**.
  Note: if the overlay has been created before calling EGE **Launch() function, you don't need to call overlay it will be called automatically.**

THE LAUNCH

When your all set, you have to call a final function called EGE.**Launch()** it will launch all EGE functionality.
And now everything done.

A GOOD WAY TO USE EGE

Here you will find how we use the EGE system.

Demo

In this document you will how use the EGE system.

We use a multi-step thread that allow use to wait asynchronous process to make sure everything is loaded.
This example is take from the Do Not Answer (Alpha 0.0.1) - Main Menu.

Download file:

Initialization

The first thing we do is calling all module from EE than we going to need then we call the

EE.**LaunchGE()** function and as a callback out main thread (See bellow).

```
1. EE0_3.Call(EE0_3.modulesList.UTIL_LOADCHECK, function(){
2.    EE0_3.Call(EE0_3.modulesList.ARRAY_GRID2D, function(){ //Launching Game Engine

3.       EE0_3.LaunchGE("#CreatorDiv", [EE0_3.EGEModulesList.GUI_OVERLAY,
   EE0_3.EGEModulesList.GUI_WIDGET, EE0_3.EGEModulesList.GUI_WIDGET_BUTTON,
     EE0_3.EGEModulesList.GUI_WIDGET_TEXT, EE0_3.EGEModulesList.LOADING_LOADING],
   MainThread);
4.    });
5. });
```

Global Variable

We also use global variable to store global data:

```
1. //Main thread globla variable
2. menu_OverallLoading;
3. mainOverlay = null;
4.
5. //Widgets
6. menu_text = null;
7. menu_button = null;
```

Main Thread

Now we use multi-step thread function: we will call all functionality we want to add and each widgets we want to create to initialize the app, and each time we need something to load (A module, an image, etc..), as callback function we use the step function we an incremented step ID.

Step undefined or null

The first step is not 0 but nothing (*undefined*) but it is working like any other step:

```
1. function MainThread(EGE, Step){
2.
3.         if(Step == undefined || Step == null){
4.
5.               //Creating loading
6.               creator_OverallLoading = EGE.AddOverAllLoading();
7.               creator_OverallLoading.AddLoadingBar(function()
   { MainThread(EGE, 0); });
8.         }
```

First thing we do here is to create a loading system it will show a loading bar over

everything (so the EGE can work behind and also show the progress to the user). We store this last into a global variable then call **AddLoadingBar()**, this function has to load an EE module so we wait the loading is over by send an lambda function with our main thread function and as parameter the EGE ref and the next index which is 0.

Step 0

```
1.  if(Step == 0){
2.
3.      //Creating loading
4.      creator_OverallLoading.SetMaxStep(4);
5.
6.      creator_OverallLoading.SetLoadingBackColor("#000000");
7.      creator_OverallLoading.SetLoadingFrontColor("#ffffff");
8.
9.      creator_OverallLoading.SetBackgroundHex("#7f7f7f");
10.
11.     creator_OverallLoading.Launch(function(){ MainThread(EGE, 1); });
12. }
```

In this step we set details about our loading system like how step will be used (See **SetMaxStep** and **Step** functions), the bar color, the background color and then we **Launch** the loading, it will now appear and be ready to use, of as you might noticed I've called the thread function with index 1.

Note: the loading system is an information display, it goes over everything.

Step 1

```
1.  if(Step == 1){
2.
3.      //Setting up GameEngine
4.      EGE.SetFullscreen(function(){ MainThread(EGE, 2); });
5.  }
```

This step is a way more lighter, we just the fullscreen button on, but since it has to wait images to load we pass the thread again with index 2.

Step 2

```
1.  if(Step == 2){
2.
3.      creator_OverallLoading.Step();
4.
5.      //Creating main overlay
```

```
6.      mainOverlay = EGE.GUI.AddNewMenuOverlay("mainOverlay");
7.
8.      //Setting Up Main Overlay
9.      mainOverlay.IsFillingViewport(true);
10.     mainOverlay.SetBackgroundImage("Src/BG/BGI_MenuBackground.png",
    function(){ MainThread(EGE, 3); });
11. }
```

First we add step in the loading system to make the bar moving. Then we create our main overlay named: "mainOverlay" so every children IDs will start by mainOverlay_.

Then we start calling setters for our mainOverlay the important IsFillingViewport and we set the background image and wait for it to load.

Step 3

The step is a long step in our initialization and the final one, for easier explanation I've separated in two:

```
1.  if(Step == 3){
2.
3.      //Creating main overlay
4.      creator_OverallLoading.Step();
5.
6.      //Adding text
7.      menu_text = mainOverlay.AddWidget(EGE.GUI.WidgetType.WIDGET_TEXT,
    "Menu_Text").SetSizePosition("100%", "4vw", 0, "calc(50% -
    6vw)").AddText("Welcom in <span style=\"text-decoration: underline;\">Do Not
    Answer</span> for the moment only the creator is available.").TextAlign("Centered",
    "Centered").SetFontSize("2vw");
8.      creator_OverallLoading.Step();
9.
10.     //Adding button
11.     menu_button =
    mainOverlay.AddWidget(EGE.GUI.WidgetType.WIDGET_BUTTON,
    "Menu_Button").SetSizePosition("10vw", "4vw", "calc(50% - 5vw)", "calc(50% +
    2vw)").SetBackgroundColor([0,0,0,0], [0,0,0,0.5], [0,0,0,0.3]).SetBorder("All", "solid",
    4, "Black", 50, "All").SetOnClicked(function(){window.location.href =
    "Game/Creator";});
12.
13.     menu_button.AddWidget(EGE.GUI.WidgetType.WIDGET_TEXT,
    "Menu_Text").SetSizePosition("100%", "100%", 0,
    0).AddText("Creator").TextAlign("Centered", "Centered");
14.     creator_OverallLoading.Step();
```

First we add a step in the loading system (for step 2). And now we create our widgets that we going to store in our global variables (Remember **addWidget@** will always send back the created widgets).

For all widgets created will call all required setters, we also create a sub widgets text for the button.
And finally add another step to the loading system.

Note: almost all widgets setters are returning themselves to you can call a function on the same line like above.

Now the second part:

```
1.      //Launching
2.      EGE.Launch();
3.      creator_OverallLoading.Step();
4.  }
```

This is an important part, here we **Launch** the EGE and add one more step to the loading system, because if you set it properly the loading system will automatically close himself.

Classes

The Classes class is the EGE class holder, when a module is called the class from the module will be stored here.

Variables (Members):

Name	Type	Default Value	Description
Loading	Class: • Loading	• null	Holder all class from Loading group
GUI	Class: • MenuOverlay • Linker • Widget • Widget_Button • Widget_TextInput • Widget_Text • Widget_Select	• null • null • null • null • null • null • null • null • null	Holder all class from GUI group

	• Widget_Layout • PopUp		
World	Class: • LayeredWorld	• null	Holder all class from World group

Groups

The EGE contain only groups unlike EE, Groups and Modules are note callable by using group.module. Refer to EGE file to know access functions or classes or directly in groups details (here groups are full classes with methods - like modules).

For now there is 3 groups:

- **LOADING**: Group holding different loading systems.
- **GUI**: Group of graphical classes
- **World**: Group that contains all world types.

GUI

The GUI groups contain all graphical interfaces usable for menus, HUD and other.

Module Details

Enumerations:

Name	List	Description
WidgetType	• WIDGET • WIDGET_BUTTON • WIDGET_TEXTINPUT • WIDGET_TEXT • WIDGET_SELECT	This list is used by the **GetWidget()** function to define which widget we require.
LayoutType	• LAYOUT_VERTICAL_GRID • LAYOUT_HORIZONTAL_GRID	This list is used by Layouts to define which type is as to be.

