jQuery custom content scroller

Last updated on Jul 11, 2016
Originally published on August 1, 2010 by malihu, under Plugins.

Highly customizable custom scrollbar jQuery plugin. Features include vertical and/or horizontal scrollbar(s), adjustable scrolling momentum, mouse-wheel (via jQuery mousewheel plugin), keyboard and touch support, ready-to-use themes and customization via CSS, RTL direction support, option parameters for full control of scrollbar functionality, methods for triggering actions like scroll-to, update, destroy etc., user-defined callbacks and more.

view demo	*download*
Scrollbar themes	*Release archive*
more	*Project on GitHub*

Current version **3.1.5** (Changelog)
Upgrading from version 2

How to use it

Get started by downloading the archive which contains the plugin files (and a large amount of HTML demos and examples). Extract and upload *jquery.mCustomScrollbar.concat.min.js*, *jquery.mCustomScrollbar.css* and *mCSB_buttons.png* to your web server (alternatively you can load plugin files from a CDN).

HTML

Include *jquery.mCustomScrollbar.css* in the head tag your HTML document (more info)

```
<link rel="stylesheet" href="/path/to/jquery.mCustomScrollbar.css" />
```

Include jQuery library (if your project doesn't use it already) and *jquery.mCustomScrollbar.concat.min.js* in the head tag or at the very bottom of your document, just before the closing body tag

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.mi
n.js"></script>
<script src="/path/to/jquery.mCustomScrollbar.concat.min.js"></script>
```

CSS

The element(s) you want to add scrollbar(s) should have the typical CSS properties of an overflowed block which are a height (or max-height) value, an overflow value of auto (or hidden) and content long enough to require scrolling. For horizontal scrollbar, the element should have a width (or max-width) value set.

If you prefer to set your element's height/width via javascript, you can use the `setHeight`/`setWidth` option parameters.

Initialization

Initialize via javascript
After files inclusion, call *mCustomScrollbar* function on the element selector you want to add the scrollbar(s)

```
<script>
```

```
    (function($){
        $(window).on("load",function(){
            $(".content").mCustomScrollbar();
        });
    })(jQuery);
</script>
```

more info

Initialize via HTML
Add the class `mCustomScrollbar` to any element you want to add custom scrollbar(s) with default options. Optionally, set its axis via the HTML data attribute `data-mcs-axis` (e.g. `"x"` for horizontal and `"y"` for vertical) and its theme via `data-mcs-theme`. For example:

```
<div class="mCustomScrollbar" data-mcs-theme="dark">
  <!-- your content -->
</div>
```

Basic configuration & option parameters

axis
By default, the script applies a vertical scrollbar. To add a horizontal or 2-axis scrollbars, invoke mCustomScrollbar function with the axis option set to `"x"` or `"yx"` respectively

```
$(".content").mCustomScrollbar({
    axis:"x" // horizontal scrollbar
});
```

```
$(".content").mCustomScrollbar({
    axis:"yx" // vertical and horizontal scrollbar
});
```

theme
To quickly change the appearance of the scrollbar, set the theme option parameter to any of the ready-to-use themes available in jquery.mCustomScrollbar.css, for example:

