Is there an "exists" function for jQuery?

Asked 16 years, 3 months ago Modified 2 months ago Viewed 870k times

3160

How can I check the existence of an element in jQuery?

The current code that I have is this:

```
if ($(selector).length > 0) {
    // Do something
}
```

Is there a more elegant way to approach this? Perhaps a plugin or a function?

javascript jquery

Share
Improve this question
Follow

edited Dec 16, 2017 at 11:29
René
6,176 ● 4 ● 25 ● 41

asked Aug 27, 2008 at 19:49

Jake McGraw
56.1k ● 10 ● 51 ● 63

17 Answers

Sorted by: Highest score (default)

2772

In JavaScript, everything is 'truthy' or 'falsy', and for numbers `0` means `false`, everything else `true`. So you could write:

```
if ($(selector).length)
```

You don't need that `>0` part.

Share
Improve this answer
Follow

edited Jun 22, 2020 at 7:57

E962
716 ● 5 ● 25

answered Feb 25, 2009 at 19:16

Tim Büthe
63.7k ● 17 ● 94 ● 130

1431

Yes!

```
jQuery.fn.exists = function(){ return this.length > 0; }

if ($(selector).exists()) {
```

```
    // Do something
}
```

This is in response to: [Herding Code podcast with Jeff Atwood]

Share

Improve this answer

Follow

edited Feb 13, 2017 at 22:37

user719662

answered Aug 27, 2008 at 19:50

Jake McGraw

56.1k ● 10 ● 51 ● 63

378 The `$.fn.exists` example is replacing a property lookup (cheap!) with two function calls, which are much more expensive—and one of those function calls recreates a jQuery object that you already have. – C Snover May 30, 2010 at 4:14

222 @redsquare: Code readability is the best rationale for adding this sort of function on jQuery. Having something called `.exists` reads cleanly, whereas `.length` reads as something semantically different, even if the semantics coincide with an identical result. – Ben Zotto Aug 2, 2010 at 20:52

54 @quixoto, sorry but .length is a standard across many languages that does not need wrapping. How else do you interpret .length? – redsquare Aug 3, 2010 at 0:13

159 In my opinion, it's at least one logical indirection from the concept of "a list length that is greater than zero" to the concept of "the element(s) I wrote a selector for exist". Yeah, they're the same thing technically, but the conceptual abstraction is at a different level. This causes some people to prefer a more explicit indicator, even at some performance cost. – Ben Zotto Aug 3, 2010 at 0:29

395

If you used

```
jQuery.fn.exists = function(){return ($(this).length > 0);}
if ($(selector).exists()) { }
```

you would imply that chaining was possible when it is not.

This would be better:

```
jQuery.exists = function(selector) {return ($(selector).length > 0);}
if ($.exists(selector)) { }
```

Alternatively, from the FAQ:

```
if ( $('#myDiv').length ) { /* Do something */ }
```

You could also use the following. If there are no values in the jQuery object array then getting the first item in the array would return undefined.

```
if ( $('#myDiv')[0] ) { /* Do something */ }
```

Share

Improve this answer

Follow

edited Jun 30, 2015 at 8:09

R3tep

12.9k • 10 • 51 • 76

answered Jan 14, 2009 at 19:46

Jon Erickson

115k • 45 • 139 • 177

24 There are already methods that aren't chainable, like attr and data functions. I do see your point though and, for what it's worth, I just test for length > 0 anyways. – Matthew Crumley Jan 16, 2009 at 5:42

44 Why on earth would you need to chain this? `$(".missing").css("color", "red")` already does the right thing… (i.e. nothing) – Ben Blank Sep 8, 2010 at 6:43

115

The fastest and most semantically self-explaining way to check for existence is actually by using plain JavaScript:

```
if (document.getElementById('element_id')) {
    // Do something
}
```

It is a bit longer to write than the jQuery length alternative, but executes faster since it is a native JS method.

And it is better than the alternative of writing your own jQuery function. That alternative is slower, for the reasons @snover stated. But it would also give other programmers the impression that the `exists()` function is something inherent to jQuery. JavaScript would/should be understood by others editing your code, without increased knowledge debt.

