

Gaspare Sganga

IT Manager, GIS Analyst & Lead Developer @setin.
Freelance Developer & Consultant.



Buy me a coffee?

[About](#)

[Labs](#)

[Posts](#)

jQuery LoadingOverlay

A flexible loading overlay jQuery plugin

4 May 2018 : Version 2.1.3 released. See release notes.

Contents

- [Quick Demo](#)
- [Get it](#)
- [Features](#)
- [Methods](#)
- [Actions](#)
- [Options and defaults values](#)
- [Animations](#)
- [Examples](#)
- [History](#)
- [Comments and Ideas](#)

Quick Demo

[Try jQuery LoadingOverlay!](#)

Get it

GitHub

View project on [GitHub](#) or download the latest [release](#).

npm

```
npm install gasparesganga-jquery-loading-overlay
```

Bower

```
bower install gasparesganga-jquery-loading-overlay
```

CDN

```
https://cdn.jsdelivr.net/npm/gasparesganga-jquery-loading-overlay@2.1.3/dist/loadingoverlay.min.js
```

CDN hosting of this library is possible thanks to [jsDelivr](#). For more details about URL structure please refer to the official documentation.

On the [CDN configuration page](#) you can find all the available files and versions of this library, select a specific version, use *range* or *latest* version aliasing, join more files in a single request and enable *subresource integrity (SRI)*.

Features

- Easy *plug-and-play* default behaviour yet fully configurable for advanced uses
- Shows a loading overlay on the whole page or over single DOM elements
- Can auto resize according to its container (very useful if used over a DOM element being filled meanwhile)
- Tracks a *counter* to allow multiple calls on single target
- Can show an image, some text, a progress bar or even a custom element to provide feedback to the user
- Compatible with [Font Awesome](#)
- No external CSS, high performances

Methods

There are three different methods, one to attach a LoadingOverlay to the `body` and thus covering the whole page, another to attach it to a single *DOM element* or a set of *DOM elements* and the last one to set the default parameters.

\$.LoadingOverlay(action [,options])

Shows the LoadingOverlay with a fixed position, covering the whole page. Optionally pass some `options` to it.

This method doesn't return anything. See [Actions](#) for details.

\$(selector).LoadingOverlay(action [,options])

Attach the LoadingOverlay to a single *DOM element* or a set of *DOM elements*. Optionally pass some `options` to it.

This method returns a *jQuery object* or a set of *jQuery objects (depending on the selector used)* and is **chainable**. See [Actions](#) for details.

\$.LoadingOverlaySetup(options)

Set default `options` for all future calls to `$.LoadingOverlay()` and `$(selector).LoadingOverlay()`.

Actions

The `$.LoadingOverlay()` and `$(selector).LoadingOverlay()` methods have five variants, corresponding to five *Actions*:

Show

```
$[(selector)].LoadingOverlay("show" [,options])
```

Shows a LoadingOverlay, or increases the *counter* if it's been already shown. Optionally you can pass a set of `options`, but note that they only take effect if the LoadingOverlay has not been shown yet on the element.

Hide

```
$(selector).LoadingOverlay("hide" [,force])
```

Hides the LoadingOverlay or decreases the *counter* if it's higher than `1`. You can optionally pass a boolean parameter `force` to hide the LoadingOverlay even if the counter hasn't reached `0`.

Resize

```
$[(selector)].LoadingOverlay("resize")
```

Force LoadingOverlay resizing. This is especially useful when you decide to disable the **auto resize** feature and you want to manually control it when you page changes.

Text

```
$[(selector)].LoadingOverlay("text", value)
```

Update the text currently shown with the one passed to the `value` parameter. Pass boolean value

`false` to hide the text element.

Note that this *action* only works if LoadingOverlay was initialized with some `text`.

Progress

`$(selector).LoadingOverlay("progress", value)`

Update the progress bar with the `value` parameter. Pass boolean value `false` to hide the progress bar.

