实验五　JavaScript 面向对象编程实验

一、教学课时

4学时

二、实验目的和要求

1. 掌握用不同方法创建对象。
2. 掌握 this、new、get、set 等关键字的使用。
3. 掌握原型的创建与修改。
4. 掌握原型调用链。

三、实验内容

第一阶段：程序验证

指导 1　创建一个计算器

知识点：

- 对象的字面量创建方式
- this 关键字的使用

⑴ 问题：创建一个有五个方法的 calculator 对象：

1) sum() 返回保存的值的和并返回计算结果。
2) sub() 返回保存的值的差并返回计算结果
3) mul() 将保存的值相乘并返回计算结果。
4) div() 将保存的值相除并返回计算结果。
5) read() 提示输入两个值，并将其保存为对象属性。

并编写代码验证。

⑵ 分析：当使用 this 关键字取变量时，会先检查变量是否存在，不存在则会在类中添加上该变量，并把对应的值上。

⑶ 解决方案：

a　创建网页文档，编写网页基本结构代码。

b　在网页<head></head>中编写如下 JavaScript 代码。

```
let calculator = {
    sum() {return this.a + this.b;},

    sub(){return this.a-this.b;},

    mul() {return this.a * this.b;},

    div(){
        if(this.b===0){
            alert("被除数不能为0");
            return null
        }else {
            return this.a/this.b;
        }
    },


    read() {
        while (this.a==null||this.b==null){
            this.a = this.a==null?prompt( message: 'a?',  _default: 0):this.a;
            this.b = this.b==null?prompt( message: 'b?',  _default: 0):this.b;
        }

        this.a=Number.parseInt(this.a);
        this.b=Number.parseInt(this.b);
    }
};

calculator.read();
let operationSymbol=prompt( message: "运算符号为", _default: "+");
```

```
switch (operationSymbol){
    case "+":alert("相加结果为"+calculator.sum());break;
    case "-":alert("相减结果"+calculator.sub());break;
    case "*":alert("相乘结果"+calculator.mul());break;
    case "/":{
        if(calculator.div()!=null){
            alert("相除结果"+calculator.div())
        }
    }break;
    default:alert("请输入正确的运算符");break;
}
```

(4) 在浏览器中运行此网页，运行结果如图 1 所示。

图 1-1 指导 1 运行截图

图 1-2 指导 1 运行截图

图 1-3 指导 1 运行截图

图 1-4 指导 1 运行截图

指导 2　隐藏属性

知识点：

- 用 Symbol 对象隐藏属性
- 用可选链?.链验证前面的属性是否存在

（1）问题：创建 Symbol 对象来隐藏 user 对象的 id 属性,属性 id 的值是一个对象，对象由 studentId 和 courseId 两个属性构成，并用?.可选链来确保 user 对象的 id 属性真实存在，并输出 studentId 的值。

(2) 分析：Symbol 允许创建对象的“隐藏”属性，代码的任何其它部分都不能意外访问或重写这些属性。如果可选链 ?. 前面部分是 undefined 或者 null，它会停止运算并返回 undefined。用 Symbol 创建的属性不能用 . 运算符的方式进行访问，必须用[]运算符进行访问。

(3) 解决方案：

a　创建网页文档，编写网页基本结构代码。

b　在网页<head></head>中编写如下 JavaScript 代码。

```
let user = {
    name: "John"
};

alert("user对象存在测试name: "+user?.name);

let id = Symbol( description: "id");

user[id] = {
    studentId:2,
    courseId:1,
};

alert("user对象存在测试[id]: "+user?.[id]?.studentId);
alert("user对象存在测试id: "+user?.id?.studentId);
alert("user对象为"+JSON.stringify(user));
// 我们可以使用 Symbol 作为键来访问数据,用?.链来能够安全地访问嵌套属性。
```