```
$(".content").mCustomScrollbar({
    theme:"dark"
});
```

Configuration

You can configure your scrollbar(s) using the following option parameters on *mCustomScrollbar* function
Usage `$(selector).mCustomScrollbar({ option: value });`

`setWidth: false`	Set the width of your content (overwrites CSS width), value in pixels (integer) or percentage (string).
`setHeight: false`	Set the height of your content (overwrites CSS height), value in pixels (integer) or percentage (string).
`setTop: 0`	Set the initial css top property of content, accepts string values (css top position). Example: `setTop: "-100px"`.
`setLeft: 0`	Set the initial css left property of content, accepts string values (css left position). Example: `setLeft: "-100px"`.
`axis: "string"`	Define content’s scrolling axis (the type of scrollbars added to the element: vertical and/of horizontal). Available values: `"y"`, `"x"`, `"yx"`. ◦ `axis: "y"` – vertical scrollbar (default) ◦ `axis: "x"` – horizontal scrollbar ◦ `axis: "yx"` – vertical and horizontal scrollbars
`scrollbarPosition: "string"`	Set the position of scrollbar in relation to content. Available values: `"inside"`, `"outside"`. Setting `scrollbarPosition: "inside"` (default) makes scrollbar appear inside the element. Setting `scrollbarPosition: "outside"` makes scrollbar appear outside the element. Note that setting the value to `"outside"` requires your element (or parent elements) to have CSS `position: relative` (otherwise the scrollbar will be positioned in relation to document’s root element).
`scrollInertia: integer`	Set the amount of scrolling momentum as animation duration in milliseconds. Higher value equals greater scrolling momentum which translates to smoother/more progressive animation. Set to `0` to disable.
`autoDraggerLength: boolean`	Enable or disable auto-adjusting scrollbar dragger length in relation to scrolling amount (same bahavior with browser’s native scrollbar). Set `autoDraggerLength: false` when you want your scrollbar to (always) have a fixed size.

`autoHideScrollbar: boolean`	Enable or disable auto-hiding the scrollbar when inactive. Setting `autoHideScrollbar: true` will hide the scrollbar(s) when scrolling is idle and/or cursor is out of the scrolling area. Please note that some special themes like “minimal” overwrite this option.
`autoExpandScrollbar: boolean`	Enable or disable auto-expanding the scrollbar when cursor is over or dragging the scrollbar.
`alwaysShowScrollbar: integer`	Always keep scrollbar(s) visible, even when there’s nothing to scroll. ◦ `alwaysShowScrollbar: 0` – disable (default) ◦ `alwaysShowScrollbar: 1` – keep dragger rail visible ◦ `alwaysShowScrollbar: 2` – keep all scrollbar components (dragger, rail, buttons etc.) visible
`snapAmount: integer`	Make scrolling snap to a multiple of a fixed number of pixels. Useful in cases like scrolling tabular data, image thumbnails or slides and you need to prevent scrolling from stopping half-way your elements. Note that your elements must be of equal width or height in order for this to work properly. To set different values for vertical and horizontal scrolling, use an array: `[y,x]`
`snapOffset: integer`	Set an offset (in pixels) for the snapAmount option. Useful when for example you need to offset the snap amount of table rows by the table header.
`mouseWheel:{ enable: boolean }`	Enable or disable content scrolling via mouse-wheel.
`mouseWheel:{ scrollAmount: integer }`	Set the mouse-wheel scrolling amount (in pixels). The default value `"auto"` adjusts scrolling amount according to scrollable content length.
`mouseWheel:{ axis: "string" }`	Define the mouse-wheel scrolling axis when both vertical and horizontal scrollbars are present. Set `axis: "y"` (default) for vertical or `axis: "x"` for horizontal scrolling.
`mouseWheel:{ preventDefault: boolean }`	Prevent the default behaviour which automatically scrolls the parent element when end or beginning of scrolling is reached (same bahavior with browser’s

	native scrollbar).
`mouseWheel:{ deltaFactor: integer }`	Set the number of pixels one wheel notch scrolls. The default value “auto” uses the OS/browser value.
`mouseWheel:{ normalizeDelta: boolean }`	Enable or disable mouse-wheel (delta) acceleration. Setting `normalizeDelta: true` translates mouse-wheel delta value to -1 or 1.
