**继承的概念**

继承是面向对象的一个非常重要的特征. 继承指的是: 子类继承父类的属性和方法.

我们可以通过继承的方式, 在父类的属性和方法基础上, 在无需修改父类的情况下, 让子类也有拥有这些属性和方法, 并可以扩展其他属性和方法.

**继承的特点**

1. 子类拥有父类的属性和方法；

2. 子类可以有自己新的属性和方法；

3. 子类可以重写(override)父类的方法 (js里面没有重载 overload,同名不同参)

**继承的优点**

1. 代码复用：子类可以继承父类的属性和方法

2. 更灵活：子类可以追加或修改属性和方法

**原型链继承**

能继承父类原型上的方法，也会带过来父类原型的属性，但是父类的属性始终用不上

注意：先继承父类的方法，再给自己添加方法，避免被原型覆盖

Student.prototype = new Person//不用传参 传了也用不上

**对象冒充继承**

function Student(sNo,name,age,sex){

this.sNo = sNo

//对象的冒充 改变this的指向

Person.bind(this,name,age,sex)()//bind不会立即执行 所以要加小括号调用

Person.call(this,name,age,sex)

Person.apply(this,[name,age,sex])

}

**组合继承**

对象冒充+原型链继承

对象冒充：实例化类的时候，都有自己的初始属性（动态传值）

原型链继承：解决父类原型的方法继承

缺点：原型上会有多余的属性，并且是undefined

function Person(name, age, sex) {

this.name = name

this.age = age

this.sex = sex

this.eat = function (food) {

console.log(`我${this.name}在吃${food}`)

}

this.run = function (road) {

console.log(`我${this.name}在${road}跑步`)

}

}

function Student(sNo, name, age, sex) {

this.sNo = sNo

Person.call(this, name, age, sex)

}

Student.prototype = new Person()

Student.prototype.study = function () {

console.log(`我爱学习`)

}

var s1 = new Student(9527, "刘德华", 42, "男")

s1.eat("牛排")///我刘德华在吃牛排

s1.study()//我爱学习

**寄生组合继承**

寄生继承+对象冒充继承

注意：先继承再添加/重写

**利用Object.create(proto)方法**

**返回一个新对象，这个对象的原型对象的传入的参数proto所指定的**

这个proto参数只能接收空对象null或者非原始包装对象，其他的会跑出typeerror

// base表示基类（父类），child表示子类

function inherit(base, child) {

// 父类的方法都放在父类的原型中 子类想继承的话创建一个新的对象

//该对象的原型为父类的原型

let baseprototype = Object.create(base.prototype)

//再让该对象的构造器constructor 和 子类的prototype形成相互引用的关系

baseprototype.constructor = child

child.prototype = baseprototype

}

function Person(name, age) {

this.name = name

this.age = age

}

Person.prototype.eat = function () {

console.log(`我${this.name}会吃东西`)

}

function Student(sNo, name, age) {

Person.call(this, name, age) //属性先继承

this.sNo = sNo //再添加

}

//再在原型中添加方法

inherit(Person, Student) //继承父类的原型

Student.prototype.study = function () {

console.log(`我${this.name}会学习`)

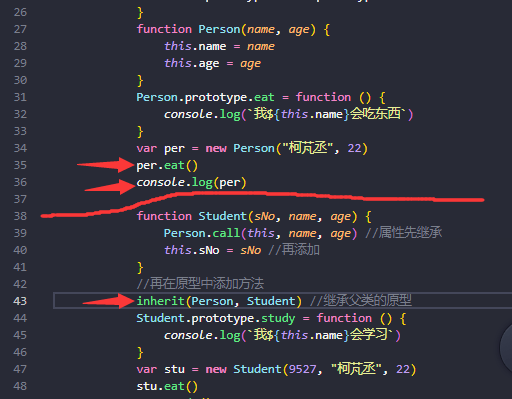
}

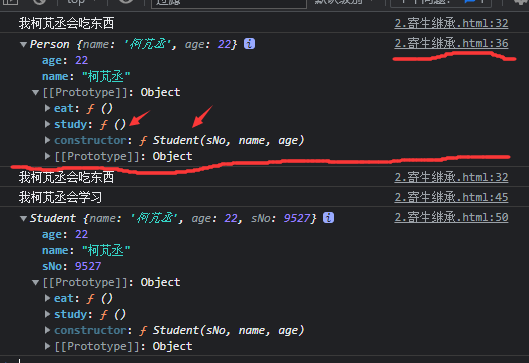
var stu = new Student(9527, "柯芃丞", 22)

stu.eat()//我柯芃丞会吃东西

stu.study()//我柯芃丞会学习

**问题**





**类的继承**

ES6 的class可以看作只是一个语法糖,而类本身可以看作一个构造函数

用typeof 类名 会返回"function"

class中的方法是绑定在构造函数的原型上的

class A {

hello() {}

}

// 等于

function A() {}

A.prototype.hello = function() {};

同样构造函数原型的constructor指向它本身

class A {

hello() {}

}

A.prototype.constructor === A // true

不难看出constructor才是实例A的构造函数，写在constructor外部的可以理解为绑定在构造函数的原型上

class A {

constructor() {

this.a = 'a';

this.b = 'b';

}

hello() {}

}

console.log(new A());

**链式编程的核心思想就是return this**