(4) 在浏览器中运行此网页，运行结果如下图 所示。

图 2-1 指导 2 运行截图

图 2-2 指导 2 运行截图

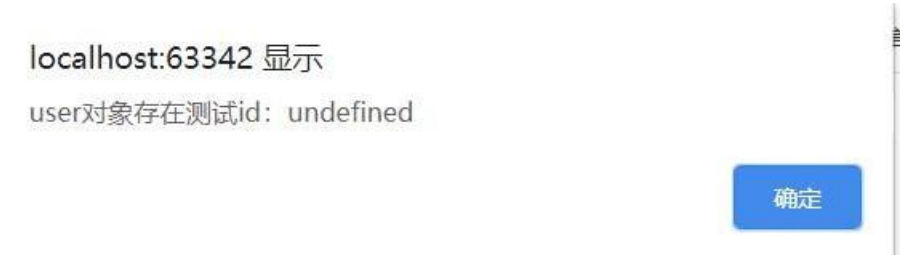

图 2-3 指导 2 运行截图

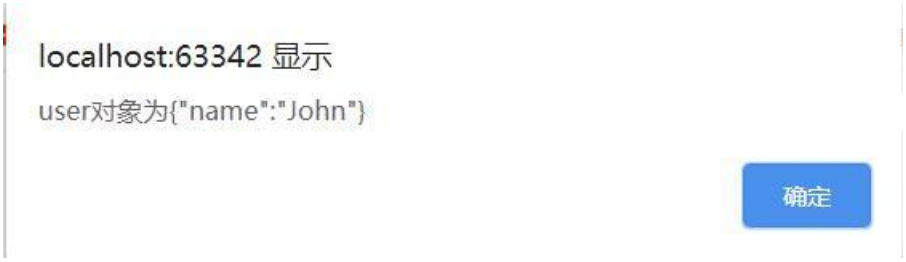

图 2-4 指导 2 运行截图

指导 3 深层克隆

知识点：

- Symbol 对象的隐藏属性的修改
- 对象的深层克隆复制

① 问题：在指导 2 的基础上，添加 size 属性，size 属性的值为由 height 与 weight 两个属性值组成的对象，克隆出另一个对象并对其所有属性进行重新赋值，并输出两个对象的值。

② 分析： 对象的深度克隆需要对每个属性的值进行克隆，每一个属性相对于原来的对象都是独立的。如果值是对象 c，则需要进入则需要进入对象 c 内部，进行一次属性值的克隆复制，从而获得一个独立的对象 c 的复制，依次类推，最终获得到一个完全独立的克隆对象。

③ 解决方案：

a 创建网页文档，编写网页基本结构代码。

b 在网页<head></head>中编写如下 JavaScript 代码。

```
let user = {
    name: "John",
    size:{
        height:185,
        weight:75,
    }
};
let id = Symbol( description: "id");

user[id] = 1;

let cloneUser=cloneDeep(user);

cloneUser.size.height=190;
cloneUser.size.weight=80;
cloneUser.name="Jack";
cloneUser[id]=2;

alert(user[id]+" "+JSON.stringify(user));
alert(cloneUser[id]+" "+JSON.stringify(cloneUser));
```

```
function cloneDeep(obj) {
    let newobj = obj.constructor === Array ? [] : {};
    //constructor 属性返回对创建此对象的数组函数的引用。创建相同类型的空数据

    if (typeof obj !== 'object') {
        return;
    } else {
        for (var i in obj) {
            if (typeof obj[i] === 'object'){//判断对象的这条属性是否为对象
                newobj[i] = cloneDeep(obj[i]);//若是对象进行嵌套调用
            }else{
                newobj[i] = obj[i];
            }
        }
    }

    return newobj;//返回深度克隆后的对象
};
```

④ 在浏览器中运行此网页，运行结果如下图 所示。

图 3-1 指导 3 运行截图

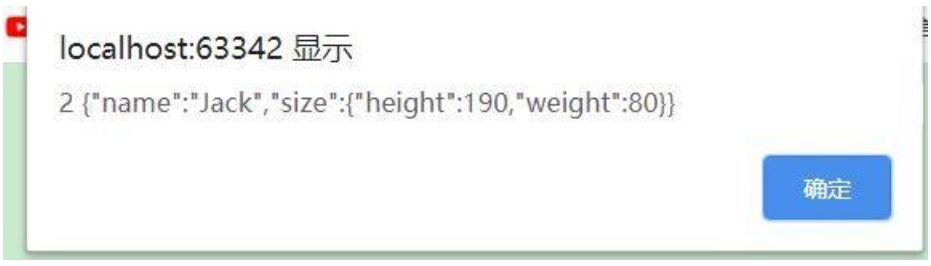

图 3-2 指导 3 运行截图

指导 4 setter 与 getter

知识点：

- getter 与 setter 方法的运用

⑴ 问题：在指导 3 的基础上，将 name 属性的值改为 fullname 访问器属性，fullname 属性是由 firstName 和 lastName 两个属性组成的，如“Alice Cooper”，fullName 属性可以通过 getter 与 setter 方法对值进行获取与修改操作。编写代码显示修改后类的所有属性值。

⑵ 分析：JavaScript 的访问器属性由 “getter” 和 “setter” 方法表示，getter 方法用来访问对象的属性，setter 方法用来修改对象的属性。在对象字面量中，它们分别用 get 和 set 表示。

