

# How do I check for null values in JavaScript?

Asked 8 years, 4 months ago   Active 3 months ago   Viewed 1.0m times

How can I check for null values in JavaScript? I wrote the code below but it didn't work.

493 

```
if (pass == null || cpass == null || email == null || cemail == null || user == null) {  
    alert("fill all columns");  
    return false;  
}
```

120

And how can I find errors in my JavaScript programs?

javascript

null

compare

edited Aug 27 '15 at 23:31



Michael Laszlo

9,816 2 18 39

asked May 14 '11 at 18:15



Mahdi\_Nine

4,989 25 73 108

Are you sure the values you are testing are actually null and not just empty string? – Jan-Peter Vos May 14 '11 at 18:18

63 testing null in js should be done with the strict operator === – davin May 14 '11 at 18:19

3 @davin - true, but not the problem here since if it were the statement would still work. – Mark Kahn May 14 '11 at 18:22

3 @cwolves, if I thought it were the problem I would have made that comment an answer. Check out my wording, I'm clearly making a general statement about the language in reference to the OP's practise, and not proposing that this solves his problem. – davin May 14 '11 at 18:24

1 @TRiG the proposed changes fundamentally alter the nature of the question to the point where the answers (not just mine) lose context and don't make sense. The edit should just be a comment. – ic3b3rg Jul 22 '14 at 4:50

## 17 Answers

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.



666



Javascript is very flexible with regards to checking for "null" values. I'm guessing you're actually looking for empty strings, in which case this simpler code will work:

```
if(!pass || !cpass || !email || !cemail || !user){
```

Which will check for empty strings ( "" ), null , undefined , false and the numbers 0 and NaN

Please note that if you are specifically checking for numbers it is a common mistake to miss 0 with this method, and num !== 0 is preferred (or num !== -1 or ~num (hacky code that also checks against -1 )) for functions that return -1 , e.g. indexOf )

edited Jun 24 '16 at 20:17

answered May 14 '11 at 18:20



Mark Kahn

65.5k 23 144 196

- 
- 4 It would be very useful to know which parts of this test for which values. Sometimes you're looking for one in particular. – [inorganik](#) Apr 19 '13 at 19:28
- 
- 2 Somewhat of a late statement, but yes, you can perform test against each one @inorganik , see my answer below – [WebWanderer](#) Dec 18 '14 at 16:02
- 
- 15 Readers, please be careful when using this type of test on numeric data. Do not use ! or !! to test for null or undefined on typically-numeric data unless you also want to throw away values of 0. – [NickS](#) Feb 24 '16 at 21:06
- 
- 4 The answer itself states this: "...and the numbers 0 ...". I believe this answer is perfectly appropriate for the question as given the context (which I'm inferring is "username, password, email"), they're not checking for 0 values. Given the popularity of this question and answer, however, I agree that it's worth mentioning in the answer itself. – [Mark Kahn](#) Jun 24 '16 at 20:13
- 
- 3 @Hendeca - *rolls eyes* Go read the code in the actual question. It's **clearly** asking about usernames and passwords. I'm not guessing at anything, I was just being polite. You're bothered by the fact that I answered what they *needed* and not what they asked, which is absurd. This is SO, most of the time people don't know what they should be asking for. In the context of the original question this answer is correct. Now stop adding noise. – [Mark Kahn](#) Jul 4 '16 at 1:34
- 



340



To check for null **SPECIFICALLY** you would use this:

```
if(variable === null && typeof variable === "object")
```

...or more simply:

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

This test will **ONLY** pass for `null` and will not pass for `""`, `undefined`, `false`, `0`, or `NaN`.

The rest of this is in response to inorganik's comment, Yes, you can check each one individually.

You need to implement use of the absolutely equals: `===` and `typeof` to be absolutely sure with your checks.