Note that this *action* only works if LoadingOverlay was initialized with `progress` option set to `true`.

Options and defaults values

```
// Background
background          : "rgba(255, 255, 255, 0.8)"      // String
backgroundClass     : ""                               // String/Boolean
// Image
image               : "<svg> ... </svg>"               // String/Boolean
imageAnimation      : "2000ms rotate_right"           // String/Boolean
imageAutoResize     : true                             // Boolean
imageResizeFactor   : 1                               // Float
imageColor          : "#202020"                       // String/Array/Boolean
imageClass          : ""                               // String/Boolean
imageOrder          : 1                               // Integer
// Font Awesome
fontawesome         : ""                               // String/Boolean
fontawesomeAnimation : ""                             // String/Boolean
fontawesomeAutoResize : true                          // Boolean
fontawesomeResizeFactor : 1                           // Float
fontawesomeColor     : "#202020"                     // String/Boolean
fontawesomeOrder     : 2                              // Integer
// Custom
custom              : ""                               // String/DOM Element/jQuery Object/Boolean
customAnimation      : ""                             // String/Boolean
customAutoResize     : true                            // Boolean
customResizeFactor   : 1                              // Float
customOrder          : 3                              // Integer
// Text
text                : ""                               // String/Boolean
textAnimation        : ""                             // String/Boolean
textAutoResize       : true                           // Boolean
textResizeFactor     : 0.5                            // Float
textColor            : "#202020"                     // String/Boolean
textClass            : ""                             // String/Boolean
textOrder            : 4                              // Integer
// Progress
progress            : false                           // Boolean
progressAutoResize   : true                           // Boolean
progressResizeFactor : 0.25                          // Float
```

```

progressColor      : "#a0a0a0"           // String/Boolean
progressClass      : ""                  // String/Boolean
progressOrder      : 5                    // Integer
progressFixedPosition : ""                // String/Boolean
progressSpeed      : 200                  // Integer
progressMin        : 0                    // Float
progressMax        : 100                  // Float
// Sizing
size               : 50                   // Float/String/Boolean
minSize            : 20                   // Integer/String
maxSize            : 120                  // Integer/String
// Misc
direction          : "column"            // String
fade               : [400, 200]          // Array/Boolean/Integer/String
resizeInterval     : 50                   // Integer
zIndex             : 2147483647           // Integer

```

background

Overlay's CSS background-color property. Use `rgba()` to set the opacity. Keep in mind that if `backgroundClass` is provided then `background` option is ignored.

backgroundClass

Sets a custom CSS class for the background. Use an empty string `""` or `false` to disable it. Keep in mind that if `backgroundClass` is provided then `background` option is ignored.

image

URL or inline representation of the image to show. It supports both raster images and vectorial SVGs. You can pass an inline SVG, a path to a file or even use a *base64-encoded* image or SVG (e.g. `"data:image/png;base64,..."`). Set to an empty string `""` or `false` to show no image.

imageAnimation

Controls the animation of the *image* element. See [animations](#).

imageAutoResize

Controls the auto resizing of the *image* element. Set to `false` to disable it.

imageResizeFactor

Controls the proportion between the *image* element and the *size* parameter.

imageColor

Image *fill* and *stroke* colors. This setting has effect only on SVG images and will be useless with raster images (*JPG, PNG, GIF, etc.*). Use a single string value to specify only the *fill* color, or a two-elements array to set *fill* and *stroke* respectively. You can use any CSS valid expression, included `rgba()`. Use an empty string `""`, empty array `[]` or `false` to leave them unspecified. Note that if `imageClass` is provided then `imageColor` is ignored.

imageClass

Sets a custom CSS class for the *image* element. Use an empty string `""` or `false` to disable it. Note that if `imageClass` is provided then `imageColor` is ignored.

imageOrder

Sets the order of the *image* element relative to the others.

fontawesome

Class(es) of the Font Awesome icon to use. Note that you must include Font Awesome in your project if you wish to use this feature. Use an empty string `""` or `false` to disable the feature.

