js实现继承的几种方式

JavaScript (/lib/tag/JavaScript) ECMAScript (/lib/tag/ECMAScript) 2016-03-30 18:56:23 发布

前言：大多OO语言都支持两种继承方式： 接口继承和实现继承 ， 而ECMAScript中无法实现接口继承，ECMAScript只支持实现继承，而且其实现继承主要是依靠 原型链 来实现。

1.原型链

基本思想：利用原型让一个引用类型继承另外一个引用类型的属性和方法。

构造函数，原型，实例之间的关系：每个构造函数都有一个原型对象，原型对象包含一个指向构造函数的指针，而实例都包含一个指向原型对象的内部指针。

原型链实现继承例子：

```
function SuperType() {
    this.property = true;
}
SuperType.prototype.getSuperValue = function() {
    return this.property;
}
function subType() {
    this.property = false;
}
//继承了SuperType
SubType.prototype = new SuperType();
SubType.prototype.getSubValue = function (){
    return this.property;
}

var instance = new SubType();
console.log(instance.getSuperValue());//true
```

2.借用构造函数

基本思想：在子类型构造函数的内部调用超类构造函数，通过使用call()和apply()方法可以在新创建的对象上执行构造函数。

例子：

```
function SuperType() {
    this.colors = ["red","blue","green"];
}
function SubType() {
    SuperType.call(this);//继承了SuperType
}
var instance1 = new SubType();
instance1.colors.push("black");
console.log(instance1.colors);//"red","blue","green","black"

var instance2 = new SubType();
console.log(instance2.colors);//"red","blue","green"
```

3.组合继承

基本思想：将原型链和借用构造函数的技术组合在一块，从而发挥两者之长的一种继承模式。

例子：

```
function SuperType(name) {
    this.name = name;
    this.colors = ["red","blue","green"];
}
SuperType.prototype.sayName = function() {
    console.log(this.name);
}
function SubType(name, age) {
    SuperType.call(this,name);//继承属性
    this.age = age;
}
//继承方法
SubType.prototype = new SuperType();
Subtype.prototype.constructor = Subtype;
Subtype.prototype.sayAge = function() {
    console.log(this.age);
}

var instance1 = new SubType("EvanChen",18);
instance1.colors.push("black");
consol.log(instance1.colors);//"red","blue","green","black"
instance1.sayName();//"EvanChen"
instance1.sayAge();//18

var instance2 = new SubType("EvanChen666",20);
console.log(instance2.colors);//"red","blue","green"
instance2.sayName();//"EvanChen666"
instance2.sayAge();//20
```

4.原型式继承

基本想法：借助原型可以基于已有的对象创建新对象，同时还不必须因此创建自定义的类型。

原型式继承的思想可用以下函数来说明：

```
function object(o) {
    function F(){}
    F.prototype = o;
    return new F();
}
```

例子：

```
var person = {
    name:"EvanChen",
    friends:["Shelby","Court","Van"];
};

var anotherPerson = object(person);
anotherPerson.name = "Greg";
anotherPerson.friends.push("Rob");

var yetAnotherPerson = object(person);
yetAnotherPerson.name = "Linda";
yetAnotherPerson.friends.push("Barbie");

console.log(person.friends);//"Shelby","Court","Van","Rob","Barbie"
```

ECMAScript5通过新增Object.create()方法规范化了原型式继承，这个方法接收两个参数：一个用作新对象原型的对象和一个作为新对象定义额外属性的对象。

```
var person = {
    name:"EvanChen",
    friends:["Shelby","Court","Van"];
};

var anotherPerson = Object.create(person);
anotherPerson.name = "Greg";
anotherPerson.friends.push("Rob");

var yetAnotherPerson = Object.create(person);
yetAnotherPerson.name = "Linda";
yetAnotherPerson.friends.push("Barbie");

console.log(person.friends);//"Shelby","Court","Van","Rob","Barbie"
```

5.寄生式继承

基本思想：创建一个仅用于封装继承过程的函数，该函数在内部以某种方式来增强对象，最后再像真正是它做了所有工作一样返回对象。

例子：

```
function createAnother(original) {
    var clone = object(original);
    clone.sayHi = function () {
        alert("hi");
    };
    return clone;
}

var person = {
    name:"EvanChen",
    friends:["Shelby","Court","Van"];
};
var anotherPerson = createAnother(person);
anotherPerson.sayHi();///"hi"
```

6.寄生组合式继承

基本思想：通过借用函数来继承属性，通过原型链的混成形式来继承方法

其基本模型如下所示：

```
function inheritProperty(subType, superType) {
    var prototype = object(superType.prototype);//创建对象
    prototype.constructor = subType;//增强对象
    subType.prototype = prototype;//指定对象
}
```

