Object.freeze()

`Object.freeze()` 方法可以冻结一个对象，冻结指的是不能向这个对象添加新的属性，不能修改其已有属性的值，不能删除已有属性，以及不能修改该对象已有属性的可枚举性、可配置性、可写性。也就是说，这个对象永远是不可变的。该方法返回被冻结的对象。

语法

```
Object.freeze(obj)
```

参数

`obj`
要被冻结的对象。

返回值

被冻结的对象。

描述

被冻结对象自身的所有属性都不可能以任何方式被修改。任何修改尝试都会失败，无论是静默地还是通过抛出 `TypeError` 异常（最常见但不仅限于strict mode）。

数据属性的值不可更改，访问器属性（有getter和setter）也同样（但由于是函数调用，给人的错觉是还是可以修改这个属性）。如果一个属性的值是个对象，则这个对象中的属性是可以修改的，除非它也是个冻结对象。

例子

```
1 var obj = {
2   prop: function() {},
3   foo: 'bar'
4 };
5
6 // New properties may be added, existing properties may be
7 // changed or removed
```

```
obj.foo = 'baz';
obj.lumpy = 'woof';
delete obj.prop;

// Both the object being passed as well as the returned
// object will be frozen. It is unnecessary to save the
// returned object in order to freeze the original.
var o = Object.freeze(obj);

o === obj; // true
Object.isFrozen(obj); // === true

// Now any changes will fail
obj.foo = 'quux'; // silently does nothing
// silently doesn't add the property
obj.quaxxor = 'the friendly duck';

// In strict mode such attempts will throw TypeErrors
function fail(){
  'use strict';
  obj.foo = 'sparky'; // throws a TypeError
  delete obj.quaxxor; // throws a TypeError
  obj.sparky = 'arf'; // throws a TypeError
}

fail();

// Attempted changes through Object.defineProperty;
// both statements below throw a TypeError.
Object.defineProperty(obj, 'ohai', { value: 17 });
Object.defineProperty(obj, 'foo', { value: 'eit' });

// It's also impossible to change the prototype
// both statements below will throw a TypeError.
Object.setPrototypeOf(obj, { x: 20 })
obj.__proto__ = { x: 20 }

// A frozen array is like a tuple.
let a=[0];
Object.freeze(a);
// The array cannot be modified now.
a[0]=1;
a.push(2);
```

```
// a=[0]
// A frozen array can be unpacked normally.
let b, c;
[b,c]=Object.freeze([1,2]);
// b=1, c=2
```

被冻结的对象是不可变的。但也不总是这样。下例展示了一个不是常量的冻结对象（浅冻结）。

```
obj1 = {
  internal: {}
};

Object.freeze(obj1);
obj1.internal.a = 'aValue';

obj1.internal.a // 'aValue'
```

对于一个常量对象，整个引用图（直接和间接引用其他对象）只能引用不可变的冻结对象。冻结的对象被认为是不可变的，因为整个对象中的整个对象状态（对其他对象的值和引用）是固定的。注意，字符串，数字和布尔总是不可变的，而函数和数组是对象。

要使对象不可变，需要递归冻结每个类型为对象的属性（深冻结）。当你知道对象在引用图中不包含任何 环 时，将根据你的设计逐个使用该模式，否则将触发无限循环。对 deepFreeze() 的增强将是具有接收路径（例如Array）参数的内部函数，以便当对象进入不变时，可以递归地调用 deepFreeze() 。你仍然有冻结不应冻结的对象的风险，例如[window]。

```
// To do so, we use this function.
function deepFreeze(obj) {

  // Retrieve the property names defined on obj
  var propNames = Object.getOwnPropertyNames(obj);

  // Freeze properties before freezing self
  propNames.forEach(function(name) {
    var prop = obj[name];

    // Freeze prop if it is an object
    if (typeof prop == 'object' && prop !== null)
      deepFreeze(prop);
  });

```

```
  // Freeze self (no-op if already frozen)
  return Object.freeze(obj);
}

obj2 = {
  internal: {}
};

deepFreeze(obj2);
obj2.internal.a = 'anotherValue';
obj2.internal.a; // undefined
```

Notes

在ES5中，如果这个方法的参数不是一个对象（一个原始值），那么它会导致 `TypeError` 。在ES2015中，非对象参数将被视为要被冻结的普通对象，并被简单地返回。

```
> Object.freeze(1)
TypeError: 1 is not an object // ES5 code

> Object.freeze(1)
1                             // ES2015 code
```

对比 `Object.seal()`

用 `Object.seal()` 密封的对象可以改变它们现有的属性。使用 `Object.freeze()` 冻结的对象中现有属性是不可变的。

规范

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

浏览器兼容性

Desktop Mobile

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

相关链接

- `Object.isFrozen`
- `Object.preventExtensions`
- `Object.isExtensible`
- `Object.seal`
- `Object.isSealed`