fontawesomeAnimation

You can rely on Font Awesome native classes to animate the icon (e.g. `fa-spin` or `fa-pulse`) and pass them directly to `fontawesome` option, but of course you can also enjoy the full power of LoadingOverlay animations as with any other element type. See [animations](#) for details.

fontawesomeAutoResize

Controls the auto resizing of the *fontawesome* element. Set to `false` to disable it.

fontawesomeResizeFactor

Controls the proportion between the *fontawesome* element and the `size` parameter.

fontawesomeColor

Sets the color of the *fontawesome* element. You can use any CSS valid expression, included `rgba()`. Use an empty string `""` or `false` to leave it unspecified.

fontawesomeOrder

Sets the order of the *fontawesome* element relative to the others.

custom

A DOM element, jQuery object or plain HTML to append to the LoadingOverlay. Use an empty string `""` or `false` to disable the feature.

customAnimation

Controls the animation of the *custom* element. See [animations](#).

customAutoResize

Controls the auto resizing of the *custom* element. Set to `false` to disable it.

customResizeFactor

Controls the proportion between the *custom* element and the `size` parameter.

customOrder

Sets the order of the *custom* element relative to the others.

text

Displays a *text* element in the LoadingOverlay. Use an empty string `""` or `false` to disable the feature.

textAnimation

Controls the animation of the *text* element. See [animations](#).

textAutoResize

Controls the auto resizing of the *text* element. Set to `false` to disable it.

textResizeFactor

Controls the proportion between the *text* element and the `size` parameter.

textColor

Sets the color of the *text* element. You can use any CSS valid expression, included `rgba()`. Use an empty string `""` or `false` to leave it unspecified. Note that if `textClass` is provided then `textColor` is ignored.

textClass

Sets a custom CSS class for the *text* element. Use an empty string `""` or `false` to disable it. Note that if `textClass` is provided then `textColor` is ignored.

textOrder

Sets the order of the *text* element relative to the others.

progress

Displays a *progress bar* element in the LoadingOverlay. Use `false` to disable the feature.

progressAutoResize

Controls the auto resizing of the *progress* element. Set to `false` to disable it.

progressResizeFactor

Controls the proportion between the *progress* element and the `size` parameter.

progressColor

Sets the color of the *progress* element. You can use any CSS valid expression, included `rgba()`. Use an empty string `""` or `false` to leave it unspecified. Note that if `progressClass` is provided then `progressColor` is ignored.

progressClass

Sets a custom CSS class for the *progress* element. Use an empty string `""` or `false` to disable it. Note that if `progressClass` is provided then `progressColor` is ignored.

progressOrder

Sets the order of the *progress* element relative to the others.

progressFixedPosition

Set a fixed position for the *progress* element. It accepts a space-separated string with **position** and optional **margin**.

Position value can be either `top` or `bottom` while optional **margin** can be expressed in any CSS unit. Note that enabling this option will take the *progress* element out of the order flow and `progressOrder` will be ignored.

Example values: `"top"`, `"bottom"`, `"top 20px"`, `"10% top"`, `"5rem bottom"`, `"bottom 2vh"`, etc.

progressSpeed

Controls the animation speed in **milliseconds** of the progress bar when its value is updated. Set to `0` to disable smooth animation.

progressMin

Sets the minimum value for the *progress* element.

progressMax

Sets the maximum value for the *progress* element.

size

Size of elements expressed in **percentage** relative to the LoadingOverlay size. Note that the computed value will be constrained between `minSize` and `maxSize`. You can specify a fixed size expressed in any CSS unit passing a string (options `minSize` and `maxSize` will be ignored in this case).

Each element will then be resized according to the computed value and its *Resize Factor*.

