How can I check if a scrollbar is visible?

Asked 8 years, 6 months ago Active 1 month ago Viewed 255k times

256

71

Is it possible to check the `overflow:auto` of a div?

For example:

HTML

```
<div id="my_div" style="width: 100px; height:100px; overflow:auto;" class="my_class">
  * content
</div>
```

JQUERY

```
$('.my_class').live('hover', function (event)
{
    if (event.type == 'mouseenter')
    {
         if( ...  if scrollbar visible ? ... )
         {
            alert('true'):
         }
         else
         {
            alert('false'):
         }
    }

});
```

Sometimes is the content short (no scrollbar) and sometimes long (scrollbar visible).

jquery scroll overflow

edited Jun 21 at 9:59 John

asked Jan 27 '11 at 9:03 Peter


6,067 7 58 115


2,453 25 77 132

17 Answers

362

a little plugin for it.

```
(function($) {
    $.fn.hasScrollBar = function() {
        return this.get(0).scrollHeight > this.height();
    }
})(jQuery);
```

use it like this,

```
$('#my_div1').hasScrollBar(); // returns true if there's a `vertical` scrollbar, false
otherwise..
```

tested working on Firefox, Chrome, IE6,7,8

but not working properly on `body` tag selector

demo

Edit

I found out that when you have horizontal scrollbar that causes vertical scrollbar to appear, this function does not work....

I found out another solution... use `clientHeight`

```
return this.get(0).scrollHeight > this.get(0).clientHeight;
```

edited Oct 30 '13 at 1:38

answered Jan 27 '11 at 9:19


Reigel
54.5k 20 102 127

21 If you have padding you need to use `> this.innerHeight();` jsfiddle.net/p3FFL/210 – jcubic Jan 3 '12 at 16:44

5 If there a way to get it to work with the body? – Kees C. Bakker Jun 27 '12 at 9:08

2 There's a problem with this, if a horizontal scroll bar also exists, then this will return false even if a vertical scroll bar exists up until the height has been shrunk by the horizontal scroll bar height. – Ally Jul 3 '12 at 10:36

8 Note that on Macs the scrollbar floats over the content and disappears when not in use. On Windows it is always visible and takes up horizontal space. Therefor, just because content can be scrolled (which this function detects) does not mean that a scrollbar is necessarily present. – Andrew Sep 3 '14 at 1:31

2 (function($) { $.fn.hasScrollBar = function() { return this.get(0).scrollWidth > this.width); } })(jQuery); this works for the horizontal overfolow. Good for checking mobile responsiveness on websites in iframes. – Alexander Nicholas Popa May 5 '15 at 13:28

50

Maybe a more simple solution.

```
if ($(document).height() > $(window).height()) {
    // scrollbar
}
```

answered May 13 '11 at 4:59

Sean Zheng
549 4 3

This Does not work in IE – Moons Nov 19 '11 at 12:56

4 This works in Firefox, IE, and Chrome. – Magmatic Apr 17 '12 at 16:32

This answer worked for me after checking if the DOM is ready using JQuery `.ready()` – Simple Sandman Nov 29 '16 at 20:23

1 This assumes the scrollbar is on the window and not a div element. Propose changing the reference to suggest: "If you only need to test the scrollbar on the main window, you could try:" – justdan23 Jun 20 '17 at 23:58

40

I should change a little thing of what Reigel said:

```
(function($) {
    $.fn.hasScrollBar = function() {
        return this[0] ? this[0].scrollHeight > this.innerHeight() : false;
```

```
    }
})(jQuery);
```

innerHeight counts control's height and its top and bottom paddings

edited Jun 26 at 21:27

SpyderScript
608 1 9 20

answered Jan 26 '12 at 9:40

kpull1
1,322 12 16

return (this.get(0))?this.get(0).scrollHeight>this.innerHeight():false; – commonpike May 9 '12 at 12:59

2 I think this should be assigned as the right answer. This worked on FF35, IE11 and Chrome39. – LucasBr Feb 2 '15 at 13:20