`mouseWheel:{ invert: boolean }`	Invert mouse-wheel scrolling direction. Set to `true` to scroll down or right when mouse-wheel is turned upwards.
`mouseWheel:{ disableOver: [array] }`	Set the tags that disable mouse-wheel when cursor is over them. Default value: `["select","option","keygen","datalist","textarea"]`
`scrollButtons:{ enable: boolean }`	Enable or disable scrollbar buttons.
`scrollButtons:{ scrollAmount: integer }`	Set the buttons scrolling amount (in pixels). The default value `"auto"` adjusts scrolling amount according to scrollable content length.
`scrollButtons:{ scrollType: "string" }`	Define the buttons scrolling type/behavior. ◦ `scrollType: "stepless"` – continuously scroll content while pressing the button (default) ◦ `scrollType: "stepped"` – each button click scrolls content by a certain amount (defined in scrollAmount option above)
`scrollButtons:{ tabindex: integer }`	Set a tabindex value for the buttons.
`keyboard:{ enable: boolean }`	Enable or disable content scrolling via the keyboard. The plugin supports the directional arrows (top, left, right and down), page-up (PgUp), page-down (PgDn), Home and End keys.
`keyboard:{ scrollAmount: integer }`	Set the keyboard arrows scrolling amount (in pixels). The default value `"auto"` adjusts scrolling amount according to scrollable content length.

`keyboard:{ scrollType: "string" }`	Define the keyboard arrows scrolling type/behavior. ◦ `scrollType: "stepless"` – continuously scroll content while pressing the arrow key (default) ◦ `scrollType: "stepped"` – each key release scrolls content by a certain amount (defined in scrollAmount option above)
`contentTouchScroll: integer`	Enable or disable content touch-swipe scrolling for touch-enabled devices. To completely disable, set `contentTouchScroll: false`. Integer values define the axis-specific minimum amount required for scrolling momentum (default: `25`).
`documentTouchScroll: boolean`	Enable or disable document touch-swipe scrolling for touch-enabled devices.
`advanced:{ autoExpandHorizontalScroll: boolean }`	Auto-expand content horizontally (for `"x"` or `"yx"` axis). If set to `true`, content will expand horizontally to accommodate any floated/inline-block elements. Setting its value to `2` (integer) forces the non scrollHeight/scrollWidth method. A value of `3` forces the scrollHeight/scrollWidth method.
`advanced:{ autoScrollOnFocus: "string" }`	Set the list of elements/selectors that will auto-scroll content to their position when focused. For example, when pressing TAB key to focus input fields, if the field is out of the viewable area the content will scroll to its top/left position (same bahavior with browser's native scrollbar). To completely disable this functionality, set `autoScrollOnFocus: false`. Default: `"input,textarea,select,button,datalist,keygen,a[tabindex],area,object,[contenteditable='true']"`
`advanced:{ updateOnContentResize: boolean }`	Update scrollbar(s) automatically on content, element or viewport resize. The value should be `true` (default) for fluid layouts/elements, adding/removing content dynamically, hiding/showing elements etc.
`advanced:{ updateOnImag`	Update scrollbar(s) automatically each time an image

`eLoad: boolean }`	inside the element is fully loaded. Default value is `auto` which triggers the function only on `"x"` and `"yx"` axis (if needed). The value should be `true` when your content contains images and you need the function to trigger on any axis.
`advanced:{ updateOnSelectorChange: "string" }`	Update scrollbar(s) automatically when the amount and size of specific selectors changes. Useful when you need to update the scrollbar(s) automatically, each time a type of element is added, removed or changes its size. For example, setting `updateOnSelectorChange: "ul li"` will update scrollbars each time list-items inside the element are changed. Setting the value to `true`, will update scrollbars each time any element is changed. To disable (default) set to `false`.
`advanced:{ extraDraggableSelectors: "string" }`	Add extra selector(s) that'll release scrollbar dragging upon mouseup, pointerup, touchend etc. Example: `extraDraggableSelectors: ".myClass, #myID"`
