

Object.keys()

Object.keys() 方法会返回一个由一个给定对象的自身可枚举属性组成的数组，数组中属性名的排列顺序和使用 `for...in` 循环遍历该对象时返回的顺序一致（两者的主要区别是一个 `for-in` 循环还会枚举其原型链上的属性）。

语法

```
Object.keys(obj)
```

参数

obj

要返回其枚举自身属性的对象。

返回值

一个表示给定对象的所有可枚举属性的字符串数组。

描述

Object.keys 返回一个所有元素为字符串的数组，其元素来自于从给定的对象上面可直接枚举的属性。这些属性的顺序与手动遍历该对象属性时的一致。

例子

```
1  /* Array 对象 */
2  let arr = ["a", "b", "c"];
3  console.log(Object.keys(arr));
4  // ['0', '1', '2']
5
6  /* 类数组 对象 */
7  let obj = { 0 : "a", 1 : "b", 2 : "c"};
8  console.log(Object.keys(obj));
9  // ['0', '1', '2']
10
```

```
11 // 类数组 对象, 随机 key 排序
12 let anObj = { 100: 'a', 2: 'b', 7: 'c' };
13
14 console.log(Object.keys(anObj));
15 // ['2', '7', '100']
16
17 /* getFoo 是个不可枚举的属性 */
18 var my_obj = Object.create(
19     {},
20     { getFoo : { value : function () { return this.foo } } }
21 );
22 my_obj.foo = 1;
23
24 console.log(Object.keys(my_obj));
25 // ['foo']
```

如果你想获取一个对象的所有属性, 甚至包括不可枚举的, 请查看 [Object.getOwnPropertyNames](#)。

注意

在ES5里, 如果此方法的参数不是对象 (而是一个原始值), 那么它会抛出 `TypeError`。在ES2015中, 非对象的参数将被强制转换为一个对象。

```
1 Object.keys("foo");
2 // TypeError: "foo" is not an object (ES5 code)
3
4 Object.keys("foo");
5 // ["0", "1", "2"] (ES2015 code)
```

Polyfill

要在原生不支持的旧环境中添加兼容的 `Object.keys`, 请复制以下代码段:

```
1 if (!Object.keys) {
2     Object.keys = (function () {
3         var hasOwnProperty = Object.prototype.hasOwnProperty,
4             hasDontEnumBug = !({toString: null}).propertyIsEnumerable('toString'),
5             dontEnums = [
6                 'toString',
7                 'toLocaleString',
```

```
7      'valueOf',
8      'hasOwnProperty',
9      'isPrototypeOf',
10     'propertyIsEnumerable',
11     'constructor'
12 ],
13     dontEnumsLength = dontEnums.length;
14
15     return function (obj) {
16         if (typeof obj !== 'object' && typeof obj !== 'function' || obj ===
17
18         var result = [];
19
20         for (var prop in obj) {
21             if (hasOwnProperty.call(obj, prop)) result.push(prop);
22         }
23
24         if (hasDontEnumBug) {
25             for (var i=0; i < dontEnumsLength; i++) {
26                 if (hasOwnProperty.call(obj, dontEnums[i])) result.push(dontEn
27             }
28         }
29         return result;
30     }
31 }());
32 };
33
```

上面的代码在IE7（也许IE8也是）下有个问题，就是如果传入一个来自其他 window 对象下的对象时，不可枚举的属性也会获取到。

另一个简单点的实现方法，来自 [Javascript - Object.keys Browser Compatibility](#)

规范


Specification	Status	Comment
ECMAScript 5.1 (ECMA-262) Object.keys	ST Standard	Initial definition. Implemented in JavaScript 1.8.5
ECMAScript 2015 (6th Edition, ECMA-262) Object.keys	ST Standard	

↗ ECMAScript Latest Draft (ECMA-262) Object.keys	 Living Standard	
---------------------------------------------------------------------	---------------------------------------------------------------------------------------------------	--

浏览器兼容性

	Desktop	Mobile				
Feature	Chrome	Edge	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	5	(Yes)	4.0 (2.0)	9	12	5

相关链接

- [Enumerability and ownership of properties](#)
- [Object.prototype.propertyIsEnumerable](#)
- [Object.create](#)
- [Object.getOwnPropertyNames](#)
- [↗ Bug 307791 - Implement ES5's Object.keys\(O\)](#)
- [Object.values\(\)](#) 
- [Object.entries\(\)](#) 