⑶ 解决方案：

a 创建网页文档，编写网页基本结构代码。

b 网页<head></head>中编写如下 JavaScript 代码。

```
let user = {
    size:{
        height:185,
        weight:75,
    },
    set fullName(value) {
        [this.firstName, this.lastName] = value.split(" ");
    },
    get fullName() {
        return `${this.firstName} ${this.lastName}`;
    },

};

alert(JSON.stringify(user));

user.fullName = "Alice Cooper";

alert("设置后:"+JSON.stringify(user));
```

⑷ 在浏览器中运行此网页，运行结果如下图 所示。

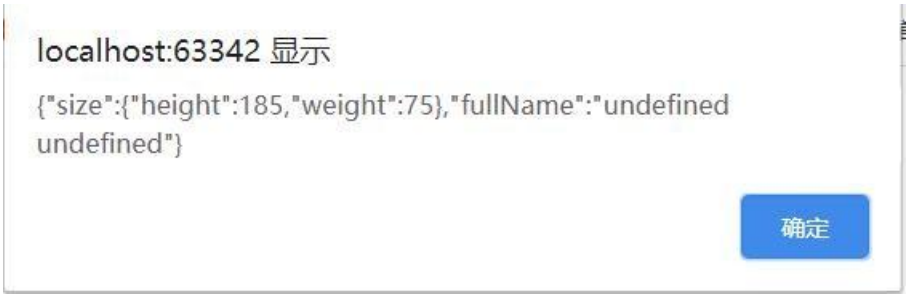

图 4-1 指导 4 运行截图

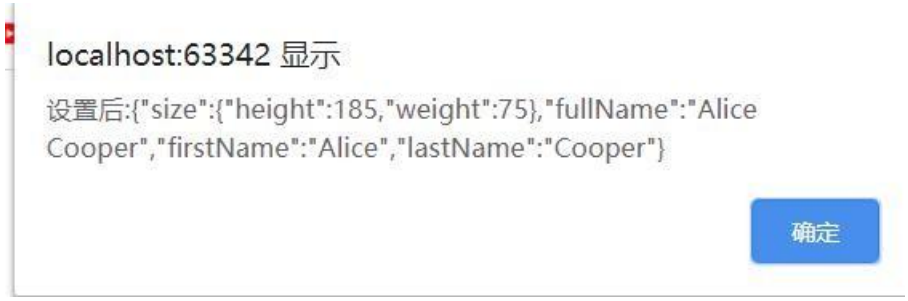

图 4-2 指导 4 运行截图

第二阶段：程序编写

1. 练习 1 用构造函数来重写计算器

⓪ 问题： 在指导 1 的基础上，改用构造函数来创建计算器对象，并用 new 操作符来新建一个对象 c，并对该对象 c 的方法进行测试。

2. 练习 2 修改原型

⓪ 问题：创建构造函数 Hamster，为 Hamster 的原型添加方法 run()，run()的内容是输出“跑得快”，在 Hamster 构造函数中添加 run（）方法，run()的内容是输出“跑得慢”，用构造函数创建对象 lazy,再用 lazy 的构造器， 创建对象 speedy,并修改 speedy 的run 属性，内容是输出“跑得很快”。执行speedy 的run()、lazy 的run(),与 lazy 的原型的 run()。

⓪ 分析：每个函数都有 "prototype" 属性，是一个只有属性 constructor 的对象，属性 constructor 指向函数自身。而当用构造函数创建对象时会调用这个属性，并给对象创建一个原型属性 proto_ ，_proto_ 属性的值为构造函数的 prototype 属性。

3. 练习 3 更聪明的 getter/setter

⓪ 问题：person 字面量对象的内容如下：

⓵ 属性的键为 Id，不能意外访问或重写 Id。

2) 属性的键为 name。

3) 属性的键为 class。该属性的值为一个对象 c，该对象 c 有 major, 与 classNumber 属性。

4) 方法 study,调用该方法时会弹出“正在学习”的弹窗。

并用代码显示该对象的属性值。

2) 问题：问题1 的基础上将 name 和 class 两个属性改为访问器属性， 并在修改属性值时对传入的值进行判断，其要求如下：

1) name 的长度不能低于四个英文字符，不能是中文。

2) class 属性的值中 classNumber 必须是数字，major 必须是中文字符串。

并用代码测试 name 和 class 两个访问器属性的修改方法(setter)。

3) 提示：判断值的条件语句建议用正则表达式进行判断，中文判断的正则表达：\^[\u4e00-\u9fa5]$\。