[I've created a JSFiddle here to show all of the individual tests working](#)

Here is all of the output of the tests:

#### Null Test:

```
if(variable === null && typeof variable === "object")  
  
- variable = ""; (false) typeof variable = string  
- variable = null; (true) typeof variable = object  
- variable = undefined; (false) typeof variable = undefined  
- variable = false; (false) typeof variable = boolean  
- variable = 0; (false) typeof variable = number  
- variable = NaN; (false) typeof variable = number
```

#### Empty String Test:

```
if(variable === "" && typeof variable === "string")  
  
- variable = ""; (true) typeof variable = string  
- variable = null; (false) typeof variable = object  
- variable = undefined; (false) typeof variable = undefined  
- variable = false; (false) typeof variable = boolean  
- variable = 0; (false) typeof variable = number  
- variable = NaN; (false) typeof variable = number
```

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

**Undefined Test:**

```
if(variable === undefined && typeof variable === "undefined")
```

- variable = ""; (false) typeof variable = string
- variable = null; (false) typeof variable = object
- variable = undefined; (true) typeof variable = undefined
- variable = false; (false) typeof variable = boolean
- variable = 0; (false) typeof variable = number
- variable = NaN; (false) typeof variable = number

**False Test:**

```
if(variable === false && typeof variable === "boolean")
```

- variable = ""; (false) typeof variable = string
- variable = null; (false) typeof variable = object
- variable = undefined; (false) typeof variable = undefined
- variable = false; (true) typeof variable = boolean
- variable = 0; (false) typeof variable = number
- variable = NaN; (false) typeof variable = number

**Zero Test:**

```
if(variable === 0 && typeof variable === "number")
```

- variable = ""; (false) typeof variable = string
- variable = null; (false) typeof variable = object
- variable = undefined; (false) typeof variable = undefined

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
- variable = NaN; (false) typeof variable = number
```

#### NaN Test:

```
if(!parseFloat(variable) && variable != 0 && typeof variable === "number")
```

```
- variable = ""; (false) typeof variable = string
```

```
- variable = null; (false) typeof variable = object
```

```
- variable = undefined; (false) typeof variable = undefined
```

```
- variable = false; (false) typeof variable = boolean
```

```
- variable = 0; (false) typeof variable = number
```

```
- variable = NaN; (true) typeof variable = number
```

As you can see, it's a little more difficult to test against NaN ;

edited Jun 21 '17 at 15:32

answered Dec 18 '14 at 16:01



**WebWanderer**

5,001 3 17 34

- 
- 7 What is the purpose of type checking if you use `===` strict equality? Thanks. Also for NaN test you can use `isNaN(value)` that will return `true` only if variable equals NaN. – [Michael Malinovskij](#) Feb 11 '15 at 13:20
- 
- 4 Sanity check, plus, you'd be surprised how many times I've used an `absolutely equals` and not received a proper value in return with checking `typeof` . lol. Also, I've had some issues with `isNaN` in the past, so I decided to provide what is an absolutely sure-fire way to check each one. There should be no case failure on any of these checks. You're welcome. – [WebWanderer](#) Feb 11 '15 at 18:10
- 
- 14 Is there a case where `variable === null` but is not of type "object"? If there is not, why not simplify the check to `variable === null` , throwing out the second conjunct? Thanks. – [Hunan Rostomyan](#) Apr 4 '15 at 22:19
- 
- 7 @HunanRostomyan Good question, and honestly, no, I do not think that there is. You are most likely safe enough using `variable === null` which I just tested [here in this JSFiddle](#). The reason I also used `&& typeof variable === 'object'` was not only to illustrate the interesting fact that a `null` value is a `typeof object` , but also to keep with the flow of the other checks. But yes, in conclusion, you are safe to use simply `variable === null` . – [WebWanderer](#) Apr 6 '15 at 14:59
- 
- 1 [jsfiddle.net/neoaptt/avt1cqem/1](https://jsfiddle.net/neoaptt/avt1cqem/1) Here is this answer broken out into functions. Not very usefull, but I did it anyways. – [Neoaptt](#) Apr 18 '16 at 18:22
- 

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

just replace the `==` with `===` in all places.

59

`==` is a loose or abstract equality comparison

`===` is a strict equality comparison

See the MDN article on [Equality comparisons and sameness](#) for more detail.

edited Jun 27 '16 at 2:15

answered May 14 '11 at 18:27



[ic3b3rg](#)

11.3k 4 21 45

2 This only works if you consider `undefined` to be not `null`. Otherwise it will lead to a lot of unexpected behavior. Generally if you're interested in both `null` / `undefined` but not falsy values, then use `==` (one of the few cases when you should do so). – [Andrew Mao](#) May 11 '16 at 22:40 ✎

2 @AndrewMao `undefined` **is not** `null` : [stackoverflow.com/a/5076962/753237](https://stackoverflow.com/a/5076962/753237) – [ic3b3rg](#) Jun 27 '16 at 2:18

1 I believe that's what @AndrewMao was saying, really. His first sentence might be rewritten "This only works in situations where `undefined` and `null` are not practical equivalents." – [BobRodes](#) Jun 30 '16 at 2:22

1 @BobRodes Thanks for clarifying my poor writing, I appreciate it :) – [Andrew Mao](#) Jun 30 '16 at 15:16 ✎

@AndrewMao You are very welcome. And I wouldn't call your writing poor, personally; I'd say it's a good deal better than average. :) – [BobRodes](#) Jun 30 '16 at 18:23

### Strict equality operator:-

27

We can check null by `===`

