

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 实验七 对象进阶 2学生姓名 吴成洋 专业班级 软件工程 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. JavaScript 面向对象编程练习题：

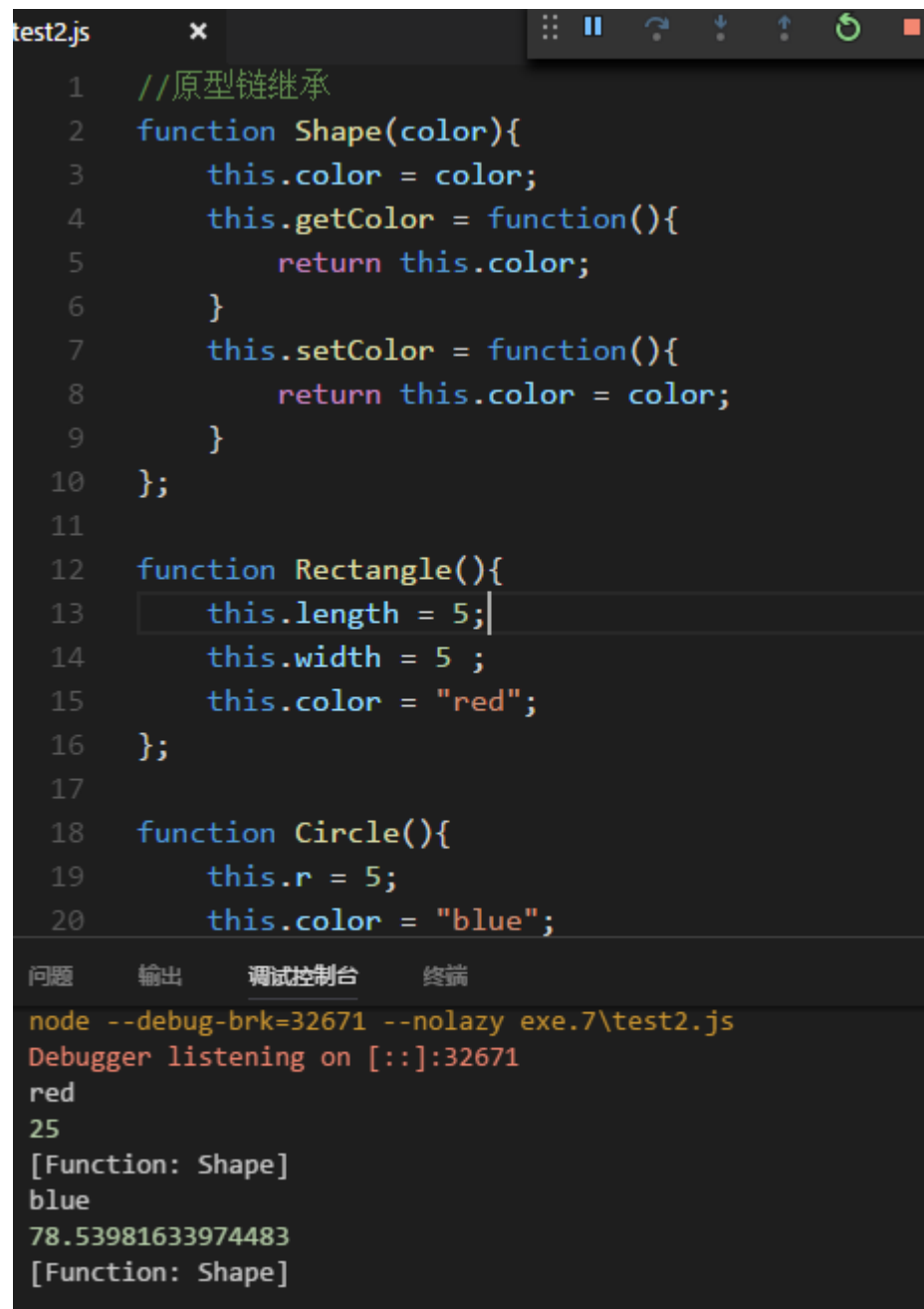
- 1) 定义父类：Shape(形状)类，Shape 只有一个属性 color，并有相应的 getColor 和 setColor 方法。
- 2) Shape 类有两个子类：Rectangle(矩形)类和 Circle(圆形)类，子类继承了父类的 color 属性和 getColor、setColor 方法。
- 3) 为两个子类增加相应的属性和 getArea 方法，可调用 getArea 方法获得矩形和圆形的面积。