It doesn't check the 'overflow' value to make sure scroll bars will appear when that scrollHeight condition is met. – B T Oct 7 '16 at 18:29

1 @BT But if I have overflow set to auto in my css, then I don't need this extra check? It compares sizes and that's enough...? – Andrew Oct 8 '16 at 11:25

An answer that only works for you isn't an answer.. how do you know what other people have in their css? Your answer doesn't mention that limitation. If someone can't drop in your answer and have it work, its not a good answer. – B T Oct 10 '16 at 3:10

33

You can do this using a combination of the `Element.scrollHeight` and `Element.clientHeight` attributes.

According to MDN:

> The **Element.scrollHeight** read-only attribute is a measurement of the height of an element's content, including content not visible on the screen due to overflow. **The scrollHeight value is equal to the minimum clientHeight the element would require in order to fit all the content in the viewpoint without using a vertical scrollbar.** It includes the element padding but not its margin.

And:

> The **Element.clientHeight** read-only property returns the inner height of an element in pixels, including padding but not the horizontal scrollbar height, border, or margin.
>
> clientHeight can be calculated as CSS height + CSS padding - height of horizontal scrollbar (if present).

Therefore, the element will display a scrollbar if the scroll height is greater than the client height, so the answer to your question is:

```
function scrollbarVisible(element) {
  return element.scrollHeight > element.clientHeight;
}
```

answered Mar 21 '15 at 21:55

Ajedi32
27.3k 12 103 137

1 example: github.com/twbs/bootstrap/blob/master/js/modal.js#L242 and +1 for the MDN quote and explanation! – lowtechsun Apr 14 '15 at 22:26

in chrome if content has border than it is not included in scrollHeight – A.T. Jan 13 '16 at 7:47

Up-voted for not laming out and using a framework/library. – John Jun 21 at 11:10

24

This expands on @Reigel's answer. It will return an answer for horizontal or vertical scrollbars.

```
(function($) {
    $.fn.hasScrollBar = function() {
        var e = this.get(0);
        return {
            vertical: e.scrollHeight > e.clientHeight,
            horizontal: e.scrollWidth > e.clientWidth
        };
    }
})(jQuery);
```

Example:

```
element.hasScrollBar()            // Returns { vertical: true/false, horizontal:
true/false }
element.hasScrollBar().vertical   // Returns true/false
element.hasScrollBar().horizontal  // Returns true/false
```

edited Aug 24 '15 at 11:46

Evan Trimboli
27.3k 5 37 56

answered Jan 29 '14 at 4:23

the4tress
432 5 17

12

You need element.scrollHeight. Compare it with `$(element).height()` .

edited Nov 19 '14 at 1:56
Francisco Costa
3,958 3 28 39

answered Jan 27 '11 at 9:07
Boldewyn
55.5k 37 126 194

7

I made a new custom :pseudo selector for jQuery to test whether an item has one of the following css properties:

1. overflow: [scroll|auto]
2. overflow-x: [scroll|auto]
3. overflow-y: [scroll|auto]

I wanted to find the closest scrollable parent of another element so I also wrote another little jQuery plugin to find the closest parent with overflow.

This solution probably doesn't perform the best, but it does appear to work. I used it in conjunction with the $.scrollTo plugin. Sometimes I need to know whether an element is inside another scrollable container. In that case I want to scroll the parent scrollable element vs the window.

I probably should have wrapped this up in a single plugin and added the psuedo selector as a part of the plugin, as well as exposing a 'closest' method to find the closest (parent) scrollable container.

Anywho....here it is.