```
if ( value === null ){
}
```

Just by using `if`

```
if( value ) {
```

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

will evaluate to true if **value is not**:

- null
- undefined
- NaN
- empty string ("")
- false
- 0

edited Jun 19 '17 at 5:14

answered Jul 5 '16 at 11:58



Arshid KV

6,194 3 23 28

Improvement over the accepted answer by explicitly checking for `null` but with a simplified syntax:

7

```
if ([pass, cpass, email, cemail, user].every(x=>x!==null)) {  
    // your code here ...  
}
```

```
// Test  
let pass=1, cpass=1, email=1, cemail=1, user=1; // just to test  
  
if ([pass, cpass, email, cemail, user].every(x=>x!==null)) {  
    // your code here ...  
    console.log ("Yay! None of them are null");  
} else {  
    console.log ("Oops! At-least one of them is null");  
}
```

Run code snippet

[Expand snippet](#)

edited Apr 21 '18 at 12:34

answered Mar 14 '18 at 3:45

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

Firstly, you have a return statement without a function body. Chances are that that will throw an error.

6

A cleaner way to do your check would be to simply use the ! operator:

```
if (!pass || !cpass || !email || !cemail || !user) {
    alert("fill all columns");
}
```

edited Dec 14 '13 at 13:55



Nikhil Agrawal

17.9k 16 77 111

answered May 14 '11 at 18:20



Joey C.

1,708 2 13 12

11 That code is probably in a function, he just didn't show it ;) – [Mark Kahn](#) May 14 '11 at 18:22

you can use try catch finally

4

```
try {
    document.getElementById("mydiv").innerHTML = 'Success' //assuming "mydiv" is
    undefined
} catch (e) {

    if (e.name.toString() == "TypeError") //evals to true in this case
        //do something

} finally {}
```

you can also throw your own errors. See [this](#).

edited Dec 14 '13 at 13:55



Nikhil Agrawal

17.9k 16 77 111

answered May 14 '11 at 18:25



DrStrangeLove

6,015 14 47 67

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



we shouldn't reach this code unless we're confident that's the case, and we'd have plenty of avenues such as response codes to make sure we're confident before attempting such a line. – [calql8edkos](#) May 5 '15 at 16:09



In JavaScript, no string is equal to `null`.

4

Maybe you expected `pass == null` to be true when `pass` is an empty string because you're aware that the loose equality operator `==` performs certain kinds of type coercion.



For example, this expression is true:

```
'' == 0
```

In contrast, the strict equality operator `===` says that this is false:

```
'' === 0
```

Given that `''` and `0` are loosely equal, you might reasonably conjecture that `''` and `null` are loosely equal. However, they are not.

This expression is false:

```
'' == null
```

The result of comparing any string to `null` is false. Therefore, `pass == null` and all your other tests are always false, and the user never gets the alert.

To fix your code, compare each value to the empty string:

```
pass === ''
```

If you're certain that `pass` is a string, `pass == ''` will also work because only an empty string is loosely equal to the empty string. On the other hand, some experts say that it's a good practice to always use strict equality in JavaScript unless you specifically want to do the type coercion that the loose equality operator performs.

If you want to know what pairs of values are loosely equal, see the table "Sameness comparisons" in the [Mozilla article on this topic](#).

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



to check for **undefined** and **null** in javascript you need just to write the following :

4

```
if (!var) {  
    console.log("var IS null or undefined");  
} else {  
    console.log("var is NOT null or undefined");  
}
```

edited May 12 '16 at 17:10



Sunny Patel

5,409 2 23 37

answered Feb 25 '15 at 10:48



Nejmeddine Jammeli

480 4 13

6 !var is true with 0, "", NaN, and false, too. – [Matt](#) Aug 13 '15 at 16:10

This is a comment on WebWanderer's solution regarding checking for NaN (I don't have enough rep yet to leave a formal comment). The solution reads as

3