Functions:

Name	Arguments	Return	Description
AddNewMenu Overlay	• **String: RegistryName** - This is the name of this overlay and must be unique, all the structure will have id starting by the RegistryName	**MenuOverlay**	Function used to create and initialize a new **MenuOverlay**. Does not launch it.
GetWidget **(Private)**	• **WidgetType: Type** - type of the require widget	**Widgets**	This function is used to get one of the **Widgets** class. **Note: function called by MenuOverlay and Widgets, this function is useless outside.**

Widgets

Widgets are the main components of the Overlays there is 5 different widgets. The Widget class is the main class all the others widgets are inheriting from it.

- **Widget**: Main widget.
- **Widget Button**: Widget composed by a button.
- **Widget Select**: Widgets offering a select box.
- **Widget Text**: Widgets for text display
- **Widget Text Input**: Widgets for text input.

Layout

Layout are used to simplify the placement of widgets by using a grid system and modifying children position automatically.

How to Initialize any Layout?

To use layouts you have to call it from the function **AddLayout** from Widgets.

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **Widgets: Owner** - Which widget will use the layout. • **LayoutType: Type** - Type of the layout.	**VOID**	This function setter main variable
Launch	• **None**	**VOID**	Launch will automatically call the **Update** function to make sure every children widgets are properly displayed.

Setters Functions

Name	Arguments	Return	Description
SetWidth	• **String: Width** - new width for each children.	**None**	Set the width of each children widgets. By using this you forcing the placement and might be overflow, if you're not using it will it evenly separate children from owner size.
SetHeight	• **String: Height** - new width for each children.	**None**	Set the height of each children widgets. By using this you forcing the placement and might be overflow, if you're not using it will it evenly separate children from owner size.
SetVerticalSpace	• **String: Size** - new width for each children.	**None**	Set the vertical space of each children widgets. By using this you forcing the placement and might be overflow, if you're not using it will it evenly separate children from owner size.
SetHorizontalSpace	• **String: Size** - new width for each children.	**None**	Set the horizontal space of each children widgets. By using this you forcing the placement and might be overflow, if you're not using it will it evenly separate children from owner size.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default	Description

		Value	
owner	Widgets	undefined	Owner of this layout.
type	LayoutType	undefined	Hold the layout type.
data	Class	{}	Class build by setters to hold information.

Functions:

Update Functions

Name	Arguments	Return	Description
Update	• **None**	**None**	This function will recalculate all size and position for all children widgets.

Compatibility

LANGUAGE CLASS: Not compatible.

Widget

The widget is most basic of all it is mostly used for layout purpose but it is also the parent class of all others widgets that use almost the same functions.

How to Initialize any Widgets?

To add a widget call AddWidget function from GUIs members that own this function.

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: BaseID** - HMTL ID defining the position in the DOM of the tab. (Automatically set by AddWidgets). • **String: RegistryName** - Name that define this overlay, all children IDs will start by this.	**VOID**	This function Initialized every part of the viewer (if **DoNotBuildDOM** is not false): HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this will start by the RegistryName defined in arguments.

	• Boolean: **DoNotBuildDOM** - Used by other widgets when creating the parent (if the children is creating it's own div).		
Launch	• **None**	**VOID**	Launch all systems required by the this and will call all children launch.

Setters Functions

Name	Arguments	Return	Description
AddLayout	• LayoutType: **Type** - Define which layout you want.	**Layout**	Add add a Layout to this widgets. **Note: By using layout avoid to resize widgets children.**
AddWidget	• WidgetType: **Type** - Define which widget you want. • String: **Name** - Name of this widgets.	**Widgets**	Will add a selected children widgets in itself and return it. **Note: The children will always they're IDs starting by the parent name, so time it's not in the same family you can add an identical name.**
Destroy	• **None**	**None**	Remove the widget from the DOM.
Clear	• **None**	**None**	Clear all widget data.

Design functions

Positions and general design:

Name	Arguments	Return	Description
Visible	• Boolean: **IsVisible** - Set you want it to appear or disappear. • Integer: **Fade** - Fading time in MS • Function: **VisCallback** - Callback function called when fade is over.	**This**	Will fade this widget, if no fade speed is set it will be instant.
SetSizePosition	• Integer - @CSS String: **Width** - of this widget (based on parent). • Integer - @CSS String: **Height** - of this widget (based on parent). • Integer - @CSS String: **X** -	**This**	Set the size and the position of this widget based on the parent widget.

	(horizontal) of this widget (based on parent). • **Integer - @CSS String: Y** - (vertical) of this widget (based on parent). • **Boolean: FromRight** - Set if x has to start from right • **Boolean: FromBottom** - Set if x has to start from bottom		
GetPosition	• **@CSS String: Where** - position in CSS	**@CSS String**	Get the position at specific location (top, left, ...).
GetSize	• **None**	**Class:** • **w: width** • **h: height**	Return the actual size of the widget.
SetPadding	• **String: At** - Where to add the padding (Top, Left, Bottom, Right or All). • **Integer - @CSS String: Value** - Size of the padding	**This**	Add a padding to the widget.
SetMargin	• **String: At** - Where to add the padding (Top, Left, Bottom, Right or All). • **Integer - @CSS String: Value** - Size of the margin	**This**	Add a margin to the widget.
AddVerticalOverflow	• **@CSS String: State** - CSS overflow state	**This**	Set the Vertical overflow of the widget.
AddHorizontalOverflow	• **@CSS String: State** - CSS overflow state	**This**	Set the Horizontal overflow of the widget.
SetRotation	• **Integer: Deg** - Degrees of rotation	**This**	Use the CSS "transform" property to rotate the widget.

Style:

Name	Arguments	Return	Description
SetBackgroundColor	• **Color: BG** - Background Color	**This**	Change the background color.
SetBackgroundImage	• **String: URL** - Image link • **Function: SBICallback** - Callback when the image has been loaded. • **Boolean: Debug** - Set if you need a debug when the image has been loaded.	**VOID**	Set a background image.

SetBGImageRepeat	• @Nullable @CSS String: **Vertical** - repeat CSS property for vertical. • @Nullable @CSS String: **Vertical** - repeat CSS property for Horizontal.	This	Set the background repeat properties, "no-repeat" is set when if *null*
SetBGImageSizeAndPosition	• **Integer** - @CSS String: **Size** - CSS background size. • **Integer** - @CSS String: **X** - CSS position • **Integer** - @CSS String: **Y** - CSS position	This	Change setting for the background image.
SetOpacity	• Float: **Opacity** - Opacity value [0,1]	**None**	Change widget opacity.

Contour:

Name	Arguments	Return	Description
SetBorder	• String: **At** - Define which border you want or enter "All" (Can can add more than one - it always start by a capital). • @CSS String: **Type** - CSS border type. • **Integer** - @CSS String: **Thickness** - CSS border size. • Color: **Color** - Border Color • **Integer** - @CSS String - Class: **Thickness** - Set the border radius (integer will be same "px" for all). To use the class just send member like: class.top_left.	This	Set a border and also set the border radius.
RemoveOutline	• None	This	Remove the default outline used by browsers.

Font:

Name	Arguments	Return	Description
SetFontSize	• **Integer** - @CSS **String** : **Value** - Set the font size based on CSS values.	**This**	Change the font size.

