Dynamically access object property using variable

Asked 8 years, 8 months ago Active 8 months ago Viewed 283k times

611

I'm trying to access a property of an object using a dynamic name. Is this possible?

```
const something = { bar: "Foobar!" };
const foo = 'bar';
something.foo; // The idea is to access something.bar, getting "Foobar!"
```

132

javascript object properties

edited Mar 22 '17 at 16:12

Taryn ♦

198k 47 304 364

asked Nov 22 '10 at 11:23

RichW

3,616 4 17 29

3 See also property access: dot notation vs. brackets? and How do I add a property to an object using a variable as the name? – Bergi Nov 18 '14 at 6:11

11 Answers

807

There are two ways to access properties of an object:

- Dot notation: `something.bar`
- Bracket notation: `something['bar']`

The value between the brackets can be any expression. Therefore, if the property name is stored in a variable, you have to use bracket notation:

```
var foo = 'bar';
something[foo];
// both x = something[foo] and something[foo] = x work as expected
```

edited Jan 2 '18 at 18:05
Salman A
192k 69 351 450

answered Nov 22 '10 at 11:25
Jan Hančič
43k 13 85 92

28 careful with this: javascript compilers will error here since they dont rename strings but they do rename object properties – chacham15 Dec 6 '11 at 8:40

6 Some more info on why this is possible: JS objects are associative arrays, that's why. Further Reading: quirksmode.org/js/associative.html stackoverflow.com/questions/14031368/… – Sudhanshu Mishra Jun 3 '14 at 9:00

@dotnetguy No they are not. Arrays are objects that inherit from the plain JS object prototype and therefore you can add properties a go-go like any plain object. The 'associative' behaviour is more object-like than array like. You can't iterate the 'associative' version by simple index so it is not displaying array-like behaviour. You can define your 'associative' array as {} or [] and treat it the same in either case as far as random property access is concerned. – Vanquished Wombat Jan 3 '17 at 16:01

3 @VanquishedWombat Not sure what your objection pertains to? I did not say that JS Objects are arrays? – Sudhanshu Mishra Jan 6 '17 at 0:30

as a reference to the correct answer , Reference – youhana Jun 8 '17 at 21:23

72

This is my solution:

```
function resolve(path, obj) {
    return path.split('.').reduce(function(prev, curr) {
        return prev ? prev[curr] : null
    }, obj || self)
}
```

Usage examples:

```
resolve("document.body.style.width")
// or
resolve("style.width", document.body)
// or even use array indexes
// (someObject has been defined in the question)
resolve("part.0.size", someObject)
// returns null when intermediate properties are not defined:
resolve('properties.that.do.not.exist', {hello:'world'})
```

edited Sep 23 '17 at 14:45
GeekyDeaks

answered Jul 26 '17 at 8:57
abahet


520 5 11


4,629 1 22 19

6 This is similar to lodash get – Moby Disk Dec 15 '17 at 14:06

Excellent answer, see also: stackoverflow.com/questions/37510640/… – Julian Knight Jan 3 at 13:45

1 You inspired me to create an enhanced version that allows bracket notation & property names with spaces as well as validating the inputs: it.knightnet.org.uk/kb/node-js/get-properties – Julian Knight Jan 3 at 14:04

33

In javascript we can access with:

- dot notation - `foo.bar`
- square brackets - `foo[someVar]` or `foo["string"]`

But only second case allows to access properties dynamically:

```
var foo = { pName1 : 1, pName2 : [1, {foo : bar }, 3] , ...}

var name = "pName"
var num  = 1;

foo[name + num]; // 1

// --

var a = 2;
var b = 1;
var c = "foo";

foo[name + a][b][c]; // bar
```

answered Jul 1 '14 at 15:40


Sonique
2,730 6 27 45

1 I'm staring at 2,000 lines of if statements because the previous dev didn't use square brackets, and statically accessed object properties by dot notation. It's for an approval process app that has 7 different approvers and the steps are all the same. /rip – Chad Jun 7 '18 at 14:28

20

Following is an ES6 example of how you can access the property of an object using a property name that has been dynamically generated by concatenating two strings.

```
var suffix = " name";

var person = {
    ["first" + suffix]: "Nicholas",
    ["last" + suffix]: "Zakas"
};

console.log(person["first name"]);      // "Nicholas"
console.log(person["last name"]);       // "Zakas"
```

This is called computed property names

edited Jul 10 '17 at 6:11
try-catch-finally
5,124 4 28 53

answered Aug 2 '16 at 19:46
zloctb
5,398 3 47 59

15

You can achieve this in quite a few different ways.

```
let foo = {
    bar: 'Hello World'
};

foo.bar;
foo['bar'];
```

The bracket notation is specially powerful as it let's you access a property based on a variable:

```
let foo = {
    bar: 'Hello World'
};

let prop = 'bar';

foo[prop];
```

This can be extended to looping over every property of an object. This can be seem redundant due to newer JavaScript constructs such as for ... of ..., but helps illustrate a use case:

```
let foo = {
    bar: 'Hello World',
    baz: 'How are you doing?',
    last: 'Quite alright'
};

for (let prop in foo.getOwnPropertyNames()) {
    console.log(foo[prop]);
}
```

