Object.isSealed()

`Object.isSealed()` 方法判断一个对象是否被密封。

语法

```
Object.isSealed(obj)
```

参数

`obj`
要被检查的对象。

返回值

表示给定对象是否被密封的一个 `Boolean` 。

描述

如果这个对象是密封的，则返回 `true`，否则返回 `false`。密封对象是指那些不可 扩展 的，且所有自身属性都不可配置且因此不可删除（但不一定是不可写）的对象。

例子

```
// 新建的对象默认不是密封的.
var empty = {};
Object.isSealed(empty); // === false

// 如果你把一个空对象变的不可扩展,则它同时也会变成个密封对象.
Object.preventExtensions(empty);
Object.isSealed(empty); // === true

// 但如果这个对象不是空对象,则它不会变成密封对象,因为密封对象的所有自身属性必须是
var hasProp = { fee: "fie foe fum" };
Object.preventExtensions(hasProp);
Object.isSealed(hasProp); // === false

```

```
// 如果把这个属性变的不可配置,则这个对象也就成了密封对象.
Object.defineProperty(hasProp, "fee", { configurable: false });
Object.isSealed(hasProp); // === true

// 最简单的方法来生成一个密封对象,当然是使用Object.seal.
var sealed = {};
Object.seal(sealed);
Object.isSealed(sealed); // === true

// 一个密封对象同时也是不可扩展的.
Object.isExtensible(sealed); // === false

// 一个密封对象也可以是一个冻结对象,但不是必须的.
Object.isFrozen(sealed); // === true ，所有的属性都是不可写的
var s2 = Object.seal({ p: 3 });
Object.isFrozen(s2); // === false，  属性"p"可写

var s3 = Object.seal({ get p() { return 0; } });
Object.isFrozen(s3); // === true ，访问器属性不考虑可写不可写,只考虑是否可配置
```

注意

在ES5中，如果这个方法的参数不是一个对象（一个原始类型），那么它会导致 `TypeError` 。在ES2015中，非对象参数将被视为是一个密封的普通对象，只返回 `true` 。

```
Object.isSealed(1);
// TypeError: 1 is not an object (ES5 code)

Object.isSealed(1);
// true                          (ES2015 code)
```

规范

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

Object.isSealed		
ECMAScript Latest Draft (ECMA-262) Object.isSealed	LS Living Standard	

浏览器兼容

Desktop Mobile

Feature	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari
Basic Support	6	(Yes)	4	9	12	5.1

相关链接

- `Object.seal()`
- `Object.preventExtensions()`
- `Object.isExtensible()`
- `Object.freeze()`
- `Object.isFrozen()`