用你学过的所有继承方法去实现上面功能。

实验步骤：

1、

//原型链



```
test2.js x [Icons]
1  //原型链继承
2  function Shape(color){
3      this.color = color;
4      this.getColor = function(){
5          return this.color;
6      }
7      this.setColor = function(){
8          return this.color = color;
9      }
10 };
11
12 function Rectangle(){
13     this.length = 5;
14     this.width = 5 ;
15     this.color = "red";
16 };
17
18 function Circle(){
19     this.r = 5;
20     this.color = "blue";
21 };
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

问题 输出 调试控制台 终端

```
node --debug-brk=32671 --nolazy exe.7\test2.js
Debugger listening on [::]:32671
red
25
[Function: Shape]
blue
78.53981633974483
[Function: Shape]
```

代码如下：

```
//原型链继承
function Shape(color){
    this.color = color;
    this.getColor = function(){
        return this.color;
    }
    this.setColor = function(){
        return this.color = color;
    }
};
```

```
function Rectangle(){
    this.length = 5;
    this.width = 5 ;
    this.color = "red";
};

function Circle(){
    this.r = 5;
    this.color = "blue";
};

Rectangle.prototype = new Shape();
Circle.prototype = new Shape();

Rectangle.prototype.getArea = function(){
    return this.length * this.width;
}

Circle.prototype.getArea = function(){
    return this.r * this.r * Math.PI;
}

var Rectangle1 = new Rectangle();
var Circle1 = new Circle();

console.log(Rectangle1.color);
console.log(Rectangle1.getArea());
console.log(Rectangle1.constructor);

console.log(Circle1.color);
console.log(Circle1.getArea());
console.log(Circle1.constructor);
```

//借用构造函数继承

```
test2.js  ×
49  function Shape(color){
50      this.color = color;
51      this.getColor = function(){
52          return this.color;
53      }
54      this.setColor = function(){
55          return this.color = color;
56      }
57  };
58
59  function Rectangle(){
60      this.length = 5;
61      this.width = 5 ;
62      Shape.call(this , "red");//对象冒充继承，可以传参
63  };
64
65  function Circle(){
66      this.r = 5;
67      Shape.call(this , "blue");//对象冒充继承，可以传参
68  };
69
70  Rectangle.prototype.getArea = function(){
71      return this.length * this.width;
72  }
73
74  Circle.prototype.getArea = function(){
75      return this.r * this.r * Math.PI;
76  }
77
78  var Rectangle1 = new Rectangle();
79  var Circle1 = new Circle();
80
```

问题 输出 调试控制台 终端

```
Debugger listening on [::]:49348
red
25
[Function: Rectangle]
blue
78.53981633974483
[Function: Circle]
false
false
```

代码如下：

```
function Shape(color){
    this.color = color;
    this.getColor = function(){
        return this.color;
    }
    this.setColor = function(){
        return this.color = color;
    }
};

function Rectangle(){
    this.length = 5;
    this.width = 5 ;
    Shape.call(this , "red");//对象冒充继承，可以传参
};

function Circle(){
    this.r = 5;
    Shape.call(this , "blue");//对象冒充继承，可以传参
};

Rectangle.prototype.getArea = function(){
    return this.length * this.width;
}

Circle.prototype.getArea = function(){
    return this.r * this.r * Math.PI;
}

var Rectangle1 = new Rectangle();
var Circle1 = new Circle();

console.log(Rectangle1.color);
console.log(Rectangle1.getArea());
console.log(Rectangle1.constructor);

console.log(Circle1.color);
console.log(Circle1.getArea());
console.log(Circle1.constructor);
```

```
console.log(Rectangle1.getColor == Circle1.getColor);  
console.log(Rectangle1.setColor == Circle1.setColor);
```

//组合继承

```
test2.js  ×
108  function Rectangle(){
109      this.length = 5;
110      this.width = 5 ;
111      Shape.call(this , "red");//对象冒充继承，可以传参
112  };
113
114  Circle.prototype = new Shape(); //一次
115  function Circle(){
116      this.r = 5;
117      Shape.call(this , "blue");//对象冒充继承，可以传参
118  };
119
120  Rectangle.prototype.getArea = function(){
121      return this.length * this.width;
122  }
123
124  Circle.prototype.getArea = function(){
125      return this.r * this.r * Math.PI;
126  }
127
128  var Rectangle1 = new Rectangle();
129  var Circle1 = new Circle();
130
131  console.log(Rectangle1.color);
132  console.log(Rectangle1.getArea());
133  console.log(Rectangle1.constructor);
134
135  console.log(Circle1.color);
136  console.log(Circle1.getArea());
137  console.log(Circle1.constructor);
138  console.log(Rectangle1.getColor == Circle1.getColor);
139  console.log(Rectangle1.setColor == Circle1.setColor);
140
```

问题 输出 调试控制台 终端