`advanced:{ releaseDraggableSelectors: "string" }`	Add extra selector(s) that'll allow scrollbar dragging upon mousemove/up, pointermove/up, touchend etc. Example: `releaseDraggableSelectors: ".myClass, #myID"`
`advanced:{ autoUpdateTimeout: integer }`	Set the auto-update timeout in milliseconds. Default timeout: `60`
`theme: "string"`	Set the scrollbar theme. View all ready-to-use themes All themes are contained in plugin's CSS file (jquery.mCustomScrollbar.css). Default theme: `"light"`

`callbacks:{`
`      onCreate: function(){}`
`}`
	A function to call when plugin markup is created.
	Example:
	`callbacks:{`
	`    onCreate:function(){`
	`      console.log("Plugin markup generated");`
	`    }`
	`}`

`callbacks:{`	A function to call when scrollbars have initialized

`onInit: function(){}` `}`	(demo). Example: `callbacks:{ onInit:function(){ console.log("Scrollbars initialized"); } }`
`callbacks:{ onScrollStart: function(){} }`	A function to call when scrolling starts (demo). Example: `callbacks:{ onScrollStart:function(){ console.log("Scrolling started..."); } }`
`callbacks:{ onScroll: function(){} }`	A function to call when scrolling is completed (demo). Example: `callbacks:{ onScroll:function(){ console.log("Content scrolled..."); } }`
`callbacks:{ whileScrolling: function(){} }`	A function to call while scrolling is active (demo). Example: `callbacks:{ whileScrolling:function(){ console.log("Scrolling..."); } }`
`callbacks:{ onTotalScroll: function(){} }`	A function to call when scrolling is completed and content is scrolled all the way to the end (bottom/right) (demo). Example: `callbacks:{ onTotalScroll:function(){ console.log("Scrolled to end of content."); } }`
`callbacks:{ onTotalScrollBack: function(){} }`	A function to call when scrolling is completed and content is scrolled back to the beginning (top/left) (demo). Example: `callbacks:{ onTotalScrollBack:function(){`

	`console.log("Scrolled back to the beginning of content.");`
	`}`
	`}`

`callbacks:{`
`onTotalScrollOffset: integer`
`}`
	Set an offset for the onTotalScroll option.
	For example, setting `onTotalScrollOffset: 100` will trigger the onTotalScroll callback 100 pixels before the end of scrolling is reached.

`callbacks:{`
`onTotalScrollBackOffset: integer`
`}`
	Set an offset for the onTotalScrollBack option.
	For example, setting `onTotalScrollBackOffset: 100` will trigger the onTotalScrollBack callback 100 pixels before the beginning of scrolling is reached.

`callbacks:{`
`alwaysTriggerOffsets: boolean`
`}`
	Set the behavior of calling onTotalScroll and onTotalScrollBack offsets.
	By default, callback offsets will trigger repeatedly while content is scrolling within the offsets.
	Set `alwaysTriggerOffsets: false` when you need to trigger onTotalScroll and onTotalScrollBack callbacks once, each time scroll end or beginning is reached.

`callbacks:{`
`onOverflowY: function(){}`
`}`
	A function to call when content becomes long enough and vertical scrollbar is added.
	Example:
	`callbacks:{`
	`onOverflowY:function(){`
	`console.log("Vertical scrolling required");`
	`}`
	`}`

`callbacks:{`
`onOverflowX: function(){}`
`}`
	A function to call when content becomes wide enough and horizontal scrollbar is added.
	Example:
	`callbacks:{`
	`onOverflowX:function(){`
	`console.log("Horizontal scrolling required");`
	`}`
	`}`

`callbacks:{`
`onOverflowYNone: function(){}`
`}`
	A function to call when content becomes short enough and vertical scrollbar is removed.
	Example:
	`callbacks:{`
	`onOverflowYNone:function(){`

	`console.log("Vertical scrolling is no`
	`t required");`
	`}`
	`}`

`callbacks:{`
`onOverflowXNone:`
`function(){}`
`}`
	A function to call when content becomes narrow enough and horizontal scrollbar is removed.
	Example:
	`callbacks:{`
	`onOverflowXNone:function(){`
	`console.log("Horizontal scrolling is`
	`not required");`
	`}`
	`}`

`callbacks:{`
`onBeforeUpdate: f`
`unction(){}`
`}`
	A function to call right before scrollbar(s) are updated.
	Example:
	`callbacks:{`
	`onBeforeUpdate:function(){`
	`console.log("Scrollbars will updat`
	`e");`
	`}`
	`}`

`callbacks:{`
`onUpdate: functio`
`n(){}`
`}`
	A function to call when scrollbar(s) are updated.
	Example:
	`callbacks:{`