$.isScrollable jQuery plugin:

```
$.fn.isScrollable = function(){
    var elem = $(this);
    return (
    elem.css('overflow') == 'scroll'
        || elem.css('overflow') == 'auto'
        || elem.css('overflow-x') == 'scroll'
        || elem.css('overflow-x') == 'auto'
        || elem.css('overflow-y') == 'scroll'
        || elem.css('overflow-y') == 'auto'
    );
};
```

$(':scrollable') jQuery pseudo selector:

```
$.expr[":"].scrollable = function(a) {
    var elem = $(a);
    return elem.isScrollable();
};
```

$.scrollableparent() jQuery plugin:

```
$.fn.scrollableparent = function(){
    return $(this).closest(':scrollable') || $(window); //default to $('html') instead?
};
```

Implementation is pretty simple

```
//does a specific element have overflow scroll?
var somedivIsScrollable = $(this).isScrollable();
//use :scrollable psuedo selector to find a collection of child scrollable elements
var scrollableChildren = $(this).find(':scrollable');
//use $.scrollableparent to find closest scrollable container
var scrollableparent = $(this).scrollableparent();
```

UPDATE: I found that Robert Koritnik already came up with a much more powerful :scrollable pseudo selector that will identify the scrollable axes and height of scrollable containers, as a part of his $.scrollintoview() jQuery plugin. [scrollintoview plugin]

Here is his fancy pseudo selector (props):

```
    $.extend($.expr[":"], {

    scrollable: function (element, index, meta, stack) {

        var direction = converter[typeof (meta[3]) === "string" &&
meta[3].toLowerCase()] || converter.both;

        var styles = (document.defaultView && document.defaultView.getComputedStyle ?
document.defaultView.getComputedStyle(element, null) : element.currentStyle);

        var overflow = {

            x: scrollValue[styles.overflowX.toLowerCase()] || false,

            y: scrollValue[styles.overflowY.toLowerCase()] || false,
```

```
            isRoot: rootrx.test(element.nodeName)
        };

        // check if completely unscrollable (exclude HTML element because it's special)
        if (!overflow.x && !overflow.y && !overflow.isRoot)
        {
            return false;
        }

        var size = {
            height: {
                scroll: element.scrollHeight,
                client: element.clientHeight
            },
            width: {
                scroll: element.scrollWidth,
                client: element.clientWidth
            },
            // check overflow.x/y because iPad (and possibly other tablets) don't dislay
scrollbars
            scrollableX: function () {
                return (overflow.x || overflow.isRoot) && this.width.scroll >
this.width.client;
            },
            scrollableY: function () {
```

```
                return (overflow.y || overflow.isRoot) && this.height.scroll >
this.height.client;

            }

        };

        return direction.y && size.scrollableY() || direction.x && size.scrollableX();

    }

});
```

edited Aug 7 '12 at 18:18

answered Aug 7 '12 at 17:50

thegrandoverseer

204 3 3

6

The first solution above works only in IE The second solution above works only in FF

This combination of both functions works in both browsers:

```
//Firefox Only!!
if ($(document).height() > $(window).height()) {
    // has scrollbar
    $("#mtc").addClass("AdjustOverflowWidth");
    alert('scrollbar present - Firefox');
} else {
    $("#mtc").removeClass("AdjustOverflowWidth");
}

//Internet Explorer Only!!
(function($) {
    $.fn.hasScrollBar = function() {
        return this.get(0).scrollHeight > this.innerHeight();
    }
})(jQuery);
if ($('#monitorWidth1').hasScrollBar()) {
    // has scrollbar
    $("#mtc").addClass("AdjustOverflowWidth");
    alert('scrollbar present - Internet Exploder');
} else {
    $("#mtc").removeClass("AdjustOverflowWidth");
}
```

- Wrap in a document ready
- monitorWidth1 : the div where the overflow is set to auto
- mtc : a container div inside monitorWidth1
- AdjustOverflowWidth : a css class applied to the #mtc div when the Scrollbar is active *Use the alert to test cross browser, and then comment out for final production code.

