

## 浙江大学城市学院实验报告

课程名称 跨平台脚本开发技术

实验项目名称 实验六 对象进阶

学生姓名 吴成洋 专业班级 软工 1404 学号 31401417

实验成绩          指导老师（签名）                      日期                 

注意：

- 务请保存好各自的源代码，以备后用。
- 请把作业保存为 pdf 上传到 BB 平台，请务必在截止日期前提交。

### 实验目的：

掌握 JS 中的对象的原型、原型链、继承的原理与应用。

### 实验内容：

#### 1. 阅读课本相关内容 回答理解以下问题

- 6.4 说明构造函数用法 构造函数的 prototype 的作用
- 6.5 说明对象属性查找机制
- 6.7 为什么需要无原型对象，好处在哪里？
- 6.9 理解 TextCell() 构造函数的写法，如何定义属性，定义方法
- 6.10 如何定义对象的 访问器 getter setter
- 6.11 如何用 原型实现继承，说明 p.85 代码中，下面两句话的作用

```
TextCell.call(this,text);
```

```
RTextCell.prototype =Object.create(TextCell.prototype);
```

2. 继承是一种代码复用机制，教材 P83 UnderlinedCell 采用了基于组合的代码复用。请说明组合与继承的不同（自行查找资料理解）。
3. 教材 P88 习题（6.14.1，6.14.2，6.14.3）

### 实验步骤：

1、

6.4

构造函数的用法：

1. 函数体内部使用了 `this` 关键字，代表了所要生成的对象实例。

`this` 关键字上所有的属性将成为被构造的对象 的属性。

2. 生成对象的时候，必需用 `new` 命令，调用 `Vehicle` 函数

构造函数的 `prototype` 的作用：

只要定义在它上面的属性和方法，能被所有实例对象共享。也就是说，构造函数生成通过 `new` 生成实例对象时，自动根据 `prototype` 属性创建对象

## 6.5 对象属性查找机制：

1. 对象访问属性的时候，先在自己查找属性，如果没有会继续在原型上查找。依次递推。

2. 对象自身的属性被修改的时候，不会影响到原型上的属性

## 6.7 为什么需要无原型对象，好处在哪里？

如果想要生成一个不继承任何属性（比如 `toString` 和 `valueOf` 方法）的对象，可以将 `Object.create` 的参数设为 `null`。好处是对象所有的属性都是对象自己的属性。

## 6.10 如何定义对象的访问器 `getter setter`

1. 在对象中，`get` 或 `set` 方法用于指定属性的读取函数和修改函数

2. 读取或修改属性时会自动调用这些函数。
3. 可以用 `defineProperty` 向函数原型中添加访问器属性。

## 6.11

```
TextCell.call(this, text);
```

重用了 `TextCell` 的构造函数、`minHeight` 和 `minWidth` 属性

```
RTextCell.prototype = Object.create(TextCell.prototype);
```

`RTextCell` 基本上就是 `TextCell`，只不过 `draw` 方法中包含了不同的函数，即原型式的继承

## 2.

组合与继承的不同：

继承的优点是子类可以重写父类的方法来方便地实现对父类的扩展。

继承的缺点有以下几点：

- ①：父类的内部细节对子类是可见的。
- ②：子类从父类继承的方法在编译时就确定下来了，所以无法在运行期间改变从父类继承的方法的行为。
- ③：如果对父类的方法做了修改的话（比如增加了一个参数），则子类的方法必须做出相应的修改。所以说子类与父类是一种高耦合，违背了面向对象思想。

