

arena teaching system

JavaScript



Java企业应用及互联网
高级工程师培训课程

达内集团教学研发部 编著

目 录

Unit01	1
1. JavaScript 概述	3
1.1. JavaScript 概述	3
1.1.1. 什么是 JavaScript	3
1.1.2. JavaScript 发展史	3
1.1.3. JavaScript 的特点	3
1.2. 使用 JavaScript	4
1.2.1. 第一个 JavaScript 程序	4
1.2.2. 事件定义方式	4
1.2.3. 嵌入式：<script> 标签	4
1.2.4. 文件调用方式：js 文件	5
1.2.5. JavaScript 的代码错误	6
2. JavaScript 基础语法	6
2.1. 语法规则	6
2.1.1. 编写 JavaScript 代码	6
2.1.2. 大小写敏感	6
2.1.3. 换行与空格	7
2.2. 标识符与变量	7
2.2.1. 标识符与关键字	7
2.2.2. 变量	8
2.3. 数据类型	8
2.3.1. 数据类型	8
2.3.2. 数据类型的隐式转换	9
2.3.3. 数据类型转换函数	10
2.3.4. 特殊数据类型	11
2.4. 运算符	11
2.4.1. 算术运算	11
2.4.2. 关系运算	11
2.4.3. 逻辑运算	12
2.4.4. 条件运算符	12
3. 流程控制	13
3.1. 控制语句	13
3.1.1. 控制语句	13
3.2. 分支结构	13
3.2.1. if 语句	13

3.2.2. switch-case 语句	13
3.3. 循环结构	14
3.3.1. for 语句	14
3.3.2. while 语句	14
3.3.3. do-while 语句	14
4. 常用内置对象.....	15
4.1. JavaScript 对象概述	15
4.1.1. 什么是 JavaScript 对象	15
4.1.2. 使用对象	15
4.1.3. 常用内置对象	15
4.2. String 对象	16
4.2.1. String 对象	16
4.2.2. String 对象的常用方法	16
4.2.3. String 对象与正则表达式	18
4.3. Array 对象	19
4.3.1. Array 对象	19
4.3.2. 创建二维数组	19
4.3.3. Array 对象的常用方法	19
4.3.4. 数组倒转与排序	20
4.4. Math 对象	21
4.4.1. Math 对象	21
4.4.2. Math 对象的常用属性和方法	21
4.5. Number 对象	22
4.5.1. Number 对象	22
4.5.2. Number 对象的常用方法	22
经典案例	23
1. javascript 的 HelloWorld	23
2. 判断数据类型，并计算平方	26
3. 猜数字	30
4. 计算阶乘.....	34
5. 字符的查询与过滤	37
6. 字符的查询与过滤（使用正则表达式）	40
7. 数组的倒转与排序	43
8. 随机数生成器	48
9. 资产折旧计算器	51
课后作业	57
Unit02	60
1. 常用内置对象.....	62

1.1. 正则表达式对象	62
1.1.1. RegExp 对象概述	62
1.1.2. RegExp 对象的常用方法	62
1.2. Date 对象	62
1.2.1. Date 对象	62
1.2.2. Date 对象的常用方法	63
1.3. Function 对象	63
1.3.1. 函数与 Function 对象	63
1.3.2. 函数的定义	63
1.3.3. 函数的调用	64
1.3.4. arguments 对象	64
1.3.5. 使用 Function 对象创建函数	64
1.3.6. 匿名函数	65
1.4. 全局函数	65
1.4.1. 全局函数概述	65
1.4.2. encodeURIComponent 与 decodeURI	65
1.4.3. eval 函数	66
2. window 对象	66
2.1. DHTML 概述	66
2.1.1. DHTML 简介	66
2.1.2. DHTML 对象模型	66
2.1.3. BOM 与 DOM	67
2.2. window 对象	67
2.2.1. window 对象	67
2.3. 窗口和对话框	68
2.3.1. 对话框	68
2.3.2. 窗口的打开和关闭	68
2.4. 定时器	68
2.4.1. 定时器	68
2.4.2. 周期性定时器	69
2.4.3. 一次性定时器	69
3. document 对象	69
3.1. document 对象概述	69
3.1.1. document 对象概述	69
3.2. DOM 概述	70
3.2.1. DOM 概述	70
3.2.2. DOM 节点树	70
3.2.3. DOM 操作	71

3.3. DOM 操作 - 读取、修改	71
3.3.1. 节点信息	71
3.3.2. 元素节点的内容	72
3.3.3. 节点属性	73
3.3.4. 元素节点的样式	73
经典案例	74
1. 输入验证	74
2. 计算查询时段	77
3. 模拟方法的重载	82
4. 数组按数值排序	85
5. 简单计算器	88
6. 删除操作的提交和取消	92
7. 动态时钟的启动和停止	94
8. 页面定时跳转	97
9. 广告轮播	100
课后作业	106
Unit03	110
1. document 对象	111
1.1. DOM 操作 - 查询	111
1.1.1. 查询节点	111
1.1.2. 根据元素 ID 查找节点	111
1.1.3. 根据层次查找节点	112
1.1.4. 根据标签名查找节点	113
1.1.5. 根据 name 属性查找节点	114
1.2. DOM 操作 - 增加	114
1.2.1. 创建新节点	114
1.2.2. 添加新节点	114
1.3. DOM 操作 - 删除	116
1.3.1. 删除节点	116
2. HTML DOM 对象	118
2.1. HTML DOM 概述	118
2.1.1. HTML DOM 概述	118
2.1.2. 常用 HTML DOM 对象	118
2.1.3. 标准 DOM 与 HTML DOM	119
2.2. Select 与 Option 对象	120
2.2.1. Select 对象	120
2.2.2. Option 对象	120
经典案例	121

1. 表单的验证与提交	121
2. 购物车 - 修改购物数量	130
3. 购物车 - 购物金额计算	136
4. 动态创建页面元素	141
5. 联动菜单	145
6. 联动菜单 (HTML DOM 实现)	150
课后作业	154
Unit04	157
1. HTML DOM 对象	159
1.1. Table 对象	159
1.1.1. Table 对象	159
1.1.2. TableRow 对象	159
1.1.3. TableCell 对象	160
2. DHTML 其他对象	160
2.1. DHTML 其他对象	160
2.1.1. DHTML 对象模型回顾	160
2.1.2. screen 对象	160
2.1.3. history 对象	161
2.1.4. location 对象	161
2.1.5. navigator 对象	161
3. 事件	162
3.1. 事件概述	162
3.1.1. 事件概述	162
3.1.2. 事件句柄	162
3.2. 事件处理	162
3.2.1. 事件定义	162
3.2.2. 事件的处理机制	163
3.3. event 对象	164
3.3.1. event 对象	164
3.3.2. 获取 event 对象	164
3.3.3. 使用 event 对象	165
4. 面向对象基础	166
4.1. 对象概述	166
4.1.1. 对象概述	166
4.2. 创建对象	166
4.2.1. 创建通用对象	166
4.2.2. 创建对象的模板	167
4.2.3. JSON	167

经典案例	168
1. 数据表格的操作	168
2. 使用 location 对象	177
3. 遍历 navigator 对象的属性	179
4. 事件的冒泡处理机制	183
5. 获取 event 对象的数据	185
6. 简单计算器	188
7. 使用 Object	192
8. 自定义对象	196
9. 使用 JSON	199
课后作业	203

JavaScript

Unit01

知识体系.....Page 3

JavaScript 概述	JavaScript 概述	什么是 JavaScript
		JavaScript 发展史
		JavaScript 的特点
	使用 JavaScript	第一个 JavaScript 程序
		事件定义方式
		嵌入式：<script> 标签
		文件调用方式：js 文件
		JavaScript 的代码错误
JavaScript 基础语法	语法规则	编写 JavaScript 代码
		大小写敏感
		换行与空格
	标识符与变量	标识符与关键字
		变量
	数据类型	数据类型
		数据类型的隐式转换
		数据类型转换函数
		特殊数据类型
	运算符	算数运算
		关系运算
		逻辑运算
		条件运算符
流程控制	控制语句	控制语句
	分支结构	if 语句
		switch-case 语句
	循环结构	for 语句
		while 语句
		do-while 语句
常用内置对象	JavaScript 对象概述	什么是 JavaScript 对象
		使用对象

		常用内置对象
	String 对象	String 对象
		String 对象的常用方法
		String 对象与正则表达式
	Array 对象	Array 对象
		创建二维数组
		Array 对象的常用方法
		数组倒转与排序
	Math 对象	Math 对象
		Math 对象的常用属性和方法
	Number 对象	Number 对象
		Number 对象的常用方法

经典案例.....Page 23

javascript 的 HelloWorld	事件定义
	嵌入式：<script> 标签
	文件调用式：js 文件
判断数据类型，并计算平方	数据类型
	数据类型的隐式转换
	数据类型转换函数
猜数字	条件运算符
计算阶乘	for 语句
字符的查询与过滤	String 对象的常用方法
字符的查询与过滤（使用正则表达式）	String 对象与正则表达式
数组的倒转与排序	Array 对象的常用方法
	数组倒转与排序
随机数生成器	Math 对象
	Math 对象的常用属性和方法
资产折旧计算器	Number 对象的常用方法

课后作业.....Page 57

1. JavaScript 概述

1.1. JavaScript 概述

1.1.1. 【JavaScript 概述】什么是 JavaScript

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">Tarena 达内科技</div> <h3>什么是 JavaScript</h3> <ul style="list-style-type: none"> JavaScript 是一种基于对象和事件驱动的解释性脚本语言，具有与Java和C语言类似的语法 <ul style="list-style-type: none"> 一种网页编程技术，用来向 HTML 页面添加交互行为 直接嵌入 HTML 页面 由浏览器解释执行代码，不进行预编译 <div style="text-align: right;">++</div>
---	---

1.1.2. 【JavaScript 概述】JavaScript 发展史

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">Tarena 达内科技</div> <h3>JavaScript 发展史</h3> <ul style="list-style-type: none"> ECMA-262 是正式的 JavaScript 标准，此标准由 ECMA 组织发展和维护 此标准基于 JavaScript (Netscape) 和 JScript (Microsoft) <ul style="list-style-type: none"> 网景公司在Netscape2.0首先推出了JavaScript 微软公司从IE3.0开始提供对客户端JavaScript的支持，并另取名为JScript <div style="text-align: right;">++</div>
---	---

1.1.3. 【JavaScript 概述】JavaScript 的特点

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">Tarena 达内科技</div> <h3>JavaScript 的特点</h3> <ul style="list-style-type: none"> 可以使用任何文本编辑工具编写 由浏览器内置的 JavaScript 引擎执行代码 <ul style="list-style-type: none"> 解释执行：事先不编译，逐行执行 基于对象：内置大量现成对象 适宜： <ul style="list-style-type: none"> 客户端数据计算 客户端表单合法性验证 浏览器事件的触发 网页特殊显示效果制作 服务器的异步数据提交 <div style="text-align: right;">++</div>
---	---

1.2. 使用 JavaScript

1.2.1. 【使用 JavaScript】第一个 JavaScript 程序

代码编辑

第一个 JavaScript 程序

Tarena
达内科技

- 事件定义方式
- 嵌入式
 - 使用 <script> 标签
- 文件调用方式
 - 代码位于单独的 .js 文件
 - html 页面引用 js 文件

++

1.2.2. 【使用 JavaScript】事件定义方式

代码编辑

事件定义方式


Tarena
达内科技

- 在定义事件时直接写入 JavaScript 脚本代码

```

<html>
  <head> </head>
  <body>
    <form>
      <input type= "button" value= "第一个
按钮" onclick=" alert( 'hello,world' );">
    </form>
  </body>
</html>
            
```

注意单双引号的使用



++

1.2.3. 【使用 JavaScript】嵌入式: <script> 标签

代码编辑

嵌入式: <script> 标签

Tarena
达内科技

- 在页面上嵌入 <script></script> 标签
 - 标签中放置 JavaScript 代码

```

<html>
  <head>
    <script type="text/javascript" language="javascript" >
      alert("Hello world.");
    </script>
  </head>
  <body>
  </body>
</html>
            
```

页面加载时运行

++

知识讲解

嵌入式：<script> 标签（续1）

```

<html>
  <head>
    <script type="text/javascript" language="javascript">
      function method1(){
        alert("hello in method1.");
      }
    </script>
  </head>
  <body>
    <form>
      <input type="button" value="第二个按钮"
        onclick="method1();">
    </form>
  </body>
</html>
          
```

<script> 标签中定义方法

按钮的事件中调用

++

1.2.4. 【使用 JavaScript】文件调用方式：js 文件

知识讲解

文件调用方式：js 文件

- 将 JavaScript 代码写入一个单独的文件，并保存为后缀为 js 的文件
 - 为纯文本文件
 - 文件中，不需要包含 <script> 标签，直接书写 js 代码

```

function method2()
{
  alert("hello word!");
}
          
```

myJs.js文件
 脚本文件中不需要脚本开始和结束声明

++

知识讲解

文件调用方式：js 文件（续1）

- html 页面的 <head> 中引用外部的 js 文件
 - 在 <head> 中添加 <script> 标签
 - 设置 <script> 标签的 "src" 属性，以指定 js 文件的 url

```

<html>
  <head>
    <script language="JavaScript" src="myJs.js"
      type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <input type="button" value="第二个按钮"
        onclick="method2();">
    </form>
  </body>
</html>
          
```

++

5

1.2.5. 【使用 JavaScript】JavaScript 的代码错误

| | |
|---|---|
| <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> | <div style="text-align: right;">Tarena 达内科技</div> <h4>JavaScript 的代码错误</h4> <ul style="list-style-type: none"> • 解释性代码，代码错误，则页面中无效果 • IE 浏览器 <ul style="list-style-type: none"> – 状态栏、开发工具 • Firefox 浏览器、Chrome 浏览器 <ul style="list-style-type: none"> – 使用错误控制台查看 <div style="text-align: right;">+</div> |
|---|---|

2. JavaScript 基础语法

2.1. 语法规范

2.1.1. 【语法规范】编写 JavaScript 代码

| | |
|---|--|
| <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> | <div style="text-align: right;">Tarena 达内科技</div> <h4>编写 JavaScript 代码</h4> <ul style="list-style-type: none"> • 由Unicode字符集编写 • 注释 <ul style="list-style-type: none"> – 单行：// – 多行：/* */ • 语句 <ul style="list-style-type: none"> – 表达式、关键字、运算符组成 – 大小写敏感 – 使用分号或者换行结束 <div style="text-align: right;">+</div> |
|---|--|

2.1.2. 【语法规范】大小写敏感

| | |
|---|--|
| <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> | <div style="text-align: right;">Tarena 达内科技</div> <h4>大小写敏感</h4> <ul style="list-style-type: none"> • 在 JavaScript 程序中大小写是敏感的 <ul style="list-style-type: none"> – 标准的 JavaScript 语法定义中，区分大小写 <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> JS 代码：
 <pre><script language="JavaScript"> function myclick() //..... } </script></pre> </div> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> HTML 文档：
 <pre><input type="button" value="play" onClick="myclick();"></pre> </div> <div style="text-align: center; margin-top: 10px;"> <div style="border: 1px solid black; padding: 2px 10px;">大小写必须一致</div> </div> <div style="text-align: right;">+</div> |
|---|--|

2.1.3. 【语法规范】换行与空格

Tarena
达内科技

换行与空格

- 换行、分号

a=1;b=2;

a=1
b=2

a=1;
b=2;

有换行，分号允许不加

推荐加上分号
减少错误和歧义的发生
- 空格、TAB

{
a=1;
b=2;
}

{
a=1;
b=2;
}

推荐加上空格或TAB
增强程序可读性

++

2.2. 标识符与变量

2.2.1. 【标识符与变量】标识符与关键字

Tarena
达内科技

标识符与关键字

- 标识符
 - 由不以数字开头的字母、数字、下划线(_)、美元符号(\$)组成
 - 常用于表示函数、变量等的名称
 - 名称最好有明确的含义
 - 建议遵守 camel 法则
 - 例如：_abc,\$abc,abc,abc123是标识符，而1abc不是
- JavaScript语言中代表特定含义的词称为保留字，不允许程序再定义为标识符
 - 保留关键字，如 break、if 等

++

Tarena
达内科技

标识符与关键字（续1）

- 标准关键字

| | | | | |
|--------|----------|--------|-----------|------------|
| break | case | catch | continue | default |
| delete | do | else | false | finally |
| for | function | if | in | instanceof |
| new | null | return | switch | this |
| throw | true | try | typeof | var |
| void | while | with | undefined | ... |
- 预保留的关键字
 - class、int、float 等

++

2.2.2. 【标识符与变量】变量

Tarena
达内科技

变量

- 变量声明
 - 使用关键字 var 声明变量，如 var x,y;
- 变量初始化
 - 使用 “=” 赋值
 - 没有初始化的变量则自动取值为 undefined
 - 如：var count = 0;
- 变量命名同标识符的规则，大小写敏感
- 变量声明时不需要指定数据类型，以赋值为准

```
var n = 12;
var str = "hello" ;
```

Tarena
达内科技

2.3. 数据类型

2.3.1. 【数据类型】数据类型

Tarena
达内科技

数据类型

JavaScript数据类型

基本类型

Number : 数字
String : 字符串
Boolean : 布尔

特殊类型

null : 空
undefined : 未定义

复杂类型

Array : 数组
Object : 对象
...


Tarena
达内科技

Tarena
达内科技

数据类型（续1）

- String 类型
 - 表示文本
 - 由Unicode字符、数字、标点符号组成的序列
 - 首尾由一对单引号或双引号括起
 - 特殊字符需要转义符\，如：\n, \, \' , \"

```
var aa = "\u4f60\u597d\u201c欢迎来到JavaScript世界\u201d";
alert(aa);
```



Tarena
达内科技

代码清单

数据类型（续2）

Tarena
达内科技

- Number 类型
 - 不区分整型数值和浮点型数值
 - 所有数字都采用 64 位浮点格式存储，类似于double 格式
- 整数
 - 10进制的整数由数字的序列组成
 - 16进制数据前面加上0x，八进制前面加0
- 浮点数
 - 使用小数点记录数据，如 3.4，5.6
 - 使用指数记录数据，如 4.3e23 = 4.3 x 1023

++

代码清单

数据类型（续3）

Tarena
达内科技

- Boolean 类型
 - 仅有两个值：true和false
 - 也代表1和0
 - 实际运算中true=1,false=0
- 多用于结构控制语句

++

2.3.2. 【数据类型】数据类型的隐式转换

代码清单

数据类型的隐式转换

Tarena
达内科技

- JavaScript 属于松散类型的程序语言
 - 变量在声明时不需要指定数据类型
 - 变量由赋值操作确定数据类型
- 不同类型数据在计算过程中会自动进行转换

数字 + 字符串：数字转换为字符串
 数字 + 布尔值：true转换为1，false转换为0
 字符串 + 布尔值：布尔值转换为字符串true或false
 布尔值 + 布尔值：布尔值转换为数值1或0

++

代码预览

数据类型的隐式转换（续1）


```
//定义不同数据类型的变量
var s1 = "a";
var n1 = 1;
var b1 = true;
var b2 = false;

//测试隐式转换
alert(s1 + n1);

alert(s1 + b1);

alert(n1 + b1);

alert(b1 + b2);
```



++

2.3.3. 【数据类型】数据类型转换函数

代码预览

数据类型转换函数

- toString
 - 转换成字符串
 - 所有数据类型均可转换为 string 类型
- parseInt
 - 强制转换成整数
 - 如果不能转换，则返回 NaN (not a number)
 - 例如 parseInt("6.12") 返回 6
- parseFloat
 - 强制转换成浮点数
 - 如果不能转换，则返回 NaN
 - 例如 parseFloat("6.12") 返回 6.12




++

代码预览



数据类型转换函数（续1）

- typeof
 - 查询数值当前类型，返回 string / number / boolean / object
 - 例如 typeof("test" + 3) = "string"
- isNaN
 - 判断是否为数值





++

2.3.4. 【数据类型】特殊数据类型



| | |
|---|---|
| <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> | <div style="text-align: right;">  </div> <h3 style="text-align: center;">特殊数据类型</h3> <ul style="list-style-type: none"> • null <ul style="list-style-type: none"> - null 在程序中代表“无值”或者“无对象” - 可以通过给一个变量赋值 null 来清除变量的内容 • undefined <ul style="list-style-type: none"> - 声明了变量但未赋值或者对象属性不存在 <div style="text-align: right;">  </div> <div style="text-align: right;">知识讲解</div> <div style="text-align: right;">+</div> |
|---|---|

2.4. 运算符

2.4.1. 【运算符】算数运算

| | |
|---|---|
| <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> | <div style="text-align: right;">  </div> <h3 style="text-align: center;">算数运算</h3> <ul style="list-style-type: none"> • 加(+)、减(-)、乘(*)、除(/)、余数(%) <ul style="list-style-type: none"> - - 可以表示减号，也可以表示负号，如：x=-y - + 可以表示加法，也可以用于字符串的连接 • 递增(++)、递减(--) <ul style="list-style-type: none"> - i++ 相当于 i=i+1, i-- 相当于 i=i-1 <div style="text-align: right;">  </div> <div style="text-align: right;">知识讲解</div> <div style="text-align: right;">+</div> |
|---|---|

2.4.2. 【运算符】关系运算

| | |
|---|---|
| <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> | <div style="text-align: right;">  </div> <h3 style="text-align: center;">关系运算</h3> <ul style="list-style-type: none"> • 关系运算符用于判断数据之间的大小关系 <ul style="list-style-type: none"> - ">" (大于), "<" (小于), ">=" (大于等于), "<=" (小于等于), "==" (等于), "!=" (不等于) • 关系表达式的值为boolean类型 ("true" 或 "false") <div style="text-align: right;">  </div> <div style="text-align: right;">知识讲解</div> <div style="text-align: right;">+</div> |
|---|---|

Tarena
达内科技

关系运算 (续1)

- 全等：===
 - 类型相同
 - 数值相同
- 不全等：!==

```
var a = "10";
var b = 10;
if (a == b)
  alert("equal");
if (a === b)
  alert("same");
```

2.4.3. 【运算符】逻辑运算

Tarena
达内科技

逻辑运算

- 逻辑非 (!)、逻辑与 (&&)、逻辑或 (||)

| b1 | b2 | b1 && b2 | b1 b2 | !b1 |
|-------|-------|----------|----------|-------|
| false | false | false | false | true |
| false | true | false | true | |
| true | false | false | true | false |
| true | true | true | true | |

2.4.4. 【运算符】条件运算符

Tarena
达内科技

条件运算符

- 条件运算符又称“三目”/“三元”运算符，其结构为：
boolean表达式 ? 表达式1 : 表达式2
 - 先计算 boolean 表达式的值，如果为true，则整个表达式的值为表达式1的值
 - 如果为false，则整个表达式的值为表达式2的值

```
var n = 10;
var b = 10;
var str = b > n ? "大了" : "小了";
var result = b == n ? "相等" : str;
alert(result);
```

3. 流程控制

3.1. 控制语句

3.1.1. 【控制语句】控制语句

Tarena
达内科技

控制语句

- 任何复杂的程序逻辑都可以通过“顺序”，“分支”，“循环”三种基本的程序结构实现
 - 语句默认为顺序执行
 - 可以使用控制语句改变程序的执行顺序

Tarena
达内科技

3.2. 分支结构

3.2.1. 【分支结构】if 语句

Tarena
达内科技

if 语句

- 在运行过程中，根据不同的条件运行不同的语句

```
if (表达式) {
    // 语句块 1
}
else {
    // 语句块 2
}
```

```
if (表达式1) {
    // 语句1;
} else if (表达式2) {
    // 语句2;
} else if (表达式3) {
    // 语句3;
} else {
    // 语句4;
}
```

Tarena
达内科技

3.2.2. 【分支结构】switch-case 语句

Tarena
达内科技

switch-case 语句

- 根据一个整数表达式的不同取值，从不同的程序入口开始执行

```
switch (表达式) {
    case 值1:
        语句1;
        break;
    case 值2:
        语句2;
        break;
    default:
        语句3;
}
```

Tarena
达内科技

3.3. 循环结构

3.3.1. 【循环结构】for 语句

for 语句

for (表达式1; 表达式2; 表达式3) {
语句块 (循环体)
}

```
var sum = 0;  
for (var i=0;i<10;i++)  
{  
    sum += i;  
}  
alert(sum);
```

3.3.2. 【循环结构】while 语句

while 语句

- while 语句是前测试循环
 - 退出条件是在执行循环内部的代码之前计算的
 - 因此，循环主体可能根本不被执行

```
while (表达式) {  
    语句块  
}
```

3.3.3. 【循环结构】do-while 语句

do-while 语句


- do-while 语句是后测试循环
 - 退出条件在执行循环内部的代码之后计算
 - 在计算表达式之前，至少会执行循环主体一次

```
do  
{  
    语句块  
} while (表达式);
```

4. 常用内置对象

4.1. JavaScript 对象概述


4.1.1. 【JavaScript 对象概述】什么是 JavaScript 对象




代码清单

什么是 JavaScript 对象

- 对象是 JavaScript 中最重要的元素
- JavaScript 包含多种对象
 - 内置对象
 - 自定义对象
 - 浏览器对象
 - HTML DOM 对象
 - ...




4.1.2. 【JavaScript 对象概述】使用对象




代码清单

使用对象

- 对象由属性和方法封装而成
- 属性的引用
 - 使用点 (.) 运算符
- 对象的方法的引用
 - ObjectName.method()




4.1.3. 【JavaScript 对象概述】常用内置对象



代码清单

常用内置对象

- 简单数据对象
 - String、Number、Boolean
- 组合对象
 - Array、Math、Date
- 高级对象
 - Function、RegExp



4.2. String 对象

4.2.1. 【String 对象】String 对象

Tarena
达内科技

String 对象

- 创建字符串对象
- String 对象的属性：length

```
var str1="hello world";
var str2= new String("hello word");

alert(str1.length);
```

++

4.2.2. 【String 对象】String 对象的常用方法

Tarena
达内科技

String 对象的常用方法

- 大小写转换方法
 - x.toLowerCase()
 - x.toUpperCase()

```
var str1="AbcdEfgh";

var str2=str1.toLowerCase();
alert(str2); //结果为"abcdefgh"

var str3=str1.toUpperCase();
alert(str3); //结果为"ABCDEFGH"
```

++

Tarena
达内科技

String 对象的常用方法（续1）

- 获取指定字符
 - x.charAt(index)：返回指定位置的字符
 - x.charCodeAt(index)：返回指定位置字符的Unicode编码
- 使用注解
 - index：字符位置

```
var str1="JavaScript网页教程";

var str2=str1.charAt(12);
alert(str2); //结果为"教"

var str3=str1.charCodeAt(12);
alert(str3); //结果为25945
```

++

String 对象的常用方法 (续2)



- 查询指定字符串
 - x.indexOf(findstr, [index])
 - x.lastIndexOf(findstr, [index])
- 使用注解
 - findstr : 查找的字符串
 - index : 开始查找的位置索引, 可以省略
 - 返回 findstr 在 x 中出现的首字符位置索引, 如果没有找到, 则返回 -1
 - lastIndexOf : 从后面找起

```
var str1="JavaScript网页教程";
var str2=str1.indexOf("a");
alert(str2); //结果为1

var str3=str1.lastIndexOf("a");
alert(str3); //结果为3
```

String 对象的常用方法 (续3)



- 获取子字符串
 - x.substring(start, [end])
- 使用注解
 - start : 开始位置
 - end : 结束位置加1, 可以省略

```
var str1="abcdefgh";
var str2=str1.substring(2,4);
alert(str2); //结果为"cd"
```

String 对象的常用方法 (续4)



- 替换子字符串
 - x.replace(findstr, tostr)
- 使用注解
 - findstr : 要找的子字符串
 - tostr : 替换为的字符串
 - 返回替换后的字符串

```
var str1="abcde";
var str2=str1.replace("cd","aaa");
alert(str2); //结果为"abaae"
```


代码清单

String 对象的常用方法 (续5)

- 拆分子字符串
 - x.split(bystr, [howmany])
- 使用注解
 - bystr : 分割用的字符串
 - howmany : 指定返回的数组的最大长度, 可以省略
 - 返回分割后的字符串数组

```
var str1="一,二,三,四,五,六,日";
var strArray=str1.split(",");
alert(strArray[1]); //结果为"二"
```

Tarena
达内科技

++

4.2.3. 【String 对象】String 对象与正则表达式

代码清单

String 对象与正则表达式

- 方法
 - x.replace(regexp, tostr)
 - x.match(regexp)
 - x.search(regexp)
- 使用注解
 - regexp代表正则表达式或字符串
 - replace 返回替换后的结果
 - match 返回匹配字符串的数组
 - search 返回匹配字符串的首字符位置索引

Tarena
达内科技

++

代码清单

String 对象与正则表达式 (续1)

```
var str1 = "abc123def";
var str2 = str1.replace(/\d/gi,"*");
alert(str2); //abc***def

var array = str1.match(/\d/g);
alert(array.toString()); //1,2,3

var index = str1.search(/\d/);
alert(index); //3
```

Tarena
达内科技

++

4.3. Array 对象

4.3.1. 【Array 对象】Array 对象

Tarena
达内科技

Array 对象

- 创建数组对象


```
var cnweek = new Array(7);
var books = new Array();
```
- 初始化数组对象


```
var test = new Array(100,"a",true);
var test1 = [100,200,300];
```

```
var cnweek = new Array(7);
cnweek[0] = "星期日";
```
- 获取数组元素的个数：length 属性

+

4.3.2. 【Array 对象】创建二维数组

Tarena
达内科技

创建二维数组

- 通过指定数组中的元素为数组的方式可以创建二维甚至多维数组

| 星期日 | Sunday |
|-----|-----------|
| 星期一 | Monday |
| 星期二 | Tuesday |
| 星期三 | Wednesday |
| 星期四 | Thursday |
| 星期五 | Friday |
| 星期六 | Saturday |

```
var cnweek=new Array(7);
for (var i=0;i<=6;i++) {
    cnweek[i]=new Array(2);
}
cnweek[0][0]="星期日";
cnweek[0][1]="Sunday";
cnweek[1][0]="星期一";
cnweek[1][1]="Monday";
...
cnweek[6][0]="星期六";
cnweek[6][1]="Saturday";
```

+

4.3.3. 【Array 对象】Array 对象的常用方法

Tarena
达内科技

Array 对象的常用方法

- 数组转换为字符串
 - x.join([bystr])
 - x.toString()
- 使用注解
 - 返回连接后的字符串
 - bystr：作为连接数组中元素的字符串，可省略
 - x.toString()：由逗号(,)连接

```
var arr1=[1, 2, 3,4];
alert(arr1.toString());      //1,2,3,4
alert(arr1.join("-"));      //1-2-3-4
```


+

代码清单

Array 对象的常用方法 (续1)

- 连接数组
 - x.concat(value,...)
- 使用注解
 - value 作为数组元素连接到数组的末尾
 - 返回连接后的数组
 - concat 方法并不改变x自身的值

```
var a = [1,2,3];
var b=a.concat(4, 5);
alert(a.toString()); //1,2,3
alert(b.toString()); //1,2,3,4,5
```



+

代码清单

Array 对象的常用方法 (续2)

- 获取子数组
 - x.slice(start, [end])
- 使用注解
 - start : 开始位置索引
 - end : 结束位置加1, 省略则相当于从start位置截取以后所有数组元素

```
var arr1=['a','b','c','d','e','f','g','h'];
var arr2=arr1.slice(2,4);
alert(arr2.toString()); //c,d

var arr3=arr1.slice(4);
alert(arr3.toString()); //e,f,g,h
```



+

4.3.4. 【Array 对象】数组倒转与排序


代码清单

数组倒转与排序

- x.reverse()
 - 反向数组
 - 改变数组 x 中数值的顺序

```
var arr1=[32, 12, 111, 444];

arr1.reverse();
alert(arr1.toString()); //444,111,12,32
```



+

代码清单

数组倒转与排序（续1）

- `x.sort([sortfunc])`：数组排序
 - `sortFunction`：可选项，用来确定元素顺序的函数的名称

```

var arr1=[32, 12, 111, 444];

arr1.sort();
alert(arr1.toString());    //111,12,32,444

arr1.sort(sortFunc);
alert(arr1.toString());    //12,32,111,444

function sortFunc(a,b){
    return a-b;
}
          
```

4.4. Math 对象

4.4.1. 【Math 对象】Math 对象

代码清单

Math 对象

- Math 对象用于执行数学任务
 - 没有构造函数 `Math()`
 - 无需创建，直接把 `Math` 作为对象使用就可以调用其所有属性和方法
 - 如：`Math.PI`、`Math.round(3.56)`


4.4.2. 【Math 对象】Math 对象的常用属性和方法

代码清单

Math 对象的常用属性和方法


- 常用属性：都是数学常数
 - `Math.E`（自然数）
 - `Math.PI`（圆周率）
 - `Math.LN2`（ $\ln 2$ ）
 - `Math.LN10`（ $\ln 10$ ）等

21


| | |
|---|---|
| <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> | <div style="text-align: right;">  </div> <h3>Math 对象的常用属性和方法 (续1)</h3> <ul style="list-style-type: none"> • 三角函数 <ul style="list-style-type: none"> – Math.sin(x)、Math.cos(x)、Math.tan(x) 等 • 反三角函数 <ul style="list-style-type: none"> – Math.asin(x)、Math.acos(x) 等 • 计算函数 <ul style="list-style-type: none"> – Math.sqrt(x)、Math.log(x)、Math.exp(x)等 • 数值比较函数 <ul style="list-style-type: none"> – Math.abs(x)、Math.max(x,y,...)、Math.random()、Math.round(x)等 <div style="text-align: right;">+</div> |
|---|---|

4.5. Number 对象

4.5.1. 【Number 对象】Number 对象

| | |
|---|---|
| <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> | <div style="text-align: right;">  </div> <h3>Number 对象</h3> <ul style="list-style-type: none"> • Number 对象是原始数值的包装对象 • 创建 Number 对象 <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <pre>var myNum = 12.567; //或者 var myNum=new Number(value); //或者 var myNum=Number(value);</pre> </div> <div style="text-align: right;">+</div> |
|---|---|

