[ScrollMagic](http://docs.google.com/index.html)

* [Classes](http://docs.google.com/classes.list.html)
  + [Controller](http://docs.google.com/ScrollMagic.Controller.html)
  + [Scene](http://docs.google.com/ScrollMagic.Scene.html)
* [Events](http://docs.google.com/events.list.html)
  + [add](http://docs.google.com/ScrollMagic.Scene.html#event:add)
  + [change](http://docs.google.com/ScrollMagic.Scene.html#event:change)
  + [destroy](http://docs.google.com/ScrollMagic.Scene.html#event:destroy)
  + [end](http://docs.google.com/ScrollMagic.Scene.html#event:end)
  + [enter](http://docs.google.com/ScrollMagic.Scene.html#event:enter)
  + [leave](http://docs.google.com/ScrollMagic.Scene.html#event:leave)
  + [progress](http://docs.google.com/ScrollMagic.Scene.html#event:progress)
  + [remove](http://docs.google.com/ScrollMagic.Scene.html#event:remove)
  + [shift](http://docs.google.com/ScrollMagic.Scene.html#event:shift)
  + [start](http://docs.google.com/ScrollMagic.Scene.html#event:start)
  + [update](http://docs.google.com/ScrollMagic.Scene.html#event:update)
* [Plugins](http://docs.google.com/mixins.list.html)
  + [GSAP](http://docs.google.com/animation.GSAP.html)
  + [Velocity](http://docs.google.com/animation.Velocity.html)
  + [addIndicators](http://docs.google.com/debug.addIndicators.html)
  + [jQuery](http://docs.google.com/framework.jQuery.html)

[**ScrollMagic**](http://docs.google.com/ScrollMagic.html)**.**Controller

new ScrollMagic.Controller**(options)**

The main class that is needed once per scroll container.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| options | object | <optional> | An object containing one or more options for the controller. Properties  | Name | Type | Argument | Default | Description | | --- | --- | --- | --- | --- | | container | string | object | <optional> | window | A selector, DOM object that references the main container for scrolling. | | vertical | boolean | <optional> | true | Sets the scroll mode to vertical (true) or horizontal (false) scrolling. | | globalSceneOptions | object | <optional> | {} | These options will be passed to every Scene that is added to the controller using the addScene method. For more information on Scene options see [ScrollMagic.Scene](http://docs.google.com/ScrollMagic.Scene.html). | | loglevel | number | <optional> | 2 | Loglevel for debugging. Note that logging is disabled in the minified version of ScrollMagic.   * 0 => silent * 1 => errors * 2 => errors, warnings * 3 => errors, warnings, debuginfo | | refreshInterval | boolean | <optional> | 100 | Some changes don't call events by default, like changing the container size or moving a scene trigger element.  This interval polls these parameters to fire the necessary events.  If you don't use custom containers, trigger elements or have static layouts, where the positions of the trigger elements don't change, you can set this to 0 disable interval checking and improve performance. | |

Source:

* [ScrollMagic/Controller.js](http://docs.google.com/ScrollMagic_Controller.js.html), [line 27](http://docs.google.com/ScrollMagic_Controller.js.html#sunlight-1-line-27)

##### Example

// basic initialization  
var controller = new ScrollMagic.Controller();  
  
// passing options  
var controller = new ScrollMagic.Controller({container: "#myContainer", loglevel: 3});

### Control Methods

#### addScene**(newScene) → {Controller}**

Add one ore more scene(s) to the controller.

This is the equivalent to Scene.addTo(controller).

##### Parameters:

| Name | Type | Description |
| --- | --- | --- |
| newScene | [ScrollMagic.Scene](http://docs.google.com/ScrollMagic.Scene.html) | array | ScrollMagic Scene or Array of Scenes to be added to the controller. |

Source:

* [ScrollMagic/Controller.js](http://docs.google.com/ScrollMagic_Controller.js.html), [line 287](http://docs.google.com/ScrollMagic_Controller.js.html#sunlight-1-line-287)

##### Returns:

Parent object for chaining.