组合的优点：

①：当前对象只能通过所包含的那个对象去调用其方法，所以所包含的对象的内部细节对当前对象是不可见的。

②：当前对象与包含的对象是一个低耦合关系，如果修改包含对象的类中代码不需要修改当前对象类的代码。

③：当前对象可以在运行时动态的绑定所包含的对象。可以通过 `set` 方法给所包含对象赋值。

组合的缺点：

①：容易产生过多的对象。②：为了能组合多个对象，必须仔细对接口进行定义。

### 3.

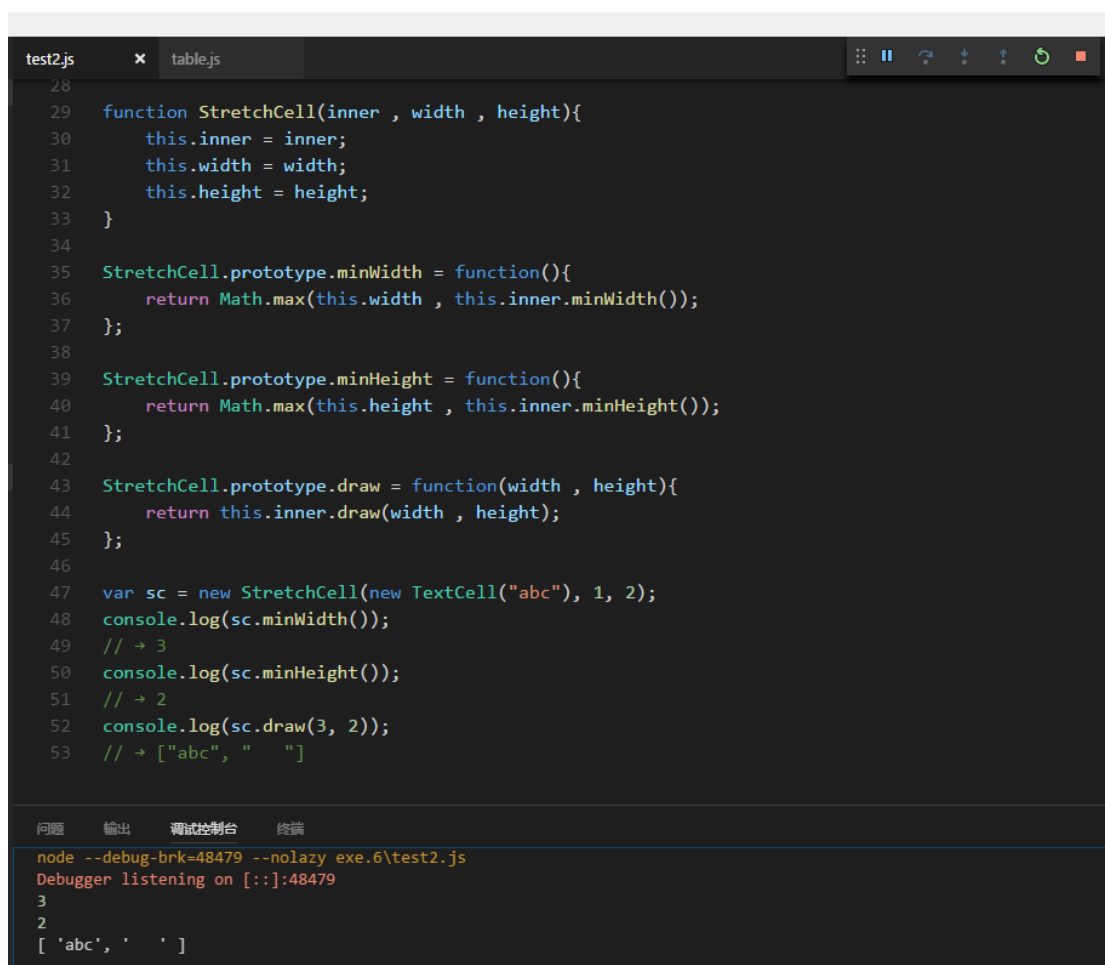
#### 3.1、

```
test.js
1 //构造函数
2 function Vector( x , y ){
3     this.x = x;
4     this.y = y;
5 }
6
7 Vector.prototype.plus = function(xiangliang){
8     return new Vector(this.x + xiangliang.x , this.y + xiangliang.y);
9 }
10
11 Vector.prototype.minus = function(xiangliang){
12     return new Vector(this.x - xiangliang.x , this.y - xiangliang.y);
13 }
14
15 Object.defineProperty(Vector.prototype, "length", {
16     get: function(){
17         return Math.sqrt(this.x * this.x + this.y * this.y);
18     }
19 });
20
21
22 console.log(new Vector(1, 2).plus(new Vector(2, 3)));
23 // → Vector{x: 3, y: 5}
24 console.log(new Vector(1, 2).minus(new Vector(2, 3)));
25 // → Vector{x: -1, y: -1}
26 console.log(new Vector(3, 4).length);
27 // → 5
```

问题 输出 调试控制台 终端

```
node --debug-brk=37008 --nolazy exe.6\test.js
Debugger listening on [::]:37008
Vector { x: 3, y: 5 }
Vector { x: -1, y: -1 }
5
```