4.5.2. 【Number 对象】Number 对象的常用方法

| | |
|---|---|
| <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> | <div style="text-align: right;">  </div> <h3>Number 对象的常用方法</h3> <ul style="list-style-type: none"> • toString() : 数值转换为字符串 • toFixed(num) : 数值转换为字符串, 并保留小数点后一定位数 <ul style="list-style-type: none"> – 如果必要, 该数字会被舍入, 也可以用 0 补足 <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <pre>var data = 23.56789; alert(data.toFixed(2)); //23.57 data = 23.5; alert(data.toFixed(2)); //23.50</pre> </div> <div style="text-align: right;">+</div> |
|---|---|

经典案例

1. javascript 的 HelloWorld

- 问题

创建 html 页面并为该页面添加三个按钮，该页面的显示效果如图 - 1 所示：

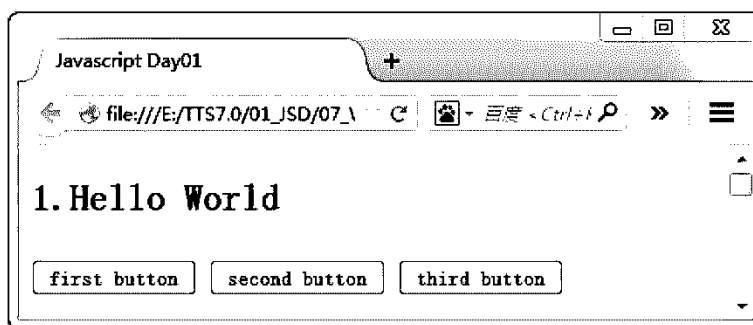


图 - 1

单击页面上的按钮“first button”，则弹出提示信息“Hello World.”；单击页面上的按钮“second button”，则弹出提示信息“Hello World in script block.”；单击页面上的按钮“third button”，则弹出提示信息“Hello World in script file.”。弹出的提示信息的效果如图 - 2 所示：



图 - 2

要求：第一个按钮的 javascript 代码直接定义在其 onclick 事件中；第二个按钮的 javascript 代码定义在 <script> 代码块中；第三个按钮的 javascript 代码定义在 js 文件中。

- 方案

为页面添加 JavaScript 代码有三种方式：

1、事件定义方式：将 JavaScript 代码写在 HTML 元素的事件中；

2、嵌入式：在页面的 <head> 中定义 <script> 标签，并在其中书写 JavaScript 代码；

3、文件调用式：JavaScript 代码位于单独的 .js 文件中，在 HTML 页面引用 js 文件。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建页面

创建文件 js_demo1.html，并在 html 页面上添加代码以创建一个标准结构的 HTML 文档。html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  </head>
  <body>
  </body>
</html>
```

步骤二：创建第一个按钮

在 <body> 元素中添加<form> 元素，然后添加第一个按钮，并为其定义单击事件的 JavaScript 代码。html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  </head>
  <body>

    <form>
      <h2>1.Hello World</h2>
      <input type="button" value="first button" onclick="alert('Hello
world.');" />
    </form>

  </body>
</html>
```

步骤三：创建第二个按钮，并添加 JavaScript 代码

在 html 页面的 <head> 元素里添加 <script> 元素，并定义方法 firstMethod。

然后在 html 页面上添加第二个按钮，并在其单击事件中调用方法 firstMethod。代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />

    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>

  </head>
  <body>
    <form>
      <h2>1.Hello World</h2>
      <input type="button" value="first button" onclick="alert('Hello
world.');" />

      <input type="button" value="second button" onclick="firstMethod();" />

    </form>
  </body>
</html>
```

步骤四：创建 js 文件

创建名称为 js_demo1.js 的文件，并在此文件中添加 javascript 代码，以定义方法 secondMethod。js_demo1.js 文件中的代码如下所示：

```
function secondMethod() {
  alert("Hello World in script file.");
}
```

步骤五：添加第三个按钮，并定义单击事件

在 html 页面上引入 js 文件，然后在 <form> 元素中添加第三个按钮，并在其单击事件中调用方法 secondMethod。代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />

    <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>

    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
```



```

        <h2>1. Hello World</h2>
        <input type="button" value="first button" onclick="alert('Hello
world.');" />
        <input          type="button"          value="second          button"
onclick="firstMethod();" />

        <input type="button" value="third button" onclick="secondMethod();" />

    </form>
</body>
</html>

```

• 完整代码

js_demo1.html 文件的代码如下所示：

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <h2>1. Hello World</h2>
      <input type="button" value="first button" onclick="alert('Hello
world.');" />
      <input          type="button"          value="second          button"
onclick="firstMethod();" />
      <input          type="button"          value="third          button"
onclick="secondMethod();" />
    </form>
  </body>
</html>

```

js_demo1.js 文件的代码如下所示：

```

function secondMethod() {
  alert("Hello World in script file.");
}

```

2. 判断数据类型，并计算平方

• 问题

有页面，可以由用户录入数据，页面效果如图 - 3 所示：

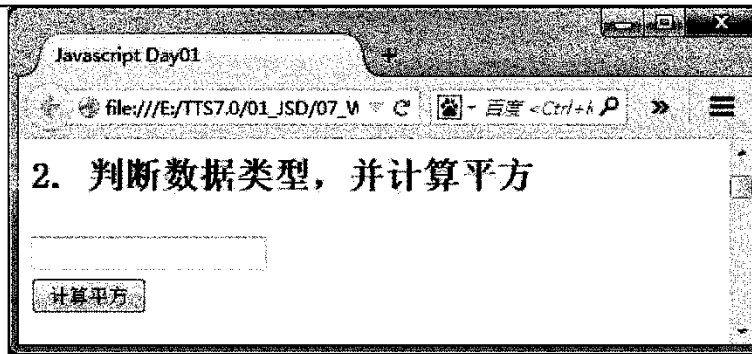


图 - 3

用户在图 - 3 所示的页面上的文本框中录入数据 ,然后单击页面上的按钮“计算平方”。在按钮的单击事件中,首先需要判断文本框中录入的文本是否为数值,如果录入的文本不能转换为数值,则提示用户重新录入。页面效果如图 - 4 所示:



图 - 4

如果录入的文本可以转换为数值,则计算录入数值的平方,并弹出计算结果。页面效果如图 - 5 所示:

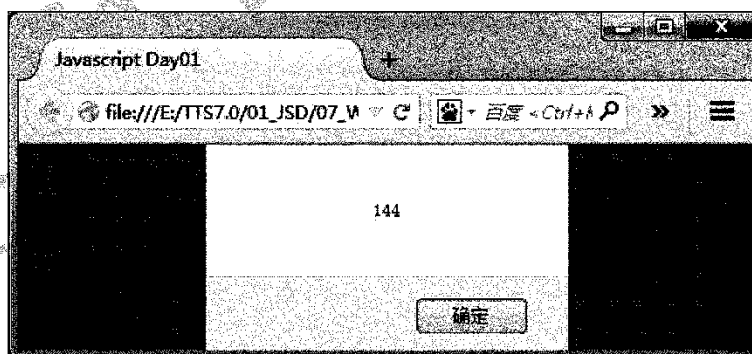


图 - 5

• 方案

实现此案例时,首先需要得到界面上文本框中录入的文本,然后使用 `isNaN` 函数判断该文本是否可以转换为数值,如果不能转换,则提示信息,如果可以转换,则使用 `parseFloat` 函数转换为数值,并计算平方,最后显示计算结果。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加文本框和按钮

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加文本框和按钮，并为文本框定义 id 属性，同时为按钮定义单击事件。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <h2>1.Hello World</h2>
      <input type="button" value="first button" onclick="alert('Hello
world.');" />
      <input type="button" value="second button"
onclick="firstMethod();" />
      <input type="button" value="third button"
onclick="secondMethod();" />

      <h2>2.判断数据类型，并计算平方</h2>
      <input type="text" id="txtData" /><br />
      <input type="button" value="计算平方" onclick="getSquare();" />

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义方法

打开文件 js_demo1.js，在其中添加代码，定义名为 getSquare 的方法，js 文件中添加的代码如下所示：

```
//转换函数：计算录入数值的平方
function getSquare() {
}
```

步骤三：获取文本框中的数据

在方法 getSquare 中添加代码，使用 document.getElementById 方法得到文本框对象，并使用 value 属性得到文本框的文本，代码如下所示：

```
//转换函数：计算录入数值的平方
function getSquare() {

    var str = document.getElementById("txtData").value;

}
```

步骤四：判断数据类型

得到文本后，首先使用 `isNaN` 方法判断文本框中的文本是否为数值。

该方法如果返回 `true`，则表示非数值，需要使用 `alert` 方法弹出提示信息“请录入数值”；如果该方法返回 `false`，则表示为数值，然后使用 `parseFloat` 方法将文本转换为数值，并计算平方后使用 `alert` 方法弹出计算结果。代码如下所示：

```
//转换函数：计算录入数值的平方
function getSquare() {
    var str = document.getElementById("txtData").value;

    if (isNaN(str))
        alert("请录入数值");
    else {
        var data = parseFloat(str);
        var result = data * data;
        alert(result);
    }

}
```

• 完整代码

`js_demo1.html` 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Javascript Day01</title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
<script language="javascript" type="text/javascript">
    function firstMethod() {
        alert("Hello World in script block.");
    }
</script>
</head>
<body>
<form>
<h2>1.Hello World</h2>
<input type="button" value="first button" onclick="alert('Hello
world.');" />
<input type="button" value="second button"
onclick="firstMethod();" />
<input type="button" value="third button"
onclick="secondMethod();" />
<h2>2.判断数据类型，并计算平方</h2>
<input type="text" id="txtData" /><br />
```

```
<input type="button" value="计算平方" onclick="getSquare();" />
</form>
</body>
</html>
```

js_demo1.js 文件的代码如下所示：

```
//js 文件中的代码
function secondMethod() {
    alert("Hello World in script file.");
}
//转换函数：计算录入数值的平方
function getSquare() {
    var str = document.getElementById("txtData").value;
    if (isNaN(str)) {
        alert("请录入数值");
    } else {
        var data = parseFloat(str);
        var result = data * data;
        alert(result);
    }
}
```

3. 猜数字

- 问题

实现猜数字的游戏功能。程序内置好一个数值作为正确结果（比如数值 10），并在页面显示一个文本框，由用户录入所猜测的数字，页面效果如图 - 6 所示：

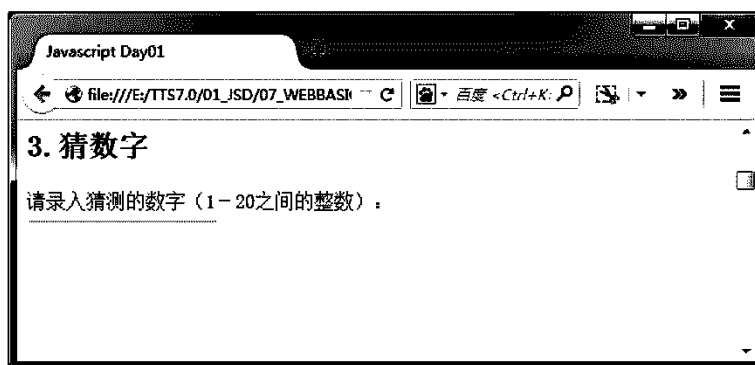


图 - 6

用户录入完毕后，如果录入的文本不能转换为数值，则提示用户重新录入，页面效果如图 - 7 所示：



图 - 7

如果可以转换为数值，则与内置好的数值进行比较，并提示比较的结果（“小了”、“大了”或者“猜对了”）。页面效果如图 - 8 所示：

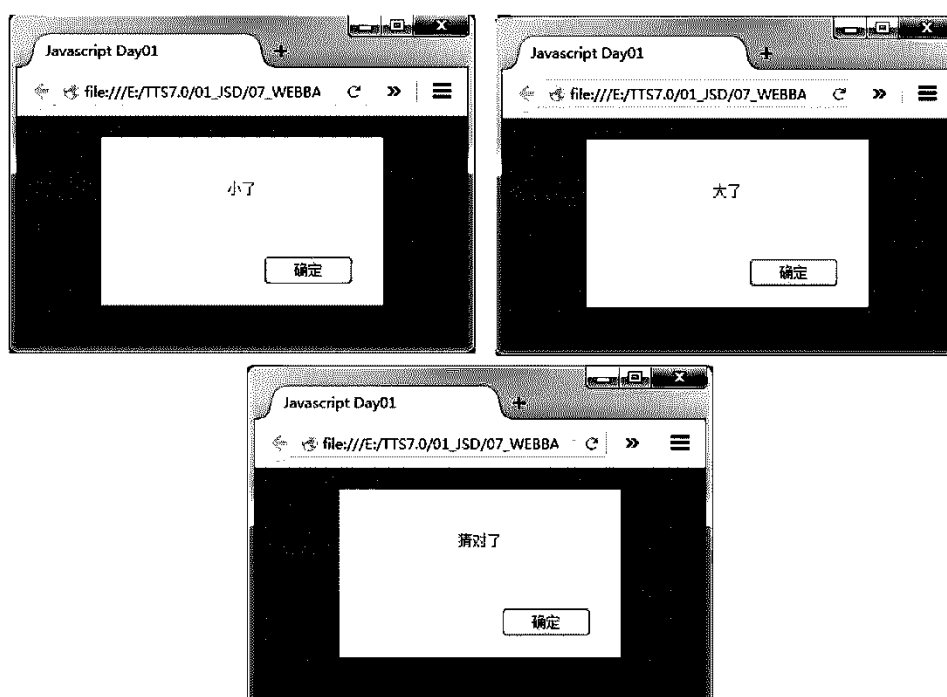


图 - 8

• 方案

本案例中要求，一旦录入完毕，立即判断用户的录入，因此，需要为文本框定义 `onblur` 事件，并在事件中添加代码实现功能。

使用条件运算符来实现比较。条件运算符又称“三目” / “三元”运算符，其结构为：

`boolean 表达式 ? 表达式 1 : 表达式 2 ;`

先计算 `boolean` 表达式的值，如果为 `true`，则整个表达式的值为表达式 1 的值；如果为 `false`，则整个表达式的值为表达式 2 的值。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：为页面添加文本框和按钮

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加文本框，并为文本框定义 id 属性，同时为文本框定义失去焦点事件。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>3.猜数字</h2>
      请录入猜测的数字 ( 1 - 20 之间的整数 ): <br />
      <input id="txtNumber" type="text" onblur="guessNumber();" />

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义方法

打开文件 js_demo1.js，在其中添加代码，定义名为 guessNumber 的方法，js 文件中添加的代码如下所示：

```
//猜数字
function guessNumber() {
}
```

步骤三：预置结果并获取文本框中的数据

在 guessNumber 方法中，首先预定义好一个用于比较判断的数值（比如 10），并获取文本框中的数据。代码如下所示：

```
//猜数字
function guessNumber() {
```

//内置结果

```
var result = 10;
//得到用户的录入
var str = document.getElementById("txtNumber").value;

}
```

步骤四：判断输入类型

使用 isNaN 方法判断文本框中的文本是否为数值。该方法如果返回 true，则表示非数值，需要使用 alert 方法弹出提示信息“请录入数值”；如果该方法返回 false，则表示为数值，然后使用 parseFloat 方法将文本框中的文本转换为数值。代码如下所示：

```
//猜数字
function guessNumber() {
    //内置结果
    var result = 10;
    //得到用户的录入
    var str = document.getElementById("txtNumber").value;

    //比较
    if (isNaN(str))
        alert("请录入数值");
    else {
        var data = parseFloat(str);
    }
}
```

步骤五：比较数值的大小

将录入的文本转换为数值之后，使用三元运算符将录入的数值与预置好的数值进行比较，并使用 alert 方法弹出比较结果。代码如下所示：

```
//猜数字
function guessNumber() {
    //内置结果
    var result = 10;
    //得到用户的录入
    var str = document.getElementById("txtNumber").value;
    //比较
    if (isNaN(str))
        alert("请录入数值");
    else {
        var data = parseFloat(str);

        var info = data > result ? "大了" : "小了";
        info = data == result ? "猜对了" : info;
        alert(info);
    }
}
```


- 完整代码

js_demo1.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->
      <h2>3.猜数字</h2>
      请录入猜测的数字（1-20 之间的整数）: <br />
      <input id="txtNumber" type="text" onblur="guessNumber();" />
    </form>
  </body>
</html>
```

js_demo1.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//猜数字
function guessNumber() {
  //内置结果
  var result = 10;
  //得到用户的录入
  var str = document.getElementById("txtNumber").value;
  //比较
  if (isNaN(str))
    alert("请录入数值");
  else {
    var data = parseFloat(str);
    var info = data > result ? "大了" : "小了";
    info = data == result ? "猜对了" : info;
    alert(info);
  }
}
```

4. 计算阶乘

- 问题

计算阶乘。为 html 页面添加一个按钮，单击该按钮，则计算 10 的阶乘，并弹出显示计算结果。页面效果如图 - 9 所示：



图 - 9

• 方案

此案例中，需要使用循环语句计算阶乘。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加按钮

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加按钮，并为按钮定义单击事件。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>4.控制语句：求阶乘</h2>
      <input type="button" value="求 10 的阶乘" onclick="getFac();" />

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义方法

打开文件 js_demo1.js，在其中添加代码，定义名为 getFac 的方法，js 文件中添加的代码如下所示：

```
//求阶乘
function getFac() {
}
```

步骤三：计算阶乘

在 getFac 方法中，使用 for 循环语句计算 10 的阶乘。代码如下所示：

```
//求阶乘
function getFac() {

    var result = 1;
    for (var i = 1; i <= 10; i++) {
        result *= i;
    }
    alert("10的阶乘为：" + result);

}
```

注意：for 语句中，使用 var 关键字定义变量 i，而不是 int 关键字。

• 完整代码

js_demo1.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->
      <h2>4.控制语句：求阶乘</h2>
      <input type="button" value="求 10 的阶乘" onclick="getFac();" />
    </form>
  </body>
</html>
```

js_demo1.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//求阶乘
function getFac() {
    var result = 1;
    for (var i = 1; i <= 10; i++) {
        result *= i;
    }
    alert("10的阶乘为: " + result);
}
```

5. 字符的查询与过滤

- 问题

页面上有文本框，用户可以在文本框中录入文本，页面效果如图 - 10 所示：



图 - 10

录入完毕，并单击图 - 10 所示页面上的按钮后，则过滤文本中的字符串 js，并将其替换为*，页面效果如图 - 11 所示：



图 - 11

• 方案

此案例中，需要使用 string 对象的 indexOf 方法判断子字符串 js 出现的位置，然后使用 string 对象的 replace 方法进行替换。因为文本中可能有多个 js，上述的查找和替换过程需要重复进行，直到文本中没有子字符串 js 为止。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加文本框和按钮

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加文本框和按钮。为了得到文本框中录入的文本，为其定义 id 属性，并为按钮定义单击事件。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>5.String 对象：字符的查找与过滤</h2>
      <input type="text" id="txtString" /><br />
      <input type="button" value="过滤特殊字符 ( js )"
        onclick="searchStringAndReplace();" />

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义方法

打开文件 js_demo1.js，在其中添加代码，定义名为 searchStringAndReplace 的方法，js 文件中添加的代码如下所示：

```
//查找并替换文本框中录入的子字符串 js 为 *
function searchStringAndReplace() {
}
```

步骤三：获取文本框中的文本并查找第一个子字符串 js

在 `searchStringAndReplace` 方法中，首先获得文本框中的文本，并查找第一个子字符串 `js`。代码如下所示：

```
//查找并替换文本框中录入的子字符串 js 为 *
function searchStringAndReplace() {

    var str = document.getElementById("txtString").value;

    var index = str.indexOf("js", 0);

}
```

步骤四：替换

如果找到了子字符串 `js`，则将其替换为 `*`，并进行下一次查询，直到全部替换完毕为止。最后，将替换后的结果显示在文本框中。代码如下所示：

```
//查找并替换文本框中录入的子字符串 js 为 *
function searchStringAndReplace() {
    var str = document.getElementById("txtString").value;

    var index = str.indexOf("js", 0);

    while (index > -1) {
        str = str.replace("js","*");
        index = str.indexOf("js", index + 1);
    }
    document.getElementById("txtString").value = str;
}
```

• 完整代码

`js_demo1.html` 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->
      <h2>5.String 对象：字符的查找与过滤</h2>
      <input type="text" id="txtString" /><br />
      <input type="button" value="过滤特殊字符(js)"
        onclick="searchStringAndReplace();" />
    </form>
  </body>
</html>
```

```
</form>
</body>
</html>
```

js_demo1.js 文件中的代码如下所示：

```
//其他案例的代码部分略

//查找并替换文本框中录入的子字符串 js 为 *
function searchStringAndReplace() {
    var str = document.getElementById("txtString").value;

    var index = str.indexOf("js", 0);
    while (index > -1) {
        str = str.replace("js", "*");
        index = str.indexOf("js", index + 1);
    }
    document.getElementById("txtString").value = str;
}
```

6. 字符的查询与过滤（使用正则表达式）

• 问题

此案例的要求和上一个案例相似，依然要求查询并替换页面文本框中的子字符串 js，要求使用正则表达式来实现。此次替换，要求忽略大小写，即，忽略子字符串 js 的大小写方式，一律替换为*。

用户可以在文本框中录入文本，页面效果如图 - 12 所示：



图 - 12

由图 - 12 可以看出，用户在文本框中录入了多个子字符串 js，且各种大小写混排。单击按钮“查找字符并过滤（使用正则表达式）”后，统计录入文本中子字符串“js”的数量（忽略 js 的大小写），并统一替换为 * 并显示在文本框中，然后弹出替换的统计结果。页面交互效果如图 - 13 所示：



图 - 13

• 方案

可以定义正则表达式 `"/js/gi"`，表示匹配字符串 `js`，不区分大小写，且实现全局匹配。然后，将正则表达式与 `string` 对象的 `match` 和 `replace` 方法结合使用，以实现子字符串的搜索和匹配。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加按钮

在上一个案例的 `html` 页面上继续添加此案例的内容。

首先，在 `<form>` 标记中添加按钮，并为其定义单击事件。修改后的 `html` 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js/demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <!-- 其他部分代码，略-->
      <h2>5.String 对象：字符的查找与过滤</h2>
      <input type="text" id="txtString" /><br />
      <input type="button" value="过滤特殊字符 ( js )"
        onclick="searchStringAndReplace();" />
    </form>
  </body>
</html>
```



```
<input type="button" value="查找字符并过滤 ( 使用正则表达式 )"
      onclick="stringByRegex();" />
```

```
</form>
</body>
</html>
```

步骤二：在 js 文件中定义方法

打开文件 js_demo1.js，在其中添加代码，定义名为 stringByRegex 的方法，js 文件中添加的代码如下所示：

```
//使用正则表达式操作文本
function stringByRegex() {
}
```

步骤三：获取文本框中的文本并查找子字符串 js

在 stringByRegex 方法中，首先获得文本框中的文本，然后调用 string 对象的 match 方法，并传入正则表达式对象，得到查找结果。代码如下所示：

```
//使用正则表达式操作文本
function stringByRegex() {

    var str = document.getElementById("txtString").value;
    var result = str.match(/js/gi);

}
```

步骤四：替换

调用 string 对象的 replace 方法，并传入正则表达式对象，最后将替换后的结果显示在文本框中。代码如下所示：

```
//使用正则表达式操作文本
function stringByRegex() {
    var str = document.getElementById("txtString").value;
    var result = str.match(/js/gi);

    document.getElementById("txtString").value = str.replace(/js/gi, "");
    alert("共替换了" + result.length + "处。");

}
```

• 完整代码

js_demo1.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
  <title>Javascript Day01</title>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
  <script language="javascript" type="text/javascript">
    function firstMethod() {
      alert("Hello World in script block.");
    }
  </script>
</head>
<body>
  <form>
    <!--其他部分代码，略-->
    <h2>5.String 对象：字符的查找与过滤</h2>
    <input type="text" id="txtString" /><br />
    <input type="button" value="过滤特殊字符 (js)"
      onclick="searchStringAndReplace();" />
    <input type="button" value="查找字符并过滤（使用正则表达式）"
      onclick="stringByRegex();" />
  </form>
</body>
</html>
```

js_demo1.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//使用正则表达式操作文本
function stringByRegex() {
  var str = document.getElementById("txtString").value;
  var result = str.match(/js/gi);
  document.getElementById("txtString").value = str.replace(/js/gi, "");
  alert("共替换了" + result.length + "处。");
}
```

7. 数组的倒转与排序

• 问题

用户在文本框中录入逗号分隔的数值文本，页面效果如图 - 14 所示：

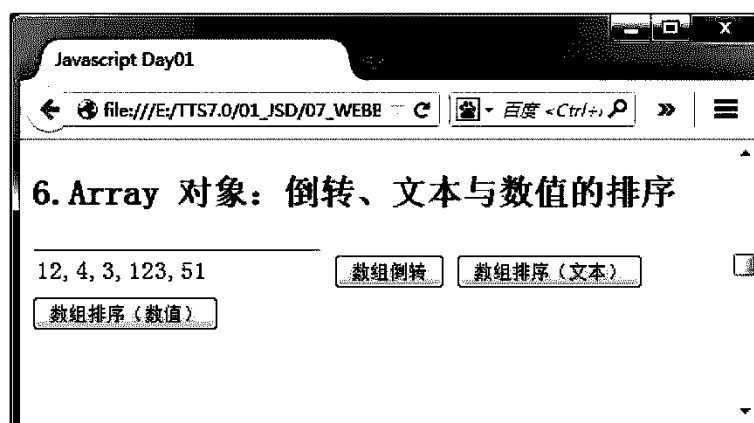


图 - 14

单击“数组倒转”按钮后，将文本框中的文本按照逗号分隔为数组，然后将数组倒转，并弹出倒转后的结果。页面交互效果如图 - 15 所示：

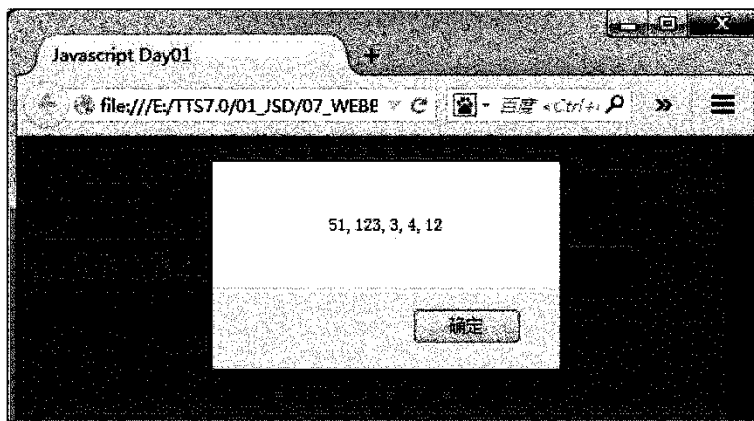


图 - 15

单击“数组排序（文本）”按钮后，将文本框中的文本按照逗号分隔为数组，并对数组按照文本比较的方式进行排序，并弹出排序后的结果。页面交互效果如图 - 16 所示：

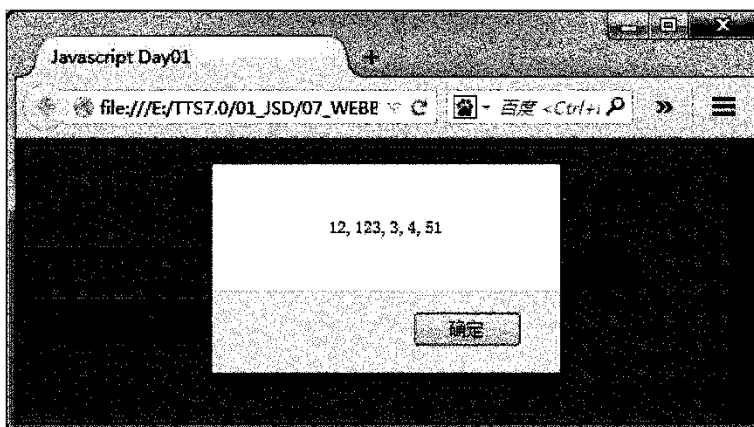


图 - 16

单击“数组排序（数值）”按钮后，将文本框中的文本按照逗号分隔为数组，然后对数组按照数值比较的方式进行排序，并弹出排序后的结果。页面交互效果如图 - 17 所示：

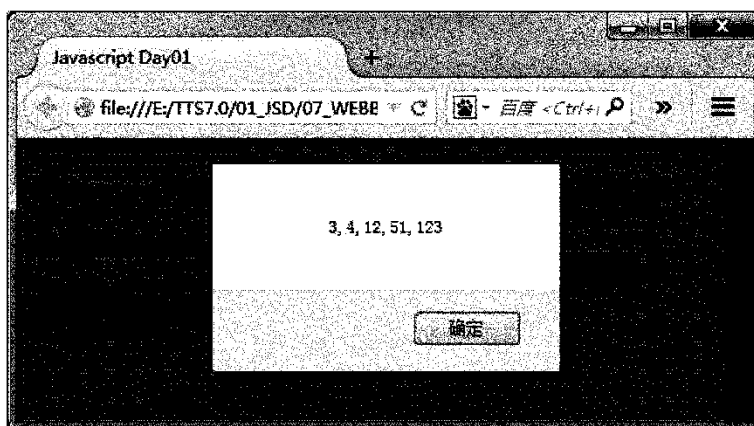


图 - 17

• 方案

实现此案例的大致步骤如下：

- 1、得到文本框中录入的文本；
- 2、将文本拆分为字符数组；
- 3、调用 Array 对象的 revert 方法实现数组的倒转；
- 4、调用 Array 对象的 sort 方法实现数组的排序（按照文本）；
- 5、调用 Array 对象的 sort 方法，并传入自定义的比较函数，对数组实现按数值排序。

因为三个按钮所实现的功能类似，可以调用同一个方法，传入不同的参数值，表示不同的操作。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加文本框和按钮

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加文本框和按钮，为文本框定义初始值和 id 属性，并分别为按钮定义单击事件。

三个按钮调用同一个方法 operateArray，分别传入不同的参数值，其中：传入参数值 1，表示实现数组倒转；参数值 2 表示按照文本排序；参数值 3 表示按照数值排序。

修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js/demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>6.Array 对象：倒转、文本与数值的排序</h2>
      <input type="text" id="txtNumbers" value="12,4,3,123,51"
style="font-size:13pt;"/>
      <input type="button" value="数组倒转" onclick="operateArray(1);" />
      <input type="button" value="数组排序（文本）" onclick="operateArray(2);"
/>

      <input type="button" value="数组排序（数值）" onclick="operateArray(3);"
```

/>

```
        </form>
    </body>
</html>
```

步骤二：在 js 文件中定义方法

打开文件 js_demo1.js，在其中添加代码，定义名为 operateArray 的方法，js 文件中添加的代码如下所示：

```
//数组操作
function operateArray(t) {
}
```

步骤三：获取数组，并判断操作类型

在 operateArray 方法中，首先获得文本框中的文本，并调用 string 对象的 split 方法，拆分为数组。然后根据所传入的参数数值判断操作类型。代码如下所示：

```
//数组操作
function operateArray(t) {

    //拆分为数组
    var array = document.getElementById("txtNumbers").value.split(",");
    //判断操作类型
    switch (t) {
        case 1:
            break;
        case 2:
            break;
        case 3:
            break;
    }

}
```

步骤四：倒转和排序（按照文本排序）

分别调用 string 对象的 revert 方法和 sort 方法，实现数组的倒转和默认排序（按照文本排序），并弹出数组的内容显示操作结果。代码如下所示：

```
//数组操作
function operateArray(t) {
    //拆分为数组
    var array = document.getElementById("txtNumbers").value.split(",");
    //判断操作类型
    switch (t) {
        case 1:

            array.reverse();
```

```

        break;
    case 2:

        array.sort();

        break;
    case 3:
        break;
    }

    alert(array.toString());
}

```

步骤五：按照数值对数组排序

定义比较规则的函数,并将该函数传入 Array 对象的 sort 方法,实现按照数值排序。
代码如下所示：

```

//数组操作
function operateArray(t) {
    //拆分为数组
    var array = document.getElementById("txtNumbers").value.split(",");
    //判断操作类型
    switch (t) {
        case 1:
            array.reverse();
            break;
        case 2:
            array.sort();
            break;
        case 3:

            array.sort(sortFunc);

            break;
    }
    alert(array.toString());
}

//自定义排序规则函数
function sortFunc(a, b) {
    return a - b;
}

```

• 完整代码

js_demo1.html 文件的代码如下所示：

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo1.js"

```

```
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
        function firstMethod() {
            alert("Hello World in script block.");
        }
    </script>
</head>
<body>
    <form>
        <!--其他部分代码，略-->
        <h2>6.Array 对象：倒转、文本与数值的排序</h2>
        <input type="text" id="txtNumbers" value="12,4,3,123,51"
style="font-size:13pt;"/>
        <input type="button" value="数组倒转" onclick="operateArray(1);" />
        <input type="button" value=" 数 组 排 序 （ 文 本 ） "
onclick="operateArray(2);" />
        <input type="button" value=" 数 组 排 序 （ 数 值 ） "
onclick="operateArray(3);" />
    </form>
</body>
</html>
```

js_demo1.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//数组操作
function operateArray(t) {
    //拆分为数组
    var array = document.getElementById("txtNumbers").value.split(",");
    //判断操作类型
    switch (t) {
        case 1:
            array.reverse();
            break;
        case 2:
            array.sort();
            break;
        case 3:
            array.sort(sortFunc);
            break;
    }
    alert(array.toString());
}
//自定义排序规则函数
function sortFunc(a, b) {
    return a - b;
}
```

8. 随机数生成器

• 问题

用户在界面录入最小值和最大值，然后单击按钮，可以获得该范围内的一个随机整数，页面效果如图 - 18 所示：

注：所得到的随机数包含下限（最小值），不包含上限（最大值）。

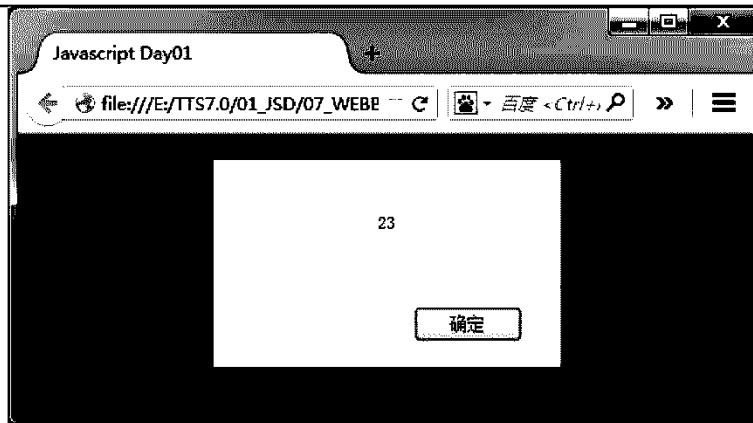


图 - 18

• 方案

Math 对象的 random 方法可以产生一个随机的小数，即随机产生一个大于等于 0，小于 1 的数值。如果要将此随机数换算到所需的数值范围，则首先需要使用如下公式进行第一步换算（得到最大值和最小值差值范围之间的一个随机数）：

随机小数 * (max - min)

上述公式可以得到差值范围内的一个随机数值，但是可能包含小数部分，因此，还需要使用 Math 对象的 floor 方法转换为整数数值（含下限，不包含上限），换算公式为：

Math.floor(随机小数 * (max - min))

最后，加上最小值，即可实现数值范围的换算，完整公式为：

Math.floor(随机小数 * (max - min)) + min

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加文本框和按钮

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加文本框和按钮，为文本框定义 id 属性，并为按钮定义单击事件。

修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
```



```
<script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
<script language="javascript" type="text/javascript">
    function firstMethod() {
        alert("Hello World in script block.");
    }
</script>
</head>
<body>
    <form>
        <!--其他部分代码，略-->

        <h2>7.Math 对象：随机数生成器</h2>
        最小值：<input type="text" id="txtMin" /><br />
        最大值：<input type="text" id="txtMax" /><br />
        <input type="button" value="得到随机数" onclick="getRandomNumber();" />

    </form>
</body>
</html>
```