NB: Notice the lack of an `#` before the `element_id` (since this is plain JS, not jQuery).

Share

Improve this answer

Follow

edited Feb 22 at 13:49

answered Jan 11, 2012 at 12:27

Magne

17.2k • 11 • 72 • 95

62 Totally not the same thing. JQuery selectors can be used for any CSS rule - for example `$('#foo a.special')`. And it can return more than one element. `getElementById` can't begin to approach that. – kikito Mar 7, 2012 at 16:30

35 @Noz `if(document.querySelector("#foo a.special"))` would work. No jQuery needed. – Blue Skies Dec 8, 2013 at 0:43

38 The argument of speed in JS engines is only dead in the mind of people who can't function without jQuery, since it's an argument they can't win. – Blue Skies Dec 8, 2013 at 0:45

27 Remember the good old days when document.getElementById was all we had? And I always forgot the document. and couldn't figure out why it didn't work. And I always spelled it wrong and got the character casing wrong. – JustJohn Nov 18, 2014 at 21:05

86

You can save a few bytes by writing:

```
if ($(selector)[0]) { ... }
```

This works because each jQuery object also masquerades as an array, so we can use the array dereferencing operator to get the first item from the *array*. It returns `undefined` if there is no item at the specified index.

Share Improve this answer Follow

answered Jan 18, 2014 at 9:04

Salman Arshad

272k ● 84 ● 441 ● 533

7 No, `jQuery.first()` or `jQuery.eq(0)` both return objects, objects are truthy even if they are empty-ish. This example should illustrate why these functions cannot be used as-is: `if(jQuery("#does-not-exist").eq(0)) alert("#does-not-exist exists")` – Salman Arshad Jul 21, 2015 at 15:16

73

You can use:

```
if ($(selector).is('*')) {
  // Do something
}
```

A **little** more elegant, perhaps.

Share Improve this answer Follow

answered Sep 17, 2008 at 17:53

Devon

5,784 ● 5 ● 40 ● 46

41 This is too much for such a simple thing. see Tim Büthe answer – vsync Nov 24, 2009 at 9:28

1 This is the correct answer. The 'length' method has the problem that it gives false positive with any number, for example: $(666).length // returns 1, but it's not a valid selector – earnaz Sep 16, 2015 at 16:23

4 This is extremely expensive for very simple task. Just look into jquery implementation if .is() and you will see how much code it needs to process to answer you this simple question. Also it is not obvious what you want to do exactly, so it is same or maybe less elegant then solution in question. – micropro.cz Feb 22, 2016 at 19:59

70

This plugin can be used in an `if` statement like `if ($(ele).exist()) { /* DO WORK */ }` or using a callback.

Plugin

```
;;(function($) {
    if (!$.exist) {
        $.extend({
            exist: function() {
                var ele, cbmExist, cbmNotExist;
                if (arguments.length) {
                    for (x in arguments) {
                        switch (typeof arguments[x]) {
                            case 'function':
                                if (typeof cbmExist == "undefined") cbmExist =
arguments[x];
                                else cbmNotExist = arguments[x];
                                break;
                            case 'object':
                                if (arguments[x] instanceof jQuery) ele =
arguments[x];
                                else {
                                    var obj = arguments[x];
                                    for (y in obj) {
                                        if (typeof obj[y] == 'function') {
                                            if (typeof cbmExist == "undefined")
```

```
cbmExist = obj[y];
                                            else cbmNotExist = obj[y];
                                        }
                                        if (typeof obj[y] == 'object' && obj[y]
instanceof jQuery) ele = obj[y];
                                        if (typeof obj[y] == 'string') ele =
$(obj[y]);
                                    }
                                }
                                break;
                            case 'string':
                                ele = $(arguments[x]);
                                break;
                        }
                    }
                }

                if (typeof cbmExist == 'function') {
                    var exist =  ele.length > 0 ? true : false;
                    if (exist) {
                        return ele.each(function(i) { cbmExist.apply(this,
[exist, ele, i]); });
                    }
                    else if (typeof cbmNotExist == 'function') {
                        cbmNotExist.apply(ele, [exist, ele]);
                        return ele;
                    }
                    else {
                        if (ele.length <= 1) return ele.length > 0 ? true :
false;
                        else return ele.length;
                    }
                }
                else {
                    if (ele.length <= 1) return ele.length > 0 ? true : false;
                    else return ele.length;
                }

                return false;
            }
        });
        $.fn.extend({
            exist: function() {
                var args = [$(this)];
                if (arguments.length) for (x in arguments)
args.push(arguments[x]);
                return $.exist.apply($, args);
            }
        });
    }
})(jQuery);
```

