How to check a not-defined variable in JavaScript

Asked 10 years, 2 months ago Active 3 months ago Viewed 1.0m times

851

293

I wanted to check whether the variable is defined or not. For example, the following throws a not-defined error

```
alert( x );
```

How can I catch this error?

javascript variables undefined

edited Mar 3 '17 at 19:35
Josh Crozier
164k 38 290 234

asked May 13 '09 at 14:09
Jineesh
4,634 5 19 23

3 http://stackoverflow.com/questions/519145/how-to-check-whether-a-javascript-variable-defined/519157#519157 – jim0thy May 13 '09 at 14:26

4 Looks like this is a granddaddy: stackoverflow.com/questions/1485840/… stackoverflow.com/questions/2647867/… - great gramps maybe this: stackoverflow.com/questions/27509/… – random Apr 19 '10 at 15:52

1 This is not a duplicate of the marked duplicate. Variable resolution and object property resolution are very different things. A better duplicate is *how to check if a variable exist in javascript?*. – RobG Nov 10 '15 at 2:04

Your error is due to the variable not being declared. Most answers are focused on assignment. See my answer for more. Additionally many of them incorrectly state that null and undefined are objects in JavaScript. They are primitives, not objects... – JBallin Jan 10 '18 at 6:43

14 Answers

In JavaScript, `null` is an object. There's another value for things that don't exist, `undefined` . The DOM returns `null` for almost all cases where it fails to find some structure in the document, but in JavaScript itself `undefined` is the value used.

```
if (yourvar === null) // Does not execute if yourvar is `undefined`
```

If you want to check if a variable exists, that can only be done with `try` / `catch` , since `typeof` will treat an undeclared variable and a variable declared with the value of `undefined` as equivalent.

But, to check if a variable is declared *and* is not `undefined` :

```
if (typeof yourvar !== 'undefined') // Any scope
```

Beware, this is nonsense, because there could be a variable with name `undefined` :

```
if (yourvar !== undefined)
```

If you want to know if a member exists independent but don't care what its value is:

```
if ('membername' in object) // With inheritance
if (object.hasOwnProperty('membername')) // Without inheritance
```

If you want to to know whether a variable is truthy:

```
if (yourvar)
```

Source

edited Apr 8 at 12:31
Lorenz Meyer
13.2k 20 56 99

answered May 13 '09 at 14:11
Natrium
24.3k 15 51 70

70 undefined is not a reserved word; you (or someone else's code) can do "undefined = 3" and that will break two of your tests. – Jason S May 13 '09 at 14:14

5 "If you know the variable exists but don't know if there's any value stored in it" -- huh?! – Jason S May 13 '09 at 14:20

35 I think he is referring to a variable declared that has not been assigned to. eg: var foo; // foo exists but does not have a value – Wally Lawless May 13 '09 at 14:29

11 "In JavaScript null is an object.", that's not actually true, and probably, the culprit of this misconception is the `typeof` operator (`typeof null == 'object'`). The `null` value is a primitive value, which is the only value of the Null type. – CMS Oct 13 '11 at 7:29

325

The only way to truly test if a variable is `undefined` is to do the following. Remember, undefined is an object in JavaScript.

```
if (typeof someVar === 'undefined') {
  // Your variable is undefined
}
```

Some of the other solutions in this thread will lead you to believe a variable is undefined even though it has been defined (with a value of NULL or 0, for instance).

edited Nov 23 '14 at 13:44
Peter Mortensen
14.3k 19 88 116

answered May 13 '09 at 14:53
Michael Wales
6,692 7 23 28

17 Because the question was IS NOT UNDEFINED here should be typeof someVar !== 'undefined', right? – eomeroff Aug 6 '12 at 9:14

1 Really, I don't think so that undefinded is an object, check documentation first developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures – Nicramus Sep 14 '14 at 15:42

2 The only test that does not produce a `ReferenceError` . – Nostalg.io Nov 17 '15 at 19:31

2 This code is correct, but I think saying `undefined` is an object in javascript is misinformation. Does this statement relate to your answer anyway? It is a value `undefined` of type `undefined` , assigned to the global identifier named `undefined` . – SimplGy Aug 15 '16 at 4:34

This seems to be a safer answer than the one given by w3schools.com: `(x === undefined)` . As pointed out in another comment, the value of `undefined` could be redefined and that test would fail. However, using the example given in this answer, that would not be a problem. – L S Jun 12 '17 at 19:58