Interaction:

Name	Arguments	Return	Description
ForceInteraction	• **None**	**This**	When you defined the overlay as not filling the viewport it will no more obstruct the mouse events and widget will not be interactive. But if you want a specific widget have it you can use this function.
AddHoveredEvent	• **Function**: **funcIn** - Function called when mouse enter • **Function**: **funcOut** - Function called when mouse leave	**VOID**	This function will set the events "mouseenter" and "mouseleave".
AddPressedEvent	• **Function**: **funcIn** - Function called when mouse is pressed • **Function**: **funcOut** - Function called when mouse is released	**VOID**	This function will set the events "mousedown" and "mouseup".
Movable	• **Boolean**: **LockHorizontal** • **Boolean**: **LockVertical**	**VOID**	If you've set the overlay as movable and you don't want a widget moving either one direction or all, use this function.
trigger	• **String** : **Trigger** - JQuery event name.	**VOID**	Force trigger a specific event.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default Value	Description
id	String	undefined	Global HMTL Id of the viewer (# + name).
registryName	String	undefined	Name of this widget.

basedId	String	undefined	Where it is located is the existing DOM (its parent).
widgetType	WidgetType	WIDGET@	Hold the type of this widget.
layout	Layout	null	Hold this widget layout (if created).

Information

Name	Type	Default Value	Description
position	Class	x: 0 and y: 0 and fromRight: false and fromBottom: false	Hold the actual position of the widget.
horizontalLock	Boolean	true	Used to check if movement is locked horizontally.
verticalLock	Boolean	true	Used to check if movement is locked vertically.
shiftX	Integer	0	Hold the actual shift X
shiftY	Integer	0	Hold the actual shift Y
backgroundImageSize	Integer - @CSS String	100%	Hold the background image size.
AllWidget	Widgets[]	[]	Hold all children Widgets.
lastZIndex	Integer	100	To make sure the Widgets are properly layered we decrement this variable each widgets.
intervals	Integer[]	[]	Hold all animation IDs.

Functions:

Interaction Functions

Name	Arguments	Return	Description
Shift	• **Integer - @CSS String: ShiftX** • **Integer - @CSS String: ShiftY**	**None**	Used by parent widget or overlay for moving based on a general shift.

Compatibility

LANGUAGE CLASS: Not compatible.

Widget_Button

The widget_text is a special module use to show text dynamically.

How to Initialize any Widgets?

To add a widgets call AddWidget function from GUIs members that own this function.

Class Details

Main Functions

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: BaseID** - HMTL ID defining the position in the DOM of the tab. (Automatically set by AddWidgets). • **String: RegistryName** - Name that define this overlay, all children IDs will start by this.	**VOID**	This function Initialized every part of the viewer : HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this will start by the RegistryName defined in arguments. It will also call the parent **Init()** and finish by adding event with the function: **InitEvents**.
InitEvents	• **None**	**VOID**	Initialize all button events.
.Launch	• **None**	**VOID**	Launch all systems required by the this and will call all children launch. It will also call the parent **Launch()**.

Button Functions

Name	Arguments	Return	Description
SetOnClicked	• **Function: FuncIn** - Function called when pressed • **Class: Args** - Arguments required by **FuncIn**	**THIS**	Setup the function called when the button is pressed.

Overrides / Inherited Functions:

Setters Functions

Name	Arguments	Return	Description

AddLayout	• LayoutType@: **Type** - Define which layout you want.	**Layout**	Same as parent.
AddWidget	• WidgetType@: **Type** - Define which widget you want. • String: **Name** - Name of this widgets.	**Widgets**	Same as parent.
Destroy	• **None**	**None**	Same as parent.
Clear	• **None**	**None**	Same as parent.

Design functions

Positions and general design:

Name	Arguments	Return	Description
Visible	• Boolean: **IsVisible** - Set you want it to appear or disappear. • Integer: **Fade** - Fading time in MS • Function: **VisCallback** - Callback function called when fade is over.	This	Same as parent.
SetSizePosition	• Integer - @CSS String: **Width** - of this widget (based on parent). • Integer - @CSS String: **Height** - of this widget (based on parent). • Integer - @CSS String: **X** - (horizontal) of this widget (based on parent). • Integer - @CSS String: **Y** - (vertical) of this widget (based on parent). • Boolean: **FromRight** - Set if x has to start from right • Boolean: **FromBottom** - Set if x has to start from bottom	This	Same as parent.
GetPosition	• @CSS String: **Where** - position in CSS	@CSS String	Same as parent.
GetSize	• **None**	Class: • w: **width** • h: **height**	Same as parent.
SetPadding	• String: **At** - Where to add the padding (Top, Left, Bottom, Right or All).	This	Same as parent.

	• **Integer - @CSS String: Value** - Size of the padding		
SetMargin	• **String: At** - Where to add the padding (Top, Left, Bottom, Right or All). • **Integer - @CSS String: Value** - Size of the margin	**This**	Same as parent.
AddVerticalOverflow	• **@CSS String: State** - CSS overflow state	**This**	Same as parent.
AddHorizontalOverflow	• **@CSS String: State** - CSS overflow state	**This**	Same as parent.
SetRotation	• **Integer: Deg** - Degrees of rotation	**This**	Same as parent.

Style:

Name	Arguments	Return	Description
SetBackgroundColor	• **Color: Normal** - Background color when normal • **Color: Hovered** - Background color when hovered • **Color: Pressed** - Background color when pressed	**This**	Same as parent but in 3 step to set all 3 button conditions
SetBackgroundImage	• **String: NormalURL** - Image link when normal • **@Nullable String: HoveredURL** - Image link when hovered • **@Nullable String: PressedURL** - Image link when pressed • **Function: SBICallback** - Callback when the image has been loaded. • **Boolean: Debug** - Set if you need a debug when the image has been loaded.	**VOID**	Same as parent but in 3 step to set all 3 button conditions
SetBGImageRepeat	• **@Nullable @CSS String: Vertical** - repeat CSS property for vertical. • **@Nullable @CSS String: Vertical** - repeat CSS property	**This**	Same as parent.

	for Horizontal.		
SetBGImageSizeAndPosition	• **Integer - @CSS String: Size** - CSS background size. • **Integer - @CSS String: X** - CSS position • **Integer - @CSS String: Y** - CSS position	This	Same as parent.
SetOpacity	• **Float: NormalOpacity** - Opacity value [0,1] • **Float: HoveredOpacity** - Opacity value [0,1] • **Float: PressedOpacity** - Opacity value [0,1]	This	Same as parent but in 3 step to set all 3 button conditions

Contour:

Name	Arguments	Return	Description
SetBorder	• **String: At** - Define which border you want or enter "All" (Can can add more than one - it always start by a capital). • **@CSS String: Type** - CSS border type. • **Integer - @CSS String: Thickness** - CSS border size. • **Color: Color** - Border Color • **Integer - @CSS String - Class: Thickness** - Set the border radius (integer will be same "px" for all). To use the class just send member like: class.top_left. • **String: On** - Setup the button condition ("Normal", "Hovered", "Pressed" or "All")	This	Same as parent but you can choose the condition you apply to
RemoveOutline	• **None**	This	Same as parent.

Font:

Name	Arguments	Return	Description
SetFontSize	• **Integer** - @CSS **String** : **Value** - Set the font size based on CSS values.	**This**	Same as parent.

Interaction:

Name	Arguments	Return	Description
ForceInteraction	• **None**	**This**	Same as parent.
AddHoveredEvent	• **Function**: **funcIn** - Function called when mouse enter • **Function**: **funcOut** - Function called when mouse leave	**VOID**	Same as parent.
AddPressedEvent	• **Function**: **funcIn** - Function called when mouse is pressed • **Function**: **funcOut** - Function called when mouse is released	**VOID**	Same as parent.
Movable	• **Boolean**: **LockHorizontal** • **Boolean**: **LockVertical**	**VOID**	Same as parent.
trigger	• **String** : **Trigger** - JQuery event name.	**VOID**	Same as parent.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default Value	Description

parent	Widget	undefined	Parent class of this widget
id	String	undefined	Global HMTL Id of the viewer (# + name).
registryName	String	undefined	Name of this widget.
basedId	String	undefined	Where it is located is the existing DOM (its parent).
widgetType	WidgetType	WIDGET	Hold the type of this widget.
layout	Layout	null	Hold this widget layout (if created).

