

Object.prototype.hasOwnProperty() - JavaScript | MDN

笔记本： 程序员

创建时间： 2020/8/20 星期四 14:49

URL: https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Ob...

hasOwnProperty() 方法会返回一个布尔值，指示对象自身属性中是否具有指定的属性（也就是，是否有指定的键）。



JavaScript Demo:

Object.prototype.hasOwnProperty()

```
1const object1 = {};  
2  
3object1.property1 = 42;  
4  
5  
6console.log(object1.hasOwnProperty('property1'));  
7//5expected output: true  
8  
9  
10console.log(object1.hasOwnProperty('toString'));  
11//8expected output: false  
12  
13  
14console.log(object1.hasOwnProperty('hasOwnProperty'));  
15//11expected output: false  
16  
17  
18
```

语法

`obj.hasOwnProperty(prop)`

参数

prop

要检测的属性的 `String` 字符串形式表示的名称，或者 `Symbol`。

返回值

用来判断某个对象是否含有指定的属性的布尔值 `Boolean`。

描述

所有继承了 `Object` 的对象都会继承到 `hasOwnProperty` 方法。这个方法可以用来检测一个对象是否含有特定的自身属性；和 `in` 运算符不同，该方法会忽略掉那些从原型链上继承到的属性。

备注

即使属性的值是 `null` 或 `undefined`，只要属性存在，`hasOwnProperty` 依旧会返回 `true`。

```
1 | o = new Object();  
2 | o.propOne = null;  
3 | o.hasOwnProperty('propOne'); // 返回 true  
4 | o.propTwo = undefined;  
5 | o.hasOwnProperty('propTwo'); // 返回 true
```

示例

使用 `hasOwnProperty` 方法判断属性是否存在

下面的例子检测了对象 `o` 是否含有自身属性 `prop`:

```
1 | o = new Object();
2 | o.hasOwnProperty('prop'); // 返回 false
3 | o.prop = 'exists';
4 | o.hasOwnProperty('prop'); // 返回 true
5 | delete o.prop;
6 | o.hasOwnProperty('prop'); // 返回 false
```

自身属性与继承属性

下面的例子演示了 `hasOwnProperty` 方法对待自身属性和继承属性的区别:

```
1 | o = new Object();
2 | o.prop = 'exists';
3 | o.hasOwnProperty('prop');           // 返回 true
4 | o.hasOwnProperty('toString');      // 返回 false
5 | o.hasOwnProperty('hasOwnProperty'); // 返回 false
```

遍历一个对象的所有自身属性

下面的例子演示了如何在遍历一个对象的所有属性时忽略掉继承属性, 注意这里 `for...in` 循环只会遍历可枚举属性, 所以不应该基于这个循环中没有不可枚举的属性而得出 `hasOwnProperty` 是严格限制于可枚举项目的 (如同 `Object.getOwnPropertyNames()`)。

```
1 | var buz = {
2 |     fog: 'stack'
3 | };
4 |
   | for (var name in buz) {
```

```

5     if (buz.hasOwnProperty(name)) {
6         console.log('this is fog (' +
7             name + ') for sure. Value: ' + buz[name]);
8     }
9     else {
10        console.log(name); // toString or something else
11    }
12 }
13

```

使用 `hasOwnProperty` 作为属性名

JavaScript 并没有保护 `hasOwnProperty` 这个属性名，因此，当某个对象可能自有一个占用该属性名的属性时，就需要使用外部的 `hasOwnProperty` 获得正确的结果：

```

1  var foo = {
2      hasOwnProperty: function() {
3          return false;
4      },
5      bar: 'Here be dragons'
6  };
7
8  foo.hasOwnProperty('bar'); // 始终返回 false
9
10 // 如果担心这种情况，
11 // 可以直接使用原型链上真正的 hasOwnProperty 方法
12 ({}).hasOwnProperty.call(foo, 'bar'); // true
13
14 // 也可以使用 Object 原型上的 hasOwnProperty 属性
15 Object.prototype.hasOwnProperty.call(foo, 'bar'); //

```














注意，只有在最后一种情况下，才不会新建任何对象。

规范

规范	状态	备注
ECMAScript (ECMA-262) Object.prototype.hasOwnProperty	<div>LS</div> Living Standard	
ECMAScript 2015 (6th Edition, ECMA-262) Object.prototype.hasOwnProperty	<div>ST</div> Standard	
ECMAScript 5.1 (ECMA-262) Object.prototype.hasOwnProperty	<div>ST</div> Standard	
ECMAScript 3rd Edition (ECMA-262)	<div>ST</div> Standard	Initial definition. Implemented in JavaScript 1.5.

浏览器兼容性

[Update compatibility data on GitHub](#)

	<div>Desktop</div>						<div>Mobile</div>					<div>Server</div>	
	 Chrome	 Edge	 Firefox	 Internet Explorer	 Opera	 Safari	 Android webview	 Chrome for Android	 Firefox for Android	 Opera for Android	 Safari on iOS	 Samsung Internet	 Node.js
hasOwnProperty	1	12	1	5.5	5	3	1	18	4	10.	1	1.0	Yes

What are we missing?



Full support

参见

- [属性的可枚举性和所有权](#)
- [Object.getOwnPropertyNames\(\)](#)
- [for...in](#)
- [in](#)
- [继承与原型链](#)