```
Debugger listening on [::]:41854
red
25
[Function: Shape]
blue
78.53981633974483
[Function: Shape]
true
true
```

代码如下：

```
function Shape(color){
    this.color = color;
```

```
};

Shape.prototype.getColor = function() {
    return this.color;
};

Shape.prototype.setColor = function(color) {
    return this.color = color;
};

Rectangle.prototype = new Shape(); //一次
function Rectangle(){
    this.length = 5;
    this.width = 5 ;
    Shape.call(this , "red");//对象冒充继承，可以传参
};

Circle.prototype = new Shape(); //一次
function Circle(){
    this.r = 5;
    Shape.call(this , "blue");//对象冒充继承，可以传参
};

Rectangle.prototype.getArea = function(){
    return this.length * this.width;
}

Circle.prototype.getArea = function(){
    return this.r * this.r * Math.PI;
}

var Rectangle1 = new Rectangle();
var Circle1 = new Circle();

console.log(Rectangle1.color);
console.log(Rectangle1.getArea());
console.log(Rectangle1.constructor);

console.log(Circle1.color);
console.log(Circle1.getArea());
console.log(Circle1.constructor);
console.log(Rectangle1.getColor == Circle1.getColor);
```



```
console.log(Rectangle1.setColor == Circle1.setColor);
```

//原型式继承

```
test2.js
142 //原型式继承
143 function obj(o){ //中转函数
144     function F(){
145         F.prototype = o ;
146         return new F();
147     }
148     var Shape = { //父类
149         color:this.color
150         ,
151         getColor: function(){
152             return this.color;
153         }
154         ,
155         setColor: function(){
156             return this.color = color;
157         }
158     }
159     var Rectangle = obj(Shape);
160     Rectangle.color = "red";
161     Rectangle.length = 5 ;
162     Rectangle.width = 5 ;
163     Rectangle.getArea = function(){
164         return this.length * this.width;
165     }
166
167     var Circle = obj (Shape);
168     Circle.color = "blue";
169     Circle.r = 5;
170     Circle.getArea = function(){
171         return this.r * this.r * Math.PI;
172     }
173 }
```

问题 输出 调试控制台 终端

```
node --debug-brk=3332 --nolazy exe.7\test2.js
Debugger listening on [::]:3332
25
78.53981633974483
red
25
[Function: Object]
blue
78.53981633974483
[Function: Object]
true
true
```

代码如下：

```
//原型式继承
function obj (o){ //中转函数
    function F(){
        F.prototype = o ;
        return new F();
    }
    var Shape = { //父类
        color:this.color
        ,
        getColor: function(){
            return this.color;
        }
        ,
        setColor: function(){
            return this.color = color;
        }
    }
    var Rectangle = obj(Shape);
    Rectangle.color = "red";
    Rectangle.length = 5 ;
    Rectangle.width = 5 ;
    Rectangle.getArea = function(){
        return this.length * this.width;
    }

    var Circle = obj (Shape);
    Circle.color = "blue";
    Circle.r = 5;
    Circle.getArea = function(){
        return this.r * this.r * Math.PI;
    }

    console.log(Rectangle.color);
    console.log(Rectangle.getArea());
    console.log(Rectangle.constructor);

    console.log(Circle.color);
    console.log(Circle.getArea());
    console.log(Circle.constructor);
    console.log(Rectangle.getColor == Circle.getColor);
```

```
console.log(Rectangle.setColor == Circle.setColor);
```

//寄生式继承

```
test2.js x [Debugger] [Pause] [Step Over] [Step Into] [Step Out] [Run] [Close]
193
194 function create (o){ //寄生函数
195     var f = obj (o);
196     f.getColor = function(){
197         return this.color;
198     }
199     f.setColor = function(){
200         return this.color = color;
201     }
202
203     return f;
204 }
205
206 var Shape = { //父类
207     color:this.color
208 }
209 var Rectangle = create(Shape);
210 Rectangle.color = "red";
211 Rectangle.length = 5 ;
212 Rectangle.width = 5 ;
213 Rectangle.getArea = function(){
214     return this.length * this.width;
215 }
216
217 var Circle = create (Shape);
218 Circle.color = "blue";
219 Circle.r = 5;
220 Circle.getArea = function(){
221     return this.r * this.r * Math.PI;
222 }
```

问题 输出 调试控制台 终端

