malihu web design

jQuery custom content scroller

Last updated on Nov 17, 2015
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
Scrollbar themes
more

download
Release archive
Project on GitHub

Current version **3.1.3** (Changelog)
Upgrading from version 2

Get started *Configuration* *Methods* *Styling*

Callbacks *Code examples and tutorials* *FAQ*

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/jqu
<script src="/path/to/jquery.mCustomScrollbar.concat.min.js"></scri
```

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).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: "s`

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: intege`

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: bo`

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: bo`

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:`

Enable or disable auto-expanding the scrollbar when cursor is over or dragging the scrollbar.

`alwaysShowScrollbar:`

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:`

Enable or disable content scrolling via mouse-wheel.

`mouseWheel:{ scrollAm`

Set the mouse-wheel scrolling amount (in pixels). The default value `"auto"` adjusts scrolling amount according to scrollable content length.

`mouseWheel:{ axis: "s`

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:{ preventD`

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:{ deltaFac`

Set the number of pixels one wheel notch scrolls. The default value “auto” uses the OS/browser value.

`mouseWheel:{ normaliz`

Enable or disable mouse-wheel (delta) acceleration. Setting `normalizeDelta: true` translates mouse-wheel delta value to -1 or 1.

`mouseWheel:{ invert:`

Invert mouse-wheel scrolling direction. Set to `true` to scroll down or right when mouse-wheel is turned upwards.

`mouseWheel:{ disableC`

Set the tags that disable mouse-wheel when cursor is over them.
Default value:

```
["select","option","keygen","datalist",
```

`scrollButtons:{ enabl`

Enable or disable scrollbar buttons.

`scrollButtons:{ scrol`

Set the buttons scrolling amount (in pixels). The default value `"auto"` adjusts scrolling amount according to scrollable content length.

`scrollButtons:{ scrol`

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:{ tabir`

Set a tabindex value for the buttons.

`keyboard:{ enable: bc`

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:{ scrollAmou`

Set the keyboard arrows scrolling amount (in pixels). The default value `"auto"` adjusts scrolling amount according to scrollable content length.

`keyboard:{ scrollType`

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: i`

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:`

Enable or disable document touch-swipe scrolling for touch-enabled devices.

`advanced:{ autoExpand`

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:{ autoScroll`

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,`
`advanced:{ updateOnCo`	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:{ updateOnIm`	Update scrollbar(s) automatically each time an image 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:{ updateOnSe`	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:{ extraDragg`	Add extra selector(s) that'll release scrollbar dragging upon mouseup, pointerup, touchend etc. Example: `extraDraggableSelectors: ".myClass, #myID"`
`advanced:{ releaseDra`	Add extra selector(s) that'll allow scrollbar dragging upon mousemove/up, pointermove/up, touchend etc. Example: `releaseDraggableSelectors: ".myClass, #myID"`

```
advanced:{ autoUpdate
```

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: funct
}
```

A function to call when plugin markup is created.
Example:

```
callbacks:{
    onCreate:function(){
      console.log("Plugin markup genera
    }
}
```

```
callbacks:{
      onInit: functic
}
```

A function to call when scrollbars have initialized (demo).
Example:

```
callbacks:{
    onInit:function(){
      console.log("Scrollbars initializ
    }
}
```

```
callbacks:{
      onScrollStart:
}
```

A function to call when scrolling starts (demo).
Example:

```
callbacks:{
    onScrollStart:function(){
      console.log("Scrolling started...
    }
}
```

```
callbacks:{
      onScroll: funct
}
```

A function to call when scrolling is completed (demo).
Example:

```
callbacks:{
    onScroll:function(){
      console.log("Content scrolled..."
    }
}
```

```
callbacks:{
	whileScrolling:
}
```

A function to call while scrolling is active (demo).
Example:

```
callbacks:{
    whileScrolling:function(){
      console.log("Scrolling...");
    }
}
```

```
callbacks:{
	onTotalScroll:
}
```

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 c
    }
}
```

```
callbacks:{
	onTotalScrollBa
}
```

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
    }
}
```

```
callbacks:{
	onTotalScrollOf
}
```