```
if(!parseInt(variable) && variable != 0 && typeof variable === "number")
```

but this will fail for rational numbers which would round to 0 , such as `variable = 0.1` . A better test would be:

```
if(isNaN(variable) && typeof variable === "number")
```

answered Feb 18 '15 at 19:10




Gabriel

460 4 11

1 Thanks for pointing out the bug Gabriel, I've put the fix into my answer. Aside from that, I was able to fix the test by changing `parseInt` to `parseFloat`

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

- 1 Great, that seems to work. Thanks for the comment on why you avoided `isNaN` as well, I can get behind that logic. There's also the `Underscore.js` method, which seems even more confusing/blackbox, but worth noting anyway because it takes advantage of `NaN !== NaN`.  
`Object.prototype.toString.call(variable) === '[object Number]' && variable !== +variable` – [Gabriel](#) Apr 18 '15 at 0:23 

Ooh! That's actually pretty cool! Thanks for the info Gabel! – [WebWanderer](#) Apr 20 '15 at 14:33

Actually I think you may need to use `if (value !== null || value !== undefined)` because if you use `if (value)` you may also filter 0 or false values.

1

Consider these two functions:

```
const firstTest = value => {  
  if (value) {  
    console.log('passed');  
  } else {  
    console.log('failed');  
  }  
}  
  
const secondTest = value => {  
  if (value !== null && value !== undefined) {  
    console.log('passed');  
  } else {  
    console.log('failed');  
  }  
}
```

```
firstTest(0);           // result: failed  
secondTest(0);          // result: passed
```

```
firstTest(false);       // result: failed  
secondTest(false);      // result: passed
```

```
firstTest('');          // result: failed  
secondTest('');         // result: passed
```

```
firstTest(null);        // result: failed  
secondTest(null);       // result: failed
```

```
firstTest(undefined);   // result: failed  
secondTest(undefined);  // result: failed
```

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

answered Nov 4 '18 at 14:52



Naeem Baghi

193 1 6 20

I found a another way to test if the value is null:

1 `if(variable >= 0 && typeof variable === "object")`

`null` acts as a number and object at the same time. Comparing `null >= 0` or `null <= 0` results in `true`. Comparing `null === 0` or `null > 0` or `null < 0` will result in `false`. But as `null` is also an object we can detect it as a null.

I made a more complex function **natureof** witch will do better than `typeof` and can be told what types to include or keep grouped

```
/* function natureof(variable, [included types])
included types are
  null - null will result in "undefined" or if included, will result in "null"
  NaN - NaN will result in "undefined" or if included, will result in "NaN"
  -infinity - will separate negative -Infinity from "Infinity"
  number - will split number into "int" or "double"
  array - will separate "array" from "object"
  empty - empty "string" will result in "empty" or
  empty=undefined - empty "string" will result in "undefined"
*/
function natureof(v, ...types){
  /*null*/      if(v === null) return types.includes('null') ? "null" : "undefined";
  /*NaN*/      if(typeof v == "number") return (isNaN(v)) ? types.includes('NaN') ?
  "NaN" : "undefined" :
  /*-infinity*/ (v+1 === v) ? (types.includes('-infinity') && v ===
  Number.NEGATIVE_INFINITY) ? "-infinity" : "infinity" :
  /*number*/   (types.includes('number')) ? (Number.isInteger(v)) ? "int" :
  "double" : "number";
  /*array*/    if(typeof v == "object") return (types.includes('array') &&
  Array.isArray(v)) ? "array" : "object";
  /*empty*/    if(typeof v == "string") return (v == "") ? types.includes('empty')
  ? "empty" :
  /*empty=undefined*/ types.includes('empty=undefined') ? "undefined" : "string" :
  "string";
  else return typeof v
}
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
let types = [null, "", "string", undefined, NaN, Infinity, -Infinity, false, "false",
true, "true", 0, 1, -1, 0.1, "test", {var:1}, [1,2], {0: 1, 1: 2, length: 2}]

for(i in types){
  console.log("natureof ", types[i], " = ", natureof(types[i], "null", "NaN", "-infinity",
"number", "array", "empty=undefined"))
}
```

[Run code snippet](#)[Expand snippet](#)

edited Mar 8 at 11:22

answered Mar 8 at 11:10

[Tarmo Saluste](#)

400 4 16

You can use lodash module to check value is null or undefined

1

`_.isNil(value)`**Example**

