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，已备后用。
- 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

```
//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";
```

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

```
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();
```

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

```
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);
```

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

```
//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;
}
```

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

```
}

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;
}
```

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

```
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);   getArea(): number
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);
```

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);
```