### 3.2、



```
test2.js x table.js
28
29 function StretchCell(inner , width , height){
30     this.inner = inner;
31     this.width = width;
32     this.height = height;
33 }
34
35 StretchCell.prototype.minWidth = function(){
36     return Math.max(this.width , this.inner.minWidth());
37 };
38
39 StretchCell.prototype.minHeight = function(){
40     return Math.max(this.height , this.inner.minHeight());
41 };
42
43 StretchCell.prototype.draw = function(width , height){
44     return this.inner.draw(width , height);
45 };
46
47 var sc = new StretchCell(new TextCell("abc"), 1, 2);
48 console.log(sc.minWidth());
49 // → 3
50 console.log(sc.minHeight());
51 // → 2
52 console.log(sc.draw(3, 2));
53 // → ["abc", " " ]

问题 输出 调试控制台 终端
node --debug-brk=48479 --nolazy exe.6\test2.js
Debugger listening on [::]:48479
3
2
[ 'abc', ' ' ]
```

代码如下：

```
function TextCell(text){
    this.text = text.split("\n");
}
function repeat(string,times){
    var result = "";
    for(var i=0;i<times;i++)
        result += string;
    return result;
}
TextCell.prototype.minWidth=function(){
    return this.text.reduce(function(width,line){
        return Math.max(width,line.length);
    },0);
};
TextCell.prototype.minHeight=function(){
    return this.text.length;
};
```

```
TextCell.prototype.draw=function(width,height){
    var result=[];
    for(var i = 0;i<height;i++){
        var line = this.text[i]||"";
        result.push(line+repeat(" ",width-line.length));
    }
    return result;
};

function StretchCell(inner , width , height){
    this.inner = inner;
    this.width = width;
    this.height = height;
}

StretchCell.prototype.minWidth = function(){
    return Math.max(this.width , this.inner.minWidth());
};

StretchCell.prototype.minHeight = function(){
    return Math.max(this.height , this.inner.minHeight());
};

StretchCell.prototype.draw = function(width , height){
    return this.inner.draw(width , height);
};

var sc = new StretchCell(new TextCell("abc"), 1, 2);
console.log(sc.minWidth());
// → 3
console.log(sc.minHeight());
// → 2
console.log(sc.draw(3, 2));
// → ["abc", "   "]
```

### 3.3、

```
test3.js • test2.js
1 function logFive(sequence){
2     for ( var i = 0 ; i < 5 ; i++){
3         if (!sequence.next())
4             break;
5         console.log(sequence.current());
6     }
7 }
8 //对象类型ArraySeq
9 function ArraySeq(array){
10     this.pos = -1 ;
11     this.array = array;
12 }
13 ArraySeq.prototype.next = function(){
14     if(this.pos >= this.array.length-1)
15         return false;
16     this.pos++;
17     return true;
18 };
19 ArraySeq.prototype.current = function(){
20     return this.array[this.pos];
21 };
22 //对象类型RangeSeq
23 function RangeSeq(from , to){
24     this.pos = from -1;
25     this.to = to ;
26 }
27 RangeSeq.prototype.next = function(){
问题 输出 调试控制台 终端
node --debug-brk=22469 --nolazy exe.6\test3.js
Debugger listening on [::]:22469
1
2
100
101
102
103
104
```

```
function logFive(sequence){
    for ( var i = 0 ; i < 5 ; i++){
        if (!sequence.next())
            break;
        console.log(sequence.current());
    }
}
//对象类型 ArraySeq
function ArraySeq(array){
    this.pos = -1 ;
    this.array = array;
}
ArraySeq.prototype.next = function(){
    if(this.pos >= this.array.length-1)
        return false;
    this.pos++;
    return true;
```



```
};
ArraySeq.prototype.current = function(){
    return this.array[this.pos];
};
//对象类型 RangeSeq
function RangeSeq(from , to){
    this.pos = from -1;
    this.to = to ;
}
RangeSeq.prototype.next = function(){
    if(this.pos >= this.to)
        return false;
    this.pos++;
    return true;
};
RangeSeq.prototype.current = function(){
    return this.pos;
};

logFive(new ArraySeq([1, 2]));
// → 1
// → 2
logFive(new RangeSeq(100, 1000));
// → 100
// → 101
// → 102
// → 103
// → 104
```