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:{
	onTotalScrollBa
}
```

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:{
	alwaysTriggerOf
}
```

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: fu
}
```

A function to call when content becomes long enough and vertical scrollbar is added.
Example:

```
callbacks:{
    onOverflowY:function(){
      console.log("Vertical scrolling r
    }
}
```

```
callbacks:{
	onOverflowX: fu
}
```

A function to call when content becomes wide enough and horizontal scrollbar is added.
Example:

```
callbacks:{
    onOverflowX:function(){
      console.log("Horizontal scrolling
    }
}
```

```
callbacks:{
	onOverflowYNone
}
```

A function to call when content becomes short enough and vertical scrollbar is removed.
Example:

```
callbacks:{
    onOverflowYNone:function(){
      console.log("Vertical scrolling i
    }
}
```

```
callbacks:{
	onOverflowXNone
}
```

A function to call when content becomes narrow enough and horizontal scrollbar is removed.
Example:

```
callbacks:{
    onOverflowXNone:function(){
      console.log("Horizontal scrolling
    }
}
```

```
callbacks:{
      onBeforeUpdate:
}
```

A function to call right before scrollbar(s) are updated.
Example:

```
callbacks:{
    onBeforeUpdate:function(){
      console.log("Scrollbars will upda
    }
}
```

```
callbacks:{
      onUpdate: funct
}
```

A function to call when scrollbar(s) are updated.
Example:

```
callbacks:{
    onUpdate:function(){
      console.log("Scrollbars updated")
    }
}
```

```
callbacks:{
      onImageLoad: fu
}
```

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:{
      onSelectorChang
}
```

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: intege`

Scroll-to duration, value in milliseconds.
Example:

```
$(selector).mCustomScrollbar("scrollTo"
    scrollInertia:3000
});
```

`scrollEasing: "string`

Scroll-to animation easing, values: `"linear"`, `"easeOut"`, `"easeInOut"`.
Example:

```
$(selector).mCustomScrollbar("scrollTo"
    scrollEasing:"easeOut"
});
```

`moveDragger: boolean`

Scroll scrollbar dragger (instead of content).
Example:

```
$(selector).mCustomScrollbar("scrollTo"
    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"
    timeout:1000
});
```

`callbacks: boolean`

Trigger user defined callbacks after scroll-to completes.
Example:

```
$(selector).mCustomScrollbar("scrollTo"
    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{ bac
.mCS-my-theme.mCSB_scrollTools .mCSB_draggerRail{ background-color:
/* and so on... */
```