HTH

edited Mar 10 '12 at 11:52

Chamika Sandamal
20k 4 53 77

answered Mar 9 '12 at 12:44

Aaron Hopkins
69 1 2

6

(scrollWidth/Height - clientWidth/Height) is a good indicator for the presence of a scrollbar, but it will give you a "false positive" answer on many occasions. if you need to be accurate i would suggest using the following function. instead of trying to guess if the element is scrollable - you can scroll it...

```
function isScrollable( el ){
  var y1 = el.scrollTop;
  el.scrollTop  += 1;
  var y2 = el.scrollTop;
  el.scrollTop  -= 1;
  var y3 = el.scrollTop;
  el.scrollTop   = y1;
  var x1 = el.scrollLeft;
  el.scrollLeft += 1;
  var x2 = el.scrollLeft;
  el.scrollLeft -= 1;
  var x3 = el.scrollLeft;
  el.scrollLeft  = x1;
  return {
    horizontallyScrollable: x1 !== x2 || x2 !== x3,
    verticallyScrollable: y1 !== y2 || y2 !== y3
  }
}
function check( id ){
  alert( JSON.stringify( isScrollable( document.getElementById( id )))));
}
```

```
#outer1, #outer2, #outer3 {
  background-color: pink;
  overflow: auto;
  float: left;
}
#inner {
  width:  150px;
  height: 150px;
}
button {  margin: 2em 0 0 1em; }


<div id="outer1" style="width: 100px; height: 100px;">
  <div id="inner">
    <button onclick="check('outer1')">check if<br>scrollable</button>
  </div>
</div>
<div id="outer2" style="width: 200px; height: 100px;">
  <div id="inner">
    <button onclick="check('outer2')">check if<br>scrollable</button>
  </div>
</div>
<div id="outer3" style="width: 100px; height: 180px;">
  <div id="inner">
    <button onclick="check('outer3')">check if<br>scrollable</button>
  </div>
</div>
```

Run code snippet Expand snippet

edited Jun 27 '18 at 10:02
biziclop
12.6k 3 38 59

answered Mar 8 '17 at 20:59
Yair Levy
185 2 7

4

Ugh everyone's answers on here are incomplete, and lets stop using jquery in SO answers already please. Check jquery's documentation if you want info on jquery.

Here's a generalized pure-javascript function for testing whether or not an element has scrollbars in a complete way:

```
// dimension - Either 'y' or 'x'
// computedStyles - (Optional) Pass in the domNodes computed styles if you already have
it (since I hear its somewhat expensive)
function hasScrollBars(domNode, dimension, computedStyles) {
    dimension = dimension.toUpperCase()
    if(dimension === 'Y') {
        var length = 'Height'
    } else {
        var length = 'Width'
    }

    var scrollLength = 'scroll'+length
    var clientLength = 'client'+length
    var overflowDimension = 'overflow'+dimension

    var hasVScroll = domNode[scrollLength] > domNode[clientLength]


    // Check the overflow and overflowY properties for "auto" and "visible" values
    var cStyle = computedStyles || getComputedStyle(domNode)
    return hasVScroll && (cStyle[overflowDimension] == "visible"
                        || cStyle[overflowDimension] == "auto"
                        )
          || cStyle[overflowDimension] == "scroll"
}
```

answered Oct 3 '16 at 23:28

B T

30.3k 29 142 173

3 Why avoid using jquery on a question flaged as jquery? Please add links to the part of jquery documentation you mention. – kpull1 Oct 6 '16 at 11:07

3 @kpull1 Too many people tag jQuery on every single javascript question they have. This question has 0 relation to jQuery. There *is* no part of the jQuery documentation that has the answer, because jQuery doesn't do this, nor should it. – B T Oct 6 '16 at 19:10

2

The solutions provided above will work in the most cases, but checking the scrollHeight and overflow is sometimes not enough and fail for body and html elements: https://codepen.io/anon/pen/EvzXZw

This solution checks if the element is scrollable:

```
function isScrollableY (element) {
  return !!(element.scrollTop || (++element.scrollTop && element.scrollTop--));
}
```

Note: elements with overflow=hidden are also treated as scrollable (more info), so you might add a condition against that too if needed:

```
function hasVerticalScrollbar (element) {
    let style = window.getComputedStyle(element);
    return !!(element.scrollTop || (++element.scrollTop && element.scrollTop--))
           && style["overflow"] !== "hidden" && style["overflow-y"] !== "hidden";
}
```