{ Controller }

##### Example

// with a previously defined scene  
controller.addScene(scene);  
  
// with a newly created scene.  
controller.addScene(new ScrollMagic.Scene({duration : 0}));  
  
// adding multiple scenes  
controller.addScene([scene, scene2, new ScrollMagic.Scene({duration : 0})]);

#### destroy**(resetScenes) → {null}**

Destroy the Controller, all Scenes and everything.

##### Parameters:

| Name | Type | Argument | Default | Description |
| --- | --- | --- | --- | --- |
| resetScenes | boolean | <optional> | false | If true the pins and tweens (if existent) of all scenes will be reset. |

Source:

* [ScrollMagic/Controller.js](http://docs.google.com/ScrollMagic_Controller.js.html), [line 658](http://docs.google.com/ScrollMagic_Controller.js.html#sunlight-1-line-658)

##### Returns:

Null to unset handler variables.

{ null }

##### Example

// without resetting the scenes  
controller = controller.destroy();  
  
// with scene reset  
controller = controller.destroy(true);

#### removeScene**(Scene) → {Controller}**

Remove one ore more scene(s) from the controller.

This is the equivalent to Scene.remove().

##### Parameters:

| Name | Type | Description |
| --- | --- | --- |
| Scene | [ScrollMagic.Scene](http://docs.google.com/ScrollMagic.Scene.html) | array | ScrollMagic Scene or Array of Scenes to be removed from the controller. |

Source:

* [ScrollMagic/Controller.js](http://docs.google.com/ScrollMagic_Controller.js.html), [line 330](http://docs.google.com/ScrollMagic_Controller.js.html#sunlight-1-line-330)

##### Returns:

Parent object for chaining.

{ Controller }

##### Example

// remove a scene from the controller  
controller.removeScene(scene);  
  
// remove multiple scenes from the controller  
controller.removeScene([scene, scene2, scene3]);

#### scrollTo**(scrollTarget, additionalParameter) → {Controller}**

Scroll to a numeric scroll offset, a DOM element, the start of a scene or provide an alternate method for scrolling.

For vertical controllers it will change the top scroll offset and for horizontal applications it will change the left offset.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| scrollTarget | mixed |  | The supplied argument can be one of these types:   1. number -> The container will scroll to this new scroll offset. 2. string or object -> Can be a selector or a DOM object. The container will scroll to the position of this element. 3. ScrollMagic Scene -> The container will scroll to the start of this scene. 4. function -> This function will be used for future scroll position modifications. This provides a way for you to change the behaviour of scrolling and adding new behaviour like animation. The function receives the new scroll position as a parameter and a reference to the container element using this. It may also optionally receive an optional additional parameter (see below) ***NOTE:*** *All other options will still work as expected, using the new function to scroll.* |
| additionalParameter | mixed | <optional> | If a custom scroll function was defined (see above 4.), you may want to supply additional parameters to it, when calling it. You can do this using this parameter – see examples for details. Please note, that this parameter will have no effect, if you use the default scrolling function. |

Since:

* 1.1.0

Source:

* [ScrollMagic/Controller.js](http://docs.google.com/ScrollMagic_Controller.js.html), [line 477](http://docs.google.com/ScrollMagic_Controller.js.html#sunlight-1-line-477)

##### Returns:

Parent object for chaining.

{ Controller }

##### Example

// scroll to an offset of 100  
controller.scrollTo(100);  
  
// scroll to a DOM element  
controller.scrollTo("#anchor");  
  
// scroll to the beginning of a scene  
var scene = new ScrollMagic.Scene({offset: 200});  
controller.scrollTo(scene);  
  
// define a new scroll position modification function (jQuery animate instead of jump)  
controller.scrollTo(function (newScrollPos) {  
 $("html, body").animate({scrollTop: newScrollPos});  
});  
controller.scrollTo(100); // call as usual, but the new function will be used instead  
  
