js继承的多种形式

jCodeLife
0.117 2020.08.07 15:47:41 字数 688 阅读 37

编辑文章

看了一些关于js继承的文章，好多都是一上来就是怎么实现继承，而从来不说什么是继承，我们还是先来聊聊什么是继承？
有强迫症的我翻了一下字典：
继：表示连续、接着的意思；承：表示承受、接受的意思
继承一词最早是在社会出现私有制,分裂为阶级后而产生的。奴隶社会最早出现是身份继承,族长死后,由其嫡长子继承其统治身份和政治地位,随之掌握财产。身份继承是财产继承的前提。封建社会实行"宗祧继承",继承人以男为限,以嫡长子为原则,有子立长,无子立嗣,成为历代的法例。到了近代社会，继承只要是指按照法律或遵照遗嘱接受死者的财产、职务、头衔、地位等。

而在计算机语言中，最早是由后端的一些程序员提出来的概念，特别是面向对象的开发语言中比如Java，封装、继承和多态是它的三个基本特征。继承可以使得子类具有父类的属性和方法或者重新定义、追加属性和方法等。

最后回到我们的前端

js的继承与后端一样，也是为了继承别人的属性和方法，实现代码复用、缩短开发周期、降低成本。下面我们一起来看看js继承的几种形式：

1. 传统形式，通过原型链实现继承

```
Grand.prototype.lastName = "wang";
function Grand() { }
var grand = new Grand();

Father.prototype = grand;
function Father() {
    this.name = "hehe";
}
var father = new Father();

Son.prototype = father;
function Son() { }
var son = new Son();
son.lastName();
```

缺点：子类对象继承了太多没有用属性；如son只想继承原型里面的lastName属性，但name属性也继承下来了。

2. 借用构造函数的方式实现继承;通过call和apply

```
function Person(name, age, sex) {
    this.name = name;
    this.age = age;
    this.sex = sex;
}
function Student(name, age, sex, grade, tel, address) {
    Person.call(this, name, age, sex);
    this.grade = grade;
    this.tel = tel;
    this.address = address;
}
var alice = new Student("alice", 20, 'famale', 88, "134****4559", "海天二路33号")
```

jCodeLife
总资产13 (约1.20元)

深入call,apply,bind到手动封装
阅读 53

梳理CSS3 Transform相关知识
阅读 1

CSS3 过渡效果transition的基本使用
阅读 8

推荐阅读

Vue干货小技巧——学会再也不做加班狗
阅读 1,122

面试了几个前端，给爷整哭了！
阅读 18,617

vue优化页面
阅读 672

typeof和instanceOf的区别
阅读 107

学会这6个强大的CSS选择器，将真正帮你写出干净的CSS代码！
阅读 219

- 不能继承借用构造函数的原型
- 每次构造函数都得多走一个函数

3. 共享原型实现继承

```
Father.prototype.lastName = "Wang";
function Father() {

}
Son.prototype = Father.prototype;//共享一个原型
function Son() {

}
var father = new Father();
var son = new Son();
console.log(father.lastName);//Wang
console.log(son.lastName);//Wang
```

缺点：共享原型后，不能随便更改自己的原型

4. 圣杯模式

最后出现的这种形式叫圣杯模式，这种模式能随便改变自己的原型的东西不影响其他

```
// 实现Son的构造函数构造出来的所以对象继承Father原型上的属性和方法
function inherit(Target, Origin) {
    //Target构造函数构造出来的对象继承自Origin的原型
    function F(){};
    F.prototype = Origin.prototype;
    Target.prototype = new F();
    Target.prototype.constructor = Target;//扭转对象的构造函数正确的指向
    Target.prototype._super = Origin.prototype;//用于查看对象真实继承自谁
}
```

```
//test code
Father.prototype.lastName = "Wang";
function Father() {
}
function Son() {
}
inherit(Son, Father);
var son = new Son();
var father = new Father();
console.log(son.lastName);//Wang
console.log(father.lastName);//Wang
Son.prototype.sex = "male";
console.log(son.sex);//male
console.log(father.sex);//undefined
```

最后再来修改成最终的形式

```
//实现让Target构造函数构造出来的对象继承Origin的原型的属性和方法
var inherit = (function () {
    function F() { };
    return function (Target, Origin) {
        F.prototype = Origin.prototype;
        Target.prototype = new F();
        Target.prototype.constructor = Target;
        Target.prototype._super = Origin.prototype;
    }
}());
```

继承模式-腾讯课堂渡一教育
百度百科-继承

3人点赞 > 笔记（1）

"小礼物走一走，来简书关注我"

赞赏支持

还没有人赞赏，支持一下

jCodeLife 书山有路勤为径，学海无涯苦作舟
总资产13 (约1.20元) 共写了11.5W字 获得214个赞 共21个粉丝

写下你的评论...

全部评论 0 只看作者 关闭评论 按时间倒序 按时间正序

被以下专题收入，发现更多相似内容 投稿管理

+收入我的专题  领跑计划笔记