深度理解原型和原型链

jCodeLife
0.077 2020.08.04 21:35:55 字数 1,331 阅读 43

编辑文章

原型

- 概念

  原型是function对象的一个属性，它定义了构造函数构造出对象的共有祖先。通过该构造函数生成的对象可以继承原型上的属性和方法。原型也是对象。

怎么理解这句话呢
我们知道构造函数是为了生产出对象，如：

```
//定义构造函数
function Person() {}
// 生成对象
var tom = new Person();
```

构造函数上有个属性prototype,这个属性是该构造函数构造的对象的共有祖先，通过该构造函数生产出的对象都可以使用原型上的属性和方法。如：

```
//定义构造函数
function Person() {}
//原型Person.prototype上的属性和方法可以给后代使用
Person.prototype.job = "teacher";
Person.prototype.school = "一中";
Person.prototype.skill= function (s){
    console.log("I have a skill:" + s);
}
// 生成对象
var tom = new Person();
var alice = new Person();
console.log(tom.job);
console.log(alice.job);
```

到这里，我们再进一步了解原型。经过上面我们知道对象可以继承原型上的属性和方法，但是为什么对象能找到对应的原型呢？他们之前应该是存在某些特殊的关系或者联系吧？对象生成的时候到底发生了什么导致这个现象

事实上，要揭开谜底，还是的回到new关键字生成对象的那一刻。
我们知道new生成出对象，其内部原理是：

- 隐式生成this对象
- 执行this.xxx=xxx
- return this

this指代的就是生成出来的对象。在第一步中，隐式生成的this对象并非空对象{}，它里面其实一开始有东西，有个属性叫`__proto__`，这个属性的值默认就是构造出该对象的构造函数的原型。

```
function Person() {}
var tom = new Person();
//new生成tom的时候
```

推荐阅读

前端面试经典Promise理解与总结
阅读 29

配置管理系统的Vue版本来了
阅读 1,283

学Vue，就要学会vue JSX（一）
阅读 418

Vue干货小技巧——学会再也不做加班狗
阅读 418

typeof和instanceOf的区别
阅读 38

tom.`__proto__` === Person.prototype
这下真相大白了。
在查找对象的属性时，它会先找到自己，如果自己身上没有，就会沿着`__proto__`的指向，找到自己的祖先。
对象如果要查找自己的原型，可以通过属性`__proto__`查找。
对象如果要查找自己的构造函数，对象身上还有一个属性叫`constructor`，用于查看该对象的构造函数。即：

```
tom.constructor === Person;//true
```

可以利用原型的特点和概念，提取对象的共有属性
例如：将学生的品质分和总分计算提到原型里，每个对象生成时都有这两个属性：

```
var rootElement = document.getElementById('table_id');
//构造器
function Student(name, gender, score) {
    this.name = name;
    this.gender = gender;
    this.score = score;
    this.mount();
}
//原型中共有属性和方法
Student.prototype.qulity = 100;
Student.prototype.sumScore = function () {
    return this.score + this.qulity;
}
//将数据挂载到页面上
Student.prototype.mount = function () {
    var tr = document.createElement('tr');
    tr.innerHTML = '<td>' + this.name + '<td>' +
        '<td>' + this.gender + '<td>' +
        '<td>' + this.score + '<td>' +
        '<td>' + this.qulity + '<td>' +
        '<td>' + this.sumScore() + '<td>';
    rootElement.appendChild(tr);
}
var alice = new Student('alice', '女', 20);
var tom = new Student('tom', '男', 22);
```

- 原型上的增删改查

这比较容易理解，简单记一下：
增：通过原型；比如：Student.prototype.qulity = 100;
删：通过原型；比如：delete Student.prototype.quity;
修改：通过原型；比如：Student.prototype.qulity = 80;
查询：通过原型或者对应构造函数产生的对象；比如：Student.prototype.qulity 或者 tom.qulity;

注意：后代能查看原型的东西，不能更改。（非绝对，引用值可以）

原型链

我们先一起来深入原型里面，看具体有什么，
先建一个空的构造函数

```
function Person() {}
```

推荐阅读

前端面试经典Promise理解与总结
阅读 29

配置管理系统的Vue版本来了
阅读 1,283

学Vue，就要学会vue JSX（一）
阅读 418

Vue干货小技巧——学会再也不做加班狗
阅读 418

typeof和instanceOf的区别
阅读 38

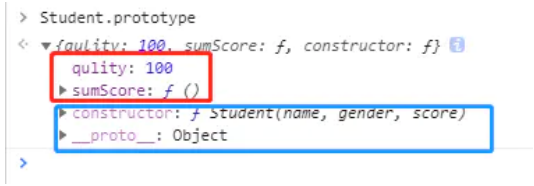
原型内部

我们会发现
原型对象里面包含两个东西我们熟悉的东西： `constructor` 和 `__proto__`

继续再来看个复杂点的

```
function Student(name, gender, score) {
    this.name = name;
    this.gender = gender;
    this.score = score;
}

Student.prototype.qulity = 100;
Student.prototype.sumScore = function () {
    return this.score + this.qulity;
}
```