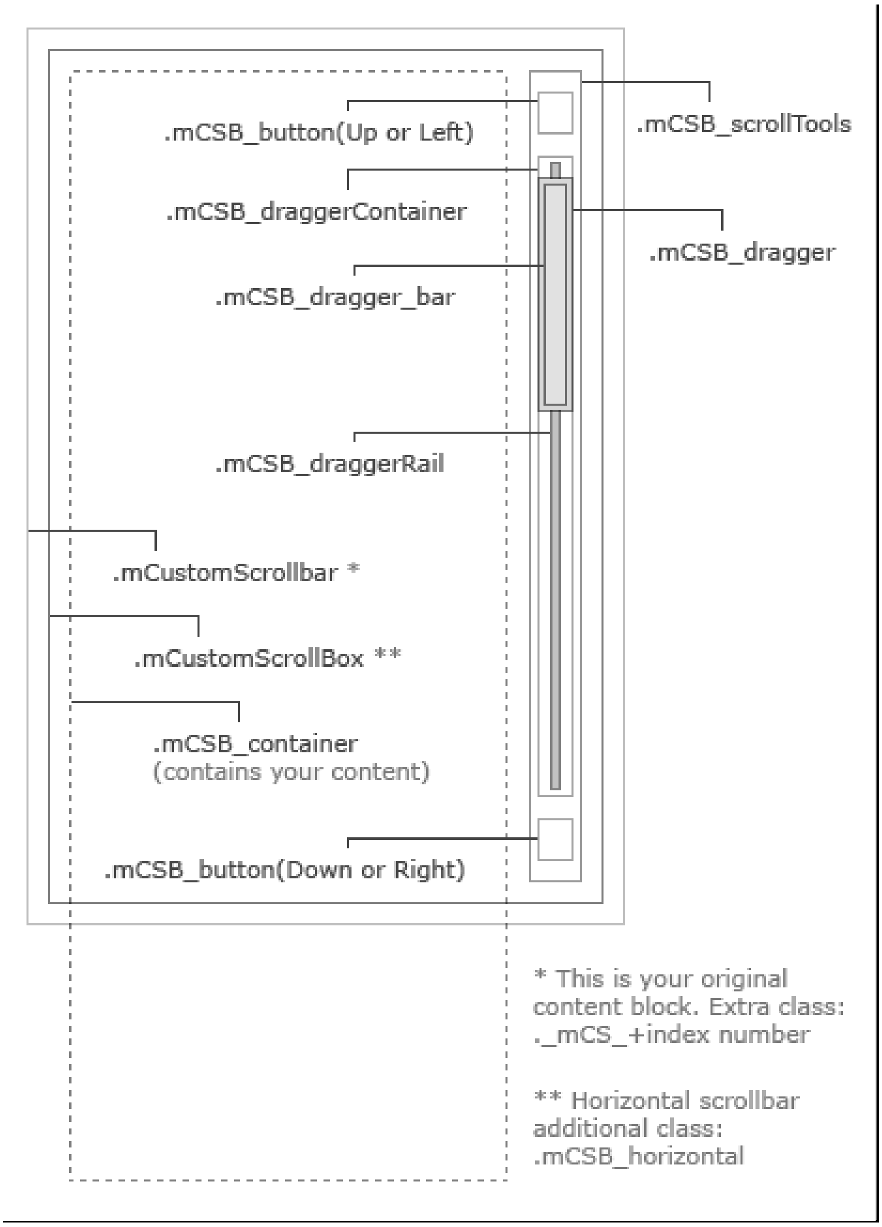
In the same manner you can clone any existing theme (e.g. “dark”), change its selector (e.g. `.mCS-dark`) to your own theme name (e.g. `.mCS-my-theme`) and modify its CSS rules.

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");`
`$(".content:mcsOverfl`	Select overflowed element(s) with visible scrollbar. As condition: `$(".content").is(":mcsOverflow");`

Plugin dependencies & requirements

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

License

This work is released under the MIT License.
You are free to use, study, improve and modify it wherever and however you like.
http://opensource.org/licenses/MIT

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)

Previous: jQuery thumbnail scroller Next: jQuery image panning

5,150 Comments

Post a comment

Comments pages: Previous 1 … 72 73 74

mrPaqui

Posted on March 14, 2016 at 02:38 *Permalink*

This is awesome!

Is there a function to add a bounce/elastic effect to the scroll?
I gave tried to add this script in but they don't work together.
http://www.onextrapixel.com/examples/scroll-overflow-effect-with-fancyscroll/index.html

Reply

Saurabh Kapoor

Posted on March 4, 2016 at 11:32 *Permalink*

possible To Focus on Active Class inside the scrolling Element

Please Sir Help ASAP on my Email

Reply

jaman7

Posted on March 8, 2016 at 21:59 *Permalink*

plugin works perfectly, but I have a little problem with the gallery, vertical scroller appears when the content on it is loaded,
http://jsfiddle.net/su1y5gct/embedded/result/

```
<div id="content-wz" class="content">
         <ul class="gallery">
                     <li><a href="#"><img src=""  a
                     .
                    <li><a href="#"><img src=""  al
                    </ul>
        <div style="clear:both"></div>
                </div>

(function($){
      $(window).load(function(){

        $("#content-wz").mCustomScrollbar({
          scrollButtons:{enable:true},
          theme:"dark-thick",
          scrollbarPosition:"outside"
        });
        $(".content:mcsOverflow")
      });
    })(jQuery);
```

Reply

Leo D.

Posted on March 4, 2016 at 04:06 *Permalink*

I just upgraded from an older version and in Safari, the scrollbars are appearing in the middle of the content rather than on the right side. The scrollbars appear correctly in chrome & IE.
Any ideas on what might be causing the issue?