步骤二：在 js 文件中定义方法

打开文件 js_demo1.js，在其中添加代码，定义名为 getRandomNumber 的方法，js 文件中添加的代码如下所示：

```
//得到随机数
function getRandomNumber() {
}
```

步骤三：获取最大值和最小值

在 getRandomNumber 方法中，首先获得文本框中录入的最大值和最小值，并转换为整数数值。代码如下所示：

```
//得到随机数
function getRandomNumber() {

    var min = parseInt(document.getElementById("txtMin").value);
    var max = parseInt(document.getElementById("txtMax").value);

}
```

注：这里省略了对录入的文本是否为数值的判断。

步骤四：创建随机数

调用 Math 对象的 random 方法产生一个随机小数，并进行换算，以得到需要的结果。代码如下所示：

```
//数组操作
//得到随机数
```

```
function getRandomNumber() {
    var min = parseInt(document.getElementById("txtMin").value);
    var max = parseInt(document.getElementById("txtMax").value);

    var ran = Math.random();
    var data = Math.floor(ran * (max - min)) + min;
    alert(data);
}
```

• 完整代码

js_demo1.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->
      <h2>7.Math 对象：随机数生成器</h2>
      最小值：<input type="text" id="txtMin" /><br />
      最大值：<input type="text" id="txtMax" /><br />
      <input type="button" value="得到随机数"
onclick="getRandomNumber();" />
    </form>
  </body>
</html>
```

js_demo1.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//得到随机数
function getRandomNumber() {
    var min = parseInt(document.getElementById("txtMin").value);
    var max = parseInt(document.getElementById("txtMax").value);
    var ran = Math.random();
    var data = Math.floor(ran * (max - min)) + min;
    alert(data);
}
```

9. 资产折旧计算器

• 问题

制作页面版的资产折旧计算器。

用户在页面上录入资产原价、折旧率以及计算年限，单击“计算”按钮后，计算该资产的折旧价值并显示在页面上。页面效果如图 - 19 所示：

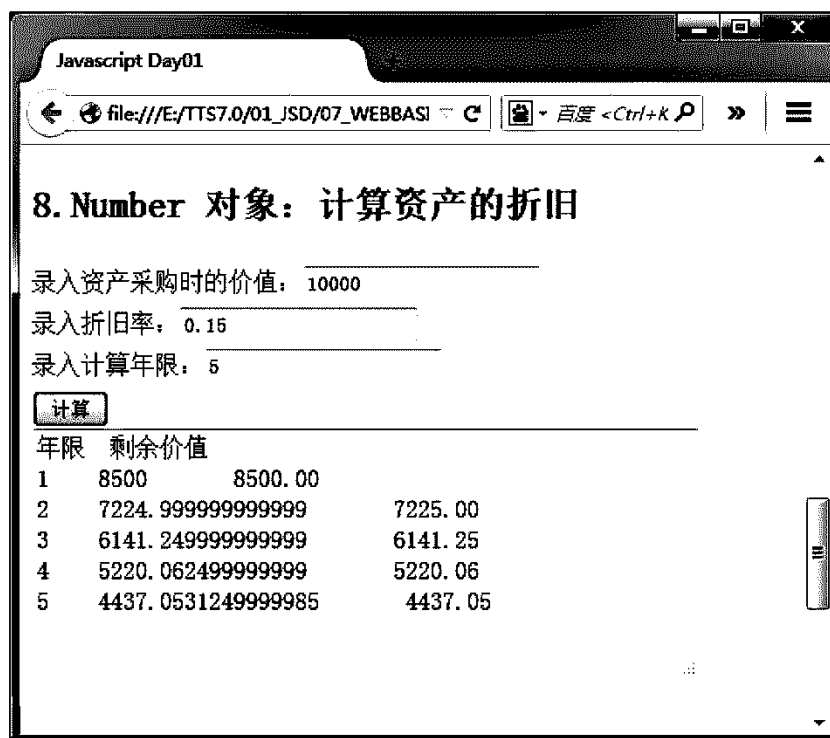


图 - 19

由图 - 19 可以看出，显示结果一定要求显示三项数据：第一列显示年限 n ，第二列和第三列显示 n 年后资产的剩余价值。其中，第二列显示计算后的结果，第三列显示保留小数点后两位的结果。

资产折旧值的计算公式为：

第 n 年的剩余价值 = 资产原价 \times (1-折旧率) 的 n 次方

由此可见，对于图 - 19 中所录入的数值，第 1 年后的资产剩余价值为： $10000 \times (1-0.15)=8500$ ；第 2 年后的资产价值为： $10000 \times (1-0.15) \times (1-0.15) = 7225$ ；以此类推。

• 方案

Math 对象的 pow 方法用于计算数值的 n 次幂，在此案例中，可以使用此方法计算 n 年后的资产剩余价值。

另外，需要注意的是，js 中，数值类型为浮点类型，其存储的是数值的近似值。比如， $10000 \times (1-0.15) \times (1-0.15)$ 计算后，结果应该为 7225，但是，实际存储的数值可能是 7224.999999999999。因此，在实际使用中，需要使用 Number 对象的 toFixed 方法进行小数位数的取舍操作。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加文本框、多行文本框和按钮

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加页面元素，为各文本框定义 id 属性并设置初始值，然后为按钮定义单击事件。

修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>8.Number 对象：计算资产的折旧</h2>
      录入资产采购时的价值：<input id="txtPrice" type="text" value="10000" /><br
/>

      录入折旧率：<input id="txtRate" type="text" value="0.15" /><br />
      录入计算年限：<input id="txtYear" type="text" value="5" /><br />
      <input type="button" value=" 计 算 " onclick="calculateDepreciation();"
/><br/>

      <textarea id="txtResult" cols="50" rows="7"></textarea>

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义方法

打开文件 js_demo1.js，在其中添加代码，定义名为 calculateDepreciation 的方法，js 文件中添加的代码如下所示：

```
//计算资产折旧
function calculateDepreciation() {
}
```

步骤三：获取各项数值

在 calculateDepreciation 方法中，首先获得文本框中录入的价值、年限和折旧率，

并转换为数值类型。代码如下所示：

```
//计算资产折旧
function calculateDepreciation() {

    //得到录入的各项数值
    var money = parseFloat(document.getElementById("txtPrice").value);
    var rate = parseFloat(document.getElementById("txtRate").value);
    var year = parseInt(document.getElementById("txtYear").value);

}
```

注：这里省略了对录入的文本是否为数值的判断。

步骤四：计算

构建循环，以计算每年后的资产剩余价值。代码如下所示：

```
//计算资产折旧
function calculateDepreciation() {
    //得到录入的各项数值
    var money = parseFloat(document.getElementById("txtPrice").value);
    var rate = parseFloat(document.getElementById("txtRate").value);
    var year = parseInt(document.getElementById("txtYear").value);

    //拼接结果字符串
    var s = "年限  剩余价值\n";
    //循环计算每年的折旧
    for (var i = 1; i <= year; i++) {
        var r = money * Math.pow((1 - rate), i);
    }

}
```

步骤五：拼接显示结果

拼接计算结果，并显示在多行文本框中。代码如下所示：

```
//计算资产折旧
function calculateDepreciation() {
    //得到录入的各项数值
    var money = parseFloat(document.getElementById("txtPrice").value);
    var rate = parseFloat(document.getElementById("txtRate").value);
    var year = parseInt(document.getElementById("txtYear").value);

    //拼接结果字符串
    var s = "年限  剩余价值\n";
    //循环计算每年的折旧
    for (var i = 1; i <= year; i++) {
        var r = money * Math.pow((1 - rate), i);
    }
}
```

//每年的结果显示为一行

```
s += i + "    " + r + "    " + r.toFixed(2) + "\n";
```

```
}
```

```
document.getElementById("txtResult").innerHTML = s;
```

```
}
```

• 完整代码

js_demo1.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day01</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo1.js"
type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
      function firstMethod() {
        alert("Hello World in script block.");
      }
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->
      <h2>8.Number 对象：计算资产的折旧</h2>
      录入资产采购时的价值： <input id="txtPrice" type="text" value="10000"
/><br />
      录入折旧率： <input id="txtRate" type="text" value="0.15" /><br />
      录入计算年限： <input id="txtYear" type="text" value="5" /><br />
      <input type="button" value="    计    算    "
onclick="calculateDepreciation();" /><br />
      <textarea id="txtResult" cols="50" rows="7"></textarea>
    </form>
  </body>
</html>
```

js_demo1.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//计算资产折旧
function calculateDepreciation() {
  //得到录入的各项数值
  var money = parseFloat(document.getElementById("txtPrice").value);
  var rate = parseFloat(document.getElementById("txtRate").value);
  var year = parseInt(document.getElementById("txtYear").value);

  //拼接结果字符串
  var s = "年限  剩余价值\n";
  //循环计算每年的折旧
  for (var i = 1; i <= year; i++) {
    var r = money * Math.pow((1 - rate), i);
    //每年的结果显示为一行
```

```
        s += i + "    " + r + "    " + r.toFixed(2) + "\n";  
    }  
    document.getElementById("txtResult").innerHTML = s;  
}
```

课后作业

1. 简要描述 JavaScript 的数据类型。
2. JavaScript 中，运算符 `==` 和 `===` 的区别是什么？
3. “百钱买百鸡”问题。

计算“百钱买百鸡”的问题：公鸡 5 文钱一只，母鸡 3 文钱一只，小鸡 1 文钱 3 只，如何用一百文钱买到一百只鸡？

单击页面上的按钮，则显示“百钱买百鸡的方案”。页面效果如图 - 1 所示：

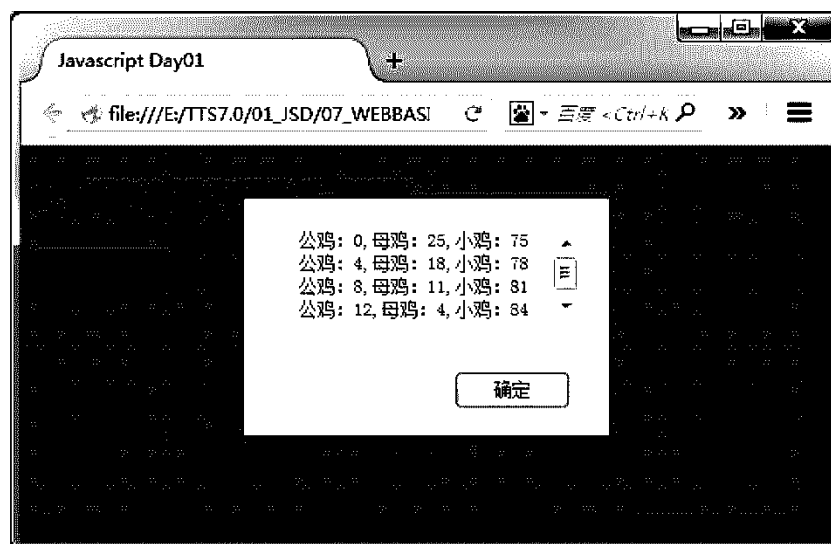


图 - 1

4. Array 对象的 join 方法和 concat 方法的区别是什么？
5. 实现双色球彩票生成器。

首先实现如图 - 2 所示的页面：

2. 彩票双色球生成器

机选

图 - 2

单击页面上的“机选”按钮，则生成一注双色球彩票号码，并弹出显示结果，效果如图

- 3 所示：

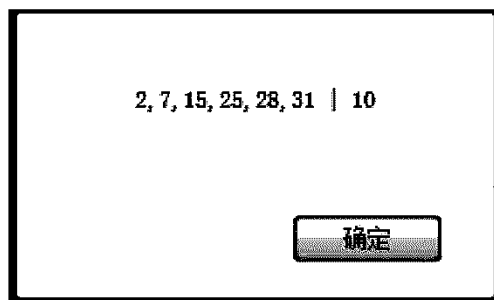


图 - 3

其中，双色球彩票号码的规则为：

- 1、红色球：从“01”到“33”中（包含 1 和 33），随机选择出 6 个数字作为红色球的号码，且这 6 个数字不能重复；
- 2、蓝色球：从“01”到“16”中（包含 1 和 16），随机选择一个数字作为蓝色球；
- 3、7 个数字合到一起作为一注双色球彩票的号码。

图 - 3 中显示的彩票号码中，前 6 个数字为红色球号码（要求按照数值大小升序排列），竖线后的数字为蓝色球号码。

6. 统计文本框中录入的各字符的个数。（提高题，选做）

首先实现如图 - 4 所示的页面：



图 - 4

在图 - 4 中的文本框中录入字符后，录入完毕，则统计文本框中录入的各字符的个数，并弹出显示，效果如图 - 5 所示：

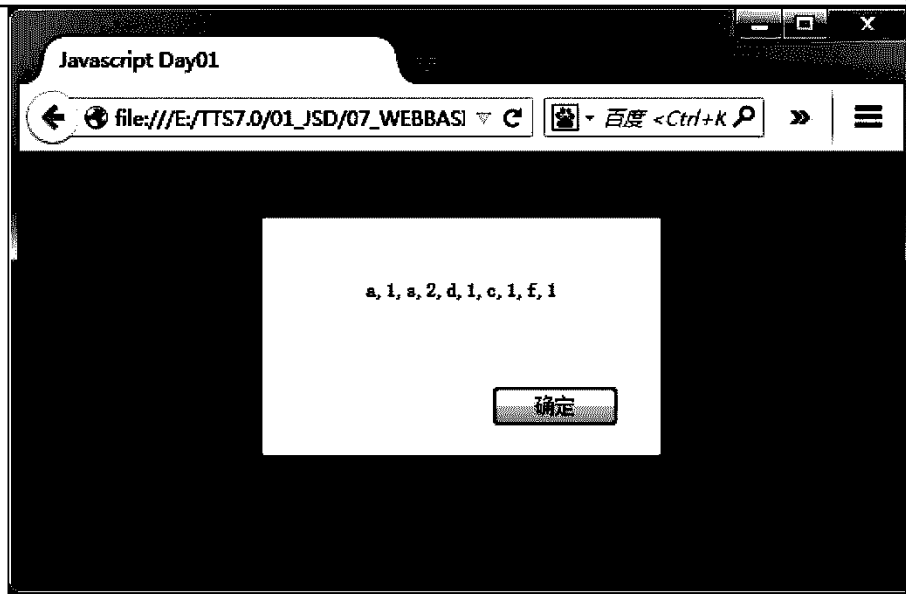


图 - 5

JavaScript

Unit02

知识体系.....Page 62

常用内置对象	正则表达式对象	RegExp 对象概述
		RegExp 对象的常用方法
	Date 对象	Date 对象
		Date 对象的常用方法
	Function 对象	函数与 Function 对象
		函数的定义
		函数的调用
		arguments 对象
		使用 Function 对象创建函数
		匿名函数
	全局函数	全局函数概述
		encodeURIComponent 与 decodeURI
		eval 函数
window 对象	DHTML 概述	DHTML 简介
		DHTML 对象模型
		BOM 与 DOM
	window 对象	window 对象
	窗口和对话框	对话框
		窗口的打开和关闭
	定时器	定时器
		周期性定时器
		一次性定时器
document 对象	document 对象概述	document 对象概述
	DOM 概述	DOM 概述
		DOM 节点树
		DOM 操作
	DOM 操作 - 读取、修改	节点信息
		元素节点的内容
		节点属性
		元素节点的样式

经典案例.....Page 74

输入验证	Regex 对象概述
	Regex 对象的常用方法
计算查询时段	Date 对象
	Date 对象的常用方法
模拟方法的重载	arguments 对象
数组按数值排序	使用 Function 对象创建函数
	匿名函数
简单计算器	eval 函数
删除操作的提交和取消	对话框
动态时钟的启动和停止	周期性定时器
页面定时跳转	一次性定时器
广告轮播	节点属性
	元素节点的样式

课后作业.....Page 106

1. 常用内置对象

1.1. 正则表达式对象

1.1.1. 【正则表达式对象】RegExp 对象概述

Tarena
达内科技

RegExp 对象概述

- RegExp 对象表示正则表达式，它是对字符串执行模式匹配的强大工具
- 创建正则表达式对象
 - var rgExp=/pattern/flags;
 - var rgExp=new RegExp("pattern",["flags"]);
- flags标识有以下几个：
 - g：设定当前匹配为全局模式
 - i：忽略匹配中的大小写检测
 - m：多行搜索模式

```
var reg1 = /^d{3,6}$/;
var reg2 = new RegExp("^d{3,6}$");
```

1.1.2. 【正则表达式对象】RegExp 对象的常用方法

Tarena
达内科技

RegExp 对象的常用方法

- RegExpObject.test(string)
 - 如果字符串 string 中含有与 RegExpObject 匹配的文本，则返回 true，否则返回 false

<i>reg.compile(pattern, [flags])</i>	编译正则表达式
<i>reg.exec(str)</i>	检索字符串中指定的值 返回找到的值，并确定其位置
<i>reg.test(str)</i>	检索字符串中指定的值 返回 true 或 false

```
var name = "aaaa" ;
var reg = new RegExp("^[a-zA-Z0-9]{3,6}$");
var isRight = reg.test(name);
if (!isRight)
    alert("录入错误！");
```

1.2. Date 对象

1.2.1. 【Date 对象】Date 对象

Tarena
达内科技

Date 对象

- Date 对象用于处理日期和时间
- 创建 Date 对象

```
//Date 对象会自动把当前日期和时间保存为其初始值
var now = new Date();
//或者
var nowd2=new Date("2013/3/20 11:12");
```

1.2.2. 【Date 对象】Date 对象的常用方法



<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">Date 对象的常用方法</h4> <ul style="list-style-type: none"> • 读取日期的相关信息 <ul style="list-style-type: none"> - getDate()、getDay()、getFullYear()等 • 修改日期 <ul style="list-style-type: none"> - setDate()、setDay()、setFullYear()等 • 转换为字符串 <ul style="list-style-type: none"> - toString() - toLocaleTimeString() - toLocaleDateString() - ... <div style="text-align: right;">  </div> <div style="text-align: right;">+</div>
---	--

1.3. Function 对象

1.3.1. 【Function 对象】函数与 Function 对象

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">函数与Function 对象</h4> <ul style="list-style-type: none"> • 函数（方法）是一个可以重复执行的代码段 <ul style="list-style-type: none"> - 一组可以运行的语句 • Function 对象可以表示开发者定义的任何函数 • 函数实际上是功能完整的对象 <div style="text-align: right;">  </div> <div style="text-align: right;">+</div>
---	--

1.3.2. 【Function 对象】函数的定义

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">函数的定义</h4> <ul style="list-style-type: none"> • 由关键字function定义 • 函数名的定义规则与标识符一致，大小写敏感 • 可以使用变量、常量或表达式作为函数调用的参数 • 返回值必须使用return <ul style="list-style-type: none"> - return 语句也可以终止函数的执行 <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <pre>function 函数名 ([参数]) { 函数体; return 返回值; }</pre> </div> <div style="text-align: right;">  </div> <div style="text-align: right;">+</div>
---	--

1.3.3. 【Function 对象】函数的调用

Tarena
达内科技

函数的调用

- 函数可以通过其名字加上括号中的参数进行调用
 - 如果有多个参数，则参数之间用逗号隔开
 - 如果函数有返回值，则可以声明变量接收即可

```
function sum(iNum1, iNum2){
    return iNum1 + iNum2;
}
```

```
var iResult = sum(1,1);
alert(iResult); //输出 "2"
```

知识讲解

+

1.3.4. 【Function 对象】arguments 对象

Tarena
达内科技

arguments 对象

- arguments 是一种特殊对象，在函数代码中，表示函数的参数数组
- 在函数代码中，可以使用 arguments 访问所有参数
 - arguments.length : 函数的参数个数
 - arguments[i] : 第 i 个参数

```
function method1(){
    alert(arguments.length);
    alert(arguments[0]);
}
```

```
method1( 10,20 ); //输出 2 , 再输出 10
```

知识讲解

+

1.3.5. 【Function 对象】使用 Function 对象创建函数

Tarena
达内科技

使用 Function 对象创建函数

- 使用 Function 对象直接创建函数


```
var functionName =
    new Function( arg1, ... argN, functionBody );
```

```
var add = new Function("x", "y", "return(x+y);");
var result = add(2,3);
alert(result); // 5
alert(add); //弹出方法的文本
```

知识讲解

+

1.3.6. 【Function 对象】匿名函数

Tarena
达内科技

匿名函数

- 创建匿名函数


```
var func = function(arg1,...,argN) {
    func_body;
};
```

```
var add = function(x,y) {
    return x + y;
};
var result = add(2,3);
alert(result); // 5
alert(add); //弹出方法的文本
```

1.4. 全局函数

1.4.1. 【全局函数】全局函数概述

Tarena
达内科技

全局函数概述

- 全局函数可用于所有的 JavaScript 对象
- 常用的全局函数有：
 - parseInt/parseFloat
 - isNaN
 - eval
 - decodeURI/encodeURI
 - 等

1.4.2. 【全局函数】encodeURIComponent 与 decodeURI

Tarena
达内科技

encodeURIComponent 与 decodeURI

- encodeURIComponent() : 把字符串作为 URI 进行编码
- decodeURI() : 对 encodeURIComponent() 函数编码过的 URI 进行解码

```
var str = "http://tts6.tarena.com.cn/index.html?name=张三";
var r1 = encodeURIComponent(str);
alert(r1);

var r2 = decodeURI(r1);
alert(r2);
```


1.4.3. 【全局函数】eval 函数

Tarena
达内科技

eval 函数

- eval 函数用于计算某个字符串，以得到结果；或者用于执行其中的 JavaScript 代码
 - 只接受原始字符串作为参数
 - 如果参数中没有合法的表达式和语句，则抛出异常

```
var str = "2+3";
alert(str); //2+3
alert(eval(str)); //5

var str1 = "alert('hello');";
eval(str1);//hello
```

++

知识讲解

2. window 对象

2.1. DHTML 概述

2.1.1. 【DHTML 概述】DHTML 简介

Tarena
达内科技

DHTML 简介

- 操作 HTML 以创造各种动态视觉效果
 - 一种浏览器端的动态网页技术
- DHTML 的功能
 - 动态改变页面元素
 - 与用户进行交互等
 - DHTML 对象模型包括浏览器对象模型和 DOM 对象模型

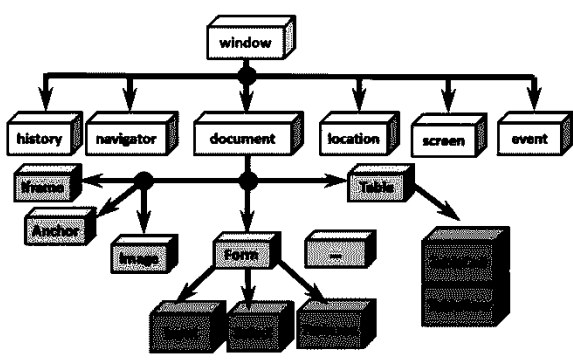
++

知识讲解

2.1.2. 【DHTML 概述】DHTML 对象模型

Tarena
达内科技

DHTML 对象模型





```

graph TD
    window[window] --> history[history]
    window --> navigator[navigator]
    window --> document[document]
    window --> location[location]
    window --> screen[screen]
    window --> event[event]
    document --> frame[frame]
    document --> image[image]
    document --> form[form]
    document --> table[table]
    form --> input[input]
    form --> button[button]
    form --> text[text]
    form --> select[select]
    form --> checkbox[checkbox]
    form --> radio[radio]
    form --> textarea[textarea]
    
```

++

知识讲解



2.1.3. 【DHTML 概述】BOM 与 DOM

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">BOM 与 DOM</h4> <ul style="list-style-type: none"> • BOM：浏览器对象模型，用来访问和操纵浏览器窗口，使 JavaScript 有能力与浏览器“对话” <ul style="list-style-type: none"> – 通过使用BOM，可移动窗口、更改状态栏文本、执行其它不与页面内容发生直接联系的操作 – 没有相关标准，但被广泛支持 • DOM：文档对象模型，用来操作文档 <ul style="list-style-type: none"> – 定义了访问和操作 HTML 文档的标准方法 – 应用程序通过对 DOM 树的操作，来实现对 HTML 文档数据的操作 <div style="text-align: right;">  </div>
---	--

2.2. window 对象



2.2.1. 【window 对象】window 对象

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">window 对象</h4> <ul style="list-style-type: none"> • window 表示浏览器窗口 <ul style="list-style-type: none"> – 所有 JavaScript 全局对象、函数以及变量均自动成为 window 对象的成员 • 常用属性 <ul style="list-style-type: none"> – document：窗口中显示的 HTML 文档对象 – history：浏览过窗口的历史记录对象 – location：窗口文件地址对象 – name：窗口名称 – opener：打开当前窗口的 window 对象 – ... <div style="text-align: right;">  </div>
---	--

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">window对象（续1）</h4> <ul style="list-style-type: none"> • 常用方法 <ul style="list-style-type: none"> – alert(),confirm,prompt()：对话框 – close(),open()：关闭、打开窗口 – focus(),blur()：窗口获得焦点或者失去焦点 – moveBy(),moveTo()移动窗口 – resizeBy(),resizeTo()：调整窗口大小 – scrollBy(),scrollTo()：滚动窗口中网页的内容 – ... <div style="text-align: right;">  </div>
---	---

2.3. 窗口和对话框

2.3.1. 【窗口和对话框】对话框



<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<div style="text-align: right;">  </div> <h3 style="margin-top: 0;">对话框</h3> <ul style="list-style-type: none"> • alert(str) <ul style="list-style-type: none"> - 提示对话框，显示str字符串的内容 • confirm(str) <ul style="list-style-type: none"> - 确认对话框，显示str字符串的内容 - 按“确定”按钮返回true，其他操作返回false • prompt(str,value) <ul style="list-style-type: none"> - 输入对话框，采用文本框输入信息 - str为提示信息，value为初始值 - 按“确定”按钮返回输入值，其他返回 undefined <div style="text-align: right; margin-top: 10px;">  </div>
---	--

2.3.2. 【窗口和对话框】窗口的打开和关闭



<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<div style="text-align: right;">  </div> <h3 style="margin-top: 0;">窗口的打开和关闭</h3> <ul style="list-style-type: none"> • window.open([url], [name], [config]) <ul style="list-style-type: none"> - url：打开的超链接 - name：窗口的名称 - config：窗口的配置参数 - 返回新窗口对象 • window.close() <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <pre>var config = "toolbar=yes,location=no,width=500,height=300"; //打开窗口 var openurl = "http://www.google.com"; var newWin = window.open(openurl,"popwin",config);</pre> </div> <div style="text-align: right; margin-top: 10px;">  </div>
---	--

2.4. 定时器



2.4.1. 【定时器】定时器

<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<div style="text-align: right;">  </div> <h3 style="margin-top: 0;">定时器</h3> <ul style="list-style-type: none"> • 多用于网页动态时钟、制作倒计时、跑马灯效果等 • 周期性时钟 <ul style="list-style-type: none"> - 以一定的间隔执行代码，循环往复 • 一次性时钟 <ul style="list-style-type: none"> - 在一个设定的时间间隔之后来执行代码，而不是在函数被调用后立即执行 <div style="text-align: right; margin-top: 10px;">  </div>
---	---

2.4.2. 【定时器】周期性定时器

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>周期性定时器</h3> <ul style="list-style-type: none"> • setInterval(exp,time) : 周期性触发代码exp <ul style="list-style-type: none"> - exp : 执行语句 - time : 时间周期, 单位为毫秒 - 返回已经启动的定时器对象 • clearInterval(tID) : 停止启动的定时器 <ul style="list-style-type: none"> - tID : 启动的定时器对象 <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <pre>window.setInterval("alert('hello');", 3000); //或者 window.setInterval(func,3000);</pre> </div> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <pre>function func() { alert("hello"); }</pre> </div> <div style="text-align: right;">  </div>
---	--



2.4.3. 【定时器】一次性定时器

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>一次性定时器</h3> <ul style="list-style-type: none"> • setTimeout(exp,time) : 一次性触发代码exp <ul style="list-style-type: none"> - exp : 执行语句 - time : 间隔时间, 单位为毫秒 - 返回已经启动的定时器 • clearTimeout(tID) : 停止启动的定时器 <ul style="list-style-type: none"> - tID : 启动的定时器对象 <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <pre>window.setTimeout("alert('hello');", 3000); //或者 window.setTimeout(func,3000);</pre> </div> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <pre>function func() { alert("hello"); }</pre> </div> <div style="text-align: right;">  </div>
---	--

3. document 对象


3.1. document 对象概述

3.1.1. 【document 对象概述】document 对象概述


<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>document 对象概述</h3> <ul style="list-style-type: none"> • 每个载入浏览器的 HTML 文档都会成为 document 对象 • 通过使用 document 对象, 可以从脚本中对 HTML 页面中的所有元素进行访问 <ul style="list-style-type: none"> - document 对象是 Window 对象的一部分, 可通过 window.document 属性对其进行访问 <div style="text-align: right;">  </div>
---	--


3.2. DOM 概述

3.2.1. 【DOM 概述】DOM 概述

<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<div style="text-align: right;"></div> <h3>DOM 概述</h3> <ul style="list-style-type: none">• DOM (Document Object Model) : 文档对象模型<ul style="list-style-type: none">- 当网页被加载时, 浏览器会创建页面的文档对象模型• 通过 DOM, 可以访问所有的 HTML 元素, 连同它们所包含的文本和属性<ul style="list-style-type: none">- 可以对其中的内容进行修改和删除, 同时也可以创建新的元素• HTML 文档中的所有节点组成了一个文档树 (或节点树)<ul style="list-style-type: none">- document 对象是一棵文档树的根 <div style="text-align: right;">+</div>
---	---

3.2.2. 【DOM 概述】DOM 节点树

<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<div style="text-align: right;"></div> <h3>DOM 节点树</h3> <ul style="list-style-type: none">• HTML 文档中的所有节点组成了一个文档树 (或节点树)• HTML 文档中的每个元素、属性、文本等都代表着树中的一个节点<ul style="list-style-type: none">- 整个文档是一个文档节点- 每个 HTML 标签是一个元素节点- 包含在 HTML 元素中的文本是文本节点- 每一个 HTML 属性是一个属性节点- 注释属于注释节点- 一切皆节点 <div style="text-align: right;">+</div>
---	---

<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<div style="text-align: right;"></div> <h3>DOM 节点树 (续1)</h3> <ul style="list-style-type: none">• 树根为 document 对象• 通过 document 对象, 来操作整个节点树• 一棵节点树中的所有节点彼此都有关系<ul style="list-style-type: none">- 父节点- 子节点- 后代- ... <div style="text-align: right;">+</div>
---	--

Technology
Tarena
达内科技

DOM 节点树 (续2)

+

3.2.3. 【DOM 概述】DOM 操作

Technology
Tarena
达内科技

DOM 操作

- 通过可编程的对象模型，JavaScript 获得了足够的能力来创建动态的 HTML
 - 查找节点
 - 读取节点信息
 - 修改节点信息
 - 创建新节点
 - 删除节点

+

3.3. DOM 操作 - 读取、修改

3.3.1. 【DOM 操作—读取、修改】节点信息

Technology
Tarena
达内科技

节点信息

- nodeName：节点名称
 - 元素节点和属性节点：标签或属性名称
 - 文本节点：永远是 #text
 - 文档节点：永远是 #document
- nodeType：节点类型
 - 返回数值
 - 元素节点：返回 1
 - 属性节点：返回 2
 - 文本节点：返回 3
 - 注释节点：返回 8
 - 文档节点：返回 9

+

节点信息（续1）

html 文档：

```
<select>
  <option id="o1" value="1">a</option>
  <option value="2">b</option>
</select>
```

js 代码：

```
var o1 = document.getElementById("o1");
var s = "option元素节点的名称" + o1.nodeName
s += "\noption" + "元素节点的类型：" + o1.nodeType;
alert(s);
```

来自网页的消息
 option元素节点的名称:OPTION
option元素节点的类型: 1

确定

3.3.2. 【DOM 操作—读取、修改】元素节点的内容

元素节点的内容

- innerText
 - 设置或获取位于对象起始和结束标签内的文本
- innerHTML
 - 设置或获取位于对象起始和结束标签内的 HTML

元素节点的内容（续1）

html 文档：

```
<div id="div1">
  第一行文本
  <p>段落中的文本</p>
</div>
```

js 代码：

```
var div = document.getElementById("div1");
var str = div.innerText;
str += "\n";
str += div.innerHTML;
alert(str);
```