	`onUpdate:function(){`
	`console.log("Scrollbars updated");`
	`}`
	`}`

`callbacks:{`
`onImageLoad: func`
`tion(){}`
`}`
	A function to call each time an image inside the element is fully loaded and scrollbar(s) are updated.
	Example:
	`callbacks:{`
	`onImageLoad:function(){`
	`console.log("Image loaded");`
	`}`
	`}`

`callbacks:{`
`onSelectorChange:`
`function(){}`
`}`
	A function to call each time a type of element is added, removed or changes its size and scrollbar(s) are updated.
	Example:
	`callbacks:{`
	`onSelectorChange:function(){`
	`console.log("Scrollbars updated");`
	`}`

	}
`live: "string"`	Enable or disable applying scrollbar(s) on all elements matching the current selector, now and in the future. Set `live: true` when you need to add scrollbar(s) on elements that do not yet exist in the page. These could be elements added by other scripts or plugins after some action by the user takes place (e.g. lightbox markup may not exist untill the user clicks a link). If you need at any time to disable or enable the live option, set `live: "off"` and `"on"` respectively. You can also tell the script to disable live option after the first invocation by setting `live: "once"`.
`liveSelector: "string"`	Set the matching set of elements (instead of the current selector) to add scrollbar(s), now and in the future.

Plugin methods

Ways to execute various plugin actions programmatically from within your script(s).

update

Usage `$(selector).mCustomScrollbar("update");`

Call the *update* method to **manually update existing scrollbars** to accommodate new content or resized element(s). This method is by default called automatically by the script (via `updateOnContentResize` option) when the element itself, its content or scrollbar size changes.

view examples

scrollTo

Usage `$(selector).mCustomScrollbar("scrollTo",position,options);`

Call the *scrollTo* method to **programmatically scroll the content** to the position parameter (demo).

position parameter

Position parameter can be:

- `"string"`
  - e.g. element selector: `"#element-id"`

        - e.g. special pre-defined position: `"bottom"`
        - e.g. number of pixels less/more: `"-=100"` / `"+=100"`
    - `integer`
        - e.g. number of pixels: `100`
    - `[array]`
        - e.g. different y/x position: `[100,50]`
    - `object/function`
        - e.g. jQuery object: `$("#element-id")`
        - e.g. js object: `document.getelementbyid("element-id")`
        - e.g. function: `function(){ return 100; }`

Pre-defined position strings:

- `"bottom"` – scroll to bottom
- `"top"` – scroll to top
- `"right"` – scroll to right
- `"left"` – scroll to left
- `"first"` – scroll to the position of the first element within content
- `"last"` – scroll to the position of the last element within content

view examples

Method options

`scrollInertia: integer`	Scroll-to duration, value in milliseconds.
	Example:
	`$(selector).mCustomScrollbar("scrollTo","bottom",{`
	`    scrollInertia:3000`
	`});`

`scrollEasing: "string"`	Scroll-to animation easing, values: `"linear"`, `"easeOut"`, `"easeInOut"`.
	Example:
	`$(selector).mCustomScrollbar("scrollTo","bottom",{`
	`    scrollEasing:"easeOut"`
	`});`

`moveDragger: boolean`	Scroll scrollbar dragger (instead of content).
	Example:
	`$(selector).mCustomScrollbar("scrollTo",80,{`
	`    moveDragger:true`
	`});`

`timeout: integer`	Set a timeout for the method (the default timeout is 60 ms in order to work with automatic scrollbar update), value in milliseconds.

	Example:
	`$(selector).mCustomScrollbar("scrollTo","top",{`
	`    timeout:1000`
	`});`

`callbacks: boolean`	Trigger user defined callbacks after scroll-to completes.
	Example:
	`$(selector).mCustomScrollbar("scrollTo","left",{`
	`    callbacks:false`
	`});`

stop

Usage `$(selector).mCustomScrollbar("stop");`

Stops any running scrolling animations (usefull when you wish to interupt a previously *scrollTo* method call).

disable

Usage `$(selector).mCustomScrollbar("disable");`

Calling *disable* method will **temporarily disable the scrollbar** (demo). Disabled scrollbars can be re-enable by calling the *update* method.

To disable the scrollbar and reset its content position, set the method’s *reset* parameter to `true`