Information

Name	Type	Default Value	Description
statusCss	Status[]	[Status, Status, Status]	Hold the condition Status for updating the CSS
onClicked	Function	null	Hold the on clicked function.
onClickedArgs	Lang	null	Hold the on clicked function arguments.

functions:

Name	Arguments	Return	Description
ChangeCSS	• **Integer: On** - Define which condition.	**None**	This function is called by button events to update all the CSS.

Overrides / Inherited functions:

Name	Arguments	Return	Description
Shift	• **Integer** - @CSS **String: ShiftX** • **Integer** - @CSS **String: ShiftY**	**None**	Same as parent.

Compatibility

LANGUAGE CLASS: Not compatible.

Status

Status is a private class used the widget_button to hold the CSS information at a specific condition.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Name	Type	Default Value	Description
image	String	null	Background image of the button
backgroundColor	String	null	Background color of the button
opacity	integer	1	Widget opacity.

borders	Class:
	• border
	• borderTop
	• borderRight
	• borderBottom
	• borderLeft
		Each:
		{
		"type": "none",
		"size": 0,
		"color": "black",
		"radius": {}
		}
			Hold all borders details

Widget_Text

The widget_text is a special module use to show text dynamically.

How to Initialize any Widgets?

To add a widgets call AddWidget function from GUIs members that own this function.

Class Details

Main Functions

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: BaseID** - HMTL ID defining the position in the DOM of the tab. (Automatically set by AddWidgets). • **String: RegistryName** - Name that define this overlay, all children IDs will start by this.	**VOID**	This function Initialized every part of the viewer : HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this will start by the RegistryName defined in arguments. It will also call the parent **Init()**.
Launch	• **None**	**VOID**	Launch all systems required by the this and will call all children launch. It will also call the parent **Launch()**.

Text Functions

Name	Arguments	Return	Description
AddLang	• **Lang: Lang** - Lang to assign with this widget.	**THIS**	Assign a Lang class to this widgets to make the function **SetText** usable.
SetText	• **String: TextID** - Text id is the JSON file • **String: TextID** - Text group is the JSON file • **String: TextID** - The default if not found	**THIS**	Set the text of this widgets (will not work if no Lang is set).
AddText	• **String: Text** - Text you want to add.	**THIS**	Add a text to the widget **Note: to use Lang class use SetText function.**
ClearText	• **NONE**	**THIS**	Will clear the text.
TextAlign	• **String: Horizontal** - Horizontal alignment ("Left", "Centered", "Right"). • **String: Vertical** - Vertical alignment ("Top", "Centered", "Botttom").	**THIS**	Change text alignment. **Note: when used before launch the functionality might not work properly don't hesitate to relaunch the text.**
Replace	• **String: Replace** - The text to replace • **String: To** - The text	**THIS**	Replace a specified text in the widget.

	change with.		

Font Functions

Name	Arguments	Return	Description
SetFontFamily	• **String: Name** - Font family name.	**THIS**	Change text font family.
SetFontColor	• **Color: Color** - Background Color	**THIS**	Change the font color.
SetBold	• **Boolean : Remove** - true if you want to remove it.	**THIS**	Add or remove the bold property of the text.
SetItalic	• **Boolean : Remove** - true if you want to remove it.	**THIS**	Add or remove the italic property of the text.
AddAnimatedGradient	• **Color[0] and Integer[+1]: Colors** - Each color take two place in array, the first is the color and the second the percent position. • **Integer: FinalDistance** - Distance between the last and the first color. • **Integer: Speed** - How many time in MS it take to make a full cycle. • **Integer: FPS** - frame rate of the animation. • **Function: AAGCallback** - Callback function when all set. • **Class: AAGArgs** - Callback function arguments	**THIS**	This function will create, launch, and apply to the text a AnimatedGradient but inside the letters.

Overrides / Inherited Functions:

Setters Functions

Name	Arguments	Return	Description
AddLayout	• **LayoutType: Type** - Define which layout you want.	**Layout**	Same as parent.
AddWidget	• **WidgetType: Type** - Define which widget you want. • **String: Name** - Name of this widgets.	**Widgets**	Same as parent.
Destroy	• **None**	**None**	Same as parent.
Clear	• **None**	**None**	Same as parent.

Design functions

Positions and general design:

Name	Arguments	Return	Description
Visible	• **Boolean: IsVisible** - Set you want it to appear or disappear. • **Integer: Fade** - Fading time in MS • **Function: VisCallback** - Callback function called when fade is over.	**This**	Same as parent.
SetSizePosition	• **Integer - @CSS String: Width** - of this widget (based on parent). • **Integer - @CSS String: Height** - of this widget (based on parent). • **Integer - @CSS String: X** - (horizontal) of this widget (based on parent). • **Integer - @CSS String: Y** - (vertical) of this widget (based on parent). • **Boolean: FromRight** - Set if x has to start from right • **Boolean: FromBottom** - Set if x has to start from bottom	**This**	Same as parent.

GetPosition	• @CSS String: **Where** - position in CSS	@CSS String	Same as parent.
GetSize	• **None**	**Class:** • **w: width** • **h: height**	Same as parent.
SetPadding	• **String: At** - Where to add the padding (Top, Left, Bottom, Right or All). • **Integer - @CSS String: Value** - Size of the padding	**This**	Same as parent.
SetMargin	• **String: At** - Where to add the padding (Top, Left, Bottom, Right or All). • **Integer - @CSS String: Value** - Size of the margin	**This**	Same as parent..
AddVerticalOverflow	• @CSS String: **State** - CSS overflow state	**This**	Same as parent.
AddHorizontalOverflow	• @CSS String: **State** - CSS overflow state	**This**	Same as parent.
SetRotation	• **Integer: Deg** - Degrees of rotation	**This**	Same as parent.

Style:

Name	Arguments	Return	Description
SetBackgroundColor	• **Color: BG** - Background Color	**This**	Same as parent.
SetBackgroundImage	• **String: URL** - Image link • **Function: SBICallback** - Callback when the image has been loaded. • **Boolean: Debug** - Set if you need a debug when the image has been loaded.	**VOID**	Same as parent.
SetBGImageRepeat	• @Nullable @CSS String: **Vertical** - repeat CSS property for vertical. • @Nullable @CSS String: **Vertical** - repeat CSS property for Horizontal.	**This**	Same as parent.
SetBGImageSizeAndPosition	• **Integer - @CSS String: Size** - CSS background size.	**This**	Same as parent.

	• **Integer** - @CSS String**: X** - CSS position • **Integer** - @CSS String**: Y** - CSS position		
SetOpacity	• **Float: Opacity** - Opacity value [0,1]	**None**	Same as parent.

Contour:

Name	Arguments	Return	Description
SetBorder	• **String: At** - Define which border you want or enter "All" (Can can add more than one - it always start by a capital). • @CSS String**: Type** - CSS border type. • **Integer** - @CSS String**: Thickness** - CSS border size. • **Color: Color** - Border Color • **Integer** - @CSS String - **Class: Thickness** - Set the border radius (integer will be same "px" for all). To use the class just send member like: class.top_left.	**This**	Same as parent.
RemoveOutline	• **None**	**This**	Same as parent.

Font:

Name	Arguments	Return	Description
SetFontSize	• **Integer** - @CSS String **: Value** - Set the font size based on CSS values.	**This**	Same as parent.