来自网页的消息
 第一行文本
段落中的文本
第一行文本
<p>段落中的文本</p>

确定

3.3.3. 【DOM 操作—读取、修改】节点属性

Tarena 达内科技

节点属性

- `getAttribute()` 方法：根据属性名称获取属性的值
- `setAttribute()`、`removeAttribute()`

html 文档：

```
<a href="Default.aspx" title="this is a link" id="a1">
Click Me
</a>
```

js 代码：

```
var a1 = document.getElementById("a1");
var href = a1.getAttribute("href");//Default.aspx
alert(href);
alert(a1.innerHTML);//Click Me
```

+

Tarena 达内科技

节点属性（续1）

- 将 HTML 标记、属性和 CSS 样式都对象化

html 文档：

```
<a href="Default.aspx" id="a1">Click Me</a>

```

js 代码：

```
var a1 = document.getElementById("a1");
alert(a1.href);
a1.innerHTML = "new text";
document.getElementById("img1").src = "clock.jpg";
```

+

3.3.4. 【DOM 操作—读取、修改】元素节点的样式

Tarena 达内科技

元素节点的样式

- style 属性
 - `node.style.color`
 - `node.style.fontSize`
- className 属性

html 文档：

```
<p id="p1">一周畅销榜</p>
```

js 代码：

```
var o = document.getElementById("p1");
o.style.color = "red" ;
o.style.fontSize = 20;
//或者
o.className = "样式类名称" ;
```

+

经典案例

1. 输入验证

- 问题

用户需要在页面上输入用户名和电话号码，录入的要求如图 - 1 所示：

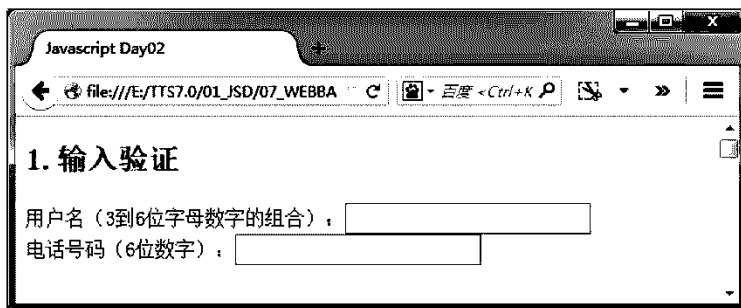


图 - 1

录入完毕后，立即检查录入的文本是否符合要求，如果不符合，则弹出提示信息，页面效果如图 - 2 所示：

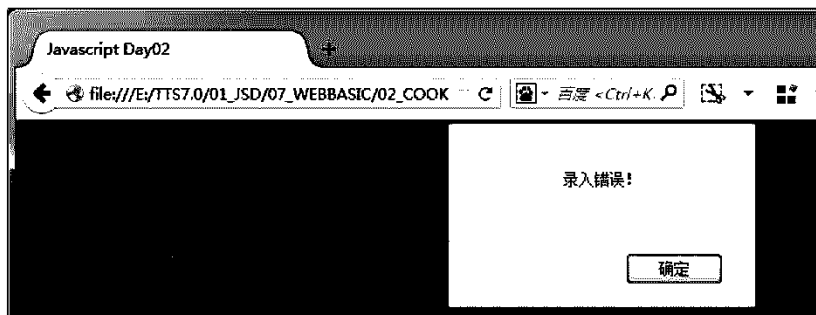


图 - 2

- 方案

在提交页面数据之前，需要对页面的输入进行验证，验证通过之后，再将数据发送到服务器，从而避免将错误的、不利的数据提交到服务器。

在 JavaScript 代码中，经常使用正则表达式对象来实现输入验证功能。首先，创建正则表达式对象，代码如下：

```
var reg = /^[a-zA-Z0-9]{3,6}$/;  
//或者  
var reg = new RegExp("^[a-zA-Z0-9]{3,6}$");
```

然后调用正则表达式对象的 test 方法进行校验，代码如下：

```
var isRight = reg.test(str);
```

该方法验证 str 表示的文本是否符合正则表达式，并返回 boolean 类型的结果。如果返回结果为 true，则表示所验证的文本符合正则表达式，否则表示不符合。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建页面

创建文件 js_demo2.html，并在 html 页面上添加代码以创建一个标准结构的 HTML 文档。html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  </head>
  <body>
  </body>
</html>
```

步骤二：创建 js 文件

创建文件 js_demo2.js，用于书写 JavaScript 代码。并在上一步所创建的 HTML 文档中引入脚本文件。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />

    <script language="javascript" src="js_demo2.js" type="text/javascript">
    </script>

  </head>
  <body>
  </body>
</html>
```

步骤三：为页面添加文本框

为 html 页面添加 <form> 标记，并在表单中添加提示文本、文本框，然后为文本框添加 onblur 事件。为便于提取文本框中录入的文本，需要为文本框定义 id 属性。

修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
```

```

<script language="javascript" src="js_demo2.js"
type="text/javascript">
</script>
</head>
<body>

    <form>
        <h2>1.输入验证</h2>
        用户名 ( 3 到 6 位字母数字的组合 ):
        <input type="text" id="txtName" onblur="validateUserName();" /><br/>
        电话号码 ( 6 位数字 ):
        <input type="text" id="txtPhone" onblur="validatePhone();" /><br/>
    </form>

</body>
</html>

```

步骤四：定义验证用户名的方法

在文件 js_demo2.js 中，创建名为 validateUserName 的方法。js 文件中的代码如下所示：

```

function validateUserName() {
}

```

步骤五：为方法添加代码，验证用户名

在方法 validateUserName 中添加代码，验证用户名。首先根据录入要求创建正则表达式对象，然后得到文本框中录入的用户名，并对录入的用户名进行验证。验证后，根据验证结果给出提示信息。代码如下所示：

```

//验证用户名
function validateUserName() {
    var name = document.getElementById("txtName").value;
    var reg = /^[a-zA-Z0-9]{3,6}$/;
    var isRight = reg.test(name);
    if (!isRight)
        alert("录入错误！");
}

```

步骤六：验证电话号码

再创建名为 validatePhone 的方法，然后为方法添加代码：根据录入要求创建正则表达式对象，并对录入的电话号码进行验证。代码如下所示：

```

//验证电话号码
function validatePhone() {
    var phone = document.getElementById("txtPhone").value;
    var reg = /^\d{6}$/;
    var isRight = reg.test(phone);
    if (!isRight)
        alert("录入错误！");
}

```

- **完整代码**

js_demo2.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <h2>1. 输入验证</h2>
      用户名（3 到 6 位字母数字的组合）：
      <input type="text" id="txtName" onblur="validateUserName();"
/><br/>
      电话号码（6 位数字）：
      <input type="text" id="txtPhone" onblur="validatePhone();"
/><br/>
    </form>
  </body>
</html>
```

js_demo2.js 文件的代码如下所示：

```
//验证用户名
function validateUserName() {
  var name = document.getElementById("txtName").value;
  var reg = /^[a-zA-Z0-9]{3,6}$/;
  var isRight = reg.test(name);
  if (!isRight)
    alert("录入错误！");
}
//验证电话号码
function validatePhone() {
  var phone = document.getElementById("txtPhone").value;
  var reg = /^\d{6}$/;
  var isRight = reg.test(phone);
  if (!isRight)
    alert("录入错误！");
}
```

2. 计算查询时段

- **问题**

有订单查询页面，页面上显示两种查询时段，显示效果如图 - 3 所示：

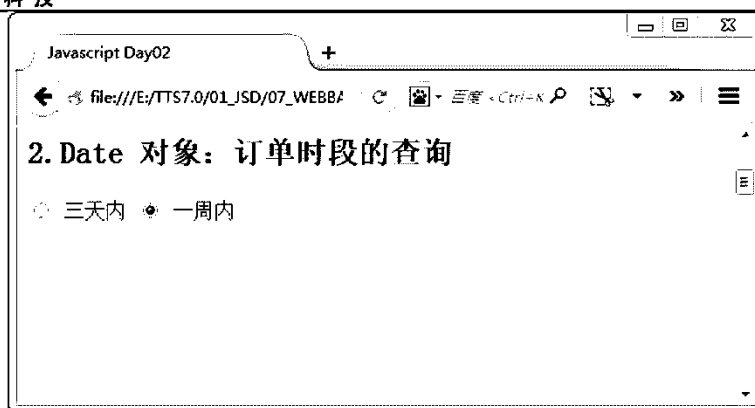


图 - 3

用户选择某种查询时段，则需要提示查询的开始日期和结束日期。比如，用户单击“三天内”，则弹出信息如图 - 4 所示：

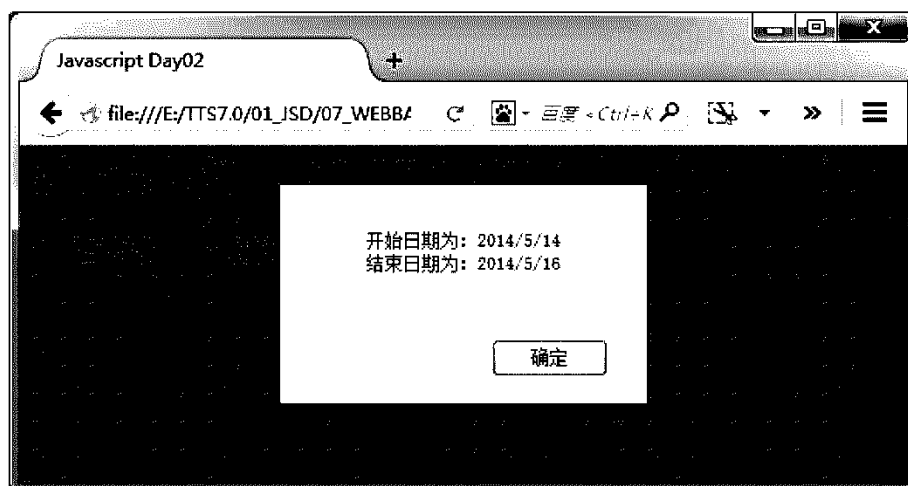


图 - 4

提示：当天日期为结束日期，向前推 3 天则为开始日期，且当天也算作一天，需要注意跨月的情况。比如，当前日期为 2014/3/2，向前推 3 天，则开始时间为 2014/2/28。三天分别为 2014/2/28、2014/3/1 和 2014/3/2。

用户单击“一周内”，也弹出所查询订单的开始日期和结束日期，页面效果如图 - 5 所示：

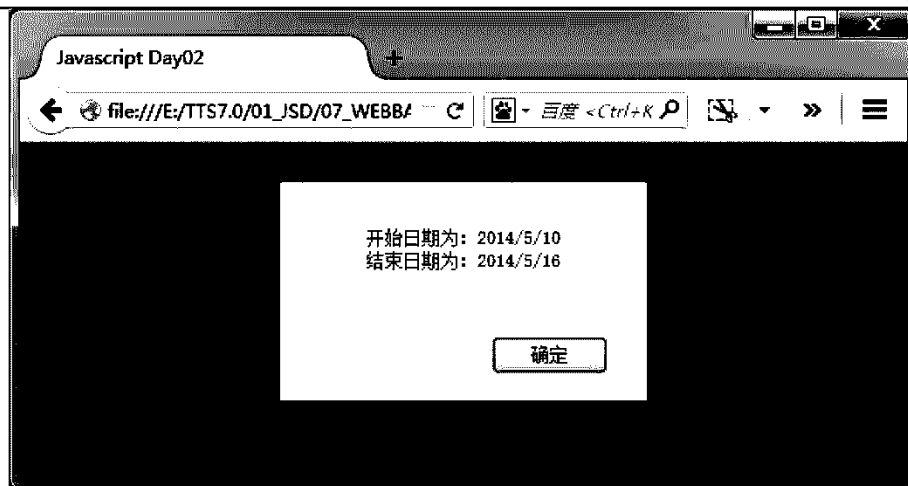


图 - 5

• 方案

实现此案例时,首先需要为页面上的两个单选框添加 onclick 事件,并为其定义方法。

两个单选框的 onclick 事件可以调用同一个方法,传入不同的天数(3 或者 7)作为参数即可。在方法中,首先需要得到当前日期作为结束日期,然后减去相应的天数得到开始日期即可。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加文本和单选框

在上一个案例的 html 页面上继续添加此案例的内容。

首先,在 <form> 标记中添加文本和单选框,然后为两个单选框定义 onclick 事件,并分别传入参数。修改后的 html 代码如下所示:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <h2>1. 输入验证</h2>
      用户名 ( 3 到 6 位字母数字的组合 ):
      <input type="text" id="txtName" onblur="validateUserName();"
./><br/>
      电话号码 ( 6 位数字 ):
      <input type="text" id="txtPhone" onblur="validatePhone();"
./><br/>
    <br />
  </body>
</html>
```

```

                <h2>2.Date 对象：订单时段的查询</h2>
                <input id="chkThree" type="radio" onclick="getDateRange(3);"
name="date" />
                <label for="chkThree">三天内</label>
                <input id="chkWeek" type="radio" onclick="getDateRange(7);" name="date"
/>
                <label for="chkWeek">一周内</label>

        </form>
</body>
</html>

```

步骤二：在 js 文件中定义方法

打开文件 js_demo2.js，在其中添加代码，定义名为 getDateRange 的方法，并为其定义参数以接收传入的数值。js 文件中添加的代码如下所示：

```

//时间的查询
function getDateRange(days) {
}

```

步骤三：得到结束日期

为方法添加代码。首先创建日期对象，以得到当前日期，也是结束日期。代码如下所示：

```

//时间的查询
function getDateRange(days) {
    //得到当前日期
    var end = new Date();
    var endString = end.toLocaleDateString();
}

```

步骤四：得到开始日期

对当前日期减去对应的天数。因为当前天也算作一天，因此首先减去参数所传入的天数，再加上一天。代码如下：

```

//时间的查询
function getDateRange(days) {
    //得到当前日期
    var end = new Date();
    var endString = end.toLocaleDateString();
    //修改日期
    end.setDate(end.getDate() - days + 1);
}

```

步骤五：显示计算的结果

最后，拼接字符串，并弹出显示计算的结果。代码如下：

```

//时间的查询
function getDateRange(days) {

```

```
//得到当前日期
var end = new Date();
var endString = end.toLocaleDateString();
//修改日期
end.setDate(end.getDate() - days + 1);
//显示结果
var s = "开始日期为：" + end.toLocaleDateString() + "\n";
s += "结束日期为：" + endString;
alert(s);
}
```

• 完整代码

js_demo2.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <h2>1. 输入验证</h2>
      用户名（3 到 6 位字母数字的组合）：
      <input type="text" id="txtName" onblur="validateUserName();"
/><br/>
      电话号码（6 位数字）：
      <input type="text" id="txtPhone" onblur="validatePhone();"
/><br/>
      <br />
      <h2>2. Date 对象：订单时段的查询</h2>
      <input id="chkThree" type="radio" onclick="getDateRange(3);"
name="date" />
      <label for="chkThree">三天内</label>
      <input id="chkWeek" type="radio" onclick="getDateRange(7);"
name="date" />
      <label for="chkWeek">一周内</label>
    </form>
  </body>
</html>
```

js_demo2.js 文件的代码如下所示：

```
//其他案例的代码部分，略

//时间的查询
function getDateRange(days) {
  //得到当前日期
  var end = new Date();
  var endString = end.toLocaleDateString();
  //修改日期
  end.setDate(end.getDate() - days + 1);
  //显示结果
  var s = "开始日期为：" + end.toLocaleDateString() + "\n";
```



```
s += "结束日期为: " + endString;  
alert(s);  
}
```

3. 模拟方法的重载

- 问题

模拟实现 javascript 中方法的重载。

定义一个方法,假如名为 myMethod。单击页面上的“计算平方”按钮,则调用该方法,并传入一个数值参数,然后计算该数值的平方,页面效果如图 - 6 所示:



图 - 6

单击页面上的“计算加法”按钮,依然调用该方法,并传入两个数值参数,然后计算两个数值的和,页面效果如图 - 7 所示:

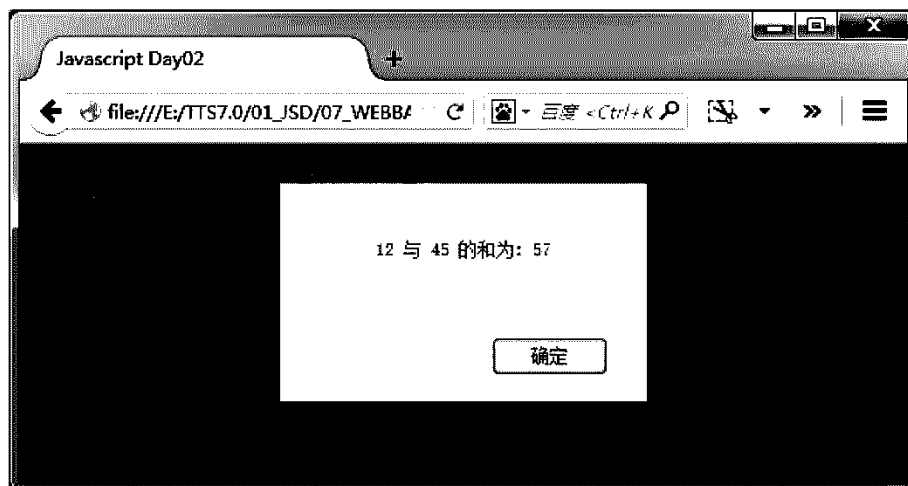


图 - 7

如果传入其他数目的参数,则不做任何计算。

- 方案

在 JavaScript 的函数代码中,arguments 表示函数的参数数组,可以用它访问所有参数。arguments 是一个数组,用 arguments.length 可以得到参数的个数,使用 arguments[i]可以得到第 i 个参数的值。

因此,可以使用 arguments 来实现方法的重载模拟。即,定义方法时,不需要明文的定义参数,而是在方法体中使用 arguments 关键字获取参数。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：定义方法

打开文件 js_demo2.js,在其中添加代码,定义名为 myMethod 的方法。js 文件中添加的代码如下所示：

```
//方法的重载
function myMethod() {
}
```

步骤二：在方法中,判断传入的参数个数

为方法 myMethod 添加代码。在方法中,使用 arguments 关键字获取传入的所有参数,并对参数的个数进行判断：只关注传入 1 个参数和 2 个参数的情况。代码如下所示：

```
//方法的重载
function myMethod() {

    if (arguments.length == 1) {
    }
    else if (arguments.length == 2) {
    }

}
```

步骤三：分别计算平方和加法

在方法中继续添加代码,分别处理不同参数个数时的情况：1 个参数时,计算平方；2 个参数时,计算数值和,需要使用索引来访问具体的参数值。代码如下所示：

```
//方法的重载
function myMethod() {
    if (arguments.length == 1) {

        //计算平方
        var n = parseInt(arguments[0]);
        alert(n + " 的平方为：" + n * n);

    }
    else if (arguments.length == 2) {
```

```
//计算和
var n = parseInt(arguments[0]);
var m = parseInt(arguments[1]);
var result = n + m;
alert(n + " 与 " + m + " 的和为：" + result);

}
}
```

步骤四：为页面添加按钮，并分别调用方法

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加两个按钮，然后为按钮定义 onclick 事件，并调用方法 myMethod。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>3.arguments：方法重载</h2>
      <input type="button" value="计算平方" onclick="myMethod(12);" />
      <input type="button" value="计算加法" onclick="myMethod(12,45);" />

    </form>
  </body>
</html>
```

• 完整代码

js_demo2.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->
      <h2>3.arguments：方法重载</h2>
      <input type="button" value="计算平方" onclick="myMethod(12);" />
```

```
<input type="button" value="计算加法" onclick="myMethod(12,45);" />
</form>
</body>
</html>
```

js_demo2.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//方法的重载
function myMethod() {
    if (arguments.length == 1) {
        //计算平方
        var n = parseInt(arguments[0]);
        alert(n + " 的平方为: " + n * n);
    }
    else if (arguments.length == 2) {
        //计算和
        var n = parseInt(arguments[0]);
        var m = parseInt(arguments[1]);
        var result = n + m;
        alert(n + " 与 " + m + " 的和为: " + result);
    }
}
```

4. 数组按数值排序

• 问题

对数组实现按照数值排序，要求如下：

- 1、定义一个无序的数组，比如内容为：[34, 2, 14, 56, 43]；
- 2、使用 Function 对象创建比较数值大小所用的方法，以实现按照数值大小对数组进行升序排列，并弹出排序后的结果在页面显示，效果如图 - 8 所示：

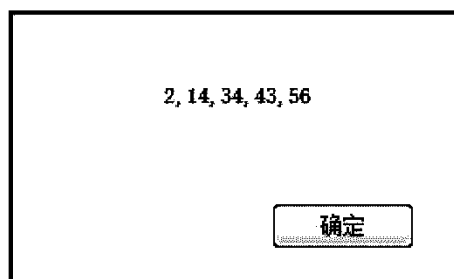


图 - 8

- 3、使用匿名函数定义比较数值大小所用的方法，实现对数组按照数值大小降序排列，并弹出排序后的结果在页面显示，效果如图 - 9 所示：

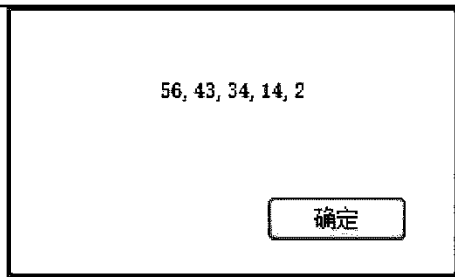


图 - 9

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加按钮

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加按钮，并为按钮定义单击事件。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>JavaScript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>4.Function 对象和匿名函数</h2>
      <input type="button" value="排序" onclick="sortArray();" />

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义方法

打开文件 js_demo2.js，在其中添加代码，定义名为 sortArray 的方法，并在方法中创建一个数组，并为数组赋初始值。js 文件中添加的代码如下所示：

```
//数组排序
function sortArray() {
  var array = [34, 2, 14, 56, 43];
}
```

步骤三：使用 Function 对象实现升序排列

在 `sortArray` 方法中,使用 `Function` 对象创建用于排序比较的方法,并调用 `Array` 对象的 `sort` 方法。代码如下所示:

```
//数组排序
function sortArray() {
    var array = [34, 2, 14, 56, 43];
    //使用 Function 对象
    array.sort(new Function("a", "b", "return a-b;"));
    alert(array.toString());
}
```

步骤四：使用匿名函数实现降序排列

在 `sortArray` 方法中,使用匿名函数创建用于排序比较的方法,并调用 `Array` 对象的 `sort` 方法。代码如下所示:

```
//数组排序
function sortArray() {
    var array = [34, 2, 14, 56, 43];
    //使用 Function 对象
    array.sort(new Function("a", "b", "return a-b;"));
    alert(array.toString());
    //使用匿名函数
    array.sort(function (a, b) { return b - a; });
    alert(array.toString());
}
```

• 完整代码

`js_demo2.html` 文件的代码如下所示:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码, 略-->
      <h2>4.Function 对象和匿名函数</h2>
      <input type="button" value="排序" onclick="sortArray();" />
    </form>
  </body>
</html>
```

`js_demo2.js` 文件中的代码如下所示:

```
//其他案例的代码部分, 略

//数组排序
```

```
function sortArray() {  
    var array = [34, 2, 14, 56, 43];  
    //使用 Function 对象  
    array.sort(new Function("a", "b", "return a-b;"));  
    alert(array.toString());  
    //使用匿名函数  
    array.sort(function (a, b) { return b - a; });  
    alert(array.toString());  
}
```

5. 简单计算器

- 问题

制作简单计算器。页面效果如图 - 10 所示：

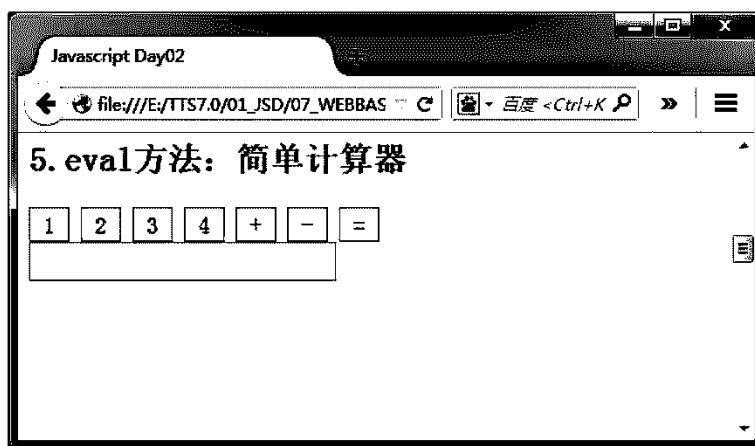


图 - 10

单击图 - 10 所示页面上的按钮（按钮 “=” 除外），比如数字以及 “+”、“-” 按钮，则拼接计算表达式并写入文本框，效果如图 - 11 所示：

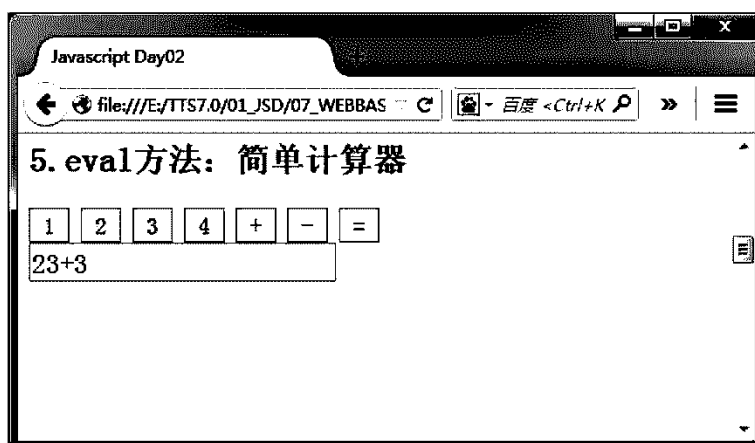


图 - 11

单击 “=” 按钮，则计算表达式并将计算的结果写入文本框，页面效果如图 - 12 所示：

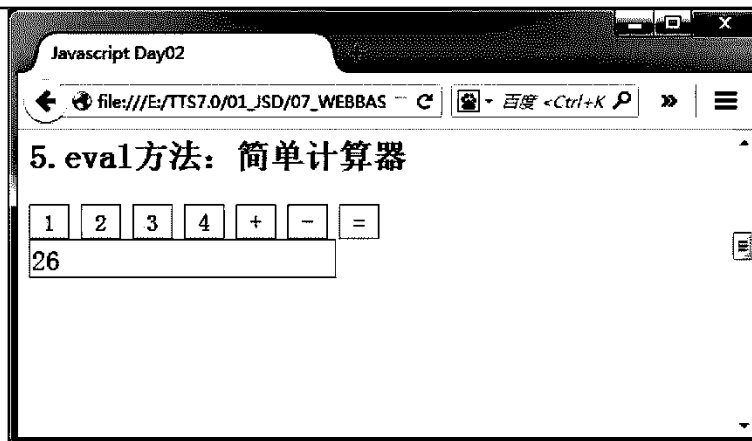


图 - 12

注：此案例实现简单计算器的功能，只考虑用户录入正确的情况，不考虑错误录入的情况。

• 方案

此案例中，首先需要使用 `this.value` 关键字将每个按钮上的文本传入方法，用于判断单击的是哪个按钮；然后在方法中，根据传入的按钮文本进行判断：如果单击的是 “=” 按钮，则使用 `eval` 方法计算文本框中的文本表达式，求出结果并显示；如果单击的是其他按钮，则将按钮上的文本追加到文本框中拼接显示。

注：如果希望实现完整的计算器功能，则需要对用户操作的判断。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加文本框和按钮

在上一个案例的 `html` 页面上继续添加此案例的内容。

首先，在 `<form>` 标记中添加文本框和按钮。为了操作文本框中的文本，为其定义 `id` 属性，并为按钮定义单击事件。在按钮的单击事件中，使用 `this.value` 关键字，将按钮文本传入方法。

修改后的 `html` 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js/demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!-- 其他部分代码，略-->
```



```
<h2>5.eval 方法：简单计算器</h2>
<input type="button" value="1" onclick="simpleCal(this.value);" />
<input type="button" value="2" onclick="simpleCal(this.value);" />
<input type="button" value="3" onclick="simpleCal(this.value);" />
<input type="button" value="4" onclick="simpleCal(this.value);" />
<input type="button" value="+" onclick="simpleCal(this.value);" />
<input type="button" value="-" onclick="simpleCal(this.value);" />
<input type="button" value="=" onclick="simpleCal(this.value);" /><br />
<input type="text" id="txtData" style="font-size:15pt;border:1px solid
gray;" />

</form>
</body>
</html>
```

步骤二：在 js 文件中定义方法

打开文件 js_demo2.js，在其中添加代码，定义名为 simpleCal 的方法，js 文件中添加的代码如下所示：

```
//简单计算器
function simpleCal(str) {
}
```

步骤三：判断所单击的按钮

在 simpleCal 方法中，使用参数传入所单击的按钮文本。因此，首先需要根据参数判断所单击的按钮，区分开 “=” 按钮和其他按钮。代码如下所示：

```
//简单计算器
function simpleCal(str) {

    if (str == "=") {
    }
    else{
    }

}
```

步骤四：计算结果或者拼接字符串并显示

如果单击的是 “=” 按钮，则使用 eval 函数计算结果并显示在文本框中；如果单击的是其他按钮，则将按钮文本拼接在文本框中并显示。代码如下所示：

```
//简单计算器
function simpleCal(str) {
    if (str == "=") {

        var result = eval(document.getElementById("txtData").value);
        document.getElementById("txtData").value = result;

    }
    else {
```

```
document.getElementById("txtData").value += str;
```

```
}  
}
```

• 完整代码

js_demo2.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->
      <h2>5.eval 方法：简单计算器</h2>
      <input type="button" value="1" onclick="simpleCal(this.value);"
/>
      <input type="button" value="2" onclick="simpleCal(this.value);"
/>
      <input type="button" value="3" onclick="simpleCal(this.value);"
/>
      <input type="button" value="4" onclick="simpleCal(this.value);"
/>
      <input type="button" value="+" onclick="simpleCal(this.value);"
/>
      <input type="button" value="-" onclick="simpleCal(this.value);"
/>
      <input type="button" value="=" onclick="simpleCal(this.value);"
/><br />
      <input type="text" id="txtData" style="font-size:15pt;border:1px
solid gray;" />
    </form>
  </body>
</html>
```

js_demo2.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//简单计算器
function simpleCal(str) {
  if (str == "=") {
    var result = eval(document.getElementById("txtData").value);
    document.getElementById("txtData").value = result;
  }
  else {
    document.getElementById("txtData").value += str;
  }
}
```

6. 删除操作的提交和取消

- 问题

为页面添加提交类型的按钮，作为删除按钮，页面效果如图 - 13 所示：

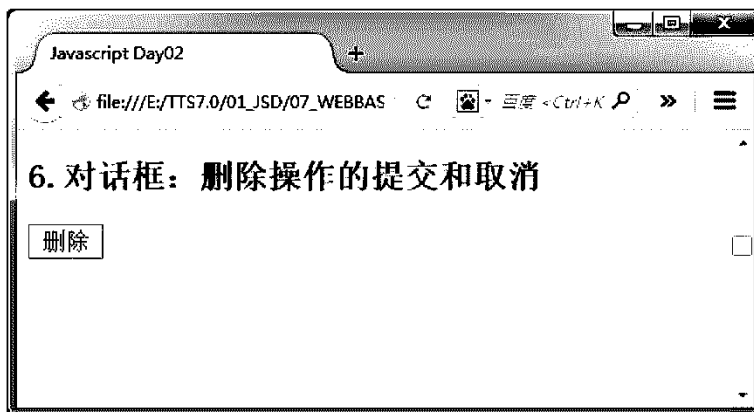


图 - 13

单击“删除”按钮，会弹出提示信息，效果如图 - 14 所示：

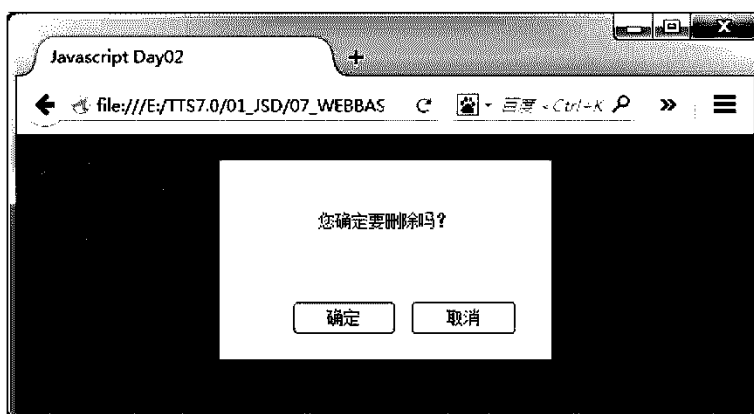


图 - 14

单击图 - 14 上的“确定”按钮，则提交信息（页面会刷新）；如果单击的是“取消”按钮，则不发生提交动作（页面不会刷新）。

- 方案

在 js 中，可以取消事件，代码如下所示：

```
<input type="button" value="aa" onclick="return false; " />
```

此案例中，提交按钮的单击事件是否取消，取决于对话框 confirm 的返回值。

- 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加按钮

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加按钮，并为其定义单击事件。需要注意的是，为按钮定义单击事件时，需要使用 return 关键字。

修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>6.对话框：删除操作的提交和取消</h2>
      <input type="submit" value="删除" onclick="return delFunc();" />

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义方法

打开文件 js_demo2.js，在其中添加代码，定义名为 delFunc 的方法，js 文件中添加的代码如下所示：

```
//模拟删除操作
function delFunc() {
}
```

步骤三：使用对话框

在 delFunc 方法中，弹出确认对话框，并返回对话框的结果。代码如下所示：

```
//模拟删除操作
function delFunc() {

  var result = confirm("您确定要删除吗?");
  return result;