Explanation: The trick is, that the attempt of scrolling down and reverting it won't be rendered by the browser. The topmost function can also be written like the following:

```
function isScrollableY (element) {
  // if scrollTop is not 0 / larger than 0, then the element is scrolled and therefore
must be scrollable
  // -> true
  if (element.scrollTop === 0) {
    // if the element is zero it may be scrollable
    // -> try scrolling about 1 pixel
    element.scrollTop++;
    // if the element is zero then scrolling did not succeed and therefore it is not
scrollable
    // -> false
    if (element.scrollTop === 0) return false;
    // else the element is scrollable; reset the scrollTop property
    // -> true
    element.scrollTop--;
  }
  return true;
}
```

edited Sep 13 '17 at 22:35

answered Sep 7 '17 at 9:24

Robbendebiene

1,998 2 11 20

I'm going to extend on this even further for those poor souls who, like me, use one of the **modern** js frameworks and not JQuery and have been wholly abandoned by the people of this thread :

2

this was written in Angular 6 but if you write React 16, Vue 2, Polymer, Ionic, React-Native, you'll know what to do to adapt it. And it's the whole component so it should be easy.

```
import {ElementRef, AfterViewInit} from '@angular/core';

@Component({
  selector: 'app',
  templateUrl: './app.html',
  styleUrls: ['./app.scss']
})
export class App implements AfterViewInit {
scrollAmount;

constructor(
  private fb: FormBuilder,
) {}

ngAfterViewInit(){
  this.scrollAmount = this.element.nativeElement.querySelector('.elem-list');
  this.scrollAmount.addEventListener('wheel', e => { //you can put () instead of e
  // but e is usefull if you require the deltaY amount.
    if(this.scrollAmount.scrollHeight > this.scrollAmount.offsetHeight){
       // there is a scroll bar, do something!
    }else{
       // there is NO scroll bar, do something!
    }
  });
}
}
```

in the html there would be a div with class "elem-list" which is stylized in the css or scss to have a `height` and an `overflow` value that isn't `hidden`. (so `auto` or `sroll`)

I trigger this eval upon a scroll event because my end goal was to have "automatic focus scrolls" which decide whether they are scrolling the whole set of components horizontally if said components have no vertical scroll available and otherwise only scroll the innards of one of the components vertically.

but you can place the eval elsewhere to have it be triggered by something else.

the important thing to remember here, is you're never **Forced** back into using JQuery, there's always a way to access the same functionalities it has without using it.

edited Oct 1 '18 at 21:47

answered Aug 7 '18 at 15:40

tatsu

986 1 13 46

Curious why you are listening for wheel events to check if there is a scrollbar or not. – mix3d Oct 1 '18 at 19:25

2 Also, since you are using arrow functions, `this` keeps the parent scope; `th = this;` is unnecessary. – mix3d Oct 1 '18 at 19:26

1 @mix3d I personally use this code to automatically switch between horizontal and vertical scrolling based on which has scroll direction at given pointed element which is dynamic – tatsu Oct 1 '18 at 21:49

1 Re: this; it's basically syntactic sugar (plus nice shorthand) for `function(){}.bind(this)` – mix3d Oct 2 '18 at 13:16

1

Here's an improved version of Evan's answer which seems to properly account for overflow logic.

```
function element_scrollbars(node) {
    var element = $(node);
    var overflow_x = element.css("overflow-x");
    var overflow_y = element.css("overflow-y");
    var overflow = element.css("overflow");
    if (overflow_x == "undefined") overflow_x == "";
    if (overflow_y == "undefined") overflow_y == "";
    if (overflow == "undefined") overflow == "";
    if (overflow_x == "") overflow_x = overflow;
    if (overflow_y == "") overflow_y = overflow;
    var scrollbar_vertical = (
        (overflow_y == "scroll")
        || (
            (
                (overflow_y == "hidden")
                || (overflow_y == "visible")
            )
            && (
                (node.scrollHeight > node.clientHeight)
            )
        )
    );
    var scrollbar_horizontal = (
        (overflow_x == "scroll")
        || (
            (
                (overflow_x == "hidden")
                || (overflow_x == "visible")
            )
            && (
                (node.scrollWidth > node.clientWidth)
```

```
                    )
                )
            );
            return {
                vertical: scrollbar_vertical,
                horizontal: scrollbar_horizontal
            };
        }
```