66

Technically, the proper solution is (I believe):

```
typeof x === "undefined"
```

```
x == null
```

but that allows both an undefined variable x, and a variable x containing null, to return true.

edited Nov 23 '14 at 13:42
Peter Mortensen
14.3k 19 88 116

answered May 13 '09 at 14:12
Jason S
110k 138 501 837

if you type `var x;` and then `typeof x;` you will get `"undefined"` just like if you did `typeof lakjdflkdsjflsj;` – Muhammad Umer Aug 26 '16 at 15:16

yes and the variable is still undefined. – Jason S Aug 26 '16 at 16:18

So there is no way to check for undefined but declared variable? – Muhammad Umer Aug 26 '16 at 16:22

1 I don't think so; I am not sure why you would want to. – Jason S Aug 26 '16 at 16:28

ujndefined shouldn't be between apices – LowFieldTheory Apr 3 '18 at 8:21

18

An even easier and more shorthand version would be:

```
if (!x) {
   //Undefined
}
```

OR

```
if (typeof x !== "undefined") {
    //Do something since x is defined.
}
```

edited Nov 23 '14 at 13:43
Peter Mortensen
14.3k 19 88 116

answered May 13 '09 at 14:26
Dmitri Farkov
7,030 1 24 43

Rajat Dec 30 '09 at 0:49

the second code can be shortened to: if(!typeof(XX)){ ... }else{ ... } – Alejandro Silva Jun 6 '14 at 21:53

2 @AlejandroSilva Sorry for late reply. That won't work since typeof returns a string, so it will return 'undefined' for an undefined variable, which in turn will evaluate as TRUE therefore leading to a false positive of a defined var. – Dmitri Farkov Mar 17 '15 at 20:29

4 Please get rid of the first snippet, it's just bad – Juan Mendes Feb 12 '16 at 12:06

1 Other comments have pointed out that the first example is bad, but not clearly why. So, for any new coders: !x doesn't test whether x is defined, but whether it's truthy. Strings, boolean true, and positive numbers are all truthy (and I might be forgetting some things), but other potentially valid values like 0, boolean false, and an empty string are not truthy. The first example can work for specific use cases (e.g., testing for a string if you can treat empty the same as undefined), but because of the many where it won't, it should not be considered the default way to check. – cfc Nov 14 '18 at 16:50

15

I've often done:

```
function doSomething(variable)
{
    var undef;

    if(variable === undef)
    {
         alert('Hey moron, define this bad boy.');
    }
}
```

edited Apr 24 '14 at 4:18

answered May 13 '09 at 14:45

Joe
1,990 12 19

9 Consider changing "==" to "===". If you call doSomething(null) you will also get the alert. Unless that's what you want. – Jason S May 13 '09 at 15:51

Yep. You have to decide if you want equivalent or exactly equal. Either case could have a use. – Joe Jul 7 '11 at 15:41

1 simplye check like this-> if(typeof variableName !== 'undefined'){ alert(variableName);} – Muhammad Sadiq Aug 19 '15 at 7:56

this is useless since you won't be able to pass an undefined var to a function anyway – avalanche1 Feb 12 '17 at 11:42

1 Sure you can. Try calling a function with no argument. – Joe Feb 12 '17 at 19:09

3

You can also use the ternary conditional-operator:

```
var a = "hallo world";
var a = !a ? document.write("i dont know 'a'") : document.write("a = " + a);
```

Run code snippet   Expand snippet

```
//var a = "hallo world";
var a = !a ? document.write("i dont know 'a'") : document.write("a = " + a);
```

Run code snippet   Expand snippet

edited Feb 12 '16 at 21:37

answered Feb 12 '16 at 12:02
John
670 4 12

What if `var a = false;` ? You should check that if `a===undefined` instead – Iter Ator Jul 14 '16 at 15:57

If a = false, then it will show "i dont know 'a'". – John Jul 14 '16 at 19:37

1 Question: check a not-defined variable..... This is undefined variable: `var x;` doing above will throw an error – Muhammad Umer Aug 26 '16 at 15:21

"If a = false, then it will show "i dont know 'a'"" – That's the problem, the question is to test if it's defined, not whether it's true. If a is defined as false, then a is not undefined. This returns the wrong result in that case. See my comment on stackoverflow.com/a/858270/2055492 for more detail on why this approach doesn't work. – cfc Nov 14 '18 at 16:54