```
$(selector).mCustomScrollbar("disable",true);
```

view examples

destroy

Usage `$(selector).mCustomScrollbar("destroy");`

Calling *destroy* method will **completely remove the custom scrollbar** and return the element to its original state (demo).

view examples

Scrollbar styling & themes

You can design and visually customize your scrollbars with **pure CSS**, using *jquery.mCustomScrollbar.css* which contains the default/basic styling and all scrollbar themes.

The easiest/quickest way is to select a ready-to-use scrollbar theme. For example:

```
$(selector).mCustomScrollbar({
  theme:"dark"
});
```

View all ready-to-use themes

You can modify the default styling or any theme either directly in *jquery.mCustomScrollbar.css* or by overwriting the CSS rules in another stylesheet.

Creating a new scrollbar theme

Create a name for your theme (e.g. "my-theme") and set it as the value of the *theme* option

```
$(selector).mCustomScrollbar({
    theme:"my-theme"
});
```

Your element will get the class "mCS-my-theme" (your theme-name with "mCS" prefix), so you can create your CSS using the `.mCS-my-theme` in your rules. For instance:

```
.mCS-my-theme.mCSB_scrollTools .mCSB_dragger .mCSB_dragger_bar{ background-color: red; }
.mCS-my-theme.mCSB_scrollTools .mCSB_draggerRail{ background-color: white; }
/* and so on... */
```

In the same manner you can clone any existing theme (e.g. "dark"), change its selector (e.g. `.mCS-dark`) to your own theme name (e.g. `.mCS-my-theme`) and modify its CSS rules.

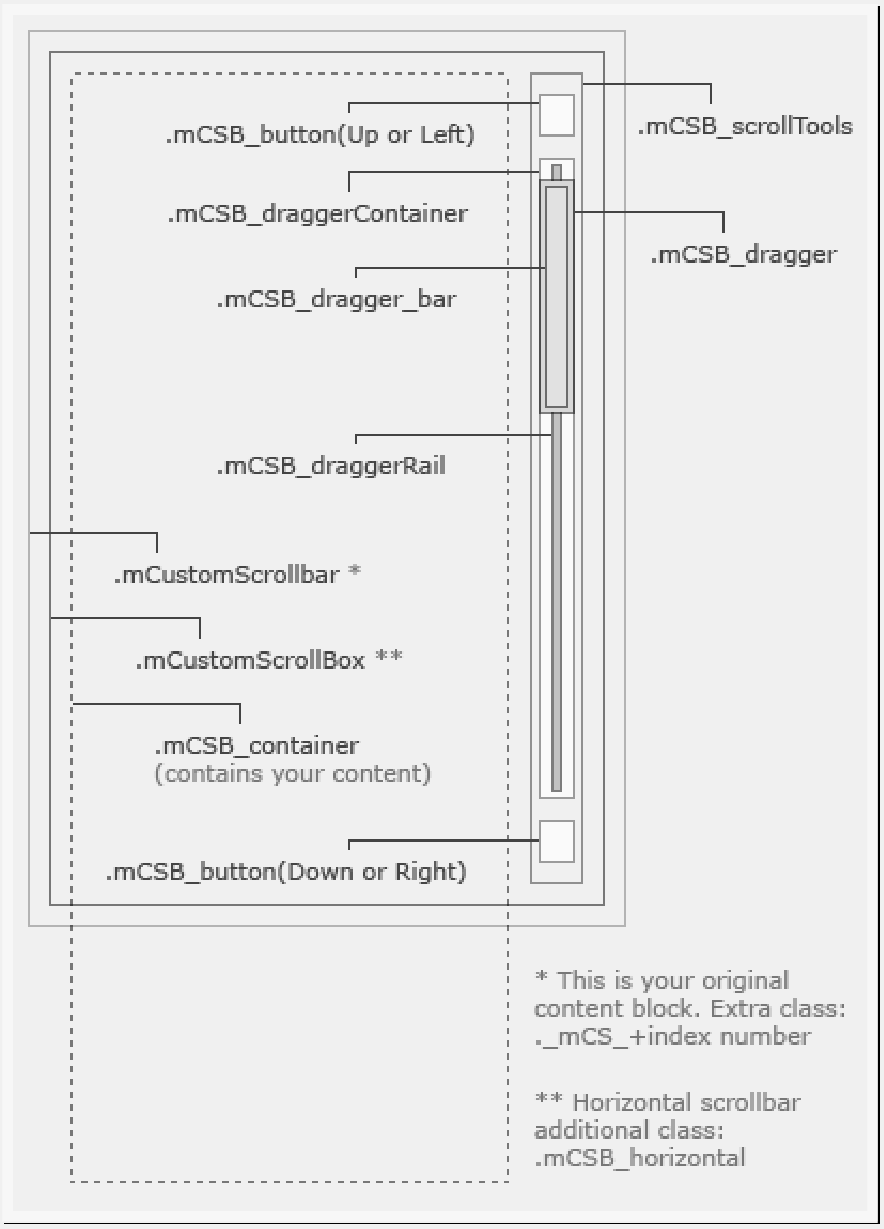
Scrollbar markup

The plugin applies specific id (unique) and/or classes to every scrollbar element/component, meaning that you can target and modify any scrollbar in more than one ways.

For example, every element with a scrollbar gets a unique class in the form of `_mCS_1`, `_mCS_2` etc. Every scrollbar container element gets a unique id in the form of `mCSB_1_scrollbar_vertical`, `mCSB_2_scrollbar_vertical` etc. Every scrollbar dragger gets a unique id in the form of `mCSB_1_dragger_vertical`, `mCSB_2_dragger_vertical` etc. in addition to the class `mCSB_dragger`. All these mean that you can do stuff like:

