Object.getOwnPropertyNames()

`Object.getOwnPropertyNames()`方法返回一个由指定对象的所有自身属性的属性名（包括不可枚举属性但不包括Symbol值作为名称的属性）组成的数组。

语法

```
Object.getOwnPropertyNames(obj)
```

参数

obj

一个对象，其自身的可枚举和不可枚举属性的名称被返回。

返回值

在给定对象上找到的属性对应的字符串数组。

描述

`Object.getOwnPropertyNames` 返回一个数组，该数组对元素是 `obj` 自身拥有的枚举或不可枚举属性名称字符串。 数组中枚举属性的顺序与通过 `for...in` 循环（或 `Object.keys` ）迭代该对象属性时一致。 数组中不可枚举属性的顺序未定义。

示例

使用 Object.getOwnPropertyNames()

```
var arr = ["a", "b", "c"];
console.log(Object.getOwnPropertyNames(arr).sort()); // ["0", "1", "2", '

// 类数组对象
var obj = { 0: "a", 1: "b", 2: "c"};
console.log(Object.getOwnPropertyNames(obj).sort()); // ["0", "1", "2"]

// 使用Array.forEach输出属性名和属性值
Object.getOwnPropertyNames(obj).forEach(function(val, idx, array) {
```

```
  console.log(val + " -> " + obj[val]);
});
// 输出
// 0 -> a
// 1 -> b
// 2 -> c

//不可枚举属性
var my_obj = Object.create({}, {
  getFoo: {
    value: function() { return this.foo; },
    enumerable: false
  }
});
my_obj.foo = 1;

console.log(Object.getOwnPropertyNames(my_obj).sort()); // ["foo", "getFo
```

如果你只要获取到可枚举属性，查看 `Object.keys` 或用 for...in循环（还会获取到原型链上的可枚举属性，不过可以使用 `hasOwnProperty()` 方法过滤掉）。

下面的例子演示了该方法不会获取到原型链上的属性：

```
function ParentClass() {}
ParentClass.prototype.inheritedMethod = function() {};

function ChildClass() {
  this.prop = 5;
  this.method = function() {};
}

ChildClass.prototype = new ParentClass;
ChildClass.prototype.prototypeMethod = function() {};

console.log(
  Object.getOwnPropertyNames(
    new ChildClass()  // ["prop", "method"]
  )
);
```

只获取不可枚举的属性

下面的例子使用了 `Array.prototype.filter()` 方法，从所有的属性名数组（使用 `Object.getOwnPropertyNames()` 方法获得 ）中去除可枚举的属性（使用 `Object.keys()` 方法获得 ），剩余的属性便是不可枚举的属性了：

```
var target = myObject;
var enum_and_nonenum = Object.getOwnPropertyNames(target);
var enum_only = Object.keys(target);
var nonenum_only = enum_and_nonenum.filter(function(key) {
    var indexInEnum = enum_only.indexOf(key);
    if (indexInEnum == -1) {
        // not found in enum_only keys mean the key is non-enumerable,
        // so return true so we keep this in the filter
        return true;
    } else {
        return false;
    }
});

console.log(nonenum_only);
```

注:

在 ES5 中，如果参数不是一个对象类型，将抛出一个 `TypeError` 异常。在 ES2015 中， non-object 参数被强制转换为 object 。

```
Object.getOwnPropertyNames('foo');
// TypeError: "foo" is not an object (ES5 code)

Object.getOwnPropertyNames('foo');
// ['length', '0', '1', '2']  (ES2015 code)
```

规范

Specification	Status	Comment
ECMAScript 5.1 (ECMA-262) Object.getOwnPropertyNames	ST Standard	Initial definition. Implemented in JavaScript 1.8.5

ECMAScript 2015 (6th Edition, ECMA-262) Object.getOwnPropertyNames	ST Standard	
ECMAScript Latest Draft (ECMA-262) Object.getOwnPropertyNames	LS Living Standard	

浏览器兼容性

Desktop Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	5	4.0 (2)	9	12	5

Based on Kangax's compat table.

SpiderMonkey-specific notes

SpiderMonkey 28 (Firefox 28 / Thunderbird 28 / SeaMonkey 2.25 / Firefox OS 1.3) 版本之前， `Object.getOwnPropertyNames` 不会获取到 `Error` 对象的属性。该 bug 在后面的版本修复了 (bug 724768)。

相关链接

- Enumerability and ownership of properties
- `Object.prototype.hasOwnProperty`
- `Object.prototype.propertyIsEnumerable`
- `Object.create`
- `Object.keys`
- `Array.forEach()`