Use `0` or `false` if you wish to fully control the size of the elements via custom classes.

minSize

Minimum size of elements in **pixels**. Set it to `0` or `false` for no limit.

maxSize

Maximum size of elements in **pixels**. Set it to `0` or `false` for no limit.

direction

Sets the arrangement of the elements in the LoadingOverlay. It can be `"column"` or `"row"`.

fade

Controls the *fade in* and *fade out* durations, expressed in **milliseconds**. Use `0` or `false` to disable it (*meaning a zero duration*), an integer or string to set equal *fade in* and *fade out* times or a two-elements array to set respectively *fade in* and *fade out* durations (e.g. `[600, 300]`). Boolean value `true` will be treated like default value `[400, 200]`.

resizeInterval

Specifies an interval in **milliseconds** to resize and reposition the LoadingOverlay according to its container. This is useful when the container element changes size and/or position while the

LoadingOverlay is being shown. Set it to `0` or `false` to disable this feature.

`zIndex`

Use this to explicitly set a `z-index` for the overlay. This is useful when LoadingOverlay is used with other *z-index intensive* libraries like Bootstrap.

Animations

LoadingOverlay takes advantage of CSS animations and offers 4 different *built-in* keyframes animations:

- `rotate_right`
- `rotate_left`
- `fadein`
- `pulse`

Elements animation properties `imageAnimation`, `fontawesomeAnimation`, `customAnimation` and `textAnimation` accept a space-separated string with **name** and **duration**.

Note that **both** parameters are optional and they default to `"rotate_right 2000ms "` when only one is specified:

```
// These are the same:  
"rotate_right 2s"  
"2000ms"  
"rotate_right"  
  
// And so are these:  
"2000ms pulse"  
"pulse"
```

If you prefer to rely on your custom animations altogether you can disable them setting `imageAnimation`, `fontawesomeAnimation`, `customAnimation` and `textAnimation` to an empty string `""` or `false`, providing animations through custom CSS classes.

Examples

Example 1 - Whole page Overlay

```
// Show full page LoadingOverlay  
$.LoadingOverlay("show");  
  
// Hide it after 3 seconds  
setTimeout(function(){
```

```
$.LoadingOverlay("hide");  
}, 3000);
```

Whole page Overlay

Example 2 - Single element Overlay

```
// Let's call it 2 times just for fun...  
$("#element").LoadingOverlay("show", {  
    background : "rgba(165, 190, 100, 0.5)"  
});  
$("#element").LoadingOverlay("show");  
  
// Here we might call the "hide" action 2 times, or simply set the "force" parameter to true:  
$("#element").LoadingOverlay("hide", true);
```

Single element Overlay

Test auto resizing

Example 3 - Showcase of different elements

```
// Font Awesome  
$.LoadingOverlay("show", {  
    image      : "",  
    fontawesome : "fa fa-cog fa-spin"  
});  
  
// Text  
$.LoadingOverlay("show", {  
    image : "",  
    text  : "Loading..."  
});  
setTimeout(function(){  
    $.LoadingOverlay("text", "Yep, still loading...");  
}, 2500);  
  
// Progress  
$.LoadingOverlay("show", {  
    image      : "",  
    progress   : true  
});  
var count = 0;  
var interval = setInterval(function(){  
    if (count >= 100) {  
        clearInterval(interval);  
        $.LoadingOverlay("hide");  
    }  
    count++;  
}, 100);
```

```

    return;
  }
  count += 10;
  $.LoadingOverlay("progress", count);
}, 300);

// Custom
var customElement = $("

", {
  "css" : {
    "border"      : "4px dashed gold",
    "font-size"   : "40px",
    "text-align"  : "center",
    "padding"     : "10px"
  },
  "class" : "your-custom-class",
  "text"  : "Custom!"
});
$.LoadingOverlay("show", {
  image      : "",
  custom     : customElement
});