Reply

Avi

Posted on March 3, 2016 at 00:52 *Permalink*

Hi Malihu,

Thanks for the great plugin. I had an issue with using the scrollbar in bootstrap modal on touch devices. This issue persists on iOS devices and I don't know about other devices. So here's the issue:

Using the scrollbar on the modal, when u swipe to the end of the scroll and try to scroll more, the background gets scrolled, its like the focus is lost. This can also be seen on the example you have for modal. (http://manos.malihu.gr/repository/custom-scrollbar/demo/examples/bootstrap_examples-extended.html).

If you have a solution, that would be great.

Thanks.

-Avi

Reply

François

Posted on February 28, 2016 at 00:50 *Permalink*

Hi 🙂
I'm using your jQuery Scrollbar and it works very well, thank you.
Currently I wont to implement an automatic paging which expands a container.

So can I determine wether I scrolled past a certain DOM-element?

Reply

Jeremy

Posted on February 27, 2016 at 13:54 *Permalink*

Can I override the colours? I have my own theme.css and would like these to drive the colours rather than rely on setting the scrollbar's theme through javascript.

Reply

Jeremy

Posted on February 27, 2016 at 13:10 *Permalink*

With autoHideScrollbar set is there any way I can make my content use the full width of the div? It seems to be reserving space for the standard scrollbar that would be appearing. I don't mind if when the custom scrollbar appears it overlays my content.

```
sa.mCustomScrollbar({
        axis: "y",
        autoHideScrollbar: true,
        autoExpandScrollbar: true,
        mouseWheel: { enable: true },
        scrollInertia: 0
    });
```

Reply

JC

Posted on February 26, 2016 at 20:54 *Permalink*

Is there any way to do a smooth scroll when both link and anchor are within the scroll container.

For example:

<div class="mCustomScrollbar
Go here
….
lots of content
…

This content down here

Reply

Prasoon

Posted on February 23, 2016 at 10:24 *Permalink*

Does this plugin work with textarea and contenteditable divs?
I could not make it work with either.

Reply

Dean

Posted on February 22, 2016 at 03:02 *Permalink*

Hi Manos,

I am in the process of trying your ajax scroller demo locally (on Mac). I am getting the extra content to load, in Safari, but Firefox and FirefoxDeveloper don't want to play along.

Are you aware of this?

Reply

Dean

Posted on February 27, 2016 at 01:32 *Permalink*

I was able to get it running. I am now looking to a way to incrementally load into the scroller more text (by chapter), both before and after the initial ajax load. Would you help me to do that?

Reply

Asher Indursky

Posted on February 21, 2016 at 19:23 *Permalink*

Hi.
I use it for a whole page and snapAmount for Full Height Sections in the Page.
how i can to change the SnapAmount when Window resize?

Reply

Erik

Posted on February 17, 2016 at 09:40 *Permalink*

Hey malihu,

This plugin works great but I am having one problem. I'm trying to port a a chat application from jquery to react.js and I have been running into a few problems. The plugin works great when there is no dynamic data but I can't add data into the scrollbar div. The reason why I can't do this is because when I call .mCustomScrollbar on an element, a child element gets created with the id of MCSB_1 or something like that. In my jquery app I just did $("#MCSB_1").innerHTML += "newmessagegoeshere" but I can't really even touch the DOM because it screws up with react components.

Any help would be greatly appreciated!

Reply

promise

Posted on February 16, 2016 at 09:31 *Permalink* hello, thanks for this awesome plugin Malihu.

i have one small issue. i have div containing two rows, each row with 2 products images and pricing and i want it to scroll horizontally, when i use your plugin, it automatically turns it into one horizontal row.

please, how can i fix this.

Reply

promise

Posted on February 16, 2016 at 10:02 *Permalink*

Never mind, solved it. Was my mistake

Reply

PooMer

Posted on February 14, 2016 at 22:32 *Permalink*

I'm using wordpress nad I know how to enqueue styles and scripts from a CDN but I don't know how to use "//malihu.github.io/custom-scrollbar/mCSB_buttons.png" in my function.php.
Any help would be appreciated

Reply

Shane Phillips

Posted on February 12, 2016 at 01:39 *Permalink*

This really is amazing functionality. Have you ever thought of translating this into a WordPress plugin? With the popularity of WP and the simple execution your scroller brings to the table, you could be up and running in a matter of hours!