}
```

• 完整代码

js_demo2.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->
      <h2>6. 对话框：删除操作的提交和取消</h2>
      <input type="submit" value="删除" onclick="return delFunc();" />
    </form>
  </body>
</html>
```

js_demo2.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//模拟删除操作
function delFunc() {
    var result = confirm("您确定要删除吗? ");
    return result;
}
```

7. 动态时钟的启动和停止

- 问题

页面上有按钮“显示时间”，单击该按钮，可以在页面上显示当前时间。页面效果如图 - 15 所示：

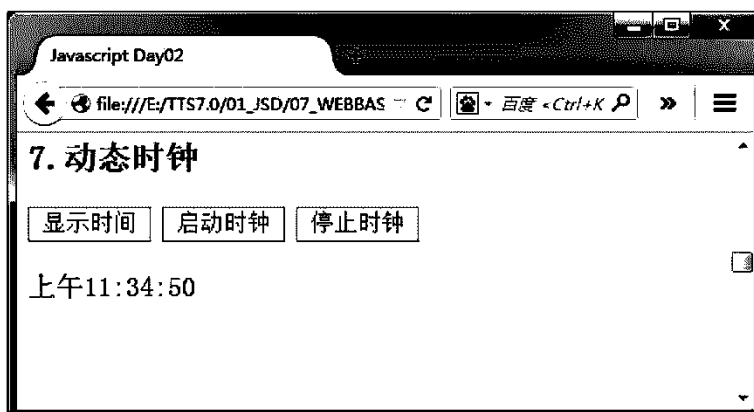


图 - 15

此时，页面上的时间仅显示单击按钮那一刻的时间。如果单击页面上的“启动时钟”按钮，则在页面上动态显示当前时间，即，时间的显示会每秒更新，从而即时的显示当前时间，

如同电子时钟。

单击页面上的“停止时钟”按钮，则停止时间的动态显示，即，不再更新时间的显示。

• 方案

实现此案例时，首先需要用到 Date 对象，以得到当前时间并显示；然后需要用到周期性定时器，即使用 window 对象的 setInterval 方法，每隔 1s 即重新显示一次当前时间，从而实现动态时钟的效果；如果需要停止时钟，则调用 window 对象的 clearInterval 方法即可。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加按钮和元素

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加按钮，并分别为按钮定义单击事件；然后添加 元素，用于显示当前时间。

修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>7.动态时钟</h2>
      <input type="button" value="显示时间" onclick="showTime();" />
      <input type="button" value="启动时钟" onclick="startClock();" />
      <input type="button" value="停止时钟" onclick="stopClock();" />
      <br /><br />
      <span id="spanTime" style="font-size:15pt;"></span>

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义方法 showTime

打开文件 js_demo2.js，在其中添加代码，首先定义名为 showTime 的方法用于显示当前时间，js 文件中添加的代码如下所示：

```
//显示时间
function showTime() {
    var now = new Date();
    document.getElementById("spanTime").innerHTML
now.toLocaleTimeString();
}
```

步骤三：在 js 文件中定义方法 startClock

打开文件 js_demo2.js，在其中添加代码，定义名为 startClock 的方法用于动态的显示当前时间。js 文件中添加的代码如下所示：

```
//启动时钟
var timer;
function startClock() {
    timer = window.setInterval(showTime,1000);
}
```

注意：为便于时钟的停止，需要将时钟对象 timer 定义为全局变量。

步骤四：在 js 文件中定义方法 stopClock

打开文件 js_demo2.js，在其中添加代码，定义名为 stopClock 的方法用于停止定时器。js 文件中添加的代码如下所示：

```
//停止时钟
function stopClock() {
    window.clearInterval(timer);
}
```

注意：本案例中，只是实现了时钟启动和停止的基本功能，关于如何防止重复启动时钟，请自行思考。

• 完整代码

js_demo2.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->
      <h2>7. 动态时钟</h2>
      <input type="button" value="显示时间" onclick="showTime();" />
      <input type="button" value="启动时钟" onclick="startClock();" />
      <input type="button" value="停止时钟" onclick="stopClock();" />
      <br /><br />
    </form>
  </body>
</html>
```

```
<span id="spanTime" style="font-size:15pt;"></span>
</form>
</body>
</html>
```

js_demo2.js 文件中的代码如下所示：

//其他案例的代码部分，略

```
//显示时间
function showTime() {
    var now = new Date();
    document.getElementById("spanTime").innerHTML =
        now.toLocaleTimeString();
}

//启动时钟
var timer;
function startClock() {
    timer = window.setInterval(showTime,1000);
}

//停止时钟
function stopClock() {
    window.clearInterval(timer);
}
```

8. 页面定时跳转

• 问题

页面上有按钮“5s 后打开新页面”，单击该按钮，则在 5 秒后打开新页面（TTS 的登录页面），页面效果如图 - 16 所示（图中红色框线圈出的部分为打开的新页面）：

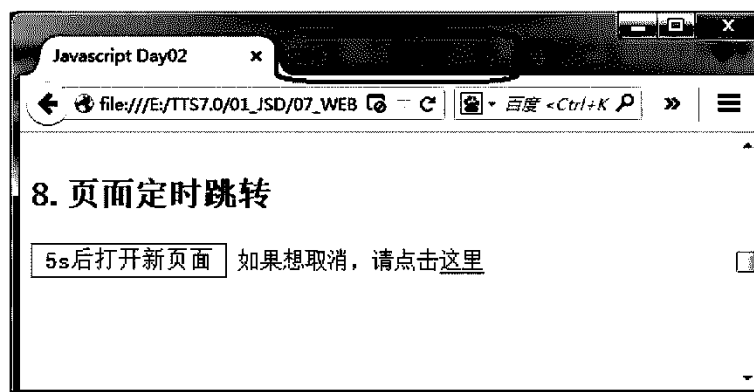


图 - 16

如果单击按钮后，在 5 秒之内单击链接文本“这里”，则取消该动作，即不再打开新页面。

• 方案

window 对象的 setTimeout 方法，可以在间隔一定时间后，一次性的触发某段代码，

从而实现类似于倒计时效果的操作。本案例中，可以使用该方法来实现 5s 后打开新页面的操作。

单击链接文本的时候，如果希望触发一段 js 代码，需要为 <a> 元素的 href 属性添加特定的声明，代码如下所示：

```
<a href="javascript:cancelOpen();">这里</a>
```

上述代码中，单击链接文本“这里”，将执行名为 cancelOpen 的 js 方法。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加按钮和链接

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加按钮和链接，为按钮定义单击事件，并为链接文本声明所需要调用的 js 方法。

修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>8. 页面定时跳转</h2>
      <input type="button" value="5s 后打开新页面" onclick="openWindowWait();"
/>

      如果想取消，请点击<a href="javascript:cancelOpen();">这里</a>

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义方法 openWindowWait

打开文件 js_demo2.js，在其中添加代码，定义名为 openWindowWait 的方法。

在该方法中，定义打开新窗口的匿名方法，并在 window 对象的 setTimeout 方法中调用该匿名方法，实现 5s 后打开新窗口。js 文件中添加的代码如下所示：

```
// 页面定时跳转
```

```
var openTimer;
function openWindowWait() {
    var openFunc = function(){
        window.open("http://tts7.tarena.com.cn");
    };
    openTimer = window.setTimeout(openFunc,5000);
}
```

注意：为便于取消定时器，需要将时钟对象 openTimer 定义为全局变量。

步骤三：在 js 文件中定义方法 cancelOpen

在文件 js_demo2.js 中继续添加代码，定义名为 cancelOpen 的方法。

在该方法中，使用 window 对象的 clearTimeout 方法取消定时器。js 文件中添加的代码如下所示：

```
//取消定时跳转
function cancelOpen() {
    window.clearTimeout(openTimer);
}
```

注意 本案例中，只是实现了时钟启动和停止的基本功能，关于如何防止重复启动时钟，请自行思考。

• 完整代码

js_demo2.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->
      <h2>8. 页面定时跳转</h2>
      <input type="button" value="5s 后 打 开 新 页 面 "
onclick="openWindowWait();" />
      如果想取消，请点击<a href="javascript:cancelOpen();">这里</a>
    </form>
  </body>
</html>
```

js_demo2.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//页面定时跳转
var openTimer;
function openWindowWait() {
```

```
var openFunc = function(){  
    window.open("http://tts7.tarena.com.cn");  
};  
openTimer = window.setTimeout(openFunc,5000);  
}  
//取消定时跳转  
function cancelOpen() {  
    window.clearTimeout(openTimer);  
}
```

9. 广告轮播

- 问题

有四张图片，其名称和图片效果如图 - 17 所示：

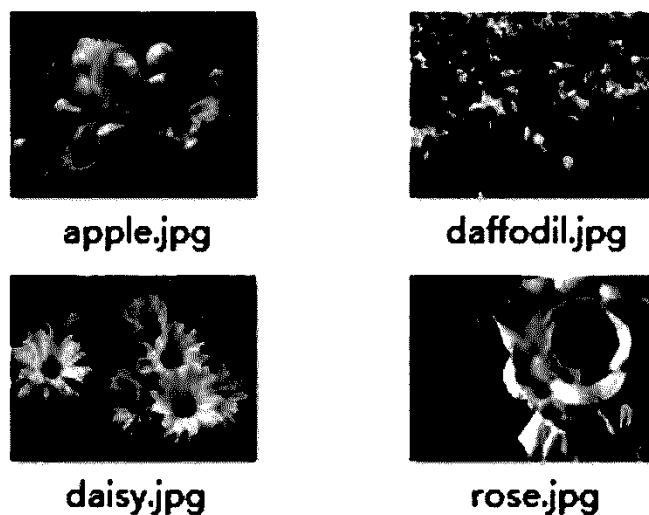


图 - 17

需要使用图 - 17 中的四张图片完成页面上的广告轮播效果，详细要求如下：

1、页面加载后，先在页面上显示某个图片，页面效果如图 - 18 所示：



图 - 18

2、每隔 3s，自动更换页面上的图片显示，效果如图 - 19 所示：



图 - 19

- 3、图片轮换显示的顺序为："rose.jpg"、"daffodil.jpg"、"apple.jpg"、"daisy.jpg"；
- 4、一轮显示完毕后，重复下一轮显示；
- 5、鼠标移入图片时，停止图片轮换；
- 6、鼠标移出图片时，继续图片轮换。

• 方案

实现本案例时，首先需要创建一个数组，用于存储需要显示的图片的路径，代码如下所示：

```
var imageArray = ["rose.jpg", "daffodil.jpg", "apple.jpg", "daisy.jpg"];
```

页面初始化时，显示第一张图片，即 "rose.jpg"；然后，启动周期性时钟，每隔 3s 更换图片的显示：显示数组中的下一个图片。因此，需要定义变量表示当前显示数组中的第几个图片，代码如下所示：

```
var imageIndex = 1;
```

上述代码中，将变量 imageIndex 赋初始值为 1，是因为：页面首次加载时显示数组中索引为 0 的图片，3s 后，进行第一次图片变换：需要显示数组中索引为 1 的图片。

另外，为实现鼠标移入图片则停止时钟、移出继续启动时钟的功能，需要用到鼠标事件 onmouseover 和 onmouseout，分别表示鼠标移入和鼠标移出事件。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：准备图片

在页面文件 js_demo2.html 所在的路径下，创建文件夹 images，用于存储页面需要的 4 个图片，存储结构如图 - 20 所示：



图 - 20

步骤二：为页面添加 元素

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加 元素，用于显示图片，并为其定义宽度；然后，设置 元素显示第一张图片（“rose.jpg”）；并为其定义鼠标移入和鼠标移出事件。

修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>JavaScript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body>
    <form>
      <!-- 其他部分代码，略-->

      <h2>9. 广告轮播</h2>
      

    </form>
  </body>
</html>
```

步骤三：在 js 文件中定义全局变量

打开文件 js_demo2.js，在其中添加代码，定义所需要的全局变量。

首先，定义数组，用于存储需要轮换显示的图片；然后定义变量用于记载当前所显示的图片的索引；并定义变量表示启动的时钟对象（便于时钟的停止）。js 文件中添加的代码如下所示：

```
// 定义数组，存储图片的路径
```

```
var imageArray = ["rose.jpg", "daffodil.jpg", "apple.jpg", "daisy.jpg"];
//当前所显示的图片的索引
var imageIndex = 1;
//定时器对象
var imageTimer;
```

步骤四：在 js 文件中定义方法 startRotate

继续在文件 js_demo2.js 中添加代码，定义名为 startRotate 的方法，实现启动广告轮播。

该方法中，首先需要定义匿名函数（用于实现图片轮换），然后，启动周期性定时器，每隔 3s 调用该匿名方法。js 文件中添加的代码如下所示：

```
//启动图片轮换
function startRotate() {
    var rotateFunc = function () {
    };
    imageTimer = window.setInterval(rotateFunc, 3000);
}
```

步骤五：实现图片轮换

为了实现图片的循环更换，首先从数组中取出当前需要显示的图片路径，并修改 元素的 src 属性，以显示新图片；然后修改变量 imageIndex 的值为下一个图片的索引位置，便于下一次的图片显示。

需要注意的是，imageIndex 的值不能超出数组的范围，因此，当 imageIndex 的值已经达到最大值（数组的最大索引）时，将其重置为数值 0，则重新开始下一轮的轮换。

代码如下所示：

```
//启动图片轮换
function startRotate() {
    var rotateFunc = function () {

        var image = document.getElementById("img1");
        image.src = "images/" + imageArray[imageIndex];
        if (imageIndex == imageArray.length - 1)
            imageIndex = 0;
        else
            imageIndex++;

    };
    imageTimer = window.setInterval(rotateFunc, 3000);
}
```

步骤六：启动图片轮换

startRotate 方法定义完毕后，在图片的鼠标移出事件中调用它，还需要在页面加载时调用它，便于在页面加载时默认启动图片轮换功能。

因此，需要在 <body> 元素的 onload 事件中调用方法 startRotate。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>

  <body onload="startRotate();">

    <form>
      <!--其他部分代码，略-->
      <h2>9. 广告轮播</h2>
      
    </form>
  </body>
</html>
```

步骤七：在 js 文件中定义方法 stopRotate

继续在文件 js_demo2.js 中添加代码，定义名为 stopRotate 的方法，停止图片的轮
换。js 文件中的代码如下所示：

```
//停止图片轮换
function stopRotate() {
  window.clearInterval(imageTimer);
}
```

• 完整代码

js_demo2.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day02</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo2.js"
type="text/javascript">
    </script>
  </head>
  <body onload="startRotate();">
    <form>
      <!--其他部分代码，略-->
      <h2>9. 广告轮播</h2>
      
    </form>
  </body>
</html>
```

js_demo2.js 文件中的代码如下所示：

//其他案例的代码部分,略

```
//定义数组,存储图片的路径
var imageArray = ["rose.jpg", "daffodil.jpg", "apple.jpg", "daisy.jpg"];
//当前所显示的图片的索引
var imageIndex = 1;
//定时器对象
var imageTimer;

//启动图片轮换
function startRotate() {
    var rotateFunc = function () {
        var image = document.getElementById("img1");
        image.src = "images/" + imageArray[imageIndex];
        if (imageIndex == imageArray.length - 1)
            imageIndex = 0;
        else
            imageIndex++;
    };
    imageTimer = window.setInterval(rotateFunc, 3000);
}

//停止图片轮换
function stopRotate() {
    window.clearInterval(imageTimer);
}
```


课后作业

1. 简述 arguments 对象的作用。
2. 列举几个 JavaScript 中常用的全局函数，并描述其作用。
3. 为 NetCTOSS 系统创建错误页面。

NetCTOSS 系统有错误页面，该页面加载后的显示效果如图 - 1 所示：



图 - 1

由图 - 1 可以看出，页面初始化时，将显示剩余的秒数。该秒数从 5 开始，其数值将每秒递减，递减到 0 秒后将自动在新窗口中打开 url 地址为 `tts7.tarena.com.cn` 的页面，页面效果如图 - 2 所示：



图 - 2

图 - 2 中红色圈出的地方，为所打开的新页面。

如果在 5s 之内，单击页面上的链接文本“返回”，则直接在新窗口中打开 url 地址为 `tts7.tarena.com.cn` 的页面，且原页面上的秒数停止变化。页面效果如图 - 3 所示：



图 - 3

4. 为 admin_list 页面添加 JavaScript 代码，实现相关功能。

有管理员列表页面 (`admin_list.html`)，该页面加载后的显示效果如图 - 4 所示：



图 - 4

单击页面上表格中的“修改”按钮，则在界面显示“修改成功”的提示信息，页面效果

如图 - 5 所示：

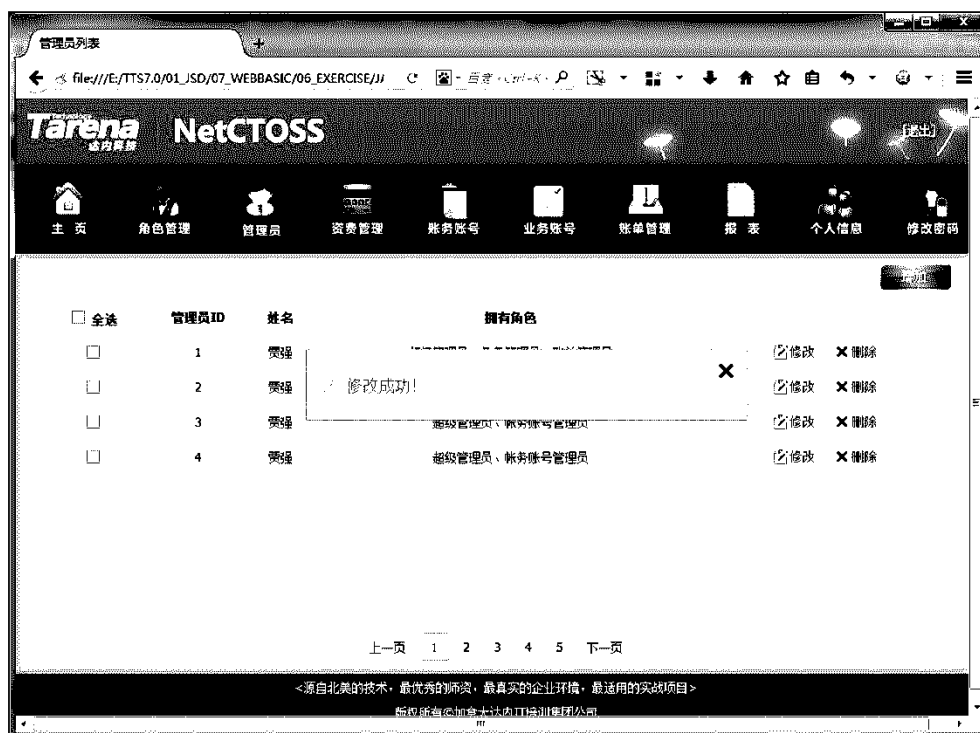


图 - 5

单击页面上表格中的“删除”按钮，先询问是否删除，页面效果如图 - 6 所示：

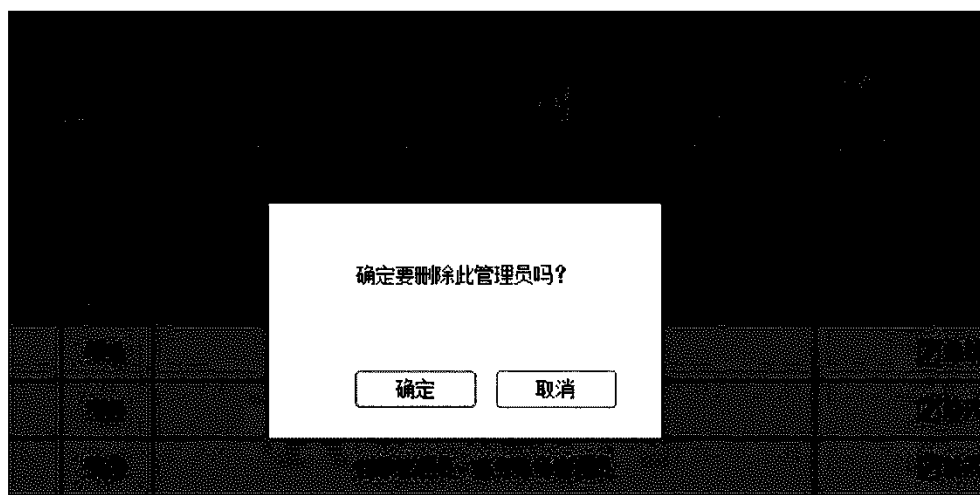


图 - 6

然后在界面显示“删除失败”的提示信息，页面效果如图 - 7 所示：



图 - 7

图 - 5 和图 - 7 所示页面上，弹出提示框的右上角有一个图片“X”，单击此图片，可以关闭提示框。

如果单击表格中的“全选”复选框，则选中表格中的所有复选框，页面效果如图 - 8 所示：

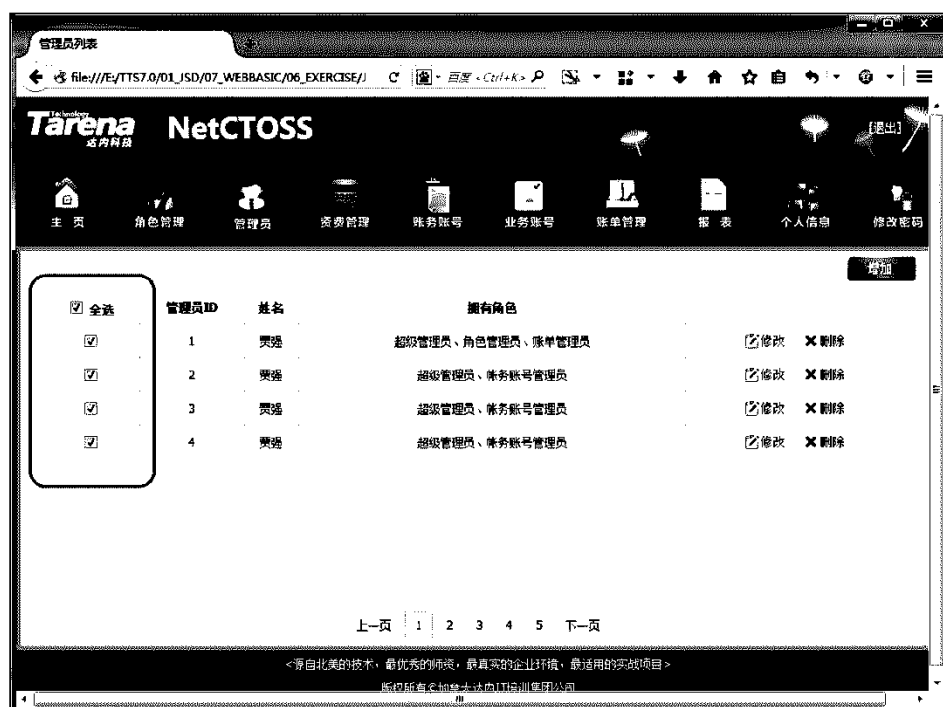


图 - 8

再单击表格中的“全选”复选框，则取消所有的选中。

JavaScript

Unit03

知识体系.....Page 111

document 对象	DOM 操作 - 查询	查询节点
		根据元素 ID 查找节点
		根据层次查找节点
		根据标签名查找节点
		根据 name 属性查找节点
	DOM 操作 - 增加	创建新节点
		添加新节点
	DOM 操作 - 删除	删除节点
HTML DOM 对象	HTML DOM 概述	HTML DOM 概述
		常用 HTML DOM 对象
		标准 DOM 与 HTML DOM
	Select 与 Option 对象	Select 对象
		Option 对象

经典案例.....Page 121

表单的验证与提交	根据元素 id 查找节点
购物车 - 修改购物数量	根据层次查找节点
购物车 - 购物金额计算	根据标签名查找节点
动态创建页面元素	创建新节点
	添加新节点
联动菜单	删除节点
联动菜单 (HTML DOM 实现)	Select 对象
	Option 对象

课后作业.....Page 154

1. document 对象

1.1. DOM 操作 - 查询

1.1.1. 【DOM 操作—查询】查询节点

Tarena 达内科技

查询节点

- 如果需要操作 HTML 元素，必须首先找到该元素
- 查询节点
 - 通过 id 查询
 - 通过层次（节点关系）查询
 - 通过标签名称查询
 - 通过 name 属性查询

+

1.1.2. 【DOM 操作—查询】根据元素 ID 查找节点

Tarena 达内科技

根据元素 ID 查找节点

- 方法：document.getElementById()
 - 通过指定的 ID 来返回元素节点，忽略文档的结构
 - 查找整个 HTML 文档中的任何 HTML 元素
 - 如果 ID 值错误，则返回 null

html 文档：

```
<p id="p1">一周畅销<span>榜</span></p>
```

js 代码：

```
var o = document.getElementById("p1");
alert(o.innerHTML);
```

+

Tarena 达内科技

根据元素 ID 查找节点（续1）

CSS 样式：

```
<style>
h1.style1 {
border-top:1px solid black;
background-color:silver;
font-weight:normal;
}
</style>
```

html 文档：

```
<input id="txt1" type="text">
<h1 id="h1">h1 text</h1>
```

js 代码：

```
var o = document.getElementById("txt1");
txt1.value = "text data";
txt1.style.color = "gray";

var o1 = document.getElementById("h1");
h1.innerHTML = "new text";
h1.className = "style1";
```

+

1.1.3. 【DOM 操作—查询】根据层次查找节点

Tarena 达内科技

根据层次查找节点

- parentNode、firstChild 和 lastChild
 - 遵循文档的上下层次结构，查找单个节点
- childNodes
 - 遵循文档的上下层次结构，查找多个节点
- previousSibling
 - 前一个同级节点
- nextSibling
 - 后一个同级节点

+

Tarena 达内科技

根据层次查找节点（续1）

```

<p>段落文本</p>
<select id="select1">
  <option value="1">a</option>
  <option value="2">b</option>
  <option value="3">c</option>
</select> <a href="me.html">click me</a>
            
```

html 文档：

```

var s1 = document.getElementById("select1");
//子节点
alert(s1.childNodes.length); //7
var o1 = s1.firstChild;
alert(o1.innerHTML); //undefined
//同级节点
alert(s1.nextSibling.innerHTML); //click me
alert(s1.previousSibling.innerHTML); //undefined
            
```

js 代码：

为什么子节点的个数为 7？

为什么是 undefined？

+

Tarena 达内科技

根据层次查找节点（续2）

- 若将 html 代码修改为如下格式

```

<p>段落文本</p> <select id="select1"> <option
value="1">a</option> <option value="2">b</option> <option
value="3">c</option> </select> <a href="me.html">click
me</a>
            
```

删除HTML 文档中的空白部分


```

var s1 = document.getElementById("select1");
//子节点
alert(s1.childNodes.length); //3
var o1 = s1.firstChild;
alert(o1.innerHTML); //a
//同级节点
alert(s1.nextSibling.innerHTML); //click me
alert(s1.previousSibling.innerHTML); //段落文本
            
```

HTML 文档中，空白部分也是节点


+

1.1.4. 【DOM 操作—查询】根据标签名查找节点




根据标签名查找节点

- `getElementsByTagName()` : 根据指定的标签名称返回所有的元素
 - 忽略文档的结构
 - 查找整个 HTML 文档中的所有元素
 - 如果标签名称错误, 则返回 长度为 0 的节点列表
- 返回一个节点列表 (数组)
 - 使用节点列表的 `length` 属性获取个数
 - `[index]` : 定位具体的元素



知识讲解

++



根据标签名查找节点 (续1)


html 文档 :

```
<body>
  <p>段落1</p>
  <div>
    <p>div 中的段落1</p>
    <p>div 中的段落2</p>
  </div>
</body>
```

js 代码 :


```
var pNodes =
  document.getElementsByTagName("p");
alert(pNodes.length); //3
alert(pNodes[1].innerHTML); //div 中的段落1
```

**getElementsByTagName 方法忽略文档的结构
对 document 进行查询, 即, 在整棵树上查询**



知识讲解

++



根据标签名查找节点 (续2)


html 文档 :

```
<body>
  <p>段落1</p>
  <div id="div1">
    <p>div 中的段落1</p>
    <p>div 中的段落2</p>
  </div>
</body>
```

js 代码 :

```
var divObj = document.getElementById("div1");
var pNodes = divObj.getElementsByTagName("p");
alert(pNodes.length); //2
alert(pNodes[1].innerHTML); //div 中的段落2
```

对 div 进行查询, 则, 只在 div 的范围内查询



知识讲解

++

1.1.5. 【DOM 操作—查询】根据 name 属性查找节点

Tarena
达内科技

根据 name 属性查找节点

- `getElementsByName()` : 根据标签的 name 属性的值进行查询

html 文档:

```
<input type="radio" name="sex" value="0" /> male
<input type="radio" name="sex" value="1" /> female
<br /> <br />
<input type="radio" name="state" value="0" /> 开通
<input type="radio" name="state" value="1" /> 暂停
<input type="radio" name="state" value="2" /> 删除
```

js 代码:

```
var nodes = document.getElementsByName("sex");
alert(nodes.length); //2

nodes = document.getElementsByName("state");
alert(nodes.length); //3
```

++

1.2. DOM 操作 - 增加

1.2.1. 【DOM 操作—增加】创建新节点

Tarena
达内科技

创建新节点

- `document.createElement(elementName)`
 - elementName : 要创建的元素标签名称
 - 返回新创建的节点
- 设置节点信息

```
//创建新节点
var newNode = document.createElement("input");

//设置节点的信息
newNode.type = "text";
newNode.value = "mary";
newNode.style.color = "red";
```

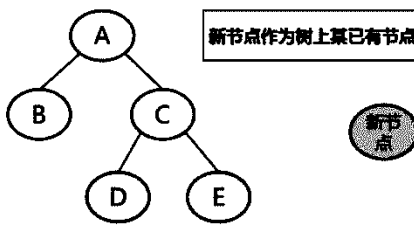
++

1.2.2. 【DOM 操作—增加】添加新节点

Tarena
达内科技

添加新节点

- `parentNode.appendChild(newNode)`
 - 追加: 新节点作为父节点的最后一个子节点存在
- `parentNode.insertBefore(newNode, refNode)`
 - refNode : 参考节点, 新节点位于此节点之前



```
graph TD
    A((A)) --- B((B))
    A --- C((C))
    C --- D((D))
    C --- E((E))
    F((新节点))
```

新节点作为树上某已有节点的子节点

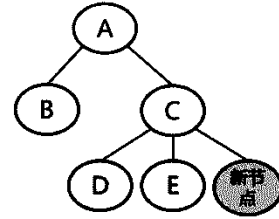
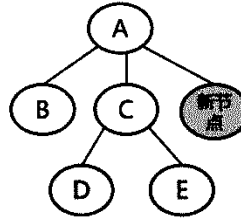
++

添加新节点 (续1)

Tarena
达内科技

A.appendChild(新节点);

C.appendChild(新节点);



打印并播放

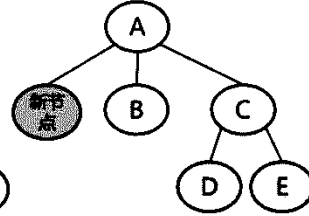
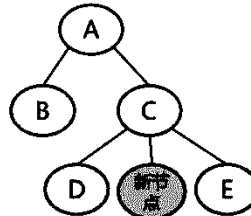


添加新节点 (续2)

Tarena
达内科技

C.insertBefore(新节点,E);

A.insertBefore(新节点,B);



B.insertBefore(新节点,D)---???? ✗

打印并播放



添加新节点 (续3)

Tarena
达内科技

html 文档:

```

<body>
  <form id="form1">
    <input id="btn1" type="button" value="
添加新节点" onclick="createNewNode();" />
  </form>
</body>
  
```



打印并播放



Tarena 达内科技

添加新节点 (续4)

js 代码:

```
function createNewNode() {
    var form = document.getElementById("form1");
    //创建一个新的文本框节点
    var textNode = document.createElement("input");
    textNode.type = "text";
    textNode.value = "mary";
    //加入到 form 中, 作为form的最后一个子节点
    form.appendChild(textNode);



    //创建一个新的 h1 节点, 加入到原有按钮之前
    var h1Node = document.createElement("h1");
    h1Node.innerHTML = "h1 text";
    form.insertBefore(h1Node,
        document.getElementById("btn1"));
}
```

+

Tarena 达内科技

添加新节点 (续5)

- 创建一个新的文本框节点
- 文本框加入到 form 中, 作为form的最后一个子节点
- 创建一个新的 h1 节点, 加入到原有按钮之前

+

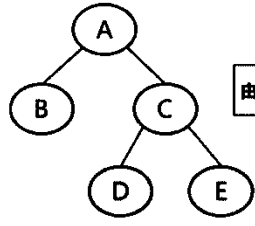
1.3. DOM 操作 - 删除

1.3.1. 【DOM 操作—删除】删除节点

Tarena 达内科技

删除节点

- node.removeChild(childNode)
 - 删除某个子节点
 - childNode 必须是 node 的子节点
- childNode.parentNode.removeChild(childNode)



由父节点删除其下属的某个子节点

+

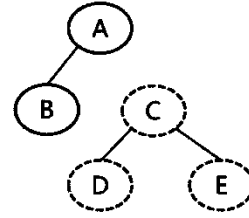
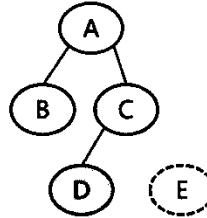
删除节点 (续1)

Tarena
达内科技

```
C.removeChild(E);
//or
E.parentNode.removeChild(E);
```

```
A.removeChild(C);
//or
C.parentNode.removeChild(C);
```

知识讲解



删除节点 (续2)

Tarena
达内科技

CSS 代码:

```
div{
    border:1px solid black;
    width:100px;
    height:70px;
}
```

html 文档:

```
<form>
  <input id="btn1" type="button" value="删除节点"
    onclick="deleteNode();" />
  <div id="div1">
    <a href="#">link1</a>
    <p>段落</p>
  </div>
</form>
```



删除节点 (续3)

Tarena
达内科技

js 代码:

```
function deleteNode()
{
    var delNode = document.getElementById( "div1");
    delNode.parentNode.removeChild(delNode);
}
```



2. HTML DOM 对象

2.1. HTML DOM 概述

2.1.1. 【HTML DOM 概述】HTML DOM 概述

Tarena
达内科技

HTML DOM概述

- HTML DOM 定义了用于 HTML 的一系列标准的对象，以及访问和处理 HTML 文档的标准方法
- HTML 标签对象化
 - 将网页中的每个元素都看作一个对象

++

2.1.2. 【HTML DOM 概述】常用 HTML DOM 对象

Tarena
达内科技

常用 HTML DOM 对象

++

Tarena
达内科技

常用 HTML DOM 对象 (续1)

js 代码:

```
function createNewNode()
{
    var form = document.getElementById("form1");
    //创建一个新的图片对象，加入到 form 中
    var newNode = new Image();
    newNode.src = "a.jpg";
    form.appendChild(newNode);
}
```

html 文档:

```
<form id= "form1" >
  <input type= "button" value= "创建新节点"
    onclick="createNewNode();" />
</form>
```

将HTML对象中的每个元素都对象化

++

2.1.3. 【HTML DOM 概述】标准 DOM 与 HTML DOM

知识讲解

标准 DOM 与 HTML DOM

Tarena
达内科技

- 标准 DOM 提供了统一的操作接口
 - createElement
 - appendChild
 - setAttribute
 - removeAttribute
 - nodeName
 - ...
- HTML DOM 提供了封装好的各种对象
 - Image
 - Select
 - Option
 - ...

++

知识讲解

标准 DOM 与 HTML DOM (续1)

Tarena
达内科技

- 标准 DOM 的实现方式

var newNode = document.createElement("img");
- HTML DOM 的实现方式

var newNode = new Image() ;

++

知识讲解

标准 DOM 与 HTML DOM (续2)

Tarena
达内科技


- 操作节点，如创建、删除、查找等
 - 使用标准 DOM 操作
- 操作属性，如读取或者修改属性的值
 - 使用 HTML DOM 操作

++

119

2.2. Select 与 Option 对象


2.2.1. 【Select 与 Option 对象】Select 对象




Select 对象

- Select 对象代表 HTML 表单中的一个下拉列表
 - <select> 标签即表示一个 Select 对象
- 常用属性
 - options、selectedIndex、size
- 常用方法
 - add(option)、remove(index)
- 事件
 - onchange

```
<select id="sel1" onchange="showInfo();">
  <option value="11">aa</option>
  <option value="22">bb</option>
</select>
```




2.2.2. 【Select 与 Option 对象】Option 对象




Option 对象

- Option 对象代表 HTML 表单中下拉列表中的一个选项
 - <option> 标签表示一个 Option 对象
- 创建对象
 - var o = new Option(text,value);
- 常用属性
 - index、text、value、selected

```
function showInfo() {
  var selObj = document.getElementById("sel1");
  var i = selObj.selectedIndex;
  alert(selObj.options[i].value);
}
```





Option 对象 (续1)


js 代码：

```
function selFunc() {
  var selObj = document.getElementById("s1");
  var value = selObj.options[selObj.selectedIndex].value;
  alert(value);

  var option = new Option("cc", "33");
  selObj.add(option);
}
```

HTML 文档：

```
<select id="s1" onchange="selFunc();">
  <option value="1">aa</option>
  <option value="2">bb</option>
</select>
```



经典案例

1. 表单的验证与提交

- 问题

用户需要在页面上录入用户名和电话号码，录入的要求如图 - 1 所示：

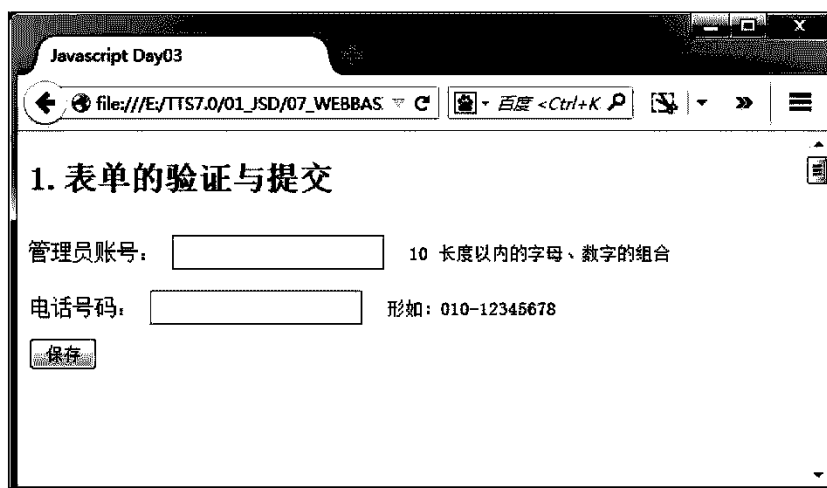


图 - 1

用户在文本框中录入文本时（文本框获得焦点），文本框的样式会发生变化。页面效果如图 - 2 所示：

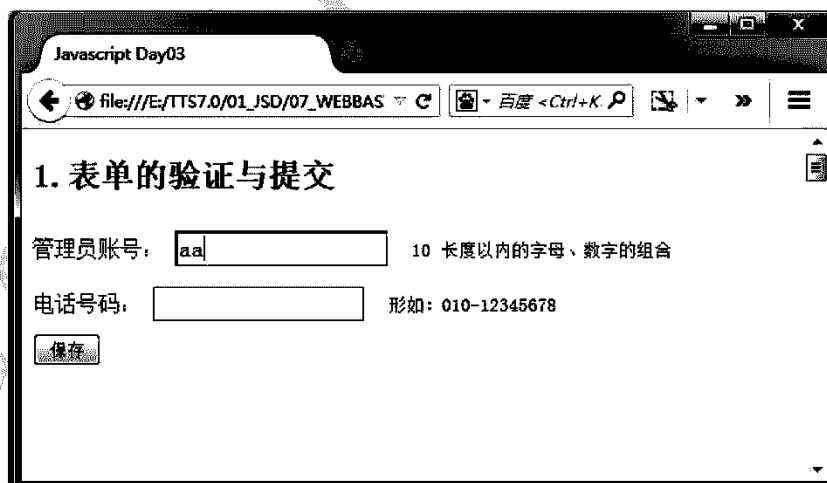


图 - 2

用户录入完毕（文本框失去焦点），文本框的样式复原，且验证文本框中的录入是否符合要求。如果符合要求，页面效果如图 - 3 所示：



图 - 3

用户录入完毕后，如果文本框中的录入没有通过验证，即不符合要求，则页面效果如图 - 4 所示：



图 - 4

如果用户单击页面上的“保存”按钮，也会逐一验证页面上两个文本框中的录入：如果验证通过，则发生提交（页面刷新）；否则，页面不提交。

• 方案

分析此案例可以看出，页面上的文本框和提示信息部分都需要两种样式，分别用于不同的业务状况。因此，首先需要为页面定义 CSS 样式：文本框获得焦点和失去焦点的样式，提示信息部分验证通过和验证失败的样式。

然后为页面元素定义事件：

1、需要为文本框定义获得焦点事件 onfocus 和失去焦点事件 onblur。在 onfocus 事件中，修改文本框的样式；在 onblur 事件中，修改文本框的样式并进行输入验证。

2、为“保存”按钮定义单击事件，并在事件中逐一验证两个文本框中的录入，并根据验证结果决定页面是否提交。

最后定义 JavaScript 代码。在代码中，需要使用正则表达式对象来实现输入验证功能，并通过 className 属性修改元素的样式类。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建页面

创建文件 js_demo3.html，并在 html 页面上添加代码以创建一个标准结构的 HTML 文档，并在文档中创建表单。html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day03</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  </head>
  <body>
    <form>
    </form>
  </body>
</html>
```

步骤二：创建 js 文件和 css 文件

此案例中需要用到 CSS 样式和 JavaScript 代码，因此，首先创建文件 js_demo3.js，用于书写 JavaScript 代码；然后创建文件 myStyle.css，用于书写 CSS 样式代码。并在上一步所创建的 HTML 文档中引入脚本文件和样式文件。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day03</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />

    <script language="javascript" src="js_demo3.js" type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />

  </head>
  <body>
    <form>
    </form>
  </body>
</html>
```

步骤三：为页面添加内容

在 <form> 元素中添加文本框、提示文本以及提交按钮。因为需要在 js 代码中获取文本框中的录入，以及修改提示信息文本，因此需要为文本框和提示信息文本分别定义 id 属性。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day03</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js/demo3.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <h2>1. 表单的验证与提交</h2>

      <!--账号-->
      管理员账号 :
      <input type="text" class="txt" id="txtAccount" />
      <span id="accountInfo">10 长度以内的字母、数字的组合</span>
      <br />
      <!--电话号码-->
      电话号码 :
      <input type="text" class="txt" id="txtPhone" />
      <span id="phoneInfo">形如：010-12345678</span>
      <br />
      <!--操作按钮-->
      <input type="submit" value="保存" />

    </form>
  </body>
</html>
```

步骤四：为文本框定义样式

在文件 myStyle.css 中添加代码，为文本框定义样式。为文本框定义两种样式类：txt 和 txt_focus，分别表示文本框失去焦点时的样式和获取焦点时的样式。

css 文件中的代码如下所示：

```
/* 文本框 */
input.txt,input.txt_focus
{
  border:1px solid gray;
  font-size:13pt;
  line-height:18pt;
  width:150px;
}
/* 文本框获得焦点时的样式 */
input.txt focus
{
  border-top:2px solid black;
  border-left:2px solid black;
}
```

步骤五：为提示信息文本定义样式

在文件 myStyle.css 中继续添加代码，为页面上的提示信息文本部分定义样式。首先

为表示提示信息的 元素定义通用的样式，如边距、字体和行高等；然后分别定义验证通过和验证失败的样式。css 文件中添加的代码如下所示：

```
/* 验证信息框 */
#accountInfo, #phoneInfo
{
    margin: 5px 10px;
    padding-top: 5px;
    padding-bottom: 5px;
    font-size: 10pt;
    background-repeat: no-repeat;
    background-position: left center;
    line-height: 40px;
}

/* 验证消息：验证通过时的样式 */
span.vali success
{
    padding-left: 20px;
    background-image: url("ok.png");
}

/* 验证消息：验证失败时的样式 */
span.vali fail
{
    padding-left: 20px;
    background-image: url("warning.png");
    border: 1px solid red;
    background-color: #fdecec;
    color: red;
}
```

步骤六：为文本框定义 onfocus 和 onblur 事件

修改 html 代码，为文本框定义 onfocus 和 onblur 事件。

在文本框的 onfocus 事件中，直接添加 js 代码，设置文本框的 className 属性的值为 txt_focus 即可；在 onblur 事件中，分别调用方法 valiAccount 和 valiPhone 实现输入的验证。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day03</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js/demo3.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <h2>1. 表单的验证与提交</h2>

      <!--账号-->
      管理员账号：
      <input type="text" class="txt" id="txtAccount" onfocus="this.className
= 'txt_focus';" onblur="valiAccount();" />
```

```

        <span id="accountInfo">10 长度以内的字母、数字的组合</span>
        <br />
        <!--电话号码-->
        电话号码：
        <input type="text" class="txt" id="txtPhone" onfocus="this.className =
'txt_focus';" onblur="valiPhone();" />
        <span id="phoneInfo">形如：010-12345678</span>

        <br />
        <!--操作按钮-->
        <input type="submit" value="保存" />
    </form>
</body>
</html>

```

步骤七：验证账号

在文件 js_demo3.js 中创建名为 valiAccount 的方法，然后为方法添加代码：复原文本框的样式，获取账号并根据录入要求创建正则表达式对象，然后进行验证，最后根据验证结果设置文本的样式以及内容，并返回验证结果。js 文件中添加的代码如下所示：

```

//验证账号的录入
function valiAccount() {
    //获取账号，并复原文本框的样式
    document.getElementById("txtAccount").className = "txt";
    var account = document.getElementById("txtAccount").value;
    //定义正则表达式进行验证
    var reg = /^[A-Za-z0-9]{1,10}$/;
    var error = reg.test(account);

    //判断验证结果
    var spanObj = document.getElementById("accountInfo");
    if (error) {
        spanObj.innerHTML = "";
        spanObj.className = "vali success";
    }
    else {
        spanObj.innerHTML = "10 长度以内的字母、数字的组合";
        spanObj.className = "vali fail";
    }

    //返回验证结果
    return error;
}

```

步骤八：验证电话号码

在文件 js_demo3.js 中再创建名为 valiPhone 的方法，然后为方法添加代码，内容与方法 valiAccount 类似。js 文件中添加的代码如下所示：

```

//验证电话号码的录入
function valiPhone() {
    //获取电话号码，并复原文本框的样式

```

```
document.getElementById("txtPhone").className = "txt";
var phone = document.getElementById("txtPhone").value;

//定义正则表达式进行验证
var reg = /^\\d{3}-\\d{8}$/;
var error = reg.test(phone);

//判断验证结果
var spanObj = document.getElementById("phoneInfo");
if (error) {
    spanObj.innerHTML = "";
    spanObj.className = "vali success";
}
else {
    spanObj.innerHTML = "形如：010-12345678";
    spanObj.className = "vali_fail";
}
//返回验证结果
return error;
}
```

步骤九：定义验证所有数据的方法

在文件 js_demo3.js 中再创建名为 validateDatas 的方法，在该方法中，逐一调用前面步骤中所定义的验证账号以及验证电话号码的方法。

valiAccount 方法和 valiPhone 方法均返回 bool 类型的结果，只有当两个方法的返回值均为 true 时，才表示验证通过。因此，在方法 validateDatas 中，需要将两个方法的返回值的逻辑“与”作为最终结果，返回给调用方。js 文件中添加的代码如下所示：

```
//验证各项的录入
function validateDatas() {
    //验证账号
    var r1 = valiAccount();
    //验证电话号码
    var r2 = valiPhone();
    //返回结果
    return r1 && r2;
}
```

步骤十：为“保存”按钮定义单击事件

修改 html 代码，为“保存”按钮定义单击事件。

在按钮的 onclick 事件中，调用上一步中所创建的方法 validateDatas，并将该方法的返回值返回给事件。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Javascript Day03</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <script language="javascript" src="js_demo3.js"
type="text/javascript">
        </script>
        <link href="myStyle.css" type="text/css" rel="stylesheet" />
```

```

</head>
<body>
  <form>
    <h2>1. 表单的验证与提交</h2>
    <!--账号-->
    管理员账号：
    <input      type="text"      class="txt"      id="txtAccount"
onfocus="this.className = 'txt_focus';" onblur="valiAccount();" />
    <span id="accountInfo">10 长度以内的字母、数字的组合</span>
    <br />
    <!--电话号码-->
    电话号码：
    <input      type="text"      class="txt"      id="txtPhone"
onfocus="this.className = 'txt_focus';" onblur="valiPhone();" />
    <span id="phoneInfo">形如：010-12345678</span>
    <br />
    <!--操作按钮-->

    <input type="submit" value="保存" onclick="return validateDatas();" />

  </form>
</body>
</html>

```

• 完整代码

js_demo3.html 文件的代码如下所示：

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day03</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script      language="javascript"      src="js_demo3.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <h2>1. 表单的验证与提交</h2>
      <!--账号-->
      管理员账号：
      <input      type="text"      class="txt"      id="txtAccount"
onfocus="this.className = 'txt_focus';" onblur="valiAccount();" />
      <span id="accountInfo">10 长度以内的字母、数字的组合</span>
      <br />
      <!--电话号码-->
      电话号码：
      <input      type="text"      class="txt"      id="txtPhone"
onfocus="this.className = 'txt_focus';" onblur="valiPhone();" />
      <span id="phoneInfo">形如：010-12345678</span>
      <br />
      <!--操作按钮-->
      <input type="submit" value=" 保 存 " onclick="return
validateDatas();" />
    </form>

```

```
</body>
</html>
```

myStyle.css 文件中的代码如下所示：

```
/* 文本框 */
input.txt,input.txt focus
{
    border:1px solid gray;
    font-size:13pt;
    line-height:18pt;
    width:150px;
}
/* 文本框获得焦点时的样式 */
input.txt_focus
{
    border-top:2px solid black;
    border-left:2px solid black;
}
/* 验证信息框 */
#accountInfo,#phoneInfo
{
    margin:5px 10px;
    padding-top:5px;
    padding-bottom:5px;
    font-size:10pt;
    background-repeat:no-repeat;
    background-position:left center;
    line-height:40px;
}

/* 验证消息：验证通过时的样式 */
span.vali_success
{
    padding-left:20px;
    background-image:url("ok.png");
}
/* 验证消息：验证失败时的样式 */
span.vali_fail
{
    padding-left:20px;
    background-image:url("warning.png");
    border:1px solid red;
    background-color:#fdecec;
    color:red;
}
```

js_demo3.js 文件的代码如下所示：

```
//验证账号的录入
function valiAccount() {
    //获取账号，并复原本文本框的样式
    document.getElementById("txtAccount").className = "txt";
    var account = document.getElementById("txtAccount").value;
    //定义正则表达式进行验证
    var reg = /^[A-Za-z0-9]{1,10}$/;
    var error = reg.test(account);

    //判断验证结果
    var spanObj = document.getElementById("accountInfo");
    if (error) {
        spanObj.innerHTML = "";
    }
}
```



```

        spanObj.className = "vali success";
    }
    else {
        spanObj.innerHTML = "10 长度以内的字母、数字的组合";
        spanObj.className = "vali_fail";
    }

    //返回验证结果
    return error;
}

//验证电话号码的录入
function valiPhone() {
    //获取电话号码，并复原本框的样式
    document.getElementById("txtPhone").className = "txt";
    var phone = document.getElementById("txtPhone").value;

    //定义正则表达式进行验证
    var reg = /^\\d{3}-\\d{8}$/;
    var error = reg.test(phone);

    //判断验证结果
    var spanObj = document.getElementById("phoneInfo");
    if (error) {
        spanObj.innerHTML = "";
        spanObj.className = "vali success";
    }
    else {
        spanObj.innerHTML = "形如: 010-12345678";
        spanObj.className = "vali_fail";
    }
    //返回验证结果
    return error;
}

//验证各项的录入
function validateDatas() {
    //验证账号
    var r1 = valiAccount();
    //验证电话号码
    var r2 = valiPhone();
    //返回结果
    return r1 && r2;
}

```

2. 购物车 - 修改购物数量

- 问题

有购物车页面，页面效果如图 - 5 所示：

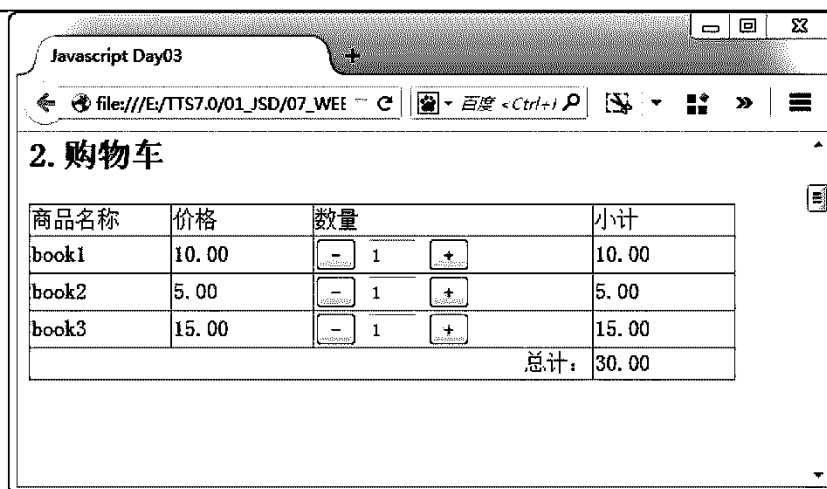


图 - 5

单击表格中的“+”、“-”按钮可以实现购买数量的增减。单击某行中的“-”按钮，则将该按钮后的文本框中的数量减1，如果数量已经为0，则不再递减；单击某行中的“+”按钮则将该按钮前的文本框中的数量加1。

注：此案例中，只实现修改购物数量的功能，暂不实现购物金额计算的功能。

• 方案

实现此案例时，比较普通的做法是，为表格中的每个文本框定义 id 属性，然后每个按钮的单击事件中，将文本框的 id 值作为参数传入 js 代码，这样，就可以在 js 代码中通过 id 找到文本框，并修改文本框中的数量。html 代码大致如下所示：

```
<td>
    <input type="button" value="-" onclick="decrease('txt1');" />
    <input type="text" value="1" id="txt1" />
    <input type="button" value="+" onclick="increase('txt1');" />
</td>
```

这种解决方案的缺点是，需要为表格中的每一个文本框定义不同的 id 值，然后需要为每个按钮传入不同的参数值。考虑到表格中的行可能比较多，这样的方式不利于代码的维护。因此，可以考虑换一种解决方案。

分析图 - 5 可以看出，如果通过单击的按钮对象就能够找到所对应的文本框对象，就可以解决上述方案的缺点：大量 id 属性的定义问题。

因此，建议使用的解决方案如下：

- 1、使用 this 关键字将按钮节点对象作为参数传入 js 方法；
- 2、在 js 方法中，通过按钮节点找到它的父节点，即 <td> 元素；
- 3、找到 <td> 节点的所有子节点（包含两个按钮、一个文本框和一些空白文本节点）；
- 4、循环所有的子节点，找到文本框节点；
- 5、修改文本框中的数量。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加购物车表格

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加购物车表格。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day03</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js/demo3.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>2.购物车</h2>
      <table id="shoppingCart">
        <tr>
          <td style="width:100px;">商品名称</td>
          <td style="width:100px;">价格</td>
          <td style="width:200px;">数量</td>
          <td style="width:100px;">小计</td>
        </tr>
        <tr>
          <td>book1</td>
          <td>10.00</td>
          <td>
            <input type="button" value="-" />
            <input type="text" value="1" />
            <input type="button" value="+" />
          </td>
          <td>10.00</td>
        </tr>
        <tr>
          <td>book2</td>
          <td>5.00</td>
          <td>
            <input type="button" value="-" />
            <input type="text" value="1" />
            <input type="button" value="+" />
          </td>
          <td>5.00</td>
        </tr>
        <tr>
          <td>book3</td>
          <td>15.00</td>
          <td>
            <input type="button" value="-" />
            <input type="text" value="1" />
            <input type="button" value="+" />
          </td>
          <td>

```

```

        <td>15.00</td>
      </tr>
      <tr>
        <td style="text-align:right;" colspan="3">总计:</td>
        <td>30.00</td>
      </tr>
    </table>

  </form>
</body>
</html>

```

步骤二：为表格定义样式

在文件 myStyle.css 中添加代码，为表格定义样式：定义单元格的边框、表格边框合并、文本框和按钮的宽度。

css 文件中添加的代码如下所示：

```

/*购物车*/
table
{
  border-collapse:collapse;
}
td
{
  border:1px solid gray;
}
table input
{
  width:30px;
}

```

步骤三：为按钮定义单击事件

为表格中的 “+”、“-” 按钮定义单击事件，调用相应的方法，并使用 this 关键字将当前按钮节点对象传入方法。html 代码中添加的事件定义如下所示：

```

<!--其他代码，略-->
<tr>
  <td>book1</td>
  <td>10.00</td>
  <td>

    <input type="button" value="-" onclick="decrease(this);" />

    <input type="text" value="1" />

    <input type="button" value="+" onclick="increase(this);" />

  </td>
  <td>10.00</td>
</tr>
<!--其他代码，略-->

```

步骤四：在 js 文件中定义方法 decrease

打开文件 js_demo3.js，在其中添加代码，定义名为 decrease 的方法，并为其定义参数以接收传入的按钮节点对象。js 文件中添加的代码如下所示：

```
//减少数量
function decrease(btnObj) {
}
```

步骤五：减少数量

为 decrease 方法添加代码。

首先通过传入的按钮节点对象得到当前单元格中的所有节点对象，得到一个节点的数组；然后循环该数组，通过节点的 nodeName 属性和 type 属性进行判断，找到当前单元格中的文本框节点；获取文本框中的文本，转换为数值类型；如果数值大于等于 1，则将数值减 1；最后，将新数值显示在文本框中。

代码如下所示：

```
//减少数量
function decrease(btnObj) {
    var nodes = btnObj.parentNode.childNodes;
    for (var i = 0; i < nodes.length; i++) {
        if (nodes[i].nodeName == "INPUT" && nodes[i].type == "text") {
            var oldCount = parseInt(nodes[i].value);
            if (oldCount >= 1) {
                var newCount = oldCount - 1;
                nodes[i].value = newCount;
            }
        }
    }
}
```

步骤六：在 js 文件中定义方法 increase

在文件 js_demo3.js 中继续添加代码，定义名为 increase 的方法，为其定义参数以接收传入的按钮节点对象，并为其添加代码实现数量的增加。js 文件中添加的代码如下所示：

```
//增加数量
function increase(btnObj) {
    var nodes = btnObj.parentNode.childNodes;
    for (var i = 0; i < nodes.length; i++) {
        if (nodes[i].nodeName == "INPUT" && nodes[i].type == "text") {
            var oldCount = parseInt(nodes[i].value);
            var newCount = oldCount + 1;
            nodes[i].value = newCount;
        }
    }
}
```

• 完整代码

js_demo3.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day03</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js/demo3.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码, 略-->

      <h2>2. 购物车</h2>
      <table id="shoppingCart">
        <tr>
          <td style="width:100px;">商品名称</td>
          <td style="width:100px;">价格</td>
          <td style="width:200px;">数量</td>
          <td style="width:100px;">小计</td>
        </tr>
        <tr>
          <td>book1</td>
          <td>10.00</td>
          <td>
            <input type="button" value="-"
onclick="decrease(this);" />
            <input type="text" value="1" />
            <input type="button" value="+"
onclick="increase(this);" />
          </td>
          <td>10.00</td>
        </tr>
        <tr>
          <td>book2</td>
          <td>5.00</td>
          <td>
            <input type="button" value="-"
onclick="decrease(this);" />
            <input type="text" value="1" />
            <input type="button" value="+"
onclick="increase(this);" />
          </td>
          <td>5.00</td>
        </tr>
        <tr>
          <td>book3</td>
          <td>15.00</td>
          <td>
            <input type="button" value="-"
onclick="decrease(this);" />
            <input type="text" value="1" />
            <input type="button" value="+"
onclick="increase(this);" />
          </td>
          <td>15.00</td>
        </tr>
        <tr>
          <td style="text-align:right;" colspan="3">总计: </td>
          <td>30.00</td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

myStyle.css 文件中的代码如下所示：

```
/* 其他样式部分，略 */

/*购物车*/
table
{
    border-collapse:collapse;
}
td
{
    border:1px solid gray;
}
table input
{
    width:30px;
}
```

js_demo3.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//减少数量
function decrease(btnObj) {
    var nodes = btnObj.parentNode.childNodes;
    for (var i = 0; i < nodes.length; i++) {
        if (nodes[i].nodeName == "INPUT" && nodes[i].type == "text") {
            var oldCount = parseInt(nodes[i].value);
            if (oldCount >= 1) {
                var newCount = oldCount - 1;
                nodes[i].value = newCount;
            }
        }
    }
}

//增加数量
function increase(btnObj) {
    var nodes = btnObj.parentNode.childNodes;
    for (var i = 0; i < nodes.length; i++) {
        if (nodes[i].nodeName == "INPUT" && nodes[i].type == "text") {
            var oldCount = parseInt(nodes[i].value);
            var newCount = oldCount + 1;
            nodes[i].value = newCount;
        }
    }
}
```

3. 购物车 - 购物金额计算

- 问题

继续修改购物车页面，为页面添加计算小计和总计的功能。即，如果通过单击界面上的“-”或者“+”按钮修改了购物数量，则更新页面的小计和总计。页面效果如图 - 6 所示：

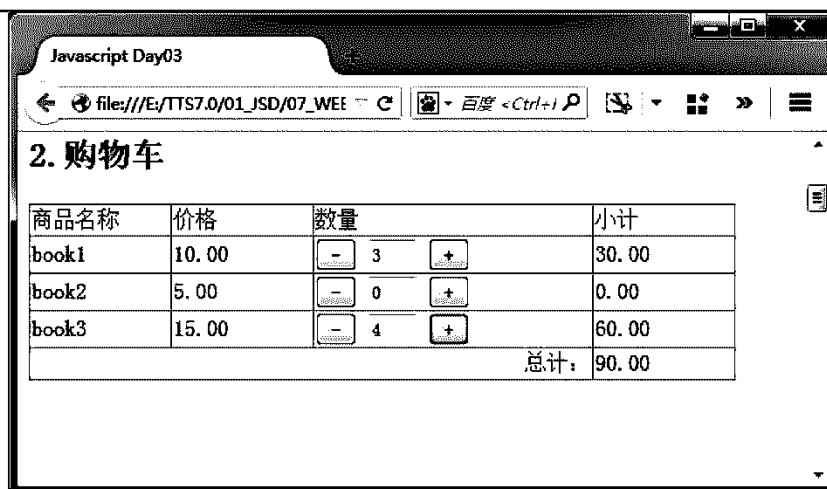


图 - 6

• 方案

本案例在上一个案例的基础上添加功能即可。解决方案是：定义用于计算小计和总计的方法 `calculate`，然后在增加数量和减少数量的方法中调用该方法即可。

在 `calculate` 方法中，需要找到购物车表格中所有的表行节点；然后从第二行开始循环表格中的行（最后一行除外）；对于每一行，需要完成如下操作：

- 1、查找当前行节点里的第二个 `td` 节点，以找到单价数值；
- 2、查找当前行节点里的第二个 `input` 节点（记载数量的文本框），以找到数量；
- 3、找到单价和数量后，转换为数值类型，然后计算小计，同时累积总计；
- 4、将小计显示在当前行的第四个 `td` 节点中；
- 5、循环完毕后，显示总计。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：在 `js` 文件中定义方法 `calculate`

打开文件 `js_demo3.js`，在其中添加代码，定义名为 `calculate` 的方法。`js` 文件中添加的代码如下所示：

```
//计算小计以及总计
function calculate() {
}
```

步骤二：循环表格的行

为 `calculate` 方法添加代码。首先定义变量用于累积总计，然后找到表格的所有表行节点对象，然后循环这些行（第一行和最后一行除外）。代码如下所示：

```
//计算小计以及总计
```



```
function calculate() {  
  
    //定义变量以存储总计  
    var total = 0;  
  
    //得到所有的行  
    var tableObj = document.getElementById("shoppingCart");  
    var trNodes = tableObj.getElementsByTagName("tr");  
  
    //从第二行开始统计，不统计最后一行  
    for (var i = 1; i < trNodes.length - 1; i++) {  
    }  
  
}
```

步骤三：查找单价和数量

查找当前行的第二个单元格中的文本，即为单价；找到当前行中的第二个 input 节点中的文本，即为数量。代码如下所示：

```
//计算小计以及总计  
function calculate() {  
    //定义变量以存储总计  
    var total = 0;  
  
    //得到所有的行  
    var tableObj = document.getElementById("shoppingCart");  
    var trNodes = tableObj.getElementsByTagName("tr");  
  
    //从第二行开始统计，不统计最后一行  
    for (var i = 1; i < trNodes.length - 1; i++) {  
  
        var tr = trNodes[i];  
  
        //得到单价和数量  
        var price = tr.getElementsByTagName("td")[1].innerHTML;  
        var quantity = tr.getElementsByTagName("input")[1].value;  
  
    }  
}
```

步骤四：计算小计并显示

根据单价和数量，计算小计，并显示在当前行的第四个单元格中。代码如下所示：

```
//计算小计以及总计  
function calculate() {  
    //定义变量以存储总计  
    var total = 0;  
  
    //得到所有的行  
    var tableObj = document.getElementById("shoppingCart");  
    var trNodes = tableObj.getElementsByTagName("tr");
```

```
//从第二行开始统计，不统计最后一行
for (var i = 1; i < trNodes.length - 1; i++) {
    var tr = trNodes[i];

    //得到单价和数量
    var price = tr.getElementsByTagName("td")[1].innerHTML;
    var quantity = tr.getElementsByTagName("input")[1].value;

    //计算小计并显示
    var sum = parseFloat(price) * parseFloat(quantity);
    tr.getElementsByTagName("td")[3].innerHTML = sum.toFixed(2);
}
}
```

步骤五：计算总计并显示

将小计的值累加起来（即为总计），循环完毕后，显示在表格最后一行的第二个单元格中。代码如下所示：

```
//计算小计以及总计
function calculate() {
    //定义变量以存储总计
    var total = 0;

    //得到所有的行
    var tableObj = document.getElementById("shoppingCart");
    var trNodes = tableObj.getElementsByTagName("tr");

    //从第二行开始统计，不统计最后一行
    for (var i = 1; i < trNodes.length - 1; i++) {
        var tr = trNodes[i];

        //得到单价和数量
        var price = tr.getElementsByTagName("td")[1].innerHTML;
        var quantity = tr.getElementsByTagName("input")[1].value;

        //计算小计并显示
        var sum = parseFloat(price) * parseFloat(quantity);
        tr.getElementsByTagName("td")[3].innerHTML = sum.toFixed(2);

        //累积总计
        total += sum;
    }

    //显示总计
    trNodes[trNodes.length - 1].getElementsByTagName("td")[1].innerHTML =
        total.toFixed(2);
}
```

步骤六：调用方法 calculate

在 increase 方法和 decrease 方法中，调用方法 calculate，从而实现，每当购物数量发生变化时，则重新计算小计和总计。修改后的代码如下所示：

```
//减少数量
function decrease(btnObj) {
    var nodes = btnObj.parentNode.childNodes;
    for (var i = 0; i < nodes.length; i++) {
        if (nodes[i].nodeName == "INPUT" && nodes[i].type == "text") {
            var oldCount = parseInt(nodes[i].value);
            if (oldCount >= 1) {
                var newCount = oldCount - 1;
                nodes[i].value = newCount;
            }
        }
    }
}

//计算
calculate();

}

//增加数量
function increase(btnObj) {
    var nodes = btnObj.parentNode.childNodes;
    for (var i = 0; i < nodes.length; i++) {
        if (nodes[i].nodeName == "INPUT" && nodes[i].type == "text") {
            var oldCount = parseInt(nodes[i].value);
            var newCount = oldCount + 1;
            nodes[i].value = newCount;
        }
    }
}

//计算
calculate();