3

Another potential "solution" is to use the `window` object. It avoids the reference error problem when in a browser.

```
if (window.x) {
    alert('x exists and is truthy');
} else {
    alert('x does not exist. or exists and is falsv'):
```

answered Nov 13 '17 at 20:16

ubershmekel
5,922 2 48 66

1

Just do something like below:

```
function isNotDefined(value) {
  return typeof value === "undefined";
}
```

and call it like:

```
isNotDefined(undefined); //return true
isNotDefined('Alireza'); //return false
```

answered Jan 31 at 12:08

Alireza
59.2k 14 195 127

0

We can check `undefined` as follows

```
var x;

if (x === undefined) {
    alert("x is undefined");
} else {
     alert("x is defined");
}
```

answered Jul 4 '16 at 17:17

Arshid KV
6,063 3 22 28

0

```
var variable;
if (variable === undefined){
    console.log('Variable is undefined');
} else {
    console.log('Variable is defined');
}
```

EDIT:

Without initializing the variable, exception will be thrown "Uncaught ReferenceError: variable is not defined..."

edited Feb 3 '17 at 19:06
simhumileco
9,806 5 65 62

answered Dec 9 '15 at 8:44
mokiSRB
922 6 15

2 `Uncaught ReferenceError: variable is not defined` – Muhammad Umer Aug 26 '16 at 15:18

@MuhammadUmer, wrong! `variable` is defined by `var variable;` . And this snippet will override `variable` in local scope. It can break logic which expects to access a closure or global variable. I.e: `var variable = 1; function test() { var variable; if (variable === undefined){ console.log('Variable is undefined'); } else { console.log('Variable is defined: ' + variable); } } test(); // Variable is undefined` – Евгений Савичев Aug 11 '17 at 16:50

0

The accepted answer is correct. Just wanted to add one more option. You also can use `try ... catch` block to handle this situation. A freaky example:

```
var a;
try {
    a = b + 1;  // throws ReferenceError if b is not defined
}
catch (e) {
    a = 1;      // apply some default behavior in case of error
}
finally {
    a = a || 0; // normalize the result in any case
}
```

Be aware of `catch` block, which is a bit messy, as it creates a block-level scope. And, of course, the example is extremely simplified to

answered Aug 11 '17 at 17:25
Евгений Савичев
156 8

0

The error is telling you that `x` doesn't even exist! It hasn't been **declared**, which is different than being **assigned** a value.

```
var x; // declaration
x = 2; // assignment
```

If you declared `x`, you wouldn't get an error. You would get an alert that says `undefined` because `x` exists/has been declared but hasn't been assigned a value.

To check if the variable has been declared, you can use `typeof`, any other method of checking if a variable exists will raise the same error you got initially.

```
if(typeof x  !==  "undefined") {
    alert(x);
}
```

This is checking the type of the value stored in `x`. It will only return `undefined` when `x` hasn't been declared OR if it *has* been declared and was not yet assigned.

edited Apr 16 '18 at 9:09
Fka
4,024 4 29 52

answered Jan 10 '18 at 6:33
JBallin
1,674 13 25

0

The `void` operator returns `undefined` for any argument/expression passed to it. so you can test against the result (actually some minifiers change your code from `undefined` to `void 0` to save a couple of characters)

For example:

```
void 0
// undefined
```

```
    // variable is undefined
}
```

answered Jul 2 '18 at 14:07
svarog
6,196 4 46 60

0

I use a small function to verify a variable has been declared, which really cuts down on the amount of clutter in my javascript files. I add a check for the value to make sure that the variable not only exists, but has also been assigned a value. The second condition checks whether the variable has also been instantiated, because if the variable has been defined but not instantiated (see example below), it will still throw an error if you try to reference it's value in your code.

Not instantiated - `var my_variable;` Instantiated - `var my_variable = "";`

```
function varExists(el) {
  if ( typeof el !== "undefined" && typeof el.val() !== "undefined" ) {
    return true;
  } else {
    return false;
  }
}
```

You can then use a conditional statement to test that the variable has been both defined AND instantiated like this...

```
if ( varExists(variable_name) ) { // checks that it DOES exist }
```

or to test that it hasn't been defined and instantiated use...

```
if( !varExists(variable_name) ) { // checks that it DOESN'T exist }
```

edited Jul 5 '18 at 18:09

answered Jul 5 '18 at 17:48
MistyDawn
545 4 8

protected by Starx Apr 25 '12 at 8:45

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?