Reply

daeyoonkim

Posted on February 11, 2016 at 10:41 *Permalink*

Mcustomscrollbar not want to have to move. When the mouseover the content

Reply

Mark-xestudio

Posted on February 9, 2016 at 07:05 *Permalink*

Love your work. but im having a bit of a problem.

http://stackoverflow.com/questions/35283943/custom-scrollbar-gets-loaded-but-not-visible/35284018#35284018

literally, the scroll bar adds class so the body, but nothing visible.

can anyone help me out?

Reply

Mark-xestudio

Posted on February 12, 2016 at 03:27 *Permalink*

It works fine for now. but now i gotta figure out how to disable scrollbars for the iframe.

Reply

SteveL

Posted on February 9, 2016 at 03:09 *Permalink*

Ok…been having a problem trying to get the scrollbar to work properly with this ajax call. It looks like the scrollbar is initializing and creating a blank space above the feed once the feed is ready to load. If I use the scrollbar and create a feed locally…its fine, because its fast enough to get the data feed and the scrollbar renders fine.

Any hints.ideas I can try to implement this scrollbar with this ajax call would be helpful….have tried looking on the site for any information on how to do this…and have tried several things…but still not working for me…gotta be simple…but not seeing it.

```
var getNews = $.ajax({
  beforeSend: function (){
    //Start the Load Spinner while getting the feed info
    $('#newsloader').show();
  },
  url: 'URL Path',
  dataType: 'json',
  type: 'GET',
  cache: false
});

getNews.done(function(nwsdata) {
  $.each(nwsdata, function(index, newsData) {
    //  Change limit number to display number of items t
    var newslimit = 4;
    if (index < newslimit){
    var news = '<li class="newsArticle"><i class="fa-li
      '<div class="newsHeadline"><a href="Path to articl
      '<div class="newsDate">Published: ' + newsData.rel
      '<div class="newsBody">' + newsData.release.body +
    $("#newsContainer").append(news);
    }
    $("div.newsBody").text(function(index, newsBodyText)
      return newsBodyText.substr(0, 170) + '...' ;
    });
  });

  // If there are no news articles to display message
  if ($('#newsContainer li').length == 0) {
    $('#newsContainer').html('<li class="noFeed">News is
  }
  //Hide the Load Spinner when the feed is loaded
  $('#newsloader').hide();

});

getNews.fail(function(){
   $('#newsContainer').html('<li class="noFeed">News dat
});
```

Reply

SteveL

Posted on February 11, 2016 at 05:19 Permalink

ok…I believe I figured this out….and maybe this will help someone else. What I did was to change the window load to a document.ready function and then wrapped that in a function and called it showScrollbar. Then after the json file was loaded…..called that function and it loaded the scrollbars.

showScrollbar();

function showScrollbar(){
(function($){
$(document).ready(function(){
$("#newsContainer").mCustomScrollbar({
theme:"3d-dark",
scrollButtons:{
enable:true
}
});
});
})(jQuery);
}

If there is a better way, let me know…but this worked for me in my testing.

Reply

Raf G

Posted on February 8, 2016 at 00:48 [Permalink]

Hello Malihu

Your plugin has been a plugin to work with but I have having problems porting an app to React.js. Right now, the scrollbar is only converting to an mCustomScrollbar when I put it in a setTimeout. Here is the code below:

```
class Scrollbar extends React.Component  {
       constructor(props)  {
           super(props);
       }
       componentDidMount()  {
             setTimeout(() =>  {

      $(this.refs.scrollbarDiv).mCustomScrollbar();
      console.log("Scrollbar converts after 825 millisec
    } , 825);
       }

       render()  {
          return (
             <div>
                <div id="scrollbarDiv" ref="scrollbarDiv
             </div>
          );
       }
   }
```

Is there any way that I can make this go instantly?

Thanks!

Reply

Viplov

Posted on February 6, 2016 at 13:16 [Permalink]

Hi,

The scrollbar does not auto position itself when I use the browser's ctrl-f feature.

