链滴

JavaScript 中的对象类型

作者：lingyundu
原文链接：https://ld246.com/article/1604759245292
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

什么是对象

简单的说，对象就是一组属性（property）的集合。每个属性包含两部分：

- 属性名（key）—— 可以是字符串或者符号（symbol）类型值。
- 特征（attributes）—— 用来描述属性的状态。

属性的分类

对象属性可以分为两类：

- **数据属性**（data property）—— 属性值是可以直接访问的。
- **访问器属性**（accessor property）—— 属性值通过**访问器函数**（accessor function）来访问的，含getter 和setter 方法。getter方法用来获取属性值，setter 方法用来设置属性值。换句话说，**访问属性**不单独保存属性值，属性值都是通过**访问器函数**来获取和设置的。

一个属性要么是**访问器属性**（具有 getter/setter 方法），要么是**数据属性**（具有 value），但不能者都是。

示例：

```
let user = {
  // name 和 surname 是数据属性
  name: "John",
  surname: "Smith",
  // fullname 是访问器属性，定义 getter 和 setter 方法不需要使用 function 关键字
  get fullName() {
    return `${this.name} ${this.surname}`;
  },

  set fullName(value) {
    [this.name, this.surname] = value.split(" ");
  }
};
user.name; // John
user.fullName; // John Smith
```

属性的特征

数据属性中的特征：

Attribute Name	Value Domain	Description
[[Value]]		属性值。
[[Writable]]	Boolean	若是true 则可以修改属性值，否则属性值不可修改。
[[Enumerable]]	Boolean	若是true，则可以使用 for-in枚举该属性值，否则该属性值不可枚举。
[[Configurable]]	Boolean	若是t

ue，则该属性可被删除，可被修改为**访问器属性**，或者可以修改该属性的部分特征（不能修改[[Value]]，不能将[[Writable]]修改为 false）

访问器属性中的特征：

Attribute Name	Value Domain	Description
[[Get]]	Object or Undefined	属性值是对象，则必须是函数对象。该函数用来获取属性值。
[[Set]]	Object or Undefined	若性值是对象，则必须是函数对象。该函数用来修改属性值。
[[Enumerable]]	Boolean	若是true，则可以使用 for-in枚举该属性值，否则该属性值不可枚举。
[[Configurable]]	Boolean	若是true，则该属性可被删除，可被修改为**数据属性**，或者可以修改该属性的特征。

用来描述属性的特征（attribute）是存放在**属性描述对象**（attributes object）中的，这些特征又被为**属性描述符**。

通过 Object.getOwnPropertyDescriptor()方法可以获取对象属性的**属性描述对象**。

```
var obj = { p: 'a' };

Object.getOwnPropertyDescriptor(obj, 'p');
// { value: "a",
//   writable: true,
//   enumerable: true,
//   configurable: true
//   __proto__: Object
// }
```

对象的创建

对象的创建有三种方式：

- 使用初始化器 —— 对象字面量。
- 使用构造函数 —— 使用 new关键字让构造函数返回一个对象实例。
- 使用 Object.create()方法 —— 以现有对象为原型，返回一个新的对象。

初始化器

一个对象初始化器，由大括号 ({}) 和其包含的零个或多个键值对构成。

示例一： ES6 之前的语法

```
var o = {}; // 这是一个空对象
var o = {a: 'foo', b: 42, c: {}};
```

示例二：ES6 新增的语法

```
// 属性可以用变量
var a = 'foo', b = 42, c = {};
var o = {a, b, c};

// 若属性值是函数，可省略 function 关键字
const o = {
  method() {
    return "Hello!";
  }
};
// 等同于
const o = {
  method: function() {
    return "Hello!";
  }
};

// 属性名可以使用变量和表达式
var prop = 'foo';
var o = {
  [prop]: 'hey',
  ['b' + 'ar']: 'there'
};
```

构造函数

在早期，JavaScript 语言使用构造函数（constructor）作为创建对象的模板。构造函数与普通函数区别是：函数体内部使用了 this关键字表示所要生成的对象实例，生成对象时必须使用 new关键字。

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

Point.prototype.toString = function () {
  return '(' + this.x + ', ' + this.y + ')';
};

var p = new Point(1, 2);
```

这种写法跟传统的面向对象语言（例如 C++ 和 Java）差异很大，让学过 Java 等语言的人感到困惑费解。后来，ES6 提供了更接近传统语言的写法，引入了 Class（类）这个概念，作为对象的模板。过 class关键字，可以定义类。ES6 的 class可以看作只是一个语法糖，它的绝大部分功能，ES5 都可做到。

```
class Point {
  // 构造器
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }

  toString() {
    return '(' + this.x + ', ' + this.y + ')';
```

```
  }
}
```

Object.create()方法

以现有的对象作为原型，生成新的实例对象。新生成的对象会继承原型对象的属性。

```
// 原型对象
var A = {
  print: function () {
    console.log('hello');
  }
};

// 实例对象
var B = Object.create(A);

Object.getPrototypeOf(B) === A // true
B.print() // hello
B.print === A.print // true
```

Object.create()方法生成的对象，会继承它的原型对象的构造函数。

```
function A() {}
var a = new A();
var b = Object.create(a);

b.constructor === A // true
b instanceof A // true
```

属性的操作

属性的查看

使用 Object.keys方法查看一个对象本身的所有属性。

```
var obj = {
  key1: 1,
  key2: 2
};

Object.keys(obj);
// ['key1', 'key2']
```

属性的读取

读取对象的属性，有两种方法，一种是使用点运算符，还有一种是使用方括号运算符。

```
var obj = {
  p: 'Hello World'
};
```

```
obj.p // "Hello World"
obj['p'] // "Hello World"
```

方括号运算符中可以使用变量和表达式。

```
var foo = 'bar';

var obj = {
  foo: 1,
  bar: 2
};

obj.foo  // 1
obj[foo]  // 2
```

属性的新增和修改

与属性的读取一样，可以使用点运算符或者方括号运算符，来新增或者修改属性。若指定的属性存在则修改该属性的值；若不存在，则新增该属性。

```
var obj = {};
// JavaScript 允许属性的“后绑定”，也就是说，你可以在任意时刻新增属性，没必要在定义对象的候，就定义好属性。
obj.foo = 'Hello';
obj['bar'] = 'World';
```

这种方法只能设置属性的值，属性的其他特征都是默认值。

若要对对象属性进行更精细的设置，则可以使用 Object.defineProperty()方法来新增或修改属性。的用法如下：

```
Object.defineProperty(object, propertyName, attributesObject)
```

Object.defineProperty方法接受三个参数：

- object：属性所在的对象
- propertyName：字符串，表示属性名
- attributesObject：属性描述对象

若属性不存在，则新增该属性；若属性存在，则更新该属性。该方法的返回值是修改后的对象。

```
var obj = Object.defineProperty({}, 'p', {
  value: 123,
  writable: false, // 将属性设置为不可修改
  enumerable: true,
  configurable: false
});

obj.p // 123

obj.p = 246;
obj.p // 123
```

属性的删除

delete命令用于删除对象的属性，删除成功后返回 true。

```
var obj = { p: 1 };
Object.keys(obj) // ["p"]

delete obj.p // true
obj.p // undefined
Object.keys(obj) // []
```

相关资料

6.1.7 The Object Type

数据类型-对象

标准库-属性描述对象

JavaScript 标准内置对象-Object

Class 的基本语法