}
```

• 完整代码

js_demo3.html 文件和 myStyle.css 文件中的代码没有变化。

js_demo3.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//减少数量
function decrease(btnObj) {
    var nodes = btnObj.parentNode.childNodes;
    for (var i = 0; i < nodes.length; i++) {
        if (nodes[i].nodeName == "INPUT" && nodes[i].type == "text") {
            var oldCount = parseInt(nodes[i].value);
            if (oldCount >= 1) {
                var newCount = oldCount - 1;
                nodes[i].value = newCount;
            }
        }
    }
}
```

```

    }
    //计算
    calculate();
}

//增加数量
function increase(btnObj) {
    var nodes = btnObj.parentNode.childNodes;
    for (var i = 0; i < nodes.length; i++) {
        if (nodes[i].nodeName == "INPUT" && nodes[i].type == "text") {
            var oldCount = parseInt(nodes[i].value);
            var newCount = oldCount + 1;
            nodes[i].value = newCount;
        }
    }
    //计算
    calculate();
}

//计算小计以及总计
function calculate() {
    //定义变量以存储总计
    var total = 0;

    //得到所有的行
    var tableObj = document.getElementById("shoppingCart");
    var trNodes = tableObj.getElementsByTagName("tr");

    //从第二行开始统计，不统计最后一行
    for (var i = 1; i < trNodes.length - 1; i++) {
        var tr = trNodes[i];

        //得到单价和数量
        var price = tr.getElementsByTagName("td")[1].innerHTML;
        var quantity = tr.getElementsByTagName("input")[1].value;

        //计算小计并显示
        var sum = parseFloat(price) * parseFloat(quantity);
        tr.getElementsByTagName("td")[3].innerHTML = sum.toFixed(2);

        //累积总计
        total += sum;
    }
    //显示总计
    trNodes[trNodes.length - 1].getElementsByTagName("td")[1].innerHTML =
    total.toFixed(2);
}

```

4. 动态创建页面元素

- 问题

页面上有按钮，用于动态的为页面添加新元素。页面效果如图 - 7 所示：



图 - 7

单击页面上的“添加新元素”按钮，则在按钮后添加一个超级链接元素，在按钮前添加一个新的按钮，页面效果如图 - 8 所示：

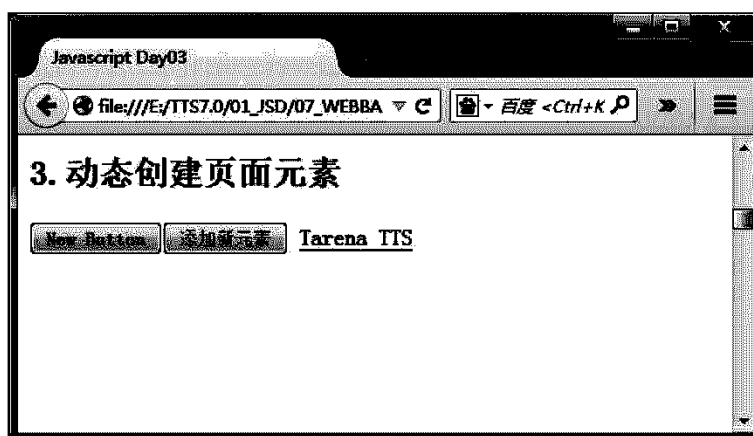


图 - 8

其中，单击图 - 8 上的新按钮“New Button”，会在界面上弹出提示框“Hello”；单击图 - 8 上的链接文本“Tarena TTS”，会在新窗口中打开 TTS 的页面。效果如图 - 9 所示：



图 - 9

• 方案

使用 `document.createElement` 方法创建新的元素节点，并设置节点的属性，然后使用 `appendChild` 或者 `insertBefore` 方法将新节点加入到节点树。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加按钮

在上一个案例的 `html` 页面上继续添加此案例的内容。

为便于定位新节点的位置，首先在 `<form>` 中添加一个 `<div>` 元素，然后在 `<div>` 元素中添加按钮，并为按钮定义单击事件。

因为新节点需要加入到 `<div>` 中，因此，为 `<div>` 元素定义 `id` 属性（用于找到该元素）；为了找到按钮“添加新元素”，从而将新节点加入到此按钮之前，也需要为按钮定义 `id` 属性（用于找到该按钮）。

修改后的 `html` 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day03</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo3.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>3. 动态创建页面元素</h2>
      <div id="div1">
        <input type="button" id="btn1" value="添加新元素"
          onclick="addNewNode();" />
      </div>

    </form>
  </body>
</html>
```

步骤二：在 `js` 文件中定义方法

打开文件 `js_demo3.js`，在其中添加代码，定义名为 `addNewNode` 的方法，并为方法添加代码，先找到 `<div>` 元素。`js` 文件中添加的代码如下所示：

```
//动态创建页面元素
function addNewNode() {
  //得到 div
```

```
var div = document.getElementById("div1");  
}
```

步骤三：创建超级链接节点并加入到按钮之后

在 addNewNode 方法中，创建 <a> 元素节点，并设置节点的信息，然后追加到 <div> 元素中，作为该元素的子节点。代码如下所示：

```
//动态创建页面元素  
function addNewNode() {  
    //得到 div  
    var div = document.getElementById("div1");  
  
    //在按钮后添加一个超级链接  
    var linkNode = document.createElement("a");  
    linkNode.href = "http://tts7.tarena.com.cn";  
    linkNode.target = "_blank";  
    linkNode.innerHTML = "Tarena TTS";  
    div.appendChild(linkNode);  
  
}
```

步骤四：创建新的按钮并添加到原有按钮之前

在 addNewNode 方法中，创建 <input> 元素节点，并设置节点的信息，然后加入到 <div> 元素中，作为该元素的子节点，并放置在原有按钮之前。代码如下所示：

```
//动态创建页面元素  
function addNewNode() {  
    //得到 div  
    var div = document.getElementById("div1");  
  
    //在按钮后添加一个超级链接  
    var linkNode = document.createElement("a");  
    linkNode.href = "http://tts7.tarena.com.cn";  
    linkNode.target = "blank";  
    linkNode.innerHTML = "Tarena TTS";  
    div.appendChild(linkNode);  
  
    //在按钮前再添加一个按钮，单击新按钮，弹出 hello  
    var newBtn = document.createElement("input");  
    newBtn.type = "button";  
    newBtn.value = "New Button";  
    newBtn.onclick = function () {  
        alert("Hello");  
    };  
    div.insertBefore(newBtn, document.getElementById("btn1"));  
  
}
```

• 完整代码

js_demo3.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day03</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo3.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>3. 动态创建页面元素</h2>
      <div id="div1">
        <input type="button" id="btn1" value=" 添 加 新 元 素 "
onclick="addNewNode();" />
      </div>
    </form>
  </body>
</html>
```

js_demo3.js 文件中的代码如下所示：

//其他案例的代码部分，略

//动态创建页面元素

```
function addNewNode() {
  //得到 div
  var div = document.getElementById("div1");

  //在按钮后添加一个超级链接
  var linkNode = document.createElement("a");
  linkNode.href = "http://tts7.tarena.com.cn";
  linkNode.target = " blank";
  linkNode.innerHTML = "Tarena TTS";
  div.appendChild(linkNode);

  //在按钮前再添加一个按钮，单击新按钮，弹出 hello
  var newBtn = document.createElement("input");
  newBtn.type = "button";
  newBtn.value = "New Button";
  newBtn.onclick = function () {
    alert("Hello");
  };
  div.insertBefore(newBtn, document.getElementById("btn1"));
}
```

5. 联动菜单

• 问题

实现页面的联动菜单。页面上有下拉框用于选择课程方向，如“JSD”、“UID”或者“PHP”。页面效果如图 - 10 所示：

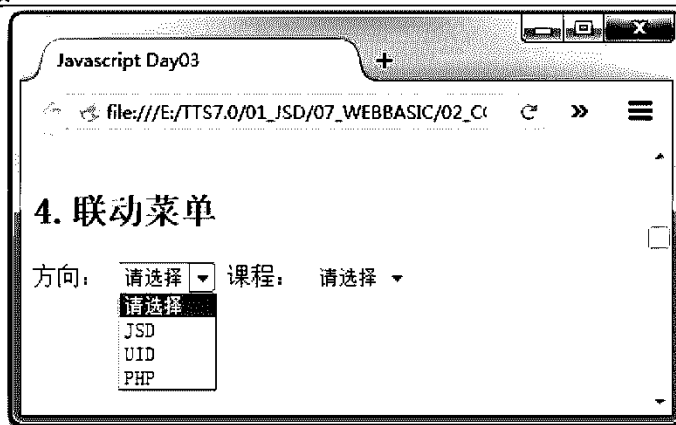


图 - 10

如果方向下拉框选择的是“请选择”，则随后的课程下拉框中也只有“请选择”选项；如果方向下拉框中选择了某课程方向（如 JSD），随后的课程下拉框则显示该课程方向下的课程名称以供用户选择。页面效果如图 - 11 所示：

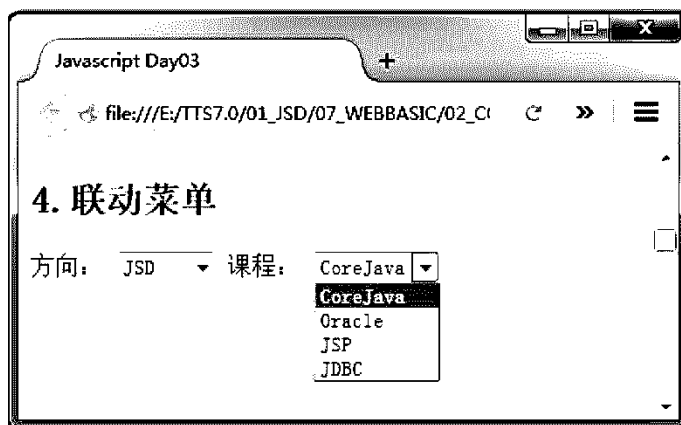


图 - 11

注：“JSD”课程方向下有课程“CoreJava”、“Oracle”、“JSP”和“JDBC”可选；“UID”课程方向下有课程“PS”和“CSS”可选；“PHP”课程方向下有课程“PHP”、“MySQL”和“Jquery”可选。

• 方案

此案例中，首先需要为方向下拉框定义 `onchange` 事件，并定义方法处理该下拉框的选择改变事件。在方法中，需要判断所选择的方向，然后为课程下拉框动态创建选项。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加下拉框

在上一个案例的 `html` 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加文本和下拉框，然后为第一个下拉框定义 onchange 事件；为了动态的操作下拉框，分别为两个下拉框定义 id 属性。

修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day03</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo3.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>4.联动菜单</h2>
      方向：
      <select id="selClass" onchange="classChanged();">
        <option>请选择</option>
        <option>JSD</option>
        <option>UID</option>
        <option>PHP</option>
      </select>
      课程：
      <select id="selProject">
        <option>请选择</option>
      </select>

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义数组存储数据

打开文件 js_demo3.js，在其中添加代码，定义一个二维数组存储课程信息。js 文件中添加的代码如下所示：

```
//二维数组：存储课程信息
var projectArray = new Array();
projectArray[0] = ["请选择"];
projectArray[1] = ["CoreJava", "Oracle", "JSP", "JDBC"];
projectArray[2] = ["PS", "CSS"];
projectArray[3] = ["PHP", "MySQL", "Jquery"];
```

步骤三：在 js 文件中定义方法

继续为文件 js_demo3.js 添加代码，定义名为 classChanged 的方法，js 文件中添加的代码如下所示：

```
//根据所选择的方向写入课程
function classChanged() {
}
```

步骤四：判断所选择的方向

在 classChanged 方法中，首先需要得到第一个下拉框（方向下拉框）中所选择的选项索引，从而得到该方向对应的课程数据。

代码如下所示：

```
//根据所选择的方向写入课程
function classChanged() {

    //得到方向的选择索引以及相应的数组
    var i = document.getElementById("selClass").selectedIndex;
    var data = projectArray[i];

}
```

步骤五：删除第二个下拉框中原有的课程数据

先删除第二个下拉框（课程下拉框）中原有的选项。代码如下所示：

```
//根据所选择的方向写入课程
function classChanged() {
    //得到方向的选择索引以及相应的数组
    var i = document.getElementById("selClass").selectedIndex;
    var data = projectArray[i];

    //得到课程选择框对象
    var selObj = document.getElementById("selProject");
    //删除原有选项
    while (selObj.childNodes.length > 0) {
        selObj.removeChild(selObj.lastChild);
    }

}
```

步骤六：为第二个下拉框创建新的课程数据

为第二个下拉框（课程下拉框）创建新的课程数据。代码如下所示：

```
//根据所选择的方向写入课程
function classChanged() {
    //得到方向的选择索引以及相应的数组
    var i = document.getElementById("selClass").selectedIndex;
    var data = projectArray[i];

    //得到课程选择框对象
    var selObj = document.getElementById("selProject");
    //删除原有选项
    while (selObj.childNodes.length > 0) {
```

```

        selObj.removeChild(selObj.lastChild);
    }

    //循环数组，写入新的 option 对象
    for (var i = 0; i < data.length; i++) {
        var optionObj = document.createElement("option");
        optionObj.value = i;
        optionObj.innerHTML = data[i];
        selObj.appendChild(optionObj);
    }
}

```

• 完整代码

js_demo3.html 文件的代码如下所示：

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Javascript Day03</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <script language="javascript" src="js_demo3.js"
type="text/javascript">
        </script>
        <link href="myStyle.css" type="text/css" rel="stylesheet" />
    </head>
    <body>
        <form>
            <!--其他部分代码，略-->
            <h2>4. 联动菜单</h2>
            方向：
            <select id="selClass" onchange="classChanged();">
                <option>请选择</option>
                <option>JSD</option>
                <option>UID</option>
                <option>PHP</option>
            </select>
            课程：
            <select id="selProject">
                <option>请选择</option>
            </select>
        </form>
    </body>
</html>

```

js_demo3.js 文件中的代码如下所示：

```

//其他案例的代码部分，略

//根据所选择的方向写入课程
function classChanged() {
    //得到方向的选择索引以及相应的数组
    var i = document.getElementById("selClass").selectedIndex;
    var data = projectArray[i];

    //得到课程选择框对象
    var selObj = document.getElementById("selProject");
    //删除原有选项

```

```
while (selObj.childNodes.length > 0) {
    selObj.removeChild(selObj.lastChild);
}

//循环数组，写入新的 option 对象
for (var i = 0; i < data.length; i++) {
    var optionObj = document.createElement("option");
    optionObj.value = i;
    optionObj.innerHTML = data[i];
    selObj.appendChild(optionObj);
}
}
```

6. 联动菜单 (HTML DOM 实现)

- 问题

使用 HTML DOM 的方式实现上一个案例中的联动菜单。

- 方案

此案例的需求和上一个案例完全相同，只是换为 HTML DOM 的方式来实现。

使用 HTML DOM 的方式创建选项 <option> 元素时，代码如下所示：

```
var optionObj = new Option(data[i], i);
```

操作下拉框的选项时，可以使用 Select 对象的 options 属性，该属性表示所有的选项数组。如果需要清空下拉框的所有选项，可以使用如下代码(selObj 表示某个 Select 对象)：

```
selObj.options.length = 0;
```

如果需要设置某个选项，可以使用如下代码：

```
selObj.options[i] = optionObj;
```

其中，selObj 表示某个 Select 对象，optionObj 表示 Option 对象。

- 步骤

实现此案例需要按照如下步骤进行。

步骤一：在 js 文件中定义方法

继续为文件 js_demo3.js 添加代码，定义名为 classChangedByHTMLDOM 的方法，js 文件中添加的代码如下所示：

```
//根据所选择的方向写入课程 (HTML DOM 方式)
function classChangedByHTMLDOM() {
}
```

步骤二：判断所选择的方向

在 classChangedByHTMLDOM 方法中，首先需要得到第一个下拉框（方向下拉框）中所选择的选项索引，从而得到该方向对应的课程数据。

代码如下所示：

```
//根据所选择的方向写入课程（HTML DOM 方式）
function classChangedByHTMLDOM() {

    //得到方向下拉框的选择索引以及相应的数组
    var i = document.getElementById("selClass").selectedIndex;
    var data = projectArray[i];

}
```

步骤三：删除第二个下拉框中原有的课程数据

先删除第二个下拉框（课程下拉框）中原有的选项。代码如下所示：

```
//根据所选择的方向写入课程（HTML DOM 方式）
function classChangedByHTMLDOM() {
    //得到方向下拉框的选择索引以及相应的数组
    var i = document.getElementById("selClass").selectedIndex;
    var data = projectArray[i];

    //得到课程选择框对象
    var selObj = document.getElementById("selProject");
    //删除原有选项
    selObj.options.length = 0;

}
```

步骤四：为第二个下拉框创建新的课程数据

为第二个下拉框（课程下拉框）创建新的课程数据。代码如下所示：

```
function classChangedByHTMLDOM() {
    //得到方向下拉框的选择索引以及相应的数组
    var i = document.getElementById("selClass").selectedIndex;
    var data = projectArray[i];

    //得到课程选择框对象
    var selObj = document.getElementById("selProject");
    //删除原有选项
    selObj.options.length = 0;

    //循环数组，写入新的 option 对象
    for (var i = 0; i < data.length; i++) {
        var optionObj = new Option(data[i],i);
        selObj.options[i] = optionObj;
    }
}
```

}

}

步骤五：修改下拉框的 onchange 事件

修改第一个下拉框（方向下拉框）的 onchange 事件，改为调用上述步骤中所创建的方法 classChangedByHTMLDOM。修改后的 HTML 代码如下所示：

```
<!--其他部分代码，略-->
```

```
<h2>4. 联动菜单</h2>
```

方向：

```
<select id=" selClass " onchange="classChangedByHTMLDOM();">
```

```
    <option>请选择</option>
```

```
    <option>JSD</option>
```

```
    <option>UID</option>
```

```
    <option>PHP</option>
```

```
</select>
```

课程：

```
<select id=" selProject ">
```

```
    <option>请选择</option>
```

```
</select>
```

```
<!--其他部分代码，略-->
```

• 完整代码

js_demo3.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day03</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo3.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>4. 联动菜单</h2>
      方向：
      <select id="selClass" onchange="classChangedByHTMLDOM();">
        <option>请选择</option>
        <option>JSD</option>
        <option>UID</option>
        <option>PHP</option>
      </select>
      课程：
      <select id="selProject">
        <option>请选择</option>
```

```
        </select>
    </form>
</body>
</html>
```

js_demo3.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//根据所选择的方向写入课程（HTML DOM 方式）
function classChangedByHTMLDOM() {
    //得到方向的选择索引以及相应的数组
    var i = document.getElementById("selClass ").selectedIndex;
    var data = projectArray[i];

    //得到课程选择框对象
    var selObj = document.getElementById("selProject ");
    //删除原有选项
    selObj.options.length = 0;

    //循环数组，写入新的 option 对象
    for (var i = 0; i < data.length; i++) {
        var optionObj = new Option(data[i],i);
        selObj.options[i] = optionObj;
    }
}
```


课后作业

1. 广告轮播。

有四张图片，其名称和图片效果如图 - 1 所示：

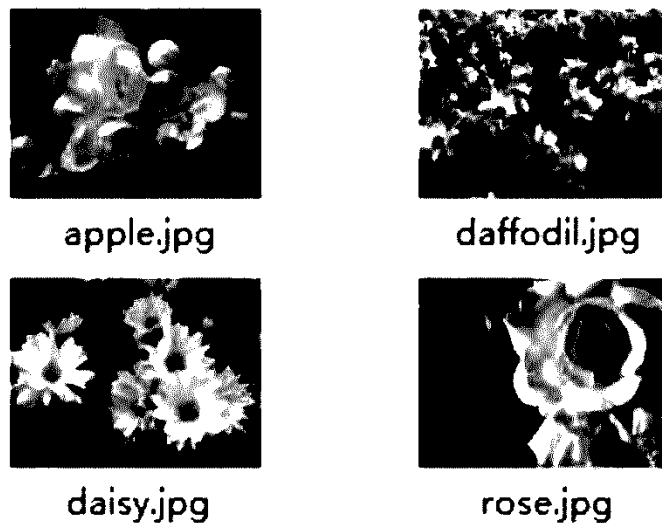


图 - 1

需要使用图 - 1 中的四张图片完成页面上的广告轮播效果，详细要求如下：

1、页面加载后，先在页面上显示第一张图片（rose.jpg），当前图片所对应的数字编号也高亮显示。页面效果如图 - 2 所示：

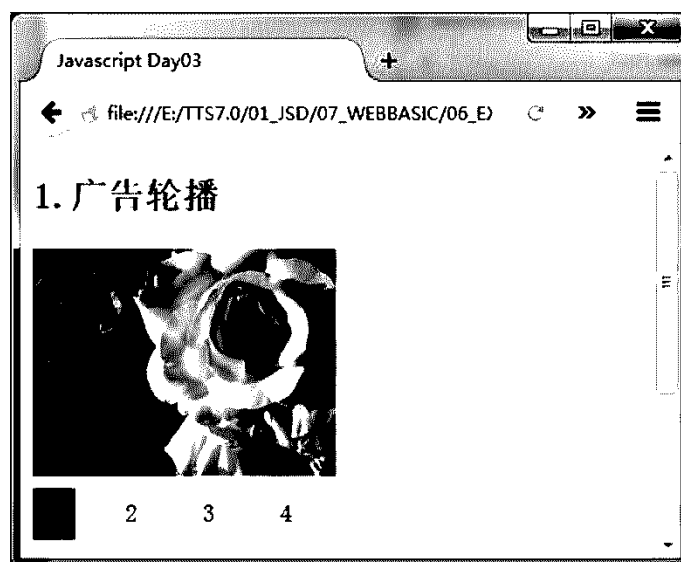


图 - 2

2、每隔 3s，自动更换页面上的图片显示，并修改图片所对应的数字编号的样式，效果如图 - 3 所示：



图 - 3

- 3、图片轮换显示的顺序为："rose.jpg"、"daffodil.jpg"、"apple.jpg"、"daisy.jpg"；
- 4、一轮显示完毕后，重复下一轮显示；
- 5、鼠标移入图片时，停止图片轮换，鼠标移出图片时，继续图片轮换；
- 6、鼠标移入图片下方的某个数字编号，则显示该数字所对应的图片，且停止图片轮换；
- 7、鼠标移出图片下方的某个数字编号，则继续图片轮换。

2. DOM 的节点操作中,appendChild 和 insertBefore 方法的区别是什么。

3. 下拉框版日历。

制作下拉框版的日历，页面加载后的显示效果如图 - 4 所示：

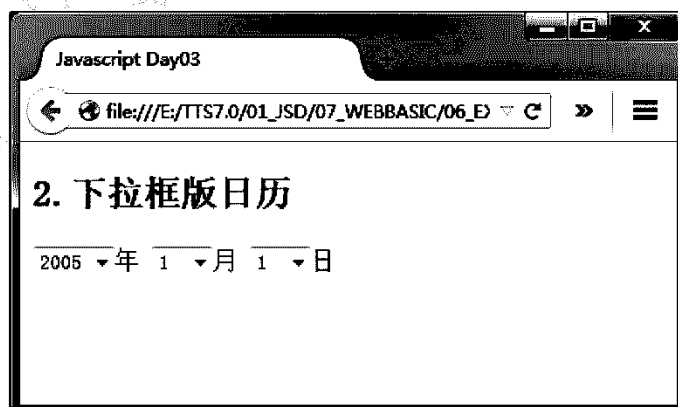


图 - 4

图 - 4 所示的页面中，年份下拉框中的数值，从 2005 年开始显示到当前年份；月份下拉框中的数值显示 1 到 12；日期下拉框中的天数需要根据用户所选择的年份以及月份显示

正确的天数。比如，年份如果选择为 2008，月份如果选择为 2，则最后一个下拉框中的最大天数应该是 29，此时，页面效果如图 - 5 所示：



图 - 5

JavaScript

Unit04

知识体系.....Page 159

HTML DOM 对象	Table 对象	Table 对象
		TableRow 对象
		TableCell 对象
DHTML 其他对象	DHTML 其他对象	DHTML 对象模型回顾
		screen 对象
		history 对象
		location 对象
		navigator 对象
事件	事件概述	事件概述
		事件句柄
	事件处理	事件定义
		事件的处理机制
	event 对象	event 对象
		获取 event 对象
面向对象基础	创建对象	使用 event 对象
		对象概述
		创建通用对象
		创建对象的模板
		JSON

经典案例.....Page 168

数据表格的操作	Table 对象
	TableRow 对象
	TableCell 对象
使用 location 对象	location 对象
遍历 navigator 对象的属性	navigator 对象
事件的冒泡处理机制	事件的处理机制
获取 event 对象的数据	获取 event 对象
简单计算器	使用 event 对象

使用 Object	创建通用对象
自定义对象	创建对象的模板
使用 JSON	JSON

课后作业.....Page 203

1. HTML DOM 对象

1.1. Table 对象

1.1.1. 【Table 对象】Table 对象

Tarena
达内科技

Table 对象

- Table 对象代表一个 HTML 表格
 - <table> 标签表示一个 Table 对象
- 常用属性
 - rows、cells
- 常用方法
 - insertRow(index) : 返回 TableRow 对象
 - deleteRow(index)

+

Tarena
达内科技

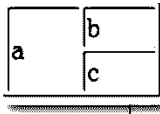
Table 对象 (续1)

js 代码 :

```
var table = document.getElementById("t1");
alert(table.rows.length); //2
alert(table.cells.length); //3
```

HTML 文档 :

```
<table id="t1" border="1" width="100">
  <tr>
    <td rowspan="2">a</td>
    <td>b</td>
  </tr>
  <tr>
    <td>c</td>
  </tr>
</table>
```



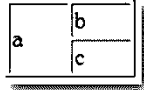
+

1.1.2. 【Table 对象】TableRow 对象

Tarena
达内科技

TableRow 对象

- TableRow 对象代表一个 HTML 表格行
 - <tr> 标签表示一个 TableRow 对象
- 常用属性
 - cells、innerHTML、rowIndex
- 常用方法
 - insertCell(index) : 返回 TableCell 对象
 - deleteCell(index)



```
var table = document.getElementById("t1");
alert(table.rows[0].cells.length); //2
alert(table.rows[1].rowIndex); //1
```

+

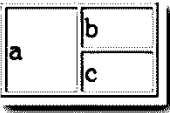
1.1.3. 【Table 对象】TableCell 对象

Tarena
达内科技

TableCell 对象

- TableCell 对象代表一个 HTML 表格单元格
 - <td> 标签表示一个 TableCell 对象
- 常用属性
 - cellIndex、innerHTML、colSpan、rowSpan

```
var table = document.getElementById("t1");
alert(table.rows[0].cells[0].rowSpan); //2
alert(table.rows[1].cells[0].innerHTML); //c
```



++

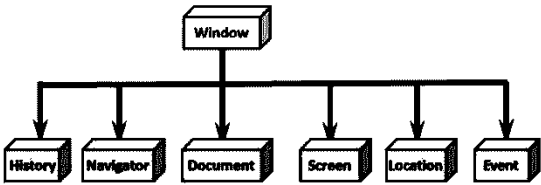
2. DHTML 其他对象

2.1. DHTML 其他对象

2.1.1. 【DHTML 其他对象】DHTML 对象模型回顾

Tarena
达内科技

DHTML 对象模型回顾



++

2.1.2. 【DHTML 其他对象】screen 对象

Tarena
达内科技

screen 对象

- Screen 对象包含有关客户端显示屏幕的信息
- 常用于获取屏幕的分辨率和色彩
- 常用属性
 - width/height
 - availWidth/availHeight

++



2.1.3. 【DHTML 其他对象】history 对象

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>history 对象</h4> <ul style="list-style-type: none"> history 对象包含用户（在浏览器窗口中）访问过的 URL <ul style="list-style-type: none"> length 属性：浏览器历史列表中的 URL 数量 方法 <ul style="list-style-type: none"> back() forward() go(num) <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <pre> alert(history.length); History.forward(); //等同于单击“前进”按钮 history.back(); //等同于单击“后退”按钮 history.go(-2); //等同于单击两次“后退”按钮 </pre> </div> <div style="text-align: right; margin-top: 10px;">  </div>
---	---

2.1.4. 【DHTML 其他对象】location 对象

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>location 对象</h4> <ul style="list-style-type: none"> location 对象包含有关当前 URL 的信息 <ul style="list-style-type: none"> 常用于获取和改变当前浏览的网址 href 属性：当前窗口正在浏览的网页地址 方法 <ul style="list-style-type: none"> replace(url)：转向到url网页地址 reload()：重新载入当前网址，同按下刷新按钮 <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <pre> location.href = "http://tts.tarena.com.cn"; //或者 location.replace("http://tts.tarena.com.cn"); </pre> </div> <div style="text-align: right; margin-top: 10px;">  </div>
---	---

2.1.5. 【DHTML 其他对象】navigator 对象

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>navigator 对象</h4> <ul style="list-style-type: none"> navigator 对象包含有关浏览器的信息 <ul style="list-style-type: none"> 常用于获取客户端浏览器和操作系统信息 <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <pre> var showText = "Navigator对象属性列表：\n"; for (var proproname in navigator) { showText += proproname + "：" + navigator[proproname] + "\n"; } alert(showText); </pre> </div> <div style="border: 1px solid black; padding: 2px; margin-top: 5px; text-align: center;"> 遍历 navigator 对象的所有属性 </div> <div style="text-align: right; margin-top: 10px;">  </div>
---	--

3. 事件

3.1. 事件概述

3.1.1. 【事件概述】事件概述

Tarena
达内科技

事件概述

- 事件：指 DHTML 对象在状态改变、操作鼠标或键盘时触发的动作
 - 鼠标事件
 - 键盘事件
 - 状态改变事件
- event 对象
 - 事件触发后将会产生一个 event 对象

++

3.1.2. 【事件概述】事件句柄

Tarena
达内科技

事件句柄

- 通过一个事件句柄，可以在某个事件发生时对某个元素进行某种操作

鼠标事件

↓

onclick
ondblclick
onmousedown
onmouseup
onmouseover
onmouseout

键盘事件

↓

onkeydown
onkeyup

状态事件

↓

onload
onunload
onchange
onfocus
onblur
onresize
onsubmit

++

3.2. 事件处理

3.2.1. 【事件处理】事件定义

Tarena
达内科技

事件定义

- 在 html 属性定义中直接处理事件

`<input type= "button" value= "按钮" onclick= "method();" />`
- js 代码中动态定义


```
//btnObj 为一个按钮对象
btnObj.onclick = method;

//或
btnObj.onclick = function(){
    alert( "hello" );
};
```

++

Tarena 达内科技

事件定义 (续1)

- 取消事件: onXXX = "return false; "

HTML 文档:

```
<input type= "submit" value= "删除"
onlick="return deleteData();" />
```

js 代码:

```
function deleteData() {
var result = confirm( "确定要删除吗? ");
return result;
}
```

++

3.2.2. 【事件处理】事件的处理机制

Tarena 达内科技

事件的处理机制

- 事件的冒泡处理机制

当处于DHTML对象模型底部对象事件发生时依次激活上面对象定义的同类事件处理

++

Tarena 达内科技

事件的处理机制 (续1)

```
<div style="border:1px solid black;height:100px;"
onlick="alert('div');">
<p style="border:1px solid red;height:50px;"
onlick="alert('p');">
ptext
<input type="button" value="button1"
onlick="alert('button');" />
</p>
</div>
```

++

Tarena 达内科技

事件的处理机制（续2）

- 可以取消事件的冒泡
 - `event.cancelBubble = true;`

```
<div style="border:1px solid black;height:100px;"
    onclick="alert('div');">
  <p style="border:1px solid red;height:50px;"
    onclick="alert('p');">
    ptext
    <input type="button" value="button1"
      onclick="event.cancelBubble=true; alert('button');" />
  </p>
</div>
```

需要使用 event 对象

3.3. event 对象

3.3.1. 【event 对象】event 对象

Tarena 达内科技

event 对象

- 任何事件触发后将会产生一个 event 对象
- event 对象记录事件发生时的鼠标位置、键盘按键状态和触发对象等信息
 - 获得 event 对象
 - 使用 event 对象获得相关信息，如单击位置、触发对象等
- 常用属性：clientX/clientY/cancelBubble 等

3.3.2. 【event 对象】获取 event 对象

Tarena 达内科技

获取 event 对象

- 需要考虑浏览器兼容性
- IE 浏览器
 - js 或者 html 代码中直接使用 event 关键字

```
<p onclick= "alert(event.clientX);" >p text</p>
<div onclick= "func();" >div text</div>
```

```
//IE 浏览器
function func() {
  alert(event.clientX + ":" + event.clientY);
}
```

直接使用 event 关键字

164



获取 event 对象 (续1)

- Firefox 浏览器
 - html 代码中直接使用 event 关键字

```
<p onclick= "alert(event.clientX);" > p text</p>
```


- js 代码中直接使用 event 关键字

```
<div onclick= "func();" >div text</div>
```

```
//firefox 浏览器
function func() {
    alert(event.clientX + ":" + event.clientY);
}
```

× ReferenceError: event is not defined

js 代码中，不能直接使用 event 关键字



获取 event 对象 (续2)

- 在 HTML 代码中，在事件句柄定义时，使用 event 关键字将事件对象作为参数传入方法

```
<div onclick= "func(event);" >div text</div>
```

```
//firefox 浏览器
function func(e) {
    alert(e.clientX + ":" + e.clientY);
}
```

可以解决浏览器兼容问题

3.3.3. 【event 对象】使用 event 对象



使用 event 对象

- 对于 event 对象，经常需要获得事件源
 - 事件源，即触发事件的元素（事件的目标节点）
- IE 浏览器：event.srcElement
- Firefox 浏览器：event.target

```
<div onclick= "func(event);" >div text</div>
```

```
//IE 浏览器
function func(e) {
    alert(e.srcElement.nodeName);    //DIV
}
```

```
//firefox 浏览器
function func(e) {
    alert(e.target.nodeName);        //DIV
}
```

如何写出各浏览器兼容的代码？

Technology
Tarena
达内科技

使用 event 对象 (续1)

HTML 文档 :

js 代码 :

```
<div onclick= "func(event);" >div text</div>
```

```
//考虑浏览器兼容问题
function func(e) {
    var obj = e.srcElement || e.target;
    alert(obj.nodeName);    //DIV
}
```

+

4. 面向对象基础

4.1. 对象概述

4.1.1. 【对象概述】对象概述

Technology
Tarena
达内科技

对象概述

- 对象是一种特殊的数据类型，由属性和方法封装而成
 - 属性指与对象有关的值：对象名.属性名
 - 方法指对象可以执行的行为或可以完成的功能：对象名.方法名()
- 定义对象
 - 创建对象的实例
 - 创建对象的模板
 - 使用 JSON

+

4.2. 创建对象

4.2.1. 【创建对象】创建通用对象

Technology
Tarena
达内科技

创建通用对象

- 使用 Object 对象创建通用的对象

```
function TestObject(){
    var personObj = new Object();

    //添加属性
    personObj.name = "John";
    personObj.age = 50;

