How to check if object property exists with a variable holding the property name?

Asked 7 years, 2 months ago Active 3 days ago Viewed 572k times

597

111

I am checking for the existence of an object property with a variable holding the property name in question.

```
var myObj;
myObj.prop = "exists";
var myProp = "p"+"r"+"o"+"p";

if(myObj.myProp){
    alert("yes, i have that property");
};
```

This is `undefined` because it's looking for `myObj.myProp` but I want it to check for `myObj.prop`

javascript object

edited Aug 10 '17 at 5:21
robinCTS
5,041 8 22 35

asked Jun 14 '12 at 19:58
Slopeside Creative
3,493 4 12 17

1 Possibly useful: From a comment by Pablo Cabrera at NCZOnline: "I think it’s worth to note that if the `hasOwnProperty` method is overwritten, you can rely on the `Object.prototype.hasOwnProperty.call(object, property)` ." – HumanInDisguise Apr 7 '15 at 8:45

10 is stackoverflow.com/questions/4244896/… a duplicate of this question? how is that? 'checking existence' and 'accessing value' are different things? Please correct me if I am wrong .... – adnan2nd May 10 '18 at 17:14

this is not a duplicate. – Jeff Clayton Sep 6 at 20:13

6 Answers

1158

✓

```
var myProp = 'prop';
if(myObj.hasOwnProperty(myProp)){
    alert("yes, i have that property");
}
```

Or

```
var myProp = 'prop';
if(myProp in myObj){
    alert("yes, i have that property");
}
```

Or

```
if('prop' in myObj){
    alert("yes, i have that property");
}
```

Note that `hasOwnProperty` doesn't check for inherited properties, whereas `in` does. For example `'constructor' in myObj` is true, but `myObj.hasOwnProperty('constructor')` is not.

edited Jul 8 at 21:51

answered Jun 14 '12 at 20:01

Rocket Hazmat
177k 34 258 304

16 `hasOwnProperty()` is better then `myObj[myProp]` (from other answers) as it works even if the value of `myProp` is 0 – Matt R Oct 26 '12 at 19:48

8 The "in" operator does not work with strings. e.g. 'length' in 'qqq' will produce an exception. So if you want a general purpose check you need to use hasOwnProperty. – Jacob Jun 19 '14 at 17:55

@Jacob what do you mean when you say 'The "in" operator does not work with strings'? with "in"' operator the left expression must be a string or value that can converts into a string. Yes, you cannot write 'length' in 'qqq' but you cannot write 'qqq'.hasOwnProperty('length') either – Wachburn Jul 21 '15 at 17:18

@Wachburn: `'qqq'.hasOwnProperty('length')` is `true`, you can do that. – Rocket Hazmat Jul 21 '15 at 17:48

You can use **hasOwnProperty**, but based on the reference you need quotes when using this method:

43

```
if (myObj.hasOwnProperty('myProp')) {
    // do something
}
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/hasOwnProperty

Another way is to use **in** operator, but you need **quotes** here as well:

```
if ('myProp' in myObj) {
    // do something
}
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/in

edited Aug 18 '15 at 10:28

answered May 10 '15 at 7:22
Adorjan Princz
10.2k 2 28 23

5 That is **not** how `hasOwnProperty()` is implemented. – canon Jul 9 '15 at 0:59

7 This is incorrect. By putting quotes around the name myProp, you are no longer referencing the value of myProp, rather you are declaring a new String() of 'myProp' and there is no such property of 'myProp' in myObj. – TriumphST Jun 11 '16 at 23:32

1 TriumpST: from MDN linked above, "prop - A string or symbol representing a property name or array index (non-symbols will be coerced to strings)." – Ben Creasy Oct 18 '17 at 18:22

This is correct. If you don't want to use a variable, but just if a specific 'myProp' is present, you need the quotes. – Katinka Hesselink Oct 24 '18 at 9:51

@KatinkaHesselink: Your comment is misleading. The question was "How to check if object property exists with a variable holding the property name?" – Herbert Van-Vliet Jan 8 at 10:36

24

Thank you for everyone's assistance and pushing to get rid of the eval statement. Variables needed to be in brackets, not dot notation. This works and is clean, proper code.

