链滴

JavaScript 中的 Object

作者：lingyundu

原文链接：https://ld246.com/article/1603512476431

来源网站：链滴

在 JavaScript 中，几乎所有的对象都是 Object 类型的实例，它们都会从Object.prototype继承属性。

Object 是 JavaScript 的一个内置对象，它是一个构造函数，但是也可以用作普通的函数。

构造函数

Object 作为构造函数，可以使用new关键字来生成一个新的对象。

```
// 这是一个空对象
var obj = new Object();
```

Object 可以接受一个参数：

- 若参数是 null或者undefined，则返回一个空对象。
- 若参数是一个对象，则直接返回这个对象；
- 若参数是一个原始类型值，则返回该值的包装对象。

```
var o1 = new Object(null); // {}
var o2 = new Object(undefined); //{}

var o3 = {a: 1};
var o4 = new Object(o3);
o3 === o4 // true

var obj = new Object(123);
obj instanceof Number // true
```

普通函数

Object 作为普通函数，它的作用是将任意值转为对象。除了语义与构造函数不同，其效果是一样的。

```
// 若无参数，或者参数是`null`或`undefined`，则返回一个空对象。
var o1 = Object(); // {}
var o2 = Object(undefined); // {}
var o3 = Object(null); // {}

o1 instanceof Object // true

// 若参数是一个对象，则直接返回这个对象，不做转换。
var arr = [];
var obj = Object(arr); // 返回原数组
obj === arr // true

// 若参数是一个原始类型值，则将其转换为对应的包装对象。
var obj = Object(1);
obj instanceof Object // true
obj instanceof Number // true
```

构造函数的方法

Object 构造函数自身的属性中包含一些函数，这些函数类似于 Java 中的静态方法，可以使用 Object.

xx()形式进行调用。

查看 Object 构造函数的所有属性：

```
Object.getOwnPropertyNames(Object);
// ["length", "name", "prototype", "assign", "getOwnPropertyDescriptor", "getOwnPropertyDe
criptors", "getOwnPropertyNames", "getOwnPropertySymbols", "is", "preventExtensions", "sea
", "create", "defineProperties", "defineProperty", "freeze", "getPrototypeOf", "setPrototypeOf",
"isExtensible", "isFrozen", "isSealed", "keys", "entries", "fromEntries", "values"]
```

与对象属性相关的方法：

- Object.keys()：获取对象的属性名
- Object.getOwnPropertyNames()：获取对象的属性名。
- Object.getOwnPropertyDescriptor()：获取某个属性的描述对象。
- Object.defineProperty()：通过描述对象，定义某个属性。
- Object.defineProperties()：通过描述对象，定义多个属性。

Object.keys()和Object.getOwnPropertyNames()两个函数都是返回对象的属性名，不同的是，Object.keys()只返回可枚举的属性名。

```
var a = ['Hello', 'World'];

Object.keys(a) // ["0", "1"]
Object.getOwnPropertyNames(a) // ["0", "1", "length"]
```

控制对象状态的方法：

- Object.preventExtensions()：防止对象扩展。
- Object.isExtensible()：判断对象是否可扩展。
- Object.seal()：防止其他代码删除对象的属性。
- Object.isSealed()：判断一个对象是否可配置。
- Object.freeze()：冻结一个对象。
- Object.isFrozen()：判断一个对象是否被冻结。

原型链相关方法：

- Object.create()：该方法可以指定原型对象和属性，返回一个新的对象。
- Object.getPrototypeOf()：获取指定对象的原型对象。

构造函数的prototype

在 JavaScript 中，对象继承采用的是基于原型的方式，而prototype属性指向的对象就是**原型对象**。

也就是说，新建对象的__proto__来源于 Object构造函数的prototype 属性。

```
var obj = new Object();
obj.__proto__ === Object.prototype; // true
```

所有的 Object 对象都会继承Object.prototype指向的**原型对象**中的属性，并且当向**原型对象**中添加的属性后，所有的 Object 对象都可以访问这个属性。

```
Object.prototype.newP = 'newV';
var obj = new Object();
obj.newP; // newV
```

Object.prototype中主要的六个方法：

- Object.prototype.valueOf()：返回当前对象对应的值。
- Object.prototype.toString()：返回当前对象对应的字符串形式。
- Object.prototype.toLocaleString()：返回当前对象对应的本地字符串形式。
- Object.prototype.hasOwnProperty()：判断某个属性是否为当前对象自身的属性，还是继承自原对象的属性。
- Object.prototype.isPrototypeOf()：判断当前对象是否为另一个对象的原型。
- Object.prototype.propertyIsEnumerable()：判断某个属性是否可枚举。

这些方法都会被所有的 Object 对象继承。

```
var num = new Number(1);
num instanceof Object; // true
num.valueOf(); // 1
```

相关资料

19.1.1 The Object Constructor

JavaScript 标准内置对象-Object

Object 对象