    //添加方法
    personObj.say =
        new Function( "alert( 'hello!' );" );

    //测试
    personObj.say();        //hello!
    alert(personObj.age);    //50
}
```

+

4.2.2. 【创建对象】创建对象的模板

Tarena
达内科技

创建对象的模板

- 定义构造函数，以创建自定义的对象
 - 语法：function ObjName(参数1,参数2,...){}

```
function Person (n,a){
    //定义 name 和 age 属性
    this.name = n;
    this.age = a;

    //定义方法 showName
    this.showName = function(){
        alert( "My Name is " + this.name);
    };
    //定义方法 introduceSelf
    this.introduceSelf = introFunc();
}
function introFunc(){
    alert(this.name + "：" + this.age);
}
```

Tarena
达内科技

创建对象的模板（续1）

```
//测试对象
function test(){
    var o1 = new Person( "mary" ,18);
    alert(o1.age);
    o1.showName();
    o1.introduceSelf();

    var o2 = new Person( "john" ,20);
    alert(o2.age);
    o2.showName();
    o2.introduceSelf();
}
```

4.2.3. 【创建对象】JSON

Tarena
达内科技

JSON

- JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式
 - 使用名 / 值对的方式定义
 - 名称需要使用 "" 引起来
 - 多对定义之间使用 , 隔开
- 名称可以是属性
 - 字符串类型的属性值，需要使用 "" 引起来

```
var obj = {
    "name" : "jerry",
    "age" : 29
};
//测试对象
alert(obj.name);
```

经典案例

1. 数据表格的操作

- 问题

界面上有数据表格，用于显示产品的名称和价格信息，并可以添加新产品，或者删除某条产品数据。页面效果如图 - 1 所示：

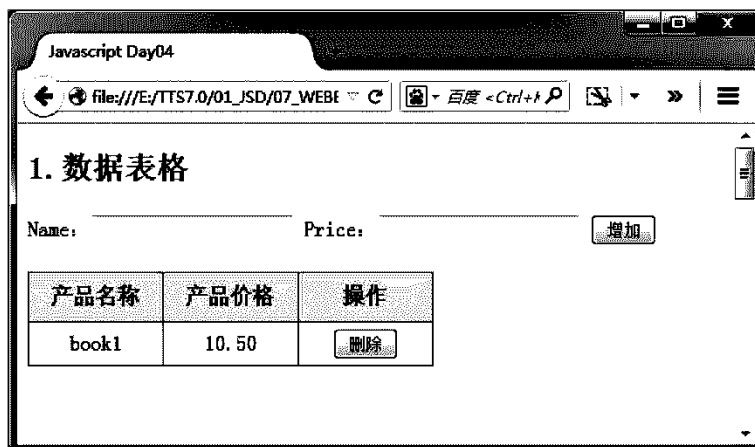


图 - 1

图 - 1 所示的界面上显示出已有的产品数据，如果在界面的文本框中录入新的名称和价格，并单击“增加”按钮，则将录入的数据加入表格，作为新行加入表格最后一行。页面效果如图 - 2 所示：

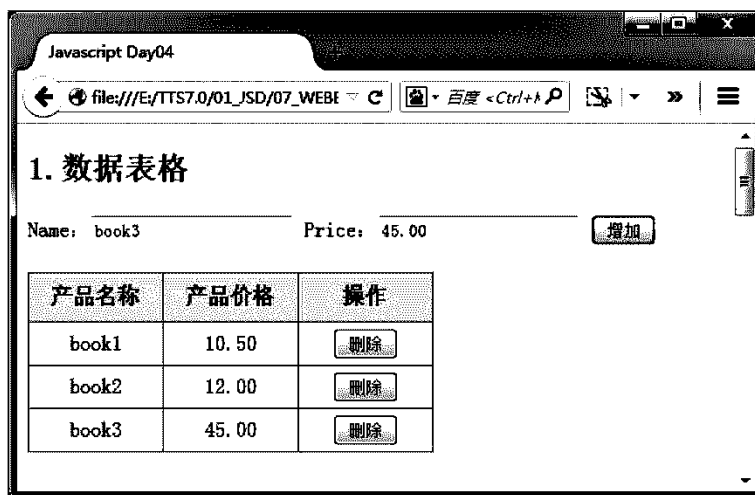


图 - 2

单击表格中某行里的“删除”按钮，首先提示是否删除，页面效果如图 - 3 所示：

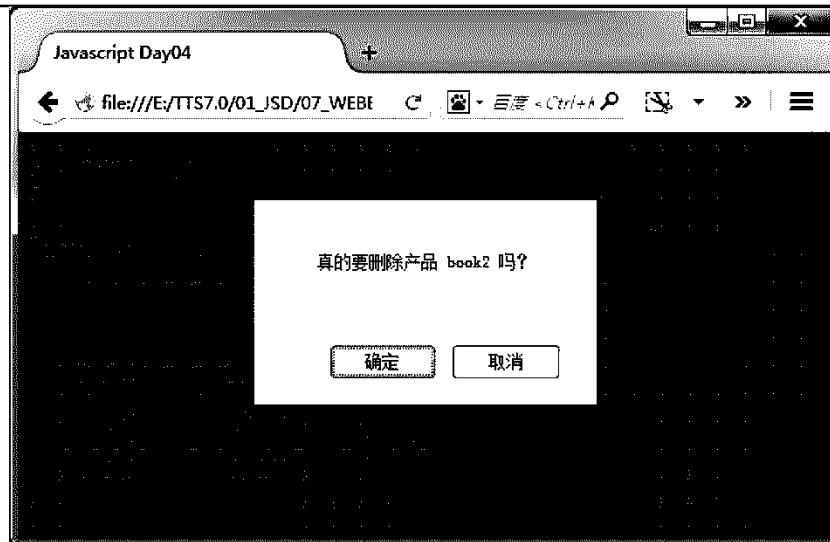


图 - 3

如果用户选择“取消”，则放弃当前的删除操作；如果用户选择“确定”，则删除该行数据，界面效果如图 - 4 所示：



图 - 4

• 方案

分析此案例可以看出，在“增加”按钮的单击事件中，需要通过 js 代码动态的为表格创建新行，并设置行中的内容。

其中，第三列中的按钮需要设置 onclick 事件，并使用 this 关键字将当前按钮节点对象传入，便于通过按钮节点找到当前的行节点，从而获取行中的信息，以及删除当前行。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建页面

创建文件 js_demo4.html , 并在 html 页面上添加代码以创建一个标准结构的 HTML 文档, 并在文档中创建表单。html 代码如下所示:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  </head>
  <body>
    <form>
    </form>
  </body>
</html>
```

步骤二: 创建 js 文件和 css 文件

此案例中需要用到 CSS 样式和 JavaScript 代码, 因此, 首先创建文件 js_demo4.js , 用于书写 JavaScript 代码; 然后创建文件 myStyle.css , 用于书写 CSS 样式代码。并在上一步所创建的 HTML 文档中引入脚本文件和样式文件。修改后的 html 代码如下所示:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />

    <script language="javascript" src="js_demo4.js" type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />

  </head>
  <body>
    <form>
    </form>
  </body>
</html>
```

步骤三: 为页面添加内容

在 <form> 元素中添加文本框、表格以及“增加”按钮。因为需要在 js 代码中获取文本框中的录入, 并操作表格, 因此需要为文本框和表格分别定义 id 属性。修改后的 html 代码如下所示:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo4.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
```

```
<body>
<form>

    <h2>1.数据表格</h2>
    Name : <input type="text" id="txtName" />
    Price : <input type="text" id="txtPrice" />
    <input type="button" value="增加" />
    <br /><br />
    <table id="table1">
        <tr class="header">
            <td>产品名称</td>
            <td>产品价格</td>
            <td>操作</td>
        </tr>
        <tr>
            <td>book1</td>
            <td>10.50</td>
            <td>
                <input type="button" value="删除" />
            </td>
        </tr>
    </table>

</form>
</body>
</html>
```

步骤四：为表格定义样式

在文件 myStyle.css 中添加代码，为表格定义样式：表格边框、单元格的高度以及表头行的背景色以及高度。

css 文件中的代码如下所示：

```
/*表格*/
table
{
    border-collapse:collapse;
}
tr.header td
{
    height:35px;
    font-weight:bold;
    background-color:#ddd;
}
td
{
    border:1px solid gray;
    width:100px;
    height:30px;
    text-align:center;
}
```

步骤五：为按钮定义 onclick 事件

修改 html 代码，首先为“增加”按钮定义 onclick 事件，然后为“删除”按钮也定义 onclick 事件。

在“删除”按钮的 onclick 事件中，需要使用 this 关键字传入当前对象。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo4.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <h2>1.数据表格</h2>
      Name:<input type="text" id="txtName" />
      Price:<input type="text" id="txtPrice" />

      <input type="button" value="增加" onclick="addRow();" />

      <br /><br />
      <table id="table1">
        <tr class="header">
          <td>产品名称</td>
          <td>产品价格</td>
          <td>操作</td>
        </tr>
        <tr>
          <td>book1</td>
          <td>10.50</td>
          <td>

              <input type="button" value="删除" onclick="delFunc(this);" />

          </td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

步骤六：在 js 文件中定义方法 addRow

在文件 js_demo4.js 中创建名为 addRow 的方法，然后为方法添加代码：找到表格对象，并为表格创建新行。js 文件中添加的代码如下所示：

```
//为表格添加行
function addRow() {
  //得到表格对象
  var table = document.getElementById("table1");

  //创建新行
  var row = table.insertRow(table.rows.length);
```

```
}
```

步骤七：添加显示名称和价格的单元格

继续为方法 `addRow` 添加代码：为新创建的行添加第一个单元格（用于显示名称）和第二个单元格（用于显示价格）。js 文件中添加的代码如下所示：

```
//为表格添加行
function addRow() {
    //得到表格对象
    var table = document.getElementById("table1");

    //创建新行
    var row = table.insertRow(table.rows.length);

    //为行创建 name 单元格
    var nameCell = row.insertCell(0);
    nameCell.innerHTML = document.getElementById("txtName").value;

    //为行创建 price 单元格
    var priceCell = row.insertCell(1);
    priceCell.innerHTML = document.getElementById("txtPrice").value;

}
```

步骤八：添加显示“删除”按钮的单元格

继续为方法 `addRow` 添加代码：为新创建的行添加第三个单元格（用于显示“删除”按钮），并为单元格添加按钮，同时设置新按钮的 `onclick` 事件。js 文件中添加的代码如下所示：

```
//为表格添加行
function addRow() {
    //得到表格对象
    var table = document.getElementById("table1");

    //创建新行
    var row = table.insertRow(table.rows.length);

    //为行创建 name 单元格
    var nameCell = row.insertCell(0);
    nameCell.innerHTML = document.getElementById("txtName").value;

    //为行创建 price 单元格
    var priceCell = row.insertCell(1);
    priceCell.innerHTML = document.getElementById("txtPrice").value;

    //为行创建操作按钮的单元格
    var buttonCell = row.insertCell(2);
    var button = document.createElement("input");
    button.type = "button";
```

```
button.value = "删除";
button.onclick = function () { delFunc(this); };
buttonCell.appendChild(button);

}
```

步骤九：在 js 文件中定义方法 delFunc

在文件 js_demo4.js 中创建名为 delFunc 的方法。js 文件中添加的代码如下所示：

```
//删除按钮的单击事件
function delFunc(btnObj) {
}
```

步骤十：找到当前行

继续为方法 delFunc 添加代码：通过传入的参数找到当前行节点对象。js 文件中添加的代码如下所示：

```
//删除按钮的单击事件
function delFunc(btnObj) {

    //找到当前行
    var trObj = btnObj.parentNode.parentNode;

}
```

步骤十一：询问是否删除

继续为方法 delFunc 添加代码：找到当前行中的产品名称，并询问是否删除。如果不删除，则放弃当前操作。js 文件中添加的代码如下所示：

```
//删除按钮的单击事件
function delFunc(btnObj) {
    //找到当前行
    var trObj = btnObj.parentNode.parentNode;

    //获取当前行中的产品名称
    var name = trObj.cells[0].innerHTML;

    //询问确认
    var isDel = confirm("真的要删除产品 " + name + " 吗?");
    if (!isDel)
        return;

}
```

步骤十二：删除行

继续为方法 delFunc 添加代码：找到当前行并删除。js 文件中添加的代码如下所示：

```
//删除按钮的单击事件
function delFunc(btnObj) {
    //找到当前行
    var trObj = btnObj.parentNode.parentNode;

    //获取当前行中的产品名称
    var name = trObj.cells[0].innerHTML;

    //询问确认
    var isDel = confirm("真的要删除产品 " + name + " 吗?");
    if (!isDel)
        return;

    //找到当前行并删除
    trObj.parentNode.removeChild(trObj);
}
```

• 完整代码

js_demo4.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Javascript Day04</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <script language="javascript" src="js_demo4.js"
type="text/javascript">
        </script>
        <link href="myStyle.css" type="text/css" rel="stylesheet" />
    </head>
    <body>
        <form>
            <h2>1. 数据表格</h2>
            Name: <input type="text" id="txtName" />
            Price: <input type="text" id="txtPrice" />
            <input type="button" value="增加" onclick="addRow();" />
            <br /><br />
            <table id="table1">
                <tr class="header">
                    <td>产品名称</td>
                    <td>产品价格</td>
                    <td>操作</td>
                </tr>
                <tr>
                    <td>book1</td>
                    <td>10.50</td>
                    <td>
                        <input type="button" value="删除"
onclick="delFunc(this);" />
                    </td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

myStyle.css 文件中的代码如下所示：

```
/*表格*/
table
{
    border-collapse:collapse;
}
tr.header td
{
    height:35px;
    font-weight:bold;
    background-color:#ddd;
}
td
{
    border:1px solid gray;
    width:100px;
    height:30px;
    text-align:center;
}
```

js_demo4.js 文件的代码如下所示：

```
//为表格添加行
function addRow() {
    //得到表格对象
    var table = document.getElementById("table1");

    //创建新行
    var row = table.insertRow(table.rows.length);

    //为行创建 name 单元格
    var nameCell = row.insertCell(0);
    nameCell.innerHTML = document.getElementById("txtName").value;

    //为行创建 price 单元格
    var priceCell = row.insertCell(1);
    priceCell.innerHTML = document.getElementById("txtPrice").value;

    //为行创建操作按钮的单元格
    var buttonCell = row.insertCell(2);
    var button = document.createElement("input");
    button.type = "button";
    button.value = "删除";
    button.onclick = function () { delFunc(this); };
    buttonCell.appendChild(button);
}

//删除按钮的单击事件
function delFunc(btnObj) {
    //找到当前行
    var trObj = btnObj.parentNode.parentNode;

    //获取当前行中的产品名称
    var name = trObj.cells[0].innerHTML;

    //询问确认
    var isDel = confirm("真的要删除产品 " + name + " 吗? ");
    if (!isDel)
        return;

    //找到当前行并删除
```

```
trobj.parentNode.removeChild(trobj);
```

2. 使用 location 对象

• 问题

为测试 location 对象的用法，创建页面如图 - 5 所示：



图 - 5

单击图 - 5 中的第一个按钮，则使用 location 对象的 href 属性实现页面的跳转，跳转到 TTS7 的登录页面；单击图 - 5 中的第二个按钮，使用 location 对象的 replace 方法实现页面的跳转，也跳转到 TTS7 的登录页面。比较两种方式的区别。

• 方案

使用 location.href 来实现页面跳转时，浏览器将保留跳转前的 URL 地址，从而可以通过浏览器上的“前进”或者“后退”来访问之前的 URL，即保留历史访问记录。

而使用 location.replace() 方式来实现页面跳转时，该方法通过指定 URL 替换当前页面，因此，浏览器中不保存跳转前的地址，不能再返回到上一个页面，即不保留历史访问记录。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加按钮

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加按钮，并分别为其定义单击事件。两个按钮的单击事件调用相同的方法，通过传入不同的参数区分不同的操作。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
```



```

    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo4.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
</head>
<body>
    <form>
        <!--其他部分代码，略-->

        <h2>2.使用 location 对象</h2>
        <input type="button" value="使用 location 对象的 href 属性"
onclick="testLocation(1);" />
        <input type="button" value="使用 location 对象的 replace 方法"
onclick="testLocation(2);" />

    </form>
</body>
</html>

```

步骤二：在 js 文件中定义方法 testLocation

打开文件 js_demo4.js，在其中添加代码，定义名为 testLocation 的方法，并为其定义参数以接收传入的操作类型数值。js 文件中添加的代码如下所示：

```

//使用 location 对象
function testLocation(index) {
}

```

步骤三：实现页面跳转

为 testLocation 方法添加代码。

如果传入的参数为数值 1，则使用 location.href 的方式实现页面跳转；如果参数为数值 2，则使用 location.replace() 的方式实现页面跳转。

代码如下所示：

```

//使用 location 对象
function testLocation(index) {

    var url = "http://tts7.tarena.com.cn";
    switch (index) {
        case 1:
            location.href = url;
            break;
        case 2:
            location.replace(url);
            break;
    }
}

```

- **完整代码**

js_demo4.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo4.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>2.使用 location 对象</h2>
      <input type="button" value="使用 location 对象的 href 属性 "
onclick="testLocation(1);" />
      <input type="button" value="使用 location 对象的 replace 方法 "
onclick="testLocation(2);" />
    </form>
  </body>
</html>
```

js_demo4.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//使用 location 对象
function testLocation(index) {
  var url = "http://tts7.tarena.com.cn";
  switch (index) {
    case 1:
      location.href = url;
      break;
    case 2:
      location.replace(url);
      break;
  }
}
```

3. 遍历 navigator 对象的属性

- **问题**

为了遍历 navigator 对象的属性，创建页面如图 - 6 所示：



图 - 6

单击图 - 6 中的按钮，则在多行文本框中显示 navigator 对象的所有属性以及属性的值，每个属性显示为一行。页面效果如图 - 7 所示：

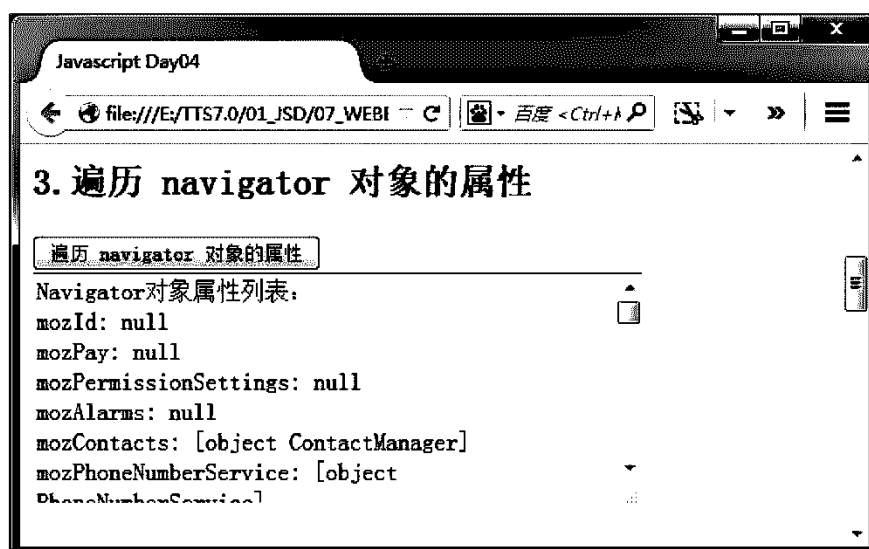


图 - 7

• 方案

在 JavaScript 中，访问对象的属性有如下两种方式：

```
var url = location.href;  
//或者  
var url = location["href"];
```

另外，可以使用 for/in 语句循环遍历对象的属性，比如，查看如下代码：

```
for (* in person)  
{
```

```
//x 表示 person 中的属性的名称
}
```

因此，结合上述两种办法，就可以实现遍历对象的属性。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加按钮

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加按钮，并为其定义单击事件，然后为页面添加多行文本框，用于显示结果。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo4.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>3.遍历 navigator 对象的属性</h2>
      <input type="button" value=" 遍 历   n a v i g a t o r   对 象 的 属 性 "
onclick="testNavigator();" />
      <br />
      <textarea id="txtInfo" style="width:400px;height:150px;"></textarea>

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义方法 testNavigator

打开文件 js_demo4.js，在其中添加代码，定义名为 testNavigator 的方法。js 文件中添加的代码如下所示：

```
//遍历 navigator 对象的属性
function testNavigator() {
}
```

步骤三：实现属性的遍历

为 testNavigator 方法添加代码，使用 for/in 语句遍历 navigator 对象的所有属性，从而逐一得到属性的名称；再使用 对象[属性名称] 的方式得到对象的属性的值，并

拼接字符串，显示在多行文本框中。

代码如下所示：

```
//遍历 navigator 对象的属性
function testNavigator() {

    var showText = "Navigator 对象属性列表：\n";
    for (var propname in navigator) {
        showText += propname + ": " +
            navigator[propname] + "\n";
    }
    document.getElementById("txtInfo").innerHTML = showText;

}
```

• 完整代码

js_demo4.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Javascript Day04</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <script language="javascript" src="js_demo4.js"
type="text/javascript">
        </script>
        <link href="myStyle.css" type="text/css" rel="stylesheet" />
    </head>
    <body>
        <form>
            <!--其他部分代码，略-->

            <h2>3.遍历 navigator 对象的属性</h2>
            <input type="button" value=" 遍 历   n a v i g a t o r   对 象 的 属 性 "
onclick="testNavigator();" />
            <br />
            <textarea id="txtInfo" style="width:400px;height:150px;">
            </textarea>
        </form>
    </body>
</html>
```

js_demo4.js 文件中的代码如下所示：

```
//遍历 navigator 对象的属性
function testNavigator() {
    var showText = "Navigator 对象属性列表：\n";
    for (var propname in navigator) {
        showText += propname + ": " +
            navigator[propname] + "\n";
    }
    document.getElementById("txtInfo").innerHTML = showText;
}
```

4. 事件的冒泡处理机制

• 问题

为测试事件的冒泡处理机制，有页面效果如图 - 8 所示：



图 - 8

图 - 8 中,最外层的黑色框线表示一个 `<div>` 元素,该 `<div>` 包含一个 `<p>` 元素(中间的绿色框线部分),而 `<p>` 元素中包含文本和一个按钮。分别为 `<div>`、`<p>`、`<input>` 元素定义单击事件，测试事件的冒泡处理机制。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加元素

在上一个案例的 html 页面上继续添加此案例的内容。

为页面添加如图 - 8 所示的页面元素，并为 `<div>` 和 `<p>` 元素定义样式：边框、高度和宽度。修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js demo4.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>4. 事件的冒泡处理机制</h2>
      <div style="width:300px;height:100px;border:1px solid black;">
```

```

        <p style="width:200px;height:50px;border:1px solid green;">
            p text<br />
            <input type="button" value="一个按钮" />
        </p>
    </div>

</form>
</body>
</html>

```

步骤二：为各元素定义 onclick 事件

分别为 <div>、<p> 和 <input> 元素定义 onclick 事件，并直接写入 js 代码，分别弹出不同的提示信息。修改的 HTML 代码如下所示：

```

<!--其他部分代码，略-->
<h2>4. 事件的冒泡处理机制</h2>

    <div onclick="alert('div click');" style="width:300px;height:100px;border:1px
solid black;">
        <p onclick="alert('p click');" style="width:200px;height:50px;border:1px solid
green;">
            p text<br />
            <input type="button" value="一个按钮" onclick="alert('input click');" />
        </p>
    </div>

<!--其他部分代码，略-->

```

步骤三：测试

在浏览器中查看页面。

单击页面上 <div> 元素中不和 <p> 元素重叠的部分，则只触发 <div> 元素的 onclick 事件，弹出“div click”；单击段落中不和按钮重叠的区域，则先触发 <p> 元素的 onclick 事件，弹出“p click”，再触发 <div> 元素的 onclick 事件，弹出“div click”。

如果单击页面上的“一个按钮”，则先触发按钮的单击事件，再触发段落的单击事件，再触发 <div> 元素的单击事件，并逐一弹出提示信息。

由此可见，当处于 DHTML 对象模型底部对象事件发生时，会依次激活上面对象定义的同类事件处理。

• 完整代码

js_demo4.html 文件的代码如下所示：

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Javascript Day04</title>
    </head>

```

```

<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<script language="javascript" src="js/demo4.js"
type="text/javascript">
</script>
<link href="myStyle.css" type="text/css" rel="stylesheet" />
</head>
<body>
<form>
<!--其他部分代码，略-->
<h2>4. 事件的冒泡处理机制</h2>
<div style="width:300px;height:100px;border:1px solid black;"
onclick="alert('div click');">
<p style="width:200px;height:50px;border:1px solid green;"
onclick="alert('p click');">
p text<br />
<input type="button" onclick="alert('input click');"
value="一个按钮" />
</p>
</div>
</form>
</body>
</html>

```

5. 获取 event 对象的数据

• 问题

利用事件的冒泡处理机制，简化上一个案例中的事件定义，详细要求如下：

- 1、只为 <div> 元素定义 onclick 事件；
- 2、单击 <div> 范围内的任何区域，弹出显示对应元素的节点名称，页面效果如图 - 9 所示：

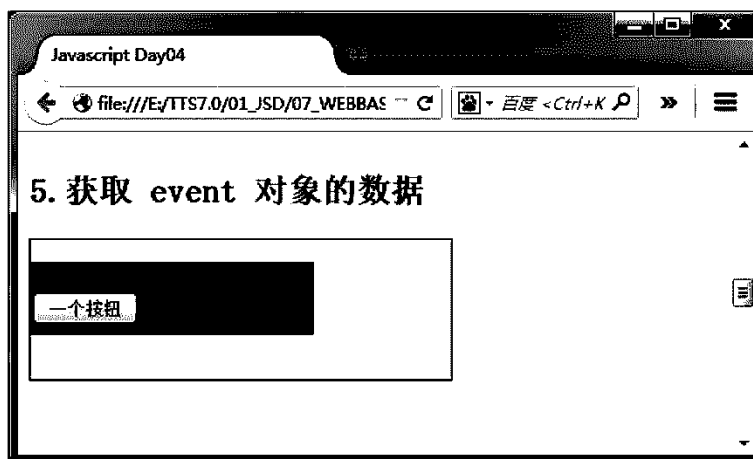


图 - 9

单击图 - 9 中的“一个按钮”位置，弹出信息的效果如图 - 10 所示（弹出“INPUT”）：

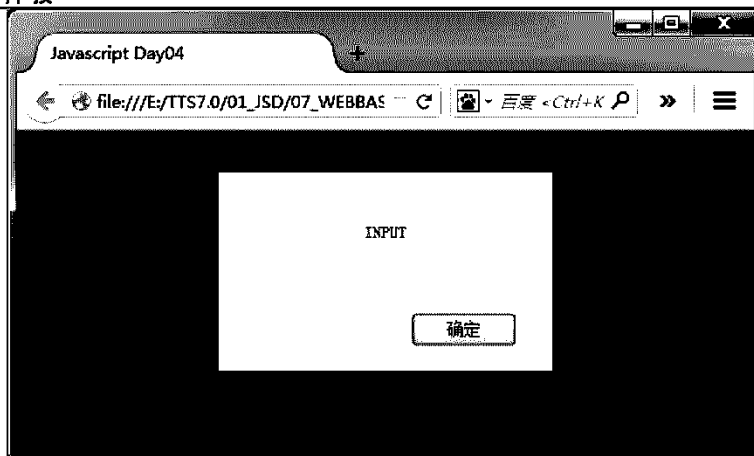


图 - 10

如果单击图 - 9 中的灰色区域 (<p> 元素中不与按钮重叠的区域), 则弹出 "P" ; 如果单击图 - 9 中的白色区域 (<div> 元素中不与 <p> 元素重叠的区域), 则弹出 "DIV"。

• 方案

利用单击事件的冒泡机制, 可以集中处理多个操作。具体方案为:

- 1、把事件定义到一个父级元素上, 避免把事件定义到多个子级元素上;
- 2、在父级元素的事件上, 使用 event 对象获取相关数据。

需要注意的是, 考虑到浏览器的兼容问题, 需要在 HTML 代码中定义事件时, 使用 event 关键字将事件对象传入 js 代码; 在 js 代码中, 使用 event 对象的 srcElement 属性或者 target 属性获取触发事件的源对象, 并使用 nodeName 属性得到其元素名称 (大写方式)。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加元素

在上一个案例的 html 页面上继续添加此案例的内容。

为页面添加如图 - 9 所示的页面元素, 并为 <div> 和 <p> 元素定义样式: 边框、高度和宽度。

注意: 为了在 js 代码中使用 event 对象, 且考虑到浏览器兼容问题, 需要在定义事件时, 使用 event 关键字传入参数。

修改后的 html 代码如下所示:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo4.js">
```

```

type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
</head>
<body>
    <form>
        <!-- 其他部分代码，略-->

        <h2>5. 获取 event 对象的数据</h2>
        <div
            style="width:300px;height:100px;border:1px solid black;"
            onclick="getEventData(event);"
            <p
                style="width:200px;height:50px;border:1px solid
                green;background-color:Gray;"
                p text<br />
                <input type="button" value="一个按钮" />
            </p>
        </div>

    </form>
</body>
</html>

```

步骤二：在 js 文件中定义方法 getEventData

打开文件 js_demo4.js，在其中添加代码，定义名为 getEventData 的方法。js 文件中添加的代码如下所示：

```

//获取 event 对象的数据
function getEventData(eObj) {
}

```

步骤三：获取 event 对象的数据

为方法 getEventData 添加代码，获取触发事件的源对象，并弹出显示对象的节点名称。代码如下所示：

```

//获取 event 对象的数据
function getEventData(eObj) {

    var node = eObj.srcElement || eObj.target;
    alert(node.nodeName);

}

```

• 完整代码

js_demo4.html 文件的代码如下所示：

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Javascript Day04</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />

```

```

<script language="javascript" src="js_demo4.js"
type="text/javascript">
</script>
<link href="myStyle.css" type="text/css" rel="stylesheet" />
</head>
<body>
<form>
<!--其他部分代码，略-->

<h2>5. 获取 event 对象的数据</h2>
<div onclick="getEventData(event);"
style="width:300px;height:100px;border:1px solid black;">
<p style="width:200px;height:50px;border:1px solid
green;background-color:Gray;">
p text<br />
<input type="button" value="一个按钮" />
</p>
</div>
</form>
</body>
</html>

```

js_demo4.js 文件中的代码如下所示：

//其他案例的代码部分，略

//获取 event 对象的数据

```

function getEventData(eObj) {
    var node = eObj.srcElement || eObj.target;
    alert(node.nodeName);
}

```

6. 简单计算器

• 问题

制作简单计算器。页面效果如图 - 11 所示：

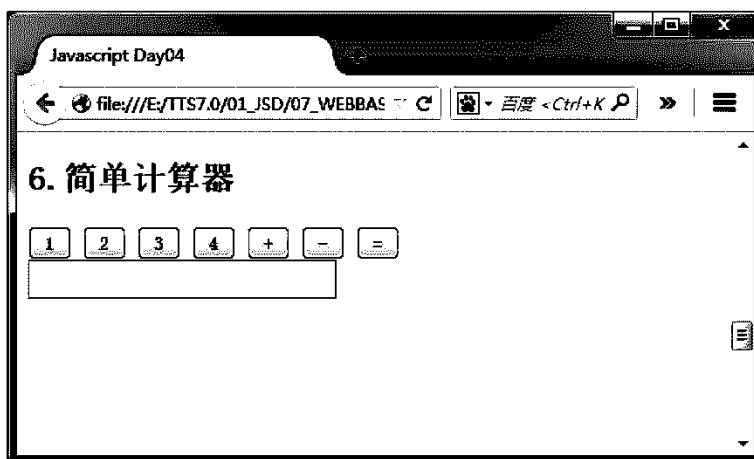


图 - 11

单击图 - 11 所示页面上的按钮（按钮 “=” 除外），比如数字以及 “+”、“-” 按钮，则拼接计算表达式并写入文本框，效果如图 - 12 所示：



图 - 12

单击 “=” 按钮，则计算表达式并将计算的结果写入文本框，页面效果如图 - 13 所示：



图 - 13

关于此案例的其他说明如下：

- 1、实现此案例时，不再为每个按钮定义单击事件，使用某单个事件统一处理；
- 2、需要考虑浏览器兼容性问题；
- 3、此案例实现简单计算器的功能，只考虑用户录入正确的情况，不考虑错误录入的情况。

• 方案

实现此案例时，如果依然使用之前案例中所使用的方式，则需要为每个按钮定义事件，这样，页面代码会过于复杂，不利于维护。

因此，可以定义一个 <div> 元素作为所有按钮以及文本框元素的父元素，并为其定义 onclick 事件，以统一处理所有按钮的点击事件，从而简化页面的代码。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加元素

在上一个案例的 html 页面上继续添加此案例的内容。

首先,在 <form> 标记中添加一个 <div> 元素作为父元素,并在其中添加简单计算器所需要的文本框和按钮。

为了操作文本框中的文本,为其定义 id 属性,并为 <div> 元素定义单击事件。在 <div> 元素的单击事件中,使用 event 关键字将事件对象传入方法。

修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo4.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码,略-->

      <h2>6.简单计算器</h2>
      <div onclick="calculator(event);">
        <input type="button" value="1" />
        <input type="button" value="2" />
        <input type="button" value="3" />
        <input type="button" value="4" />
        <input type="button" value="+" />
        <input type="button" value="-" />
        <input type="button" value="=" /><br />
        <input type="text" id="txtData" style="font-size:15pt;border:1px
solid gray;" />
      </div>

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义方法 calculator

打开文件 js_demo4.js,在其中添加代码,定义名为 calculator 的方法,并为该方法定义参数,以获取传入的事件对象。js 文件中添加的代码如下所示：

```
//简单计算器
function calculator(e) {
}
```

步骤三：判断引发事件的元素

为 calculator 方法添加代码,首先获取触发事件的对象,然后通过节点名称和类型进行判断：只处理按钮的单击事件。代码如下所示：

```
//简单计算器
function calculater(e) {

    //得到激发事件的节点对象
    var node = e.srcElement || e.target;
    //只处理按钮
    if (node.nodeName == "INPUT" && node.type == "button") {
    }

}
```

步骤四：实现计算器功能

继续为 calculater 方法