Interaction:

Name	Arguments	Return	Description

ForceInteraction	• **None**	**This**	Same as parent.
AddHoveredEvent	• **Function: funcIn** - Function called when mouse enter • **Function: funcOut** - Function called when mouse leave	**VOID**	Same as parent.
AddPressedEvent	• **Function: funcIn** - Function called when mouse is pressed • **Function: funcOut** - Function called when mouse is released	**VOID**	Same as parent.
Movable	• **Boolean: LockHorizontal** • **Boolean: LockVertical**	**VOID**	Same as parent.
trigger	• **String : Trigger** - JQuery event name.	**VOID**	Same as parent.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default Value	Description
parent	Widget	undefined	Parent class of this widget
id	String	undefined	Global HMTL Id of the viewer (# + name).
registryName	String	undefined	Name of this widget.
basedId	String	undefined	Where it is located is the existing DOM (its parent).
widgetType	WidgetType	WIDGET	Hold the type of this widget.
divID	String	undefined	Hold the text ("p") id.
layout	Layout	null	Hold this widget layout (if created).

Information

Name	Type	Default Value	Description
verticalAlign	String	"top"	Hold the actual text align set for the vertical.
horizontalAlign	String	"left"	Hold the actual text align set for the horizontal.

lang	Lang	null	Hold the Lang class used for the text.
gradient	AnimatedGradient	null	Hold the actual gradient animation if called.

Overrides / Inherited functions:

Name	Arguments	Return	Description
Shift	• **Integer - @CSS String: ShiftX** • **Integer - @CSS String: ShiftY**	**None**	Same as parent.

Compatibility

LANGUAGE CLASS: Compatible.

Widget_TextInput

The widget_TextInput is a special module use that use and enhance the HTML text input.

How to Initialize any Widgets?

To add a widgets call AddWidget function from GUIs members that own this function.

Class Details

Enumerations:

Name	List	Description
inputType	• TEXT • PASSWORD	This is used to defined the input type in **SetType**.

Main Functions

Initialization & Main Functions

Name	Arguments	Return	Description

Init	• **String: BaseID** - HMTL ID defining the position in the DOM of the tab. (Automatically set by AddWidgets). • **String: RegistryName** - Name that define this overlay, all children IDs will start by this.	**VOID**	This function Initialized every part of the viewer : HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this will start by the RegistryName defined in arguments. It will also call the parent **Init()**.
Launch	• **None**	**VOID**	Launch all systems required by the this and will call all children launch. It will also call the parent **Launch()**.
SetType	• **inputType@: Type**	**THIS**	Change the input type.

Text Functions

Name	Arguments	Return	Description
SetLang	• **Lang: Lang** - Lang to assign with this widget.	**THIS**	Assign a Lang class to this widgets to make the function **SetTextHolder_L** usable.
SetTextHolder_L	• **String: TextID** - Text id is the JSON file • **String: TextID** - Text group is the JSON file • **String: TextID** - The default if not found	**THIS**	Set the text holder of this widgets (will not work if no Lang is set).
SetTextHolder	• **String: Text** - Text you want to add.	**THIS**	Set the text holder to the widget. **Note: to use Lang class use SetTextHolder_L function.**
GetText	• **None**	**String**	Get the entered text.
SetText	• **String: Text** - Text you want to add.	**THIS**	Change the input text (not the text holder).
SetFont	• **String: Name** - Font family name.	**THIS**	Change text font family.

Interaction Functions

Name	Arguments	Return	Description

SetFocus	• **Function: FocusIn** - Function called when focused • **Function: FocusOut** - Function called when focus is lost.	**THIS**	Add events on "focusin" or "focusout".
SetOnTextChanged	• **Function: FuncIn** - Function called when text has changed.	**THIS**	Set up the event "input" to detect when the text has been changed.

Overrides / Inherited Functions:

Setters Functions

Name	Arguments	Return	Description
AddLayout	• **LayoutType: Type** - Define which layout you want.	**Layout**	Same as parent.
AddWidget	• **WidgetType: Type** - Define which widget you want. • **String: Name** - Name of this widgets.	**Widgets**	Same as parent.
Destroy	• **None**	**None**	Same as parent.
Clear	• **None**	**None**	Same as parent.

Design functions

Positions and general design:

Name	Arguments	Return	Description
Visible	• **Boolean: IsVisible** - Set you want it to appear or disappear. • **Integer: Fade** - Fading time in MS • **Function: VisCallback** - Callback function called when fade is over.	**This**	Same as parent.
SetSizePosition	• **Integer - @CSS String: Width** - of this widget (based on parent). • **Integer - @CSS String: Height** - of this widget (based on parent).	**This**	Same as parent.

	• **Integer - @CSS String: X** - (horizontal) of this widget (based on parent). • **Integer - @CSS String: Y** - (vertical) of this widget (based on parent). • **Boolean: FromRight** - Set if x has to start from right • **Boolean: FromBottom** - Set if x has to start from bottom		
GetPosition	• **@CSS String: Where** - position in CSS	@CSS String	Same as parent.
GetSize	• **None**	**Class:** • **w: width** • **h: height**	Same as parent.
SetPadding	• **String: At** - Where to add the padding (Top, Left, Bottom, Right or All). • **Integer - @CSS String: Value** - Size of the padding	This	Same as parent.
SetMargin	• **String: At** - Where to add the padding (Top, Left, Bottom, Right or All). • **Integer - @CSS String: Value** - Size of the margin	This	Same as parent.
AddVerticalOverflow	• **@CSS String: State** - CSS overflow state	This	Same as parent.
AddHorizontalOverflow	• **@CSS String: State** - CSS overflow state	This	Same as parent.
SetRotation	• **Integer: Deg** - Degrees of rotation	This	Same as parent.

Style:

Name	Arguments	Return	Description
SetBackgroundColor	• **Color: BG** - Background Color	This	Same as parent.
SetBackgroundImage	• **String: URL** - Image link • **Function: SBICallback** - Callback when the image has been loaded. • **Boolean: Debug** - Set if you need a debug when the image has been loaded.	**VOID**	Same as parent.

SetBGImageRepeat	• @Nullable @CSS String: **Vertical** - repeat CSS property for vertical. • @Nullable @CSS String: **Vertical** - repeat CSS property for Horizontal.	**This**	Same as parent.
SetBGImageSizeAndPosition	• **Integer** - @CSS String: **Size** - CSS background size. • **Integer** - @CSS String: **X** - CSS position • **Integer** - @CSS String: **Y** - CSS position	**This**	Same as parent.
SetOpacity	• Float: **Opacity** - Opacity value [0,1]	**None**	Same as parent.

Contour:

Name	Arguments	Return	Description
SetBorder	• String: **At** - Define which border you want or enter "All" (Can can add more than one - it always start by a capital). • @CSS String: **Type** - CSS border type. • **Integer** - @CSS String: **Thickness** - CSS border size. • Color: **Color** - Border Color • **Integer** - @CSS String - **Class**: **Thickness** - Set the border radius (integer will be same "px" for all). To use the class just send member like: class.top_left.	**This**	Same as parent.
RemoveOutline	• **None**	**This**	Same as parent.

Font:

Name	Arguments	Return	Description
SetFontSize	• **Integer** - @CSS **String** : **Value** - Set the font size based on CSS values.	**This**	Same as parent.

Interaction:

Name	Arguments	Return	Description
ForceInteraction	• **None**	**This**	Same as parent.
AddHoveredEvent	• **Function**: **funcIn** - Function called when mouse enter • **Function**: **funcOut** - Function called when mouse leave	**VOID**	Same as parent.
AddPressedEvent	• **Function**: **funcIn** - Function called when mouse is pressed • **Function**: **funcOut** - Function called when mouse is released	**VOID**	Same as parent.
Movable	• **Boolean**: **LockHorizontal** • **Boolean**: **LockVertical**	**VOID**	Same as parent.
trigger	• **String** : **Trigger** - JQuery event name.	**VOID**	Same as parent.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default Value	Description
parent	Widget	undefined	Parent class of this widget
id	String	undefined	Global HMTL Id of the viewer (# + name).
registryName	String	undefined	Name of this widget.
basedId	String	undefined	Where it is located is the existing DOM (its parent).

widgetType	WidgetType	WIDGET	Hold the type of this widget.
layout	Layout	null	Hold this widget layout (if created).