```
country= "Abc"
_.isNil(country)
//false
```

```
state= null
_.isNil(state)
//true
```

```
city= undefined
_.isNil(state)
//true
```

```
pin= true
_.isNil(pin)
// false
```

Reference link: <https://lodash.com/docs/#isNil>

edited May 19 at 8:01

answered May 19 at 6:34

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

This will not work in case of Boolean values coming from DB for ex:

0

```
value = false
```

```
if(!value) {  
  // it will change all false values to not available  
  return "not available"  
}
```

answered May 23 '16 at 6:48



Codiee

1,763

2

12

15

Please view carefully before downvote.

0

AFAIK in **JAVASCRIPT** when a variable is **declared** but has not assigned value, its type is `undefined` . so we can check variable even if it would be an `object` holding some **instance** in place of **value**.

create a helper method for checking nullity that returns `true` and use it in your API.

**helper function to check if variable is empty:**

```
function isEmpty(item){  
  if(item){  
    return false;  
  }else{  
    return true;  
  }  
}
```

**try-catch exceptional API call:**

```
try {
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```

    if(isEmpty(pass) || isEmpty(cpass) || isEmpty(email) || isEmpty(cemail) ||
isEmpty(user)){
        console.log("One or More of these parameter contains no vlaue. [pass] and-or
[cpass] and-or [email] and-or [cemail] and-or [user]");
    }else{
        // do stuff
    }

} catch (e) {
    if (e instanceof ReferenceError) {
        console.log(e.message); // debugging purpose
        return true;
    } else {
        console.log(e.message); // debugging purpose
        return true;
    }
}

```

### some test cases:

```

var item = ""; // isEmpty? true
var item = " "; // isEmpty? false
var item; // isEmpty? true
var item = 0; // isEmpty? true
var item = 1; // isEmpty? false
var item = "AAAAA"; // isEmpty? false
var item = NaN; // isEmpty? true
var item = null; // isEmpty? true
var item = undefined; // isEmpty? true

console.log("isEmpty? "+isEmpty(item));

```

edited Jun 25 '18 at 12:32

answered Oct 5 '15 at 14:46



Kaleem Ullah

3,938 1 32 35

1 What? This answer has nothing to do with this post. Did you post this answer on this thread by accident? – [WebWanderer](#) Nov 2 '15 at 20:32

I made this very simple function that works wonders:

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



```
function safeOrZero(route) {
  try {
    Function(`return (${route})`)();
  } catch (error) {
    return 0;
  }
  return Function(`return (${route})`)();
}
```

The route is whatever chain of values that can blow up. I use it for jQuery/cheerio and objects and such.

Examples 1: a simple object such as this `const testObj = {items: [{ val: 'haya' }, { val: null }, { val: 'hum!' }]};` .

But it could be a very large object that we haven't even made. So I pass it through:

```
let value1 = testobj.items[2].val; // "hum!"
let value2 = testobj.items[3].val; // Uncaught TypeError: Cannot read property 'val' of
undefined

let svalue1 = safeOrZero(`testobj.items[2].val`) // "hum!"
let svalue2 = safeOrZero(`testobj.items[3].val`) // 0
```

Of course if you prefer you can use `null` or `'No value'` ... Whatever suit your needs.

Usually a DOM query or a jQuery selector may throw an error if it's not found. But using something like:

```
const bookLink = safeOrZero($('span.guidebook > a')[0].href);
if(bookLink){
  [...]
}
```

answered Mar 12 at 6:56



[Carles Alcolea](#)

3,743 4 17 41



Try this:

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).





}

edited Nov 3 '15 at 14:43



THX-1138

11.1k

19

83

141

answered Nov 3 '15 at 14:23



user5520516

11

---

null is only thing that is "falsy" and typeof returns "object". – [user5520516](#) Nov 3 '15 at 14:25

---

- 1 How is that better than `if (variable === null)` ? Also someone already provided that answer last year: [stackoverflow.com/a/27550756/218196](https://stackoverflow.com/a/27550756/218196) . – [Felix Kling](#) Nov 3 '15 at 14:44
- 

**protected** by [Mark Rotteveel](#) May 19 at 8:01

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?