Each of these are variables: appChoice, underI, underObstr.

edited Jun 15 '12 at 15:21

answered Jun 15 '12 at 15:15

Slopeside Creative
3,493 4 12 17

This looks like a problem to me. If `tData.tonicdata[appChoice]` results in a value that doesn't have a property/index that matches `underI` , then this will result in an `TypeError` being thrown. – Ynot Mar 26 at 22:37

Despite your intentions with your initial post, you actually asked a different question than the one for which you provided this answer. You wanted to check the existence of a property, you don't mention anything about how to access it. Which makes this answer unrelated to the actual question. – Forage Apr 11 at 11:34

10

A much more secure way to check if property exists on the object is to use empty object or object prototype to call `hasOwnProperty()`

```
var foo = {
  hasOwnProperty: function() {
    return false;
  },
  bar: 'Here be dragons'
};

foo.hasOwnProperty('bar'); // always returns false

// Use another Object's hasOwnProperty and call it with 'this' set to foo
({}).hasOwnProperty.call(foo, 'bar'); // true

// It's also possible to use the hasOwnProperty property from the Object
// prototype for this purpose
Object.prototype.hasOwnProperty.call(foo, 'bar'); // true
```

Reference from [MDN Web Docs - Object.prototype.hasOwnProperty()]()

edited Aug 14 '17 at 11:33

answered Jul 25 '16 at 12:36

skmasq
2,431 6 33 65

If you are incorporating JavaScript that might do something evil like override `hasOwnProperty` , no amount of guards like this will make your code safe property name, which not necessarily mean their are doing something evil intentionally. – skmasq Jan 11 at 18:16

9

For own property :

```
var loan = { amount: 150 };
if(Object.prototype.hasOwnProperty.call(loan, "amount"))
{
   //will execute
}
```

Note: using **Object.prototype.hasOwnProperty** is better than loan.hasOwnProperty(..), in case a custom hasOwnProperty is defined in the prototype chain (which is not the case here), like

```
var foo = {
      hasOwnProperty: function() {
        return false;
      },
      bar: 'Here be dragons'
    };
```

// https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/hasOwnProperty

To include inherited properties in the finding use the **in** operator: (but you must place an object at the right side of 'in', primitive values will throw error, e.g. **'length' in 'home'** will throw error, but **'length' in new String('home')** won't)

```
const yoshi = { skulk: true };
const hattori = { sneak: true };
const kuma = { creep: true };
if ("skulk" in yoshi)
    console.log("Yoshi can skulk");

if (!("sneak" in yoshi))
    console.log("Yoshi cannot sneak");

if (!("creep" in yoshi))
    console.log("Yoshi cannot creep");

Object.setPrototypeOf(yoshi, hattori);
```

```
if (!("creep" in hattori))
    console.log("Hattori cannot creep");

Object.setPrototypeOf(hattori, kuma);

if ("creep" in hattori)
    console.log("Hattori can now creep");
if ("creep" in yoshi)
    console.log("Yoshi can also creep");
```

// https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/in

Note: One may be tempted to use typeof and [ ] property accessor as the following code which **doesn't work always** ...

```
var loan = { amount: 150 };

loan.installment = undefined;

if("installment" in loan) // correct
{
    // will execute
}

if(typeof loan["installment"] !== "undefined") // incorrect
{
    // will not execute
}
```

edited Sep 19 '18 at 16:13

answered Oct 15 '17 at 12:57

adnan2nd

723 8 12

2

You can use `hasOwnProperty()` as well as `in` operator.

edited Nov 21 '15 at 17:21

Martijn Pieters ♦

759k 164 2759 2476

answered Nov 7 '15 at 15:38

Simran Kaur

505 3 13 36