例子：

```
function SuperType(name){
    this.name = name;
    this.colors = ["red","blue","green"];
}
SuperType.prototype.sayName = function (){
    alert(this.name);
};

function SubType(name,age){
    SuperType.call(this,name);
    this.age = age;
}
inheritProperty(SubType,SuperType);
SubType.prototype.sayAge = function() {
    alert(this.age);
}
```

来自： https://segmentfault.com/a/1190000004730936 (https://segmentfault.com/a/1190000004730936)

同类热门经验

1. Node.js 初体验 (/lib/view/open1326870121968.html)
2. JavaScript开发规范要求 (/lib/view/open1352263831610.html)
3. 使用拖拉操作来自定义网页界面布局并保存结果 (/lib/view/open1325064347889.html)
4. Nodejs入门学习，nodejs web开发入门，npm、express、socket配置安装、nodejs聊天室开发 (/lib/view/open1329050007640.html)
5. 利用HTML5同时上传多个文件 - resumable.js (/lib/view/open1327591300671.html)
6. nide：一个不错的Node.js开发工具IDE (/lib/view/open1325834128750.html)

阅读目录

- JavaScript权威指南(第6版).pdf (http://www.open-open.com/doc/view/467b98eb4bcd4b0fb15b2f82f3457dfa)
- JavaScript权威指南(第六版)中文版.pdf (http://www.open-open.com/doc/view/f4ff0e7cd16b478bae6f5dffd38ba757)
- Functional JavaScript 中文版.pdf (http://www.open-open.com/doc/view/28dff3d278a343d386a1389bef376b0a)
- 使用Struts2提供的JSON插件.doc (http://www.open-open.com/doc/view/5acc81f5d6744fa28e2175aed96dc70c)
- Spring获取容器的几种方式.doc (http://www.open-open.com/doc/view/1f9614bd47b443b2abf8f060ce3334a4)
- Spring定时任务的几种实现.doc (http://www.open-open.com/doc/view/4007fb65ebd540e3b96ae47b66dc344b)
- Javascript 闭包.pdf (http://www.open-open.com/doc/view/fd2e6c26ba63455ebfc96ce6410ef59b)
- JavaScript高级程序设计（第3版）.pdf (http://www.open-open.com/doc/view/a42e96698e66474aba7aaf854f6a3680)
- JavaScript 高级程序设计（第3版）.pdf (http://www.open-open.com/doc/view/ff1758993ecf479693f01a37a620a468)
- ECMAScript 6入门.pdf (http://www.open-open.com/doc/view/5d48755f235843b88d5e87001492fe6a)
- 浅析 Javascript 原型继.pdf (http://www.open-open.com/doc/view/c137e448f6c3426bb0c8c7acff447bad)
- JavaScript 权威指南(第六版)中文版.pdf (http://www.open-open.com/doc/view/a81462eac97c4b0b9b590deaa29510b5)
- [JavaScript权威指南(第6版)].JavaScript:The DefinitiveGuide.pdf (http://www.open-open.com/doc/view/b25771d26d2744d5ab9aa8a723ff5866)
- Json2.js手册.pdf (http://www.open-open.com/doc/view/a6b36a3816394f449248603b38aa6567)
- Node.js高级编程Professional Node.js.pdf (http://www.open-open.com/doc/view/312ef192e9d84c9bbd86e6964e454203)
- javascript module设计模式.ppt (http://www.open-open.com/doc/view/1c88c56f4f8f45c286fc8d4c15ffa905)

(/lib/view/open1442147356707.html)

- ES6简介 (/lib/view/open1434350249505.html)
- 给 JavaScript 初心者的 ES2015 实战 (/lib/view/open1447222864319.html)
- JavaScript小技巧介绍 (/lib/view/open1453728914105.html)
- ES6：下一版本的JavaScript的新特性 (/lib/view/open1453297166620.html)
- 结合个人经历总结的前端入门方法 (/lib/view/open1449542023941.html)
- jQuery 中的编程范式 (/lib/view/open1324949159702.html)
- 深入浅出 React Native：使用 JavaScript 构建原生应用 (/lib/view/open1428648433796.html)
- 前端面试问题(二)-史上最全 前端开发面试问题及答案整理 (/lib/view/open1460612941431.html)
- 史上最全 前端开发面试问题及答案整理 (/lib/view/open1437483697115.html)
- 2015前端组件化框架之路 (/lib/view/open1427350415652.html)

open.com/solution/view/1447672613728)

- 16则极具内涵的程序员笑话 (http://www.open-open.com/solution/view/1391841126957)
- 再谈JavaScript的数据类型问题 (http://www.open-open.com/solution/view/1318472797249)
- 一个“三端”开发者眼中的React Native (http://www.open-open.com/solution/view/1438406299050)
- 程序员技术练级攻略 (http://www.open-open.com/solution/view/1319276210452)