Information

Name	Type	Default Value	Description
lang	Lang	null	Hold the Lang class used for the text.
isFocus	Boolean	undefined	Use to check if the widget is actually focused.

Overrides / Inherited functions:

Name	Arguments	Return	Description
Shift	• **Integer** - @CSS **String**: **ShiftX** • **Integer** - @CSS **String**: **ShiftY**	**None**	Same as parent.

Compatibility

LANGUAGE CLASS: Compatible.

Widget_Select

The widget_select offer a widget composed by select box to extend you input selection.

How to Initialize any Widgets?

To add a widgets call AddWidget function from GUIs members that own this function.

Class Details

Main Functions

Initialization & Main Functions

Name	Arguments	Return	Description

Init	• **String: BaseID** - HMTL ID defining the position in the DOM of the tab. (Automatically set by AddWidgets). • **String: RegistryName** - Name that define this overlay, all children IDs will start by this.	**VOID**	This function Initialized every part of the viewer : HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this will start by the RegistryName defined in arguments. It will also call the parent **Init()**.
Launch	• **None**	**VOID**	Launch all systems required by the this and will call all children launch. It will also call the parent **Launch()**.

Options Functions

Name	Arguments	Return	Description
AddOption	• **String: Name** - Name of this option and the content.	**THIS**	Add a new option to the select box.
GetOption	• **None**	**String**	Get the active option name.
SetOption	• **String: Option** - Name of the option you force to force select.	**THIS**	Force to select a specified option.
SetOnChange	• **Function: FuncIn** - Function called when selection changed.	**THIS**	This function set the event: "change" to detect when a selection has been made.

Overrides / Inherited Functions:

Setters Functions

Name	Arguments	Return	Description
AddLayout	• **LayoutType: Type** - Define which layout you want.	**Layout**	Same as parent.
AddWidget	• **WidgetType: Type** - Define which widget you want. • **String: Name** - Name of this widgets.	**Widgets**	Same as parent.
Destroy	• **None**	**None**	Same as parent.
Clear	• **None**	**None**	Same as parent.

Design functions

Positions and general design:

Name	Arguments	Return	Description
Visible	• **Boolean: IsVisible** - Set you want it to appear or disappear. • **Integer: Fade** - Fading time in MS • **Function: VisCallback** - Callback function called when fade is over.	**This**	Same as parent.
SetSizePosition	• **Integer - @CSS String: Width** - of this widget (based on parent). • **Integer - @CSS String: Height** - of this widget (based on parent). • **Integer - @CSS String: X** - (horizontal) of this widget (based on parent). • **Integer - @CSS String: Y** - (vertical) of this widget (based on parent). • **Boolean: FromRight** - Set if x has to start from right • **Boolean: FromBottom** - Set if x has to start from bottom	**This**	Same as parent.
GetPosition	• **@CSS String: Where** - position in CSS	**@CSS String**	Same as parent.
GetSize	• **None**	**Class:** • **w: width** • **h: height**	Same as parent.
SetPadding	• **String: At** - Where to add the padding (Top, Left, Bottom, Right or All). • **Integer - @CSS String: Value** - Size of the padding	**This**	Same as parent.
SetMargin	• **String: At** - Where to add the padding (Top, Left, Bottom, Right or All). • **Integer - @CSS String: Value** - Size of the margin	**This**	Same as parent.
AddVerticalOverflow	• **@CSS String: State** - CSS overflow state	**This**	Same as parent.
AddHorizontalOverflow	• **@CSS String: State** - CSS overflow state	**This**	Same as parent.
SetRotation	• **Integer: Deg** - Degrees of	**This**	Same as parent.

	rotation		

Style:

Name	Arguments	Return	Description
SetBackgroundColor	• **Color: BG** - Background Color	**This**	Same as parent.
SetBackgroundImage	• **String: URL** - Image link • **Function: SBICallback** - Callback when the image has been loaded. • **Boolean: Debug** - Set if you need a debug when the image has been loaded.	**VOID**	Same as parent.
SetBGImageRepeat	• **@Nullable @CSS String: Vertical** - repeat CSS property for vertical. • **@Nullable @CSS String: Vertical** - repeat CSS property for Horizontal.	**This**	Same as parent.
SetBGImageSizeAndPosition	• **Integer - @CSS String: Size** - CSS background size. • **Integer - @CSS String: X** - CSS position • **Integer - @CSS String: Y** - CSS position	**This**	Same as parent.
SetOpacity	• **Float: Opacity** - Opacity value [0,1]	**None**	Same as parent.

Contour:

Name	Arguments	Return	Description
SetBorder	• **String: At** - Define which border you want or enter "All" (Can can add more than one - it always start by a capital). • **@CSS String: Type** -	**This**	Same as parent.

	CSS border type. • **Integer - @CSS String: Thickness** - CSS border size. • **Color: Color** - Border Color • **Integer - @CSS String - Class: Thickness** - Set the border radius (integer will be same "px" for all). To use the class just send member like: class.top_left.		
RemoveOutline	• **None**	**This**	Same as parent.

Font:

Name	Arguments	Return	Description
SetFontSize	• **Integer - @CSS String : Value** - Set the font size based on CSS values.	**This**	Same as parent.

Interaction:

Name	Arguments	Return	Description
ForceInteraction	• **None**	**This**	Same as parent.
AddHoveredEvent	• **Function: funcIn** - Function called when mouse enter • **Function: funcOut** - Function called when mouse leave	**VOID**	Same as parent.
AddPressedEvent	• **Function: funcIn** - Function called when mouse is pressed • **Function: funcOut** - Function called when mouse is released	**VOID**	Same as parent.
Movable	• **Boolean: LockHorizontal** • **Boolean: LockVertical**	**VOID**	Same as parent.
trigger	• **String : Trigger** - JQuery event name.	**VOID**	Same as parent.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default Value	Description
parent	Widget	undefined	Parent class of this widget
id	String	undefined	Global HMTL Id of the viewer (# + name).
registryName	String	undefined	Name of this widget.
basedId	String	undefined	Where it is located is the existing DOM (its parent).
widgetType	WidgetType	WIDGET	Hold the type of this widget.
layout	Layout	null	Hold this widget layout (if created).

Overrides / Inherited functions:

Name	Arguments	Return	Description
Shift	• **Integer - @CSS String: ShiftX** • **Integer - @CSS String: ShiftY**	**None**	Same as parent.

Compatibility

LANGUAGE CLASS: Not compatible.

MenuOverlay

The Menu Overlay is the main overlays it allow you to create HUD, Menus, Inventories, ETC... This is also what you will use to create web app with the EGE

How to Initialize the Menu Overlay?

To initialize the Menu Overlay simply call the function AddNewMenuOverlay from the GUI group.

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: BaseID** - HMTL ID defining the position in the DOM of the tab. (Automatically set by AddNewMenuOverlay) . • **String: RegistryName** - Name that define this overlay, all children IDs will start by this.	**VOID**	This function Initialized every part of the viewer: HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this will start by the RegistryName defined in arguments.
Launch	• **None**	**VOID**	Launch all systems required by the this and will call all children launch.

Setters Functions

Name	Arguments	Return	Description
IsFillingViewport	• **Boolean: IsFilling** - Specified if filling (See description)	**VOID**	Setup an Overlay IsFilling mean it cover the whole viewport, so it will lock all interaction behind it. If you set it has fault all the interaction will go through (no more event on the overlay - in that case use ForceInteraction function to make it interactive).
SetBackgroundColor	• **Color: BG** - Background Color	**VOID**	Setup the background color.
SetBackgroundImage	• **String: URL** - link of the image • **@Nullable Function: SBICallback** - function called when the image is loaded.	**VOID**	Setup the background image.
Reset	• **None**	**VOID**	Will destroy and remove everything in the overlay.