```
node --debug-brk=9673 --nolazy exe.7\test2.js
Debugger listening on [::]:9673
red
25
[Function: Object]
blue
78.53981633974483
[Function: Object]
false
false
```

代码如下：

```
function obj (o){ //中转函数
    function F(){
        F.prototype = o ;
        return new F();
    }

    function create (o){ //寄生函数
        var f = obj (o);
        f.getColor = function(){
            return this.color;
        }
        f.setColor = function(){
            return this.color = color;
        }

        return f;
    }

    var Shape = { //父类
        color:this.color
    }
    var Rectangle = create(Shape);
    Rectangle.color = "red";
    Rectangle.length = 5 ;
    Rectangle.width = 5 ;
    Rectangle.getArea = function(){
        return this.length * this.width;
    }

    var Circle = create (Shape);
    Circle.color = "blue";
    Circle.r = 5;
    Circle.getArea = function(){
        return this.r * this.r * Math.PI;
    }

    console.log(Rectangle.color);
    console.log(Rectangle.getArea());
    console.log(Rectangle.constructor);
```

```
console.log(Circle.color);  
console.log(Circle.getArea());  
console.log(Circle.constructor);  
console.log(Rectangle.getColor == Circle.getColor);  
console.log(Rectangle.setColor == Circle.setColor);
```

//寄生组合式继承

```
test2.js x [debugger icons]
270     Shape.call(this , color);//对象冒充继承，可以传参
271 };
272 create_rectangle(Shape , Rectangle);
273 Rectangle.prototype.getArea = function(){
274     return this.length * this.width;
275 }
276
277
278 Circle.prototype = new Shape(); //一次
279 function Circle(color){
280     this.r = 5;
281     Shape.call(this , color);//对象冒充继承，可以传参
282 };
283 create_circle(Shape , Circle);
284 Circle.prototype.getArea = function(){
285     return this.r * this.r * Math.PI;
286 }
287
288 var Rectangle1 = new Rectangle("red");
289 var Circle1 = new Circle("blue");
290
291 console.log(Rectangle1.color);
292 console.log(Rectangle1.getArea());
293
294 console.log(Circle1.color);
295 console.log(Circle1.getArea());
296
297 console.log(Rectangle1.constructor);
298 console.log(Circle1.constructor);
```

getArea(): number

问题	输出	调试控制台	终端
			node --debug-brk=6703 --nolazy exe.7\test2.js
			Debugger listening on [::]:6703
			red
			25
			blue
			78.53981633974483
			[Function: Rectangle]
			[Function: Circle]

代码如下：

```
function obj (o){ //中转函数
    function F(){
        F.prototype = o ;
        return new F();
    }

    function create_rectangle(Shape , Rectangle ){//寄生函数 1
        var f = obj (Shape.prototype);
        f.constructor = Rectangle;
        Rectangle.prototype = f;
    }

    function create_circle(Shape , Circle){//寄生函数 2
        var f = obj (Shape.prototype);
        f.constructor = Circle;
        Circle.prototype = f;
    }

    function Shape(color){
        this.color = color;
    }

    Shape.prototype.getColor = function() {
        return this.color;
    };

    Shape.prototype.setColor = function(color) {
        return this.color = color;
    };

    //Rectangle.prototype = new Shape(); //一次
    function Rectangle(color){
        this.length = 5;
        this.width = 5 ;
        Shape.call(this , color);//对象冒充继承，可以传参
    };
    create_rectangle(Shape , Rectangle);
    Rectangle.prototype.getArea = function(){
        return this.length * this.width;
    }
}
```

```
Circle.prototype = new Shape(); //一次
function Circle(color){
    this.r = 5;
    Shape.call(this , color); //对象冒充继承，可以传参
};
create_circle(Shape , Circle);
Circle.prototype.getArea = function(){
    return this.r * this.r * Math.PI;
}

var Rectangle1 = new Rectangle("red");
var Circle1 = new Circle("blue");

console.log(Rectangle1.color);
console.log(Rectangle1.getArea());

console.log(Circle1.color);
console.log(Circle1.getArea());

console.log(Rectangle1.constructor);
console.log(Circle1.constructor);
```