// define a new scroll function with an additional parameter  
controller.scrollTo(function (newScrollPos, message) {  
 console.log(message);  
 $(this).animate({scrollTop: newScrollPos});  
});  
// call as usual, but supply an extra parameter to the defined custom function  
controller.scrollTo(100, "my message");  
  
// define a new scroll function with an additional parameter containing multiple variables  
controller.scrollTo(function (newScrollPos, options) {  
 someGlobalVar = options.a + options.b;  
 $(this).animate({scrollTop: newScrollPos});  
});  
// call as usual, but supply an extra parameter containing multiple options  
controller.scrollTo(100, {a: 1, b: 2});  
  
// define a new scroll function with a callback supplied as an additional parameter  
controller.scrollTo(function (newScrollPos, callback) {  
 $(this).animate({scrollTop: newScrollPos}, 400, "swing", callback);  
});  
// call as usual, but supply an extra parameter, which is used as a callback in the previously defined custom scroll function  
controller.scrollTo(100, function() {  
 console.log("scroll has finished.");  
});

#### update**(immediately) → {Controller}**

Updates the controller params and calls updateScene on every scene, that is attached to the controller.

See Controller.updateScene() for more information about what this means.

In most cases you will not need this function, as it is called constantly, whenever ScrollMagic detects a state change event, like resize or scroll.

The only application for this method is when ScrollMagic fails to detect these events.

One application is with some external scroll libraries (like iScroll) that move an internal container to a negative offset instead of actually scrolling. In this case the update on the controller needs to be called whenever the child container's position changes.

For this case there will also be the need to provide a custom function to calculate the correct scroll position. See Controller.scrollPos() for details.

##### Parameters:

| Name | Type | Argument | Default | Description |
| --- | --- | --- | --- | --- |
| immediately | boolean | <optional> | false | If true the update will be instant, if false it will wait until next update cycle (better performance) |

Source:

* [ScrollMagic/Controller.js](http://docs.google.com/ScrollMagic_Controller.js.html), [line 408](http://docs.google.com/ScrollMagic_Controller.js.html#sunlight-1-line-408)

##### Returns:

Parent object for chaining.

{ Controller }

##### Example

// update the controller on next cycle (saves performance due to elimination of redundant updates)  
controller.update();  
  
// update the controller immediately  
controller.update(true);

#### updateScene**(Scene, immediately) → {Controller}**

Update one ore more scene(s) according to the scroll position of the container.

This is the equivalent to Scene.update().

The update method calculates the scene's start and end position (based on the trigger element, trigger hook, duration and offset) and checks it against the current scroll position of the container.

It then updates the current scene state accordingly (or does nothing, if the state is already correct) – Pins will be set to their correct position and tweens will be updated to their correct progress.

***Note:*** *This method gets called constantly whenever Controller detects a change. The only application for you is if you change something outside of the realm of ScrollMagic, like moving the trigger or changing tween parameters.*

##### Parameters:

| Name | Type | Argument | Default | Description |
| --- | --- | --- | --- | --- |
| Scene | [ScrollMagic.Scene](http://docs.google.com/ScrollMagic.Scene.html) |  |  | ScrollMagic Scene or Array of Scenes that is/are supposed to be updated. |
| immediately | boolean | <optional> | false | If true the update will be instant, if false it will wait until next update cycle.  This is useful when changing multiple properties of the scene - this way it will only be updated once all new properties are set (updateScenes). |

Source:

* [ScrollMagic/Controller.js](http://docs.google.com/ScrollMagic_Controller.js.html), [line 369](http://docs.google.com/ScrollMagic_Controller.js.html#sunlight-1-line-369)

##### Returns:

Parent object for chaining.