Design & Other Functions

Name	Arguments	Return	Description

AddWidget	• **WidgetType: Type** - Define which widget you want. • **String: Name** - Name of this widgets.	**Widgets**	Will add a selected children widgets in itself and return it. **Note: The children will always they're IDs starting by the parent name, so time it's not in the same family you can add an identical name.**
AddPopUp	• **String: Name** - Name of this popup.	**PopUp**	Add a new PopUp window to the overlay.

Interaction Functions

Name	Arguments	Return	Description
Movable	• **Integer: WhichButton** - The mouse button ID that will move the overlay.	**VOID**	This function will authorize the overlays to move freely on a 2D plane when the selected mouse button is hold. This function will create all events and will change cursor type to *grab* .
AddClick	• **Function: Down** - function called when mouse down. • **Function: Up** - function called when mouse up.	**VOID**	Set the mouse down and up events.
AddMouseMove	• **Function: Func** - function called when mouse moves.	**VOID**	Set the mouse move events.
AddInteractiveToolBox	• **Function: AITBCallback** - function called when all EE modules are loaded.	**InteractiveToolBox**	Set an **InteractiveToolBox** working with this overlay (it return it for setters).

Linker Functions

Name	Arguments	Return	Description
CreateLink	• **Position: Pos** - Where the linker starting.	**Boolean**	It create create a linker at **Pos**. This linker will now be attached to the mouse. Send back a Boolean if the linker has been created.
CancelLink	• **None**	**VOID**	If the link is still active (still following cursor) then use this function to cancel this.
ValidateLink	• **Position: Pos** - Where the linker finishing.	**Boolean**	Will lock the linker on the defined position. Send back a Boolean if the linker has been locked.
GetLinker	• **Position: Pos** - Where the linker	**Linker**	You can retrieved an existing linker by using it's position and offset position.

	starting. • **Position: Offset** - Where the linker finishing.		
RemoveLinker	• **Position: Pos** - Where the linker starting. • **Position: Offset** - Where the linker finishing.	**None**	Will use the position and the offset position to find the linker then it will remove and destroy it.
ClearLinker	• **None**	**None**	Will remove all linker.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default Value	Description
id	String	undefined	Global HMTL Id of the viewer (# + name).
registryName	String	undefined	Name of this overlay.
basedId	String	undefined	Where it is located is the existing DOM (its parent).

Information

Name	Type	Default Value	Description
lastMousePos	Position	x: 0 and y: 0	Hold where the mouse in the last move.
shiftX	Integer	0	Hold the x shift the overlays made (When user is moving the overlay). It is used by widgets to recalculate position.
shiftX	Integer	0	Hold the y shift the overlays made (When user is moving the overlay). It is used by widgets to recalculate position.
isMoving	Boolean	false	Used to check if the overlay is moving.
mouseMoveButton	Integer	2	Used to defined which mouse button make the overlay moving

Widgets

Name	Type	Default Value	Description

AllWidget	Widgets[]	[]	Array holding all the Widgets in the overlay.
lastZIndex	Integer	false	To make sure the Widgets are properly layered we decrement this variable each widgets.

Linker

Name	Type	Default Value	Description
allLinker	Linker[]	[]	Array holding all the Linker in the overlay.
lastLinkerID	Integer	0	To make sure the Linker are properly layered we increment this variable each widgets.
doesLinkerActive	Boolean	false	Used to check if a Linker is moving with the cursor.

Tool Box

Name	Type	Default Value	Description
toolBox	toolBox	null	Hold the active toolbox set by AddInteractiveToolBox.
toolBoxIsOpen	Boolean	false	Used to check if toolBox is open.

Functions:

Interaction Functions

Name	Arguments	Return	Description
MoveAll	• **None**	**None**	This function is called when the overlay is moving to move all overlays components.

Linker Functions

Name	Arguments	Return	Description
MouseMove	• **JS Event Class: e**	**None**	This function is called when the mouse is moving and a linker is active to make it follow it.

Util Functions

Name	Arguments	Return	Description
GetLinkersID	• **None**	**String**	Return the linker div ID.

Compatibility

LANGUAGE CLASS: Not compatible.

Popup

Popup are the is a special components of the Overlays for now only is available but other will come

- **PopUp**: Main PopUp

PopUp

The PopUp is a special window component used by Overlays it will open a full customizable window using the widgets system.

How to Initialize any PopUp?

PopUp has to be called by a Overlays with the function **AddPopUp**.
Also call **SetOpening** to setup the animation details, the PopUp will not open without this function.

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: BaseID** - HMTL ID defining the position in the DOM of the popup. • **String: RegistryName** - Name that define this popup, all children IDs will start by this.	**VOID**	This function Initialized every part of the viewer: HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this will start by the RegistryName defined in arguments.
Launch	• **None**	**VOID**	Launch all systems required by the this and will call all children launch.
SetOpening	• **Curves: CurveType** - opening animation curve type. • **Integer: Duration** - Opening time. • **Integer: FPS** -	**VOID**	Will set all details about the opening animation.

	• Opening frame rate • **Class: CurveDetails** - Details required by some curves.		
Open	• **None**	**VOID**	Open the popup
Close	• **None**	**VOID**	Close the popup

Setters Functions

Name	Arguments	Return	Description
SetSizePosition	• **Integer - @CSS String: Width** - of this widget (based on parent). • **Integer - @CSS String: Height** - of this widget (based on parent). • **Integer - @CSS String: X** - (horizontal) of this widget (based on parent). • **Integer - @CSS String: Y** - (vertical) of this widget (based on parent). • **Boolean: FromRight** - Set if x has to start from right • **Boolean: FromBottom** - Set if x has to start from bottom	This	Set the size and the position of this popup.
SetBackgroundColor	• **Color: Color** - Background Color	This	Will change the popup background color
SetBorder	• **@CSS String: Type** - CSS border type. • **Integer - @CSS String: Thickness** - CSS border size. • **Color: Color** - Border Color • **Integer - @CSS String - Class: Thickness** - Set the border radius (integer will be same "px" for all). To use the class just send member like: class.top_left.	This	Add a border to the popup
SetBackColor	• **Color: Color** - Back	This	Set the color of behind the popup (Used to

	color		hide app will the popup is active).
AddWidget	• **WidgetType**: **Type** - Define which widget you want. • **String**: **Name** - Name of this widgets.	**Widgets**	Add a widget inside the popup.

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default Value	Description
id	String	undefined	Global HMTL Id of the viewer (# + name).
registryName	String	undefined	Name of this widget.
basedId	String	undefined	Where it is located is the existing DOM (its parent).

Information

Name	Type	Default Value	Description
position	Class	x: 0 and y: 0 and fromRight: false and fromBottom: false	Hold the actual position of the widget.
scale	Integer	0	Hold what is this actual scale of the popup (for animations).
animation	Integer	null	Actual animation ID.
curve	Class	{}	Hold curves details, set by **SetOpening** function.
AllWidget	Widgets[]	[]	Hold all children Widgets.
lastZIndex	Integer	100	To make sure the Widgets are properly layered we decrement this variable each widgets.

Functions:

Interaction Functions

Name	Arguments	Return	Description

GetContent	• **None**	**JQuery DOM Object**	Return the content **JQuery DOM Object**
GetBack	• **None**	**JQuery DOM Object**	Return the back **JQuery DOM Object**

Compatibility

LANGUAGE CLASS: Not compatible.

Other

The sub-group has been created to stored all individual component that doesn't fit another group.