answered Oct 12 '16 at 22:50

JBlitzen
60 11

0

Here's my improvement: added parseInt. for some weird reason it didn't work without it.

```
// usage: jQuery('#my_div1').hasVerticalScrollBar();
// Credit: http://stackoverflow.com/questions/4814398/how-can-i-check-if-a-scrollbar-is-
visible
(function($) {
    $.fn.hasVerticalScrollBar = function() {
        return this.get(0) ? parseInt( this.get(0).scrollHeight ) > parseInt(
this.innerHeight() ) : false;
    };
})(jQuery);
```

answered Aug 18 '16 at 14:48

Svetoslav Marinov
1,153 10 11

0

Works on **Chrome**, **Edge**, **Firefox** and **Opera**, at least in the newer versions.

Using JQuery...

Setup this function to fix the footer:

```
function fixFooterCaller()
{
    const body = $('body');
```

```
    const footer = $('body footer');

    return function ()
    {
        // If the scroll bar is visible
        if ($(document).height() > $(window).height())
        {
            // Reset
            footer.css('position', 'inherit');
            // Erase the padding added in the above code
            body.css('padding-bottom', '0');
        }
        // If the scrollbar is NOT visible
        else
        {
            // Make it fixed at the bottom
            footer.css('position', 'fixed');
            // And put a padding to the body as the size of the footer
            // This makes the footer do not cover the content and when
            // it does, this event fix it
            body.css('padding-bottom', footer.outerHeight());
        }
    }
}
```

It returns a function. Made this way just to set the body and footer once.

And then, set this when the document is ready.

```
$(document).ready(function ()
{
    const fixFooter = fixFooterCaller();

    // Put in a timeout call instead of just call the fixFooter function
    // to prevent the page elements needed don't be ready at this time
    setTimeout(fixFooter, 0);
    // The function must be called every time the window is resized
    $(window).resize(fixFooter);
});
```

Add this to your footer css:

```
footer {
    bottom: 0;
}
```

edited Jun 24 '17 at 17:56

answered Jun 24 '17 at 17:45

Mateus Pires

386 1 4 21

0

Most of the answers presented got me close to where I needed to be, but not quite there.

We basically wanted to assess if the scroll bars -would- be visible in a normal situation, by that definition meaning that the size of the body element is larger than the view port. This was not a presented solution, which is why I am submitting it.

Hopefully it helps someone!

```
(function($) {
    $.fn.hasScrollBar = function() {
        return this.get(0).scrollHeight > $(window).height();
    }
})(jQuery);
```

Essentially, we have the `hasScrollbar` function, but returning if the requested element is larger than the view port. For view port size, we just used `$(window).height()` . A quick compare of that against the element size, yields the correct results and desirable behavior.

answered Dec 22 '17 at 4:15

Barry Chapman

5,647 2 26 49

0

Find a parent of current element that has vertical scrolling or body.

```
$.fn.scrollableParent = function() {
    var $parents = this.parents();

    var $scrollable = $parents.filter(function(idx) {
        return this.scrollHeight > this.offsetHeight && this.offsetWidth !==
this.clientWidth;
    }).first();

    if ($scrollable.length === 0) {
        $scrollable = $('html, body');
    }
```

```
    return $scrollable;
};
```

It may be used to autoscroll to current element via:

```
var $scrollable = $elem.scrollableParent();
$scrollable.scrollTop($elem.position().top);
```

answered Feb 14 '18 at 13:39

Dmitriy Sintsov
1,536 14 14

---

protected by Reigel Feb 15 '13 at 3:14

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?