**组合原型链继承**

**一、原型链继承**

**套路：**

1.定义父类型构造函数

2.给父类型的原型添加方法

3.定义子类型的构造函数

4.创建父类型的对象赋值给予类型的原型

5.将子类型原型的构造属性设置为子类型

6.给予类型添加方法

7.创建子类型对象：可以调用父类型的方法

**关键：**子类型的原型为父类型的一个实例对象 、 **实际：**使用子函数能调用父函数的方法

**实例：让子函数的原型对象指向父元素**

function Supper() {

this.supProp = 'Supper property'

}

Supper.prototype.showSupperProp = function () {

console.log(this.supProp)

}

//子类型

function Sub(){

this.supProp = 'Sub property'

}

Sub.prototype = new Supper() // 子类型的原型为父类型的一个实例对象

Sub.prototype.constructor = Sub// 让子类型的原型的constructor指向子类型

Sub.prototype.showSubProp = function(){

console.log(this.supProp)

}

var sub = new Sub()

sub.showSupperProp() //Supper property

sub.showSubProp() //Sub property

**二、借用构造函数继承**

**套路：**

1.定义父类型构造函数

2.定义子类型构造函数

3.在子类型构造函数中调用父类型构造

**关键：在子类型构造函数中通用super()调用父类型构造函数**

**实例：让子函数使用父函数的属性**

function Person(name,age){

this.name = name

this.age = age

}

function Student(name,age,price){

Person.call(this,name,age)

this.price = price

}

var b = new Student("DZL",50,"SB")

console.log(b)

**三、原型链+借用构造函数的组合继承**

1. 利用原型链实现对父类型对象的方法继承

2. 利用call()借用父类型构建函数初始化相同属性

**实例：让子函数的原型对象指向父元素,并且子函数使用父函数的属性**

function Person(name,age){

this.name = name

this.age = age

}

Person.prototype.setName = function(name){

this.name = name

}

function Student(name,age,price){

Person.call(this,name,age) //为了得到属性

this.price = price

}

Student.prototype =new Person //为了能看到父类型的方法

Student.prototype.constructor = Student //修正constructor属性

Student.prototype.setPrice = function(price){

this.price = price

}

var b = new Student("DZL",50,"SB")

b.setName("QAQ")

b.setPrice(1800000)

console.log(b)