- **Linker**: small straight line that can take the path from a point to another.

Linker

Linker are made to show a line between two point it will automatically adjust its size, position, and rotation.

How to Initialize any Linker?

Linker have to be handle by overlays with several functions.

Class Details

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: BaseID** - HMTL ID defining the position in the DOM of the linker. • **String: RegistryName** - Name that define this popup, all children IDs will start by this.	**VOID**	This function Initialized every part of the viewer: HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this will start by the RegistryName defined in arguments.
Launch	• **None**	**VOID**	N/A
SetPos	• **Integer: X** • **Integer: Y**	**VOID**	Set the starting point based on overlay size and position.

SetOffset	• **Integer: OffsetX** • **Integer: OffsetY**	**VOID**	Set the finishing point based on overlay size and position.
Remove	• **None**	**VOID**	Delete this linker from DOM

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default Value	Description
id	String	undefined	Global HMTL Id of the viewer (# + name).
registryName	String	undefined	Name of this widget.
basedId	String	undefined	Where it is located is the existing DOM (its parent).

Information

Name	Type	Default Value	Description
position	Class	x: 0 and y: 0 and fromRight: false and fromBottom: false	Hold the actual position of the widget.

Compatibility

LANGUAGE CLASS: Not compatible.

Legend

All along this documentation you will a color code, some special annotations, and specific classes to help understanding the doc. This color color code is summarize here:

Color

Main:

- **EngineEngine (#BF0000)**: Color of the EE class.
- **EngineGameEngine (#7F0000)**: Color of the EGE class.

Web languages:

- **CSS (#588877)**: CSS class or name.

- **JQuery Classes (#A40694)**: All JQuery Classes.
- **JS Functions (#23503F)**: JavaScript method or function

EE & Main:

- **Class / Modules (#7030A0)**: Default classes / Modules color.
- **Group (#825BC1)**: Default groups color.

- **Method / Functions (#00B050)**: Refer to any type of functions.

- **Event (#FF0000)**: Refer events names.

Structures:

- **Enumerator (#FF8040):** Refer to a enumerator structure or its content.

Programming identifier:

- **Keyword (#00007F)**: Programming Key words

- **Integer (#3333FF)**: String variable
- **Float (#6666FF)**: String variable

- **String (#007F7F)**: String variable
- **BOOLEAN (#B96666)**: Boolean variable

Other:

- **WEB ADDRESS (#00B0F0)**: Used to specify a web address (this is not a link).

Specials methods (some function are use for very specific cases):

- **EVENT FUNCTION (#C80000)**: Called by a window event.
- **ANIMATION FUNCTION (#007BBF)**: Basics structure use by animation process, usually it contain the three arguments: Percent, X, ExtraData.

Annotations

Annotations are special remarks identified by the symbol @:

@Nullable: The argument can be null.
@CSS: You have to enter a value matching with CSS systems (correct units, matching description, etc...).
@Non-Module: Those are functions where don't need to call a modules, there are already loaded with EE, it use for faster access while programming

Common Classes / Structures

Some classes and structure are used by different modules all around EE here is a some of them:

Position (Pos):

Name	Type	Default Value	Description
x	Float / Integer	null	X of the point
y	Float / Integer	null	Y of the point

Position are used to keep track of a point into a 2D space.

Undefined identifier

Some identifier are multi-types and can be used different ways:

Color

This identifier can accept:

- @CSS String: Defining the color as used by CSS.
- String: used for hex colors.
- Integer[3]: used to define RGB color.
- Integer & Float[4]: used to define RGBA color.
- Class: class with members - .r for red, .g for green, .b for blue and .a for alpha Chanel (Optional).

Loading

The Loading groups all loading systems. For now only compose this group:

- **Loading**: The main loading class

Note: this group is not present in the EGE, only the function AddOverAllLoading can give an access to the loading systems.

Loading

The loading class offer you the possibility to add loading bar overall hide the initialization and make sure the user see the progress.

How to Initialize the Loading?

To initialize the Menu Overlay simply call the function AddOverAllLoading from the EGE group.

Class Details

Enumerations:

Name	List	Description

LoadingTypes	• OVERALL_LOADING	This is used to defined the input type in **SetLoadingType**.

Functions:

Initialization & Main Functions

Name	Arguments	Return	Description
Init	• **String: BaseID** - HMTL ID defining the position in the DOM of the tab. (Automatically set by AddOverAllLoading). • **String: RegistryName** - Name that define this overlay, all children IDs will start by this.	**VOID**	This function Initialized every part of the viewer: HMTL, CSS, Data, Events, etc... It will be generated in the defined ID and all HTML IDs used for this will start by the RegistryName defined in arguments.
Launch	• **Function: Callback** - Function called when everything is loaded. • **Integer: Time** - Time the loading take fade out.	**VOID**	Launch all systems required by the this and will call all children launch. And load the animation (if set).

Functionalities

Name	Arguments	Return	Description
AddLoadingBar	• **Function: Callback** - Function called when everything is loaded.	**VOID**	Call this function to add a **LoadingBar** to this loading systems.
SetMaxStep	• **Integer: Max**	**VOID**	Set how many steps the loading has.
Step	• **@Nullable Integer: Step**	**VOID**	Call this function to specified which step you are, if you're not defining the step it will just increment by one. If the step value goes the maximum it will consider as done and the loading will fade out.

Setters Functions

Name	Arguments	Return	Description
SetLoadingType	• **LoadingTypes: Type** - Specified the type.	**VOID**	Setup the type of loading, for now only OVERALL_LOADING is available, it is use to cover everything and allow EGE to load behind.

SetBackgroundRGB	• **Integer: R** - Red • **Integer: G** - Green • **Integer: B** - Blue	**VOID**	Setup the background color.
SetBackgroundHex	• **String: Hex** - Color hexa	**VOID**	Setup the background color.
SetBackgroundColorName	• **@CSS String: Name** - Color	**VOID**	Setup the background color.
SetBackgroundImage	• **String: URL** - link of the image	**VOID**	Setup the background image.
SetLoadingBackColor	• **@CSS String: Color** - Color	**VOID**	Set loading bar back color (unload part).
SetLoadingFrontColor	• **@CSS String: Color** - Color	**VOID**	Set loading bar front color (load part).
SetAnimationGIF	• **String: URL** - link of the animation	**VOID**	Set a loading animation (GIF). **Note: it might be heavy**

Accessor Functions

Name	Arguments	Return	Description
GetLoadingBar	• **None**	**LoadingBar**	Get the **LoadingBar** from the class if you want to modify some settings

Private Details

All private details are usable but its recommended to not use any of those. It might modify the proper functioning of the class.

Variables (Members):

Main Members

Name	Type	Default Value	Description
id	String	undefined	Global HMTL Id of the viewer (# + name).
registryName	String	undefined	Name of this overlay.
basedId	String	undefined	Where it is located is the existing DOM (its parent).
step	Integer	0	Actual step
maxStep	Integer	1	Maximum step (above it will close the loading).

Information

Name	Type	Default Value	Description

loadingType	LoadingTypes	OVERALL_LOADING	Hold the type of this loading system.
backgroundURL	String	null	Hold the background image link (if any).
loadingBar	LoadingBar	null	Hold the LoadingBar class (if any)
hasAnimation	Boolean	false	Used to check if there is an animated GIF.
animationURL	String	null	Link of the animated GIF.

Functions:

LoadingBar Functions

Name	Arguments	Return	Description
AddLoadingBarCallback	• **None**	**None**	This function is called by **AddLoadingBar** to setup the loading bar at the middle.

Util Functions

Name	Arguments	Return	Description
GetCenterID	• **None**	**String**	Return the center div ID.
GetAnimationID	• **None**	**String**	Return the animation div ID.

Compatibility

LANGUAGE CLASS: Not compatible.