实验七 JavaScript 函数进阶编程实验

一、教学课时

2学时。

二、实验目的和要求

1 掌握 JavaScript 中函数的定义、调用。
2 掌握 JavaScript 中闭包的概念和使用。
3 掌握 JavaScript 中简单正则表达式的使用。
4 掌握函数内置函数call（）、bind（）的使用。

三、实验内容

第一阶段：程序验证

1 指导 1 实现电子邮件验证函数

知识点：

- 函数的定义和调用
- 字符串对象的属性和方法

⑴ 问题：在会员注册时一般要求用户提供自己的邮箱。需要对表示邮箱的址的字符串进行验证。假设的规则包含：不予许为空字符串，必须包含一个且仅能包含一个@符号，@符号不能位于开头，必须包含一个[.]号，@符号必须比第一个[.]号先出现，最后一个点号不能位于末尾，@符号和[.]号之间至少有一个字符。编写一个函数，实现电子邮件的格式验证功能，并测试这个函数。

⑵ 分析：此函数应接受一个字符串作为参数，此字符串代表要被验证的字符串数据；此函数应返回一个布尔值，它表示验证的结果。函数体中验证过程的逻辑主要使用字符串对象的一系列方法。

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。

```
<script>
    function chkEmail(email) {
        return email != "" && email.indexOf("@") > 0
            && email.indexOf("@") == email.lastIndexOf("@")
            && email.indexOf(".") > email.indexOf("@") + 1
            && email.lastIndexOf(".") != length - 1;
    }
    var myemail = "myOEAC@21cn.com";
    document.write(chkEmail(myemail));
</script>
```

(4) 在浏览器中运行这个网页，输出结果应为”true”。修改变量 myemail 的值，测试其他情形。

2 指导 2 将数字字符转化为数字以参与数学运算

知识点：

- 使用全局函数实现字符串转化为数字

(1) 问题：假书已有两个变量 price 和 num，它们储存的值是从表单的文本框获取的商品单价和用户购买此商品的数量，它们都是字符串类型的。现在要计算金额 amount，但是显然字符串类型的值是不能参与乘法预算。需要先将它们转化为数字类型。

(2) 分析：有两个全局函数能实现数字的转化，分别是 parseFloat 和 parseInt。

(3) 解决方案：

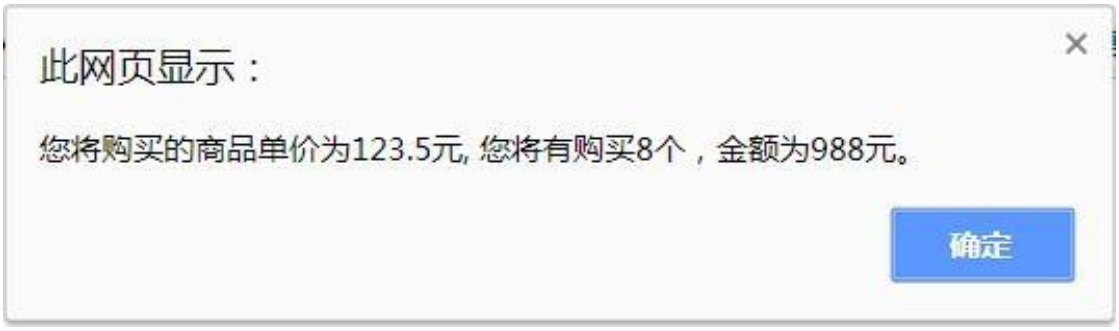
a) 创建网页，编写基本结构代码。

b) 在网页正文<body></body>标签内编写 JavaScript 如下代码。

```
<script>
    var price = "123.5";
    var num = "8";
    var amount;

    amount = parseFloat(price) * parseInt(num);
    window.alert("您将购买的商品单价为" + price + "元，您将有购买"
                + num + "个，金额为" + amount + "元。");
</script>
```

(4) 在浏览器中运行此网页，如图上机 2 所示。



图上机 2

3. 指导 3　验证字符串是否全由数字字符组成

知识点：

- 正则表达式及正则表达式对象

(1) 问题：已知 qq 储存了用户在文本框输入的 QQ 密码，它是字符串类型的。现在要验证它包含的所有字符是否都是数字字符。

(2) 分析：使用正则表达式可测试一个字符串是否满足某种匹配模式。以正则表达式作为参数构造 TegExp 对象，再调用它的 test()方法。

(3) 解决方案：

a) 创建网页，编写基本结构代码。

b) 在网页正文<body></body>标签内编写 JavaScript 代码，如下：

```
<script>
    var qq = "20534123";
    var result;
    result = (new RegExp("^[0-9]*$")).test(qq);
    window.alert("使用正则表达式兑现RegExp检测qq号码是否全部为数字，结果为：" + result);
</script>
```

图上机 3

(4) 在浏览器中运行此网页，如图上机 3 所示。

第二阶段：程序编写

1. 下面透明缓存的代码在执行过程中报错，请分析错误的原因，并改正。

```
<!DOCTYPE html>
<script>
"use strict";

// 对 worker.slow 的结果进行缓存
let worker = {
  someMethod() {
    return 1;
  },
```

```
  slow(x) {
    // 可怕的 CPU 过载任务
    alert("Called with " + x);
    return x * this.someMethod(); // (*)
  }
};

//透明缓存
function cachingDecorator(func) {
  let cache = new Map();
  return function(x) {
    if (cache.has(x)) {
      return cache.get(x);
    }
    let result = func(x); // (**)
    cache.set(x, result);
    return result;
  };
}

alert( worker.slow(1) ); // 原始方法有效

worker.slow = cachingDecorator(worker.slow); // 现在对其进行缓存

alert( worker.slow(2) ); // Error: Cannot read property 'someMethod'
 of undefined
</script>
```

2.下面代码中对askPassword()的调用将会检查 password，然后基于结果调用 user.loginOk/loginFail。但是它导致了一个错误。请分析错误原因，并改正。

```
 <!DOCTYPE html>
<script>
"use strict";

function askPassword(ok, fail) {
  let password = prompt("Password?", '');
  if (password == "rockstar") ok();
  else fail();
}

let user = {
  name: 'John',

  loginOk() {
    alert(`${this.name} logged in`);
  },
```

```
  loginFail() {
    alert(`${this.name} failed to log in`);
  },

};

askPassword(user.loginOk, user.loginFail);
</script>
```

3.下面代码每次调用add()方法输出均是“2”，请分析执行结果的原因。并请用闭包改写程序，使其每次执行加1。

```
function add() {
  let x = 1;
  console.log(++x);
}

add(); //执行输出2
add(); //执行还是输出2
```