```
._mCS_1 .mCSB_dragger .mCSB_dragger_bar{ background-color: red; }

._mCS_2 .mCSB_dragger .mCSB_dragger_bar{ background-color: green; }
```

```
#mCSB_3_dragger_vertical .mCSB_dragger_bar{ background-color: blue; }

#mCSB_1_scrollbar_vertical .mCSB_dragger{ height: 100px; }

#mCSB_1_scrollbar_horizontal .mCSB_dragger{ width: 100px; }

.mCSB_1_scrollbar .mCSB_dragger .mCSB_draggerRail{ width: 4px; }
```



User-defined callbacks

You can trigger your own js function(s) by calling them inside *mCustomScrollbar* callbacks option parameter

```
$(".content").mCustomScrollbar({
    callbacks:{
        onScroll:function(){
            myCustomFn(this);
        }
    }
});

function myCustomFn(el){
    console.log(el.mcs.top);
}
```

In the example above, each time a scroll event ends and content has stopped scrolling, the content’s top position will be logged in browser’s console. There are available callbacks for each step of the scrolling event:

- `onScrollStart` – triggers the moment a scroll event starts
- `whileScrolling` – triggers while scroll event is running
- `onScroll` – triggers when a scroll event completes
- `onTotalScroll` – triggers when content has scrolled all the way to bottom or right
- `onTotalScrollBack` – triggers when content has scrolled all the way back to top or left

You can set an offset value (pixels) for both `onTotalScroll` and `onTotalScrollBack` by setting `onTotalScrollOffset` and `onTotalScrollBackOffset` respectively (view example).

By default, `onTotalScroll` and `onTotalScrollBack` callbacks are triggered repeatedly. To prevent multiple calls when content is within their offset, set `alwaysTriggerOffsets` option to `false` (view example).

Additional callbacks:

- `onInit`
- `onOverflowY`
- `onOverflowX`
- `onOverflowYNone`
- `onOverflowXNone`
- `onUpdate`
- `onImageLoad`
- `onSelectorChange`

Returning values

The script returns a number of values and objects related to scrollbar that you can use in your own functions

- `this` – the original element containing the scrollbar(s)
- `this.mcs.content` – the original content wrapper as jquery object
- `this.mcs.top` – content’s top position (pixels)
- `this.mcs.left` – content’s left position (pixels)
- `this.mcs.draggerTop` – scrollbar dragger’s top position (pixels)
- `this.mcs.draggerLeft` – scrollbar dragger’s left position (pixels)
- `this.mcs.topPct` – content vertical scrolling percentage
- `this.mcs.leftPct` – content horizontal scrolling percentage
- `this.mcs.direction` – content’s scrolling direction (y or x)

view examples

Plugin-specific jQuery expressions

`$("#myID:mcsInView")`	Select element(s) in your content that are within scrollable viewport. As condition: `$("#myID").is(":mcsInView");`
`$(".content:mcsOverflow")`	Select overflowed element(s) with visible scrollbar. As condition: `$(".content").is(":mcsOverflow");`
`$("#myID:mcsInSight")` `$("#myID:mcsInSight(exact)")`	Select element(s) in your content that are in view of the scrollable viewport. Using the `exact` parameter will include elements that have any part of them (even 1 pixel) in view of the scrollable viewport. As condition: `$("#myID").is(":mcsInSight");`, `$("#myID").is(":mcsInSight(exact)");`

Plugin dependencies & requirements