经过观察
原型对象里面包含三个部分：

- 我们定义在原型上的东西
- constructor
- `__proto__`

这里会有几个疑问等确定
（1）这里面constructor是什么，是构造函数自身吗
（2）原型对象Student.prototype还有原型，指向Object，指向的就是Object对象吗
（3）Object.protoype里面还有原型吗？这是原型链的终端吗？难道所以的对象都会继承Object.prototype？

先来解决第一个问题
原型对象中的constructor是不是该构造函数，我们验证一下：

推荐阅读

前端面试经典Promise理解与总结
阅读 29

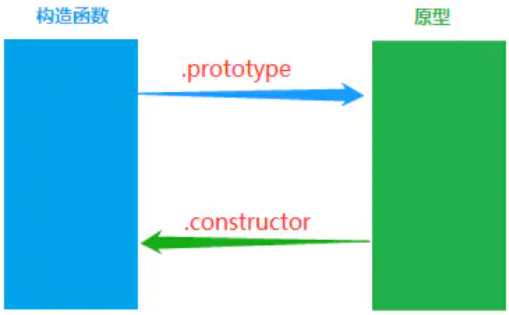
配置管理系统的Vue版本来了
阅读 1,283

学Vue，就要学会vue JSX（一）
阅读 418

Vue干货小技巧——学会再也不做加班狗
阅读 418

typeof和instanceOf的区别
阅读 38

有个属性指向你。



再来看第二个问题
原型对象Student.prototype还有原型，指向Object，指向的就是Object对象吗？
在试试呗

```
> Student.prototype.__proto__ === Object
< false
> Student.prototype.__proto__ === Object.prototype
< true
>
```

我们发现，构造函数的原型里面还有原型，这个原型指向的不是Object，而是Object的原型：Object.prototype

像这种原型里面还有原型，他们之间通过`__proto__`形成一条链，我们称为`原型链`

再来看最后一个问题
Object.protoype里面还有原型吗？这是原型链的终端吗？难道所以的对象都会继承Object.prototype？
我们观察一下

```
> Student.prototype
< ▼{qulity: 100, sumScore: ƒ, constructor: ƒ}
    qulity: 100
  ▶ sumScore: ƒ ()
  ▶ constructor: ƒ Student(name, gender, score)
  ▼ __proto__:
    ▶ constructor: ƒ Object()
    ▶ hasOwnProperty: ƒ hasOwnProperty()
    ▶ isPrototypeOf: ƒ isPrototypeOf()
    ▶ propertyIsEnumerable: ƒ propertyIsEnumerable()
    ▶ toLocaleString: ƒ toLocaleString()
    ▶ toString: ƒ toString()
    ▶ valueOf: ƒ valueOf()
    ▶ __defineGetter__: ƒ __defineGetter__()
    ▶ __defineSetter__: ƒ __defineSetter__()
    ▶ __lookupGetter__: ƒ __lookupGetter__()
    ▶ __lookupSetter__: ƒ __lookupSetter__()
    ▶ get __proto__: ƒ __proto__()
    ▶ set __proto__: ƒ __proto__()
```

发现，Object.protoype里面没有原型了，也就是Object.protoype就是这原型链的终端了。那所有的对象都会继承自Object.prototype吗？答案不是的。可以说绝大数对象最终都会继承自Object.prototype,但是有例外。我们再创建对象的时候，处理字面量和构造函数new出来以外，

推荐阅读

前端面试经典Promise理解与总结
阅读 29

配置管理系统的Vue版本来了
阅读 1,283

学Vue，就要学会vue JSX（一）
阅读 418

Vue干货小技巧——学会再也不做加班狗
阅读 418

typeof和instanceOf的区别
阅读 38

```
Object.create(null)
▼{}
    No properties
```

所以我们了解了，原型链描述了这一系列复杂的继承关系

jCodeLife 书山有路勤为径，学海无涯苦作舟

总资产12 (约1.19元) 共写了10.7W字 获得206个赞 共20个粉丝