链滴

JavaScript 基于原型的继承

作者：lingyundu

原文链接：https://ld246.com/article/1601899168616

来源网站：链滴

面向对象中的继承特性，有三种实现方案：

- 基于类（class-based）
- 基于原型（prototype-based）
- 基于元类（metaclass-based）

JavaScript 采用的是基于原型的实现方案，这也是 JavaScript 最重要的语言特性之一。

ES6之前，在 JavaScript 中没有类（class）的概念，对象实例是通过构造函数来创建的。构造函数中个 prototype属性，该属性的值就是原型。

原型也是一个对象实例，当使用构造函数创建对象实例时，新创建的对象实例就是从这个原型实例“制”出来的。

每个对象都有一个 __proto__属性，当用构造函数创建对象实例时，构造函数的属性 prototype的值赋给对象实例的 __proto__属性。

空的对象

所有的对象都继承自 Object.prototype，而 Object.prototype的原型是 null。

```
Object.prototype.__proto__;  // null
```

Object.prototype是一个“空”的对象，“空”的对象只有预定义的属性（包括方法）。这些属性会所有的对象继承，使用下面的语句可以查看这些预定义的属性。

```
Object.getOwnPropertyNames(Object.prototype).forEach(function log(element, index, array) {
  console.log('[' + index + '] = ' + element);
});
[0] = constructor
[1] = __defineGetter__
[2] = __defineSetter__
[3] = hasOwnProperty
[4] = __lookupGetter__
[5] = __lookupSetter__
[6] = isPrototypeOf
[7] = propertyIsEnumerable
[8] = toString
[9] = valueOf
[10] = __proto__
[11] = toLocaleString
```

原型链

__proto__ 主要是安放在一个实际的对象中，用它来产生一个原型链，用于寻找方法名或属性，等等。

示例：

```
var a = {
x: 10,
calculate: function (z) {
return this.x + this.y + z;
```

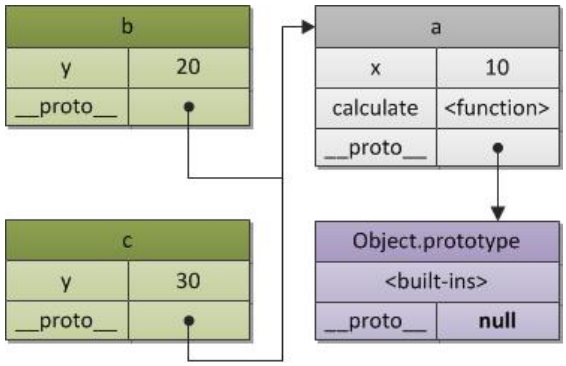
```
}
};

var b = {
y: 20,
__proto__: a
};

var c = {
y: 30,
__proto__: a
};

// call the inherited method
b.calculate(30); // 60
c.calculate(40); // 80
```

以上示例中的原型链如下图所示：



注意：ES5 中，规定原型继承需要使用 Object.create() 函数。如下所示：

```
var b = Object.create(a, {y: {value: 20}});
var c = Object.create(a, {y: {value: 30}});
```

相关资料

36 | 编程范式游记（7）- 基于原型的编程范式

A prototype chain

《JavaScript语言精髓与编程实践》