```

Font Awesome

Text

Progress

Custom

Example 4 - Complete playground

background	:	<input type="text" value="rgba(255, 255, 255, 0.8)"/>
image	:	<input type="text" value="<svg xmlns='http://www.w3."/>
imageAnimation	:	<input type="text" value="2000ms"/> <input type="text" value="rotate_right"/> ▼
imageAutoResize	:	<input type="text" value="true"/> ▼
imageResizeFactor	:	<input type="text" value="1"/>
imageColor	:	<input type="text" value="#202020"/> <input type="text"/>
imageOrder	:	<input type="text" value="1"/>
fontawesome	:	<input type="text"/>
fontawesomeAnimation	:	<input type="text"/> <input type="text"/> ▼
fontawesomeAutoResize	:	<input type="text" value="true"/> ▼
fontawesomeResizeFactor	:	<input type="text" value="1"/>
fontawesomeColor	:	<input type="text" value="#202020"/>
fontawesomeOrder	:	<input type="text" value="2"/>

text	:	<input type="text"/>
textAnimation	:	<input type="text"/> <input type="button" value="▼"/>
textAutoResize	:	<input type="text" value="tru"/> <input type="button" value="▼"/>
textResizeFactor	:	<input type="text" value="0.5"/>
textColor	:	<input type="text" value="#202020"/>
textOrder	:	<input type="text" value="4"/>
progress	:	<input type="text" value="fal"/> <input type="button" value="▼"/>
progressAutoResize	:	<input type="text" value="tru"/> <input type="button" value="▼"/>
progressResizeFactor	:	<input type="text" value="0.25"/>
progressColor	:	<input type="text" value="#a0a0a0"/>
progressOrder	:	<input type="text" value="5"/>
progressFixedPosition	:	<input type="text"/> <input type="button" value="▼"/> <input type="text"/>
progressSpeed	:	<input type="text" value="200"/>
size	:	<input type="text" value="50"/>
maxSize	:	<input type="text" value="120"/>
minSize	:	<input type="text" value="20"/>
direction	:	<input type="text" value="column"/> <input type="button" value="▼"/>
fade	:	<input type="text" value="400"/> <input type="text" value="200"/>
resizeInterval	:	<input type="text" value="50"/>
Hide after	:	<input type="text" value="5000"/> ms

Show LoadingOverlay

Example 5 - Set Defaults

```
$.LoadingOverlaySetup({
  background      : "rgba(0, 0, 0, 0.5)",
  image           : "img/custom.svg",
  imageAnimation  : "1.5s fadein",
  imageColor      : "#ffcc00"
});

$.LoadingOverlay("show");
```

Try new defaults

Example 6 - Display a LoadingOverlay during each Ajax request

You can rely on `.ajaxStart()` and `.ajaxStop()` to show and hide the LoadingOverlay during every Ajax request:

```
$(document).ajaxStart(function(){
    $.LoadingOverlay("show");
});
$(document).ajaxStop(function(){
    $.LoadingOverlay("hide");
});
// Now try to make a few Ajax calls, a LoadingOverlay will be shown until the last call is complete
```

Or, in case you need some more sophisticated control/filter, you can use `.ajaxSend()` and `.ajaxComplete()` in the same way. LoadingOverlay will take care of multiple calls thanks to its internal counter feature.

```
$(document).ajaxSend(function(event, jqxhr, settings){
    $.LoadingOverlay("show");
});
$(document).ajaxComplete(function(event, jqxhr, settings){
    $.LoadingOverlay("hide");
});
// Now try to make a few Ajax calls, a LoadingOverlay will be shown until the last call is complete
```

History

4 May 2018 - Version 2.1.3

26 April 2018 - Version 2.1.2

22 April 2018 - Version 2.1.1

4 April 2018 - Version 2.1.0

20 March 2018 - Version 2.0.2

16 March 2018 - Version 2.0.1

16 March 2018 - Version 2.0.0

10 February 2018 - Version 1.6.0

29 September 2017 - Version 1.5.4

27 January 2017 - Version 1.5.3

9 December 2016 - Version 1.5.2

11 November 2016 - Version 1.5.1

11 November 2016 - Version 1.5.0

5 August 2016 - Version 1.4.1