Both dot and bracket notation also work as expected for nested objects:

```
let foo = {
    bar: {
        baz: 'Hello World'
    }
};

foo.bar.baz;
foo['bar']['baz'];
foo.bar['baz'];
foo['bar'].baz;
```

Object destructuring

We could also consider object destructuring as a means to access a property in an object, but as follows:

```
let foo = {
    bar: 'Hello World',
    baz: 'How are you doing?',
    last: 'Quite alright'
};

let prop = 'last';
let { bar, baz, [prop]: customName } = foo;

// bar = 'Hello World'
// baz = 'How are you doing?'
// customName = 'Quite alright'
```

edited Dec 13 '17 at 8:06

answered Mar 8 '17 at 11:30

Gorka Hernandez

2,507 12 24

9

You can do it like this using Lodash get

```
_.get(object, 'a[0].b.c');
```

edited Sep 6 '18 at 15:57

JJJ

29.5k 16 78 93

answered Sep 6 '18 at 6:36

shalonteoh

453 1 6 11

There are many situations, such as deep nested object lookups, where this is the only option. – Jonathan Kempf Jun 27 at 19:51

8

UPDATED

I have take comments below into consideration and agreed. Eval is to be avoided.

Accessing root properties in object is easily achieved with `obj[variable]` , but getting nested complicates thing. Not to write already written code I suggest to use `lodash.get` .

Example

```
// Accessing root property
var rootProp = 'rootPropert';
_.get(object, rootProp, defaultValue);

// Accessing nested property
var listOfNestedProperties = [var1, var2];
_.get(object, listOfNestedProperties);
```

Lodash get can be used on different ways, here is link to the documentation lodash.get

edited Jun 19 '18 at 9:29

answered Jun 22 '15 at 8:10

Mr Br

2,077 13 17

4 It's best to avoid using `eval` whenever possible. stackoverflow.com/questions/86513/… – Luke Jun 23 '15 at 18:07

8 Using `eval` for something as trivial as accessing properties is plain overkill and hardly advisable under any circumstance. What's "trouble"? `obj['nested']['test']` works very well and doesn't require you to embed code in strings. – Paul Stenne Oct 23 '15 at 10:14

3 eval is three times slower or more, I wouldn't recommend this to newbies because it might teach them bad habits. I use `obj['nested']['value']` - remember kids, eval is evil! – jaggedsoft Nov 26 '15 at 1:25

1 @Luke He's now the only want to bring Lodash `_.get` to the table. I think this answer deserves now upvotes instead of downvotes. It may be overkill, but it's good to know it exists. – Emile Bergeron Dec 20 '16 at 21:42

1 Thank you for introducing lodash for this. I came here by google looking for a method to set a value deep in an object, and used their _.set method (which is identical to above but with the extra arguement for the value to set). – TPHughes Jul 3 '18 at 9:03

5

Whenever you need to access property dynamically you have to use square bracket for accessing property not "." operator
Syntax: object[propery}

```
const something = { bar: "Foobar!" };
const foo = 'bar';
// something.foo; -- not correct way at it is expecting foo as proprty in  something={
foo: "value"};
// correct way is  something[foo]
alert( something[foo])
```

Run code snippet Expand snippet

edited Jul 31 '18 at 8:25

Manasi
558 4 16

answered Jul 31 '18 at 8:24

Rupesh Agrawal
698 5 20

2

It gets interesting when you have to pass parameters to this function as well.

Code *jsfiddle*

```
var obj = {method:function(p1,p2,p3){console.log("method:",arguments)}}

var str = "method('p1', 'p2', 'p3');"

var match = str.match(/^\s*(\S+)\((.*)\);\s*$/);

var func = match[1]
var parameters = match[2].split(',');
for(var i = 0; i < parameters.length; ++i) {
  // clean up param begninning
    parameters[i] = parameters[i].replace(/^\s*['"]?/,'');
  // clean up param end
  parameters[i] = parameters[i].replace(/['"]?\s*$/,'');
}

obj[func](parameters); // sends parameters as array
obj[func].apply(this, parameters); // sends parameters as individual values
```

answered Nov 13 '16 at 17:37

Jacksonkr
18.7k 35 145 244

-2

You should use `JSON.parse` , take a look at https://www.w3schools.com/js/js_json_parse.asp

```
const obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}')
console.log(obj.name)
console.log(obj.age)
```

answered Jun 14 '17 at 21:42

onmyway133
27.4k 15 178 207

-3

```
const something = { bar: "Foobar!" };
const foo = 'bar';

something[\`${foo}\`];
```

edited Sep 19 '17 at 17:12

Cody Gray ♦
199k 37 401 483

answered Jul 2 '17 at 19:04

Sergey
173 1 4

7 Why on earth would you do that? Your `foo` is already a string, so `` `${foo}` `` is exactly the same as `foo`. (Also, your code seems to have some extra backslashes that don't belong there. But it would still be pointless even if you fixed that syntax error.) – Ilmari Karonen Sep 19 '17 at 18:49

protected by Samuel Liew ♦ Oct 5 '15 at 9:00

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?