{ Controller }

##### Example

// update a specific scene on next cycle  
controller.updateScene(scene);  
  
// update a specific scene immediately  
controller.updateScene(scene, true);  
  
// update multiple scenes scene on next cycle  
controller.updateScene([scene1, scene2, scene3]);

### Parameters (getter / setter)

#### enabled**(newState) → {boolean|Controller}**

**Get** or **Set** the current enabled state of the controller.

This can be used to disable all Scenes connected to the controller without destroying or removing them.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| newState | boolean | <optional> | The new enabled state of the controller true or false. |

Source:

* [ScrollMagic/Controller.js](http://docs.google.com/ScrollMagic_Controller.js.html), [line 634](http://docs.google.com/ScrollMagic_Controller.js.html#sunlight-1-line-634)

##### Returns:

Current enabled state or parent object for chaining.

{ boolean | Controller }

##### Example

// get the current value  
var enabled = controller.enabled();  
  
// disable the controller  
controller.enabled(false);

#### loglevel**(newLoglevel) → {number|Controller}**

**Get** or **Set** the current loglevel option value.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| newLoglevel | number | <optional> | The new loglevel setting of the Controller. [0-3] |

Source:

* [ScrollMagic/Controller.js](http://docs.google.com/ScrollMagic_Controller.js.html), [line 608](http://docs.google.com/ScrollMagic_Controller.js.html#sunlight-1-line-608)

##### Returns:

Current loglevel or parent object for chaining.

{ number | Controller }

##### Example

// get the current value  
var loglevel = controller.loglevel();  
  
// set a new value  
controller.loglevel(3);

#### scrollPos**(scrollPosMethod) → {number|Controller}**

**Get** the current scrollPosition or **Set** a new method to calculate it.

-> **GET**:

When used as a getter this function will return the current scroll position.

To get a cached value use Controller.info("scrollPos"), which will be updated in the update cycle.

For vertical controllers it will return the top scroll offset and for horizontal applications it will return the left offset.

-> **SET**:

When used as a setter this method prodes a way to permanently overwrite the controller's scroll position calculation.

A typical usecase is when the scroll position is not reflected by the containers scrollTop or scrollLeft values, but for example by the inner offset of a child container.

Moving a child container inside a parent is a commonly used method for several scrolling frameworks, including iScroll.

By providing an alternate calculation function you can make sure ScrollMagic receives the correct scroll position.

Please also bear in mind that your function should return y values for vertical scrolls an x for horizontals.

To change the current scroll position please use Controller.scrollTo().

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| scrollPosMethod | function | <optional> | The function to be used for the scroll position calculation of the container. |

Source:

* [ScrollMagic/Controller.js](http://docs.google.com/ScrollMagic_Controller.js.html), [line 542](http://docs.google.com/ScrollMagic_Controller.js.html#sunlight-1-line-542)

##### Returns:

Current scroll position or parent object for chaining.

{ number | Controller }

##### Example

// get the current scroll Position  
var scrollPos = controller.scrollPos();  
  
// set a new scroll position calculation method  
controller.scrollPos(function () {  
 return this.info("vertical") ? -mychildcontainer.y : -mychildcontainer.x  
});

### Properties (getter)

#### info**(about) → {mixed|object}**

**Get** all infos or one in particular about the controller.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| about | string | <optional> | If passed only this info will be returned instead of an object containing all.  Valid options are:   * "size" => the current viewport size of the container * "vertical" => true if vertical scrolling, otherwise false * "scrollPos" => the current scroll position * "scrollDirection" => the last known direction of the scroll * "container" => the container element * "isDocument" => true if container element is the document. |

Source:

* [ScrollMagic/Controller.js](http://docs.google.com/ScrollMagic_Controller.js.html), [line 575](http://docs.google.com/ScrollMagic_Controller.js.html#sunlight-1-line-575)

##### Returns:

The requested info(s).

{ mixed | object }

##### Example

// returns the current scroll position (number)  
var scrollPos = controller.info("scrollPos");  
  
// returns all infos as an object  
var infos = controller.info();

© Jan Paepke 2015

Documentation generated by [JSDoc 3.5.5](https://github.com/jsdoc3/jsdoc) using a customized version of the [DocStrap template](https://github.com/terryweiss/docstrap).