[jsFiddle]

You may specify one or two callbacks. The first one will fire if the element exists, the second one will fire if the element does *not* exist. However, if you choose to pass only one function, it will only fire when the element exists. Thus, the chain will die if the

selected element does *not* exist. Of course, if it does exist, the first function will fire and the chain will continue.

Keep in mind that using the **callback variant helps maintain chainability** – the element is returned and you can continue chaining commands as with any other jQuery method!

Example Uses

```
if ($.exist('#eleID')) {    /*    DO WORK    */ }        //    param as STRING
if ($.exist($('#eleID'))) { /*    DO WORK    */ }        //    param as jQuery
OBJECT
if ($('#eleID').exist()) {  /*    DO WORK    */ }        //    enduced on
jQuery OBJECT

$.exist('#eleID', function() {            //    param is STRING && CALLBACK
METHOD
    /*    DO WORK    */
    /*    This will ONLY fire if the element EXIST    */
}, function() {            //    param is STRING && CALLBACK METHOD
    /*    DO WORK    */
    /*    This will ONLY fire if the element DOES NOT EXIST    */
})

$('#eleID').exist(function() {            //    enduced on jQuery OBJECT with
CALLBACK METHOD
    /*    DO WORK    */
    /*    This will ONLY fire if the element EXIST    */
})

$.exist({                        //    param is OBJECT containing 2 key|value
pairs: element = STRING, callback = METHOD
    element: '#eleID',
    callback: function() {
        /*    DO WORK    */
        /*    This will ONLY fire if the element EXIST    */
    }
})
```

Share

Improve this answer

Follow

edited Jun 12, 2016 at 9:10

community wiki
22 revs, 6 users 64%
SpYk3HH

1 On the callback version, shouldn't the `Has Items` callback actually pass in the object as an argument? – Chris Marisic Jun 16, 2016 at 17:46

69

I see most of the answers here are **not accurate** as they should be, they check element length, it can be **OK** in many cases, but **not 100%**, imagine if number pass to

the function instead, so I prototype a function which check all conditions and return the answer as it should be:

```
$.fn.exists = $.fn.exists || function() {
  return !!(this.length && (this[0] instanceof HTMLDocument || this[0]
instanceof HTMLElement));
}
```

This will check both length and type, Now you can check it this way:

```
$(1980).exists(); //return false
$([1,2,3]).exists(); //return false
$({name: 'stackoverflow', url: 'http://www.stackoverflow.com'}).exists();
//return false
$([{nodeName: 'foo'}]).exists() // returns false
$('div').exists(); //return true
$('.header').exists(); //return true
$(document).exists(); //return true
$('body').exists(); //return true
```

Share

Improve this answer

Follow

edited Jul 17, 2019 at 11:59

answered May 20, 2017 at 9:21

Alireza

105k ● 27 ● 277 ● 173

58

There's no need for jQuery really. With plain JavaScript it's easier and semantically correct to check for:

```
if(document.getElementById("myElement")) {
    //Do something...
}
```

If for any reason you don't want to put an id to the element, you can still use any other JavaScript method designed to access the DOM.

jQuery is really cool, but don't let pure JavaScript fall into oblivion...

Share

Improve this answer

Follow

edited May 10, 2013 at 7:48

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Nov 14, 2011 at 14:20

amypellegrini

1,026 ● 9 ● 13

5 I know: it doesn't answer directly the original question (which asks for a jquery function), but in that case the answer would be "No" or "not a semantically correct solution". – amypellegrini Nov 14, 2011 at 14:24

48

The reason all of the previous answers require the `.length` parameter is that they are mostly using jquery's `$()` selector which has querySelectorAll behind the curtains (or they are using it directly). This method is rather slow because it needs to parse the entire DOM tree looking for **all** matches to that selector and populate an array with them.