- jQuery version 1.6.0 or higher
- Mouse-wheel support
  - jQuery mousewheel plugin

License

Donating helps greatly in developing and updating free software and running this blog 🙂

Donate

Pages: 1 2 3 4

Tags
jquery

Posts related to this article

Add a custom scrollbar to your twitter widget
Responsive custom scrollbar with CSS3 media queries
Scroll to id within element with custom scrollbar(s)   Horizontal custom scrollbar tutorial

Previous: jQuery thumbnail scroller   Next: jQuery image panning

5,490 Comments

Post a comment

Comments pages: Previous 1 … 79 80 81

rizwan
Posted on November 22, 2017 at 12:26 *Permalink*

scrollbar not working in IE-Edge latest browser any solution how to overcome this problem??

Reply

Cyril
Posted on November 16, 2017 at 19:54 *Permalink*

Hi,

I've have an issue when i use this plugin with jquery sortable on scrolling up and down during sort.
With natural scrollbar no problem. But when the plugin is in use, the scroll doesn't work.

Any idea ?

Reply

pipes-manysman

Posted on November 15, 2017 at 10:42 *Permalink*

Hi
I'm using your jQuery Scrollbar and it works very well
thanks

Reply

Arun

Posted on November 8, 2017 at 22:01 *Permalink*

Hi,

I am loading a div's dynamically onclick of a button inside the scroll content (horizontal scroll). But if I have content above the scroll div and then I append the div, whole page jumps a little bit to top.Kind off annoying.

Any help?

Same issue could be recreated in our "max-width_example" too. If we add some dummy text on top of the scroll parent div and then click add image link, the page jumps slightly to top. Sorry if the issue is already been asked in this forum.

Reply

Andrea

Posted on October 28, 2017 at 17:11 *Permalink*

Hi guys, can you tell me why when I click on a input text of a form the site returns on the first page? Sorry for my bad english

Reply

Julien

Posted on October 6, 2017 at 11:47 *Permalink*

For those who came into the same issue (everything works except one finger scrolling on Mozilla Firefox 45) this is due to Mozilla configuration. Go into about:config then setting dom.w3c_pointer_events.enabled to true did the trick.

Reply

Comments pages: Previous 1 … 79 80 81

Post a comment

*Your e-mail is **never** published nor shared. Required fields are marked* *

Name *

E-mail *

Website

Comment

Post code (html, css, js etc.), using the code tag: <code>your code here...</code> (see info below)

You may use these HTML tags and attributes:
`<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em> <i> <q cite=""> <strike> <strong>`
You can write or copy/paste code directly in your comment using the `<code>` *tag:*
`<code>code here...</code>`
You may also use the `data-lang` *attribute to determine the code language like so:*
`<code data-lang-html>, <code data-lang-css>, <code data-lang-js> and <code data-lang-php>`

Post comment

> «Free software» means software that respects users' freedom and community. Roughly, the users have the freedom to run, copy, distribute, study, change and improve the software. With these freedoms, the users (both individually and collectively) control the program and what it does for them.
>
> — GNU, *The Free Software Definition*