For ex: If I type “John” and its not visible on the screen now, usually it would auto scroll to the text. But that doesn’t seem to be the case here.

Is this something you could help with?

Reply

Troy Zirk

Posted on February 5, 2016 at 22:48 *Permalink*

How would I auto-scroll/pan without focus? I want to be able to control the scrolling interval, speed and inertia parameters dynamically, I believe.

Reply

Dasra Khadka

Posted on February 4, 2016 at 20:25 *Permalink*

I was wondering if there is a way to fixed one heading in your plugin. I was trying to have a fixed header while other elements can be scroll able.

Reply

Samir

Posted on February 4, 2016 at 12:43 *Permalink*

Great Plugin-in!

Thank you.

Reply

max

Posted on February 4, 2016 at 12:19 *Permalink*

Sorry, I do not speak English very well

I would want to know if the plugin have an implementation of keyboard scroll with mcustomscrollbar under typescript. i use the mcustomscrollbar.d.ts

Reply

max

Posted on February 11, 2016 at 13:00 *Permalink*

If somebody need keyboard scroll implementation for typescript(meteor), i have it. the plugin it so cool. thanks

Reply

hossein

Posted on February 3, 2016 at 07:55 *Permalink*

hi.
thanks for good jquery plugin.

Reply

Fabiano

Posted on February 2, 2016 at 16:13 *Permalink*

Great plug-in, but I wonder how to make a button on the bottom of the container when clicked back to the beginning .
is posível do this ?

Reply

Fabiaano

Posted on February 2, 2016 at 16:12 *Permalink*

Ótimo plug-in, mas gostaria de saber como fazer um botão no rodapé da container quando clicado voltar para o início. é posível fazer isso?

Reply

sham

Posted on February 1, 2016 at 12:52 *Permalink*

Thank you for this script and it is very use full me. I have one special requirement and i have tried it but it does not work properly. Could you please help me.

I have developed the video listing website and its single page i have used this script to list out the videos. The left side all the video titles will be displayed and if the users click on the left side title, the corresponding video content will displayed on the right side. I have used the scrolling script in the left side and it is working fine.
My requirement:
==========
if the user click on the video title on the left side, it should scroll to top or first position automatically. Is there any option for developing the same.

Reply

malihu

Posted on February 5, 2016 at 19:47 *Permalink*

Hello,

You could normally use the *scrollTo* method to trigger a scroll action programmatically (I can’t really tell how to do this exactly as I’d need to see your page/code). This of course assumes you’re using the custom scrollbar on the right side as well(?)

If you could send me a link I'd be able to help

Reply

Alan

Posted on January 30, 2016 at 08:22 Permalink

I am using this plugin and it works great.
Just a small problem: When autoScrollOnFocus is enabled, sometimes click event may not be detected.
Here is an example: https://jsfiddle.net/ououuxeo/
Click on 1st button, click event is detected.
Click on 2nd button, the scrollbar auto scrolls, then click event is missed.
Anyway to solve this issue?

Reply

malihu

Posted on February 4, 2016 at 03:23 Permalink

Hello and thanks for the feedback.

I tested the jsfiddle and what happens is that the click event does not register when the mouseup action is out of the button (while the mousedown action was on the button).

The scenario is this:

Button "BBBBBBB" doesn't have focus and it's out-of-view. This means that the plugin script will scroll it into view (autoScrollOnFocus) on mousedown (js focusin event). So, on mousedown the content scrolls and the button "changes" position. The cursor hasn't moved and the mouseup action happens on some other element (or nowhere), which results the click event not to register!
Does this make sense?

This case is specific to the button and scrollable area width. For example, if you increase the width of button "BBBBBBB" from 130 to 170 pixels the click event will always register, because the mousedown and mouseup actions of the click happen when the cursor is on the button.

I'll investigate this further and try to do more tests but I'm not sure if a solution for such case is possible.

Reply

Comments pages: Previous 1 … 72 73 74

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

Manos Malihutsakis blog. Creating and sharing (completely free) all kinds of web stuff.
Contact me at manos@malihu.gr.

Like to help? contribute code on Github or make a donation :)

> «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*

↑