The ['length'] parameter is not needed or useful and the code will be a lot faster if you directly use `document.querySelector(selector)` instead, because it returns the first element it matches or null if not found.

```
function elementIfExists(selector){  //named this way on purpose, see below
    return document.querySelector(selector);
}
/* usage: */
var myelement = elementIfExists("#myid") || myfallbackelement;
```

However this method leaves us with the actual object being returned; which is fine if it isn't going to be saved as variable and used repeatedly (thus keeping the reference around if we forget).

```
var myel=elementIfExists("#myid");
// now we are using a reference to the element which will linger after removal
myel.getParentNode.removeChild(myel);
console.log(elementIfExists("#myid")); /* null */
console.log(myel); /* giant table lingering around detached from document */
myel=null; /* now it can be garbage collected */
```

In some cases this may be desired. It can be used in a for loop like this:

```
/* locally scoped myel gets garbage collected even with the break; */
for (var myel; myel = elementIfExist(sel); myel.parentNode.removeChild(myel))
    if (myel == myblacklistedel) break;
```

If you don't actually need the element and want to get/store just a true/false, just double not it !! It works for shoes that come untied, so why knot here?

```
function elementExists(selector){
    return !!document.querySelector(selector);
}
/* usage: */
var hastables = elementExists("table");  /* will be true or false */
if (hastables){
    /* insert css style sheet for our pretty tables */
}
setTimeOut(function (){if (hastables && !elementExists("#mytablecss"))
                           alert("bad table layouts");},3000);
```

Share

Improve this answer

Follow

edited Oct 5 at 6:32

answered Aug 11, 2014 at 23:42

technosaurus

7,792 ● 1 ● 31 ● 53

Nice first part of the answer, but I think this would be improved by removing the section about holding references around, which applies to literally every JavaScript value and doesn't have anything to do with existence checking in particular. (Things also tend to be naturally unreferenced once they become unused, and so not need to be explicitly set to `null`.) (Also, `getParentNode` isn't a thing.) – Ry- ♦ Feb 10 at 0:52

45

Is `$.contains()` what you want?

> `jQuery.contains( container, contained )`
>
> The `$.contains()` method returns true if the DOM element provided by the second argument is a descendant of the DOM element provided by the first argument, whether it is a direct child or nested more deeply. Otherwise, it returns false. Only element nodes are supported; if the second argument is a text or comment node, `$.contains()` will return false.
>
> ***Note***: The first argument must be a DOM element, not a jQuery object or plain JavaScript object.

Share

Improve this answer

Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 ● 1

answered Oct 23, 2013 at 5:46

hiway

3,976 ● 11 ● 36 ● 59

4 This doesn't accept a selector, which means he would have to select it, which means he could just check the result of his selection. – user1106925 Jun 4, 2016 at 13:28

41

I have found `if ($(selector).length) {}` to be insufficient. It will silently break your app when `selector` is an empty object `{}`.

```
var $target = $({});
console.log($target, $target.length);

// Console output:
// ------------------------------------
// [▼ Object              ] 1
//    ▶ __proto__: Object
```

My only suggestion is to perform an additional check for `{}`.

```
if ($.isEmptyObject(selector) || !$(selector).length) {
    throw new Error('Unable to work with the given selector.');
}
```

I'm still looking for a better solution though as this one is a bit heavy.

Edit: **WARNING!** This doesn't work in IE when `selector` is a string.

```
$.isEmptyObject('hello') // FALSE in Chrome and TRUE in IE
```

Share

Improve this answer

Follow

edited Feb 7, 2012 at 17:43

answered Feb 6, 2012 at 20:37

Oleg

9,361 ● 2 ● 44 ● 59

14 How often do you find yourself calling `$()` with an empty object as an argument? – nnnnnn Dec 22, 2014 at 11:24

@nnnnnn Actually never (I don't use jQuery anymore). But I guess 3 years ago I had a case of exposing an API that would take a selector and return the number of elements for that selector. If another dev would pass in an empty object, it would incorrectly respond with 1. – Oleg Dec 22, 2014 at 15:03

4 Why on earth would you pass an empty object `{}` to `$()`? – ohmu Mar 26, 2015 at 15:46

9 @cpburnz why do you ask me? I was just an API provider... People pass all kinds of stupid things to APIs. – Oleg Mar 26, 2015 at 15:48

4 Just noticed, the jquery issue thread that @FagnerBrack referenced was updated shortly after his comment; it looks like it's not going away after all. – Joseph Gabriel Apr 18, 2016 at 21:09

38

Checking for existence of an element is documented neatly in the official jQuery website itself!

> Use the .length property of the jQuery collection returned by your selector:
>
> ```
> if ($("#myDiv").length) {
>     $("#myDiv").show();
> }
> ```
>
> Note that it isn't always necessary to test whether an element exists. The following code will show the element if it exists, and do nothing (with no errors) if it does not:

```
$("#myDiv").show();
```

Share

Improve this answer

Follow

edited Sep 10, 2017 at 10:31

Palec

13.5k ● 8 ● 74 ● 142

answered Mar 22, 2017 at 14:32

Tilak Madichetti

4,326 ● 5 ● 36 ● 56

33

No need for jQuery (basic solution)

```
if(document.querySelector('.a-class')) {
  // do something
}
```

Much more performant option below (notice the lack of a dot before `a-class`).

```
if(document.getElementsByClassName('a-class')[0]) {
  // do something
}
```

`querySelector` uses a proper matching engine like `$()` (sizzle) in jQuery and uses more computing power but in 99% of cases will do just fine. The second option is more explicit and tells the code exactly what to do. It's much faster according to JSBench https://jsbench.me/65l2up3t8i

Share

Improve this answer

Follow

edited May 6, 2022 at 21:30

tagurit

548 ● 6 ● 16

answered Nov 28, 2016 at 10:43

Pawel

18.1k ● 5 ● 77 ● 76

27

Inspired by hiway's answer I came up with the following:

```
$.fn.exists = function() {
    return $.contains( document.documentElement, this[0] );
}
```

jQuery.contains takes two DOM elements and checks whether the first one contains the second one.

Using `document.documentElement` as the first argument fulfills the semantics of the `exists` method when we want to apply it solely to check the existence of an element in the current document.

Below, I've put together a snippet that compares `jQuery.exists()` against the `$(sel)[0]` and `$(sel).length` approaches which both return `truthy` values for `$(4)` while

`$(4).exists()` returns `false`. In the context of **checking for existence** of an element in the DOM this seems to be the **desired result**.

Show code snippet

Share

Improve this answer

Follow

edited May 23, 2017 at 11:47

Community Bot

1 ● 1

answered Oct 13, 2015 at 20:01

Oliver

9,488 ● 9 ● 75 ● 102

This is wasteful when used on a selector. If you just selected it from the document, you know it's in the document. – Ry- ♦ Feb 10 at 0:57

I would argue that by the same standard, many usages of jQuery are wasteful - but for some people it's still a valid choice and enough. I appreciate you taking the time to comment on this answer, but I think the downvote is not really appropriate, since I've answered the OP's question "How can I check the existence of an element in jQuery?" quite to the point. What do you think, @Ry-? – Oliver Feb 16 at 13:46

I disagree. Often, uses of jQuery are a necessary tradeoff between efficiency and expressiveness. Not only is this one *unnecessary*, it also confuses the reader. – Ry- ♦ Feb 16 at 17:12

26

I just like to use plain vanilla javascript to do this.

```
function isExists(selector){
  return document.querySelectorAll(selector).length>0;
}
```

Share Improve this answer Follow

answered Jun 18, 2016 at 22:37

Sanu Uthaiah Bollera

937 ● 10 ● 13

20

I had a case where I wanted to see if an object exists inside of another so I added something to the first answer to check for a selector inside the selector..

```
// Checks if an object exists.
// Usage:
//
//     $(selector).exists()
//
// Or:
//
//     $(selector).exists(anotherSelector);
jQuery.fn.exists = function(selector) {
    return selector ? this.find(selector).length : this.length;
};
```

Share Improve this answer Follow

answered Apr 5, 2012 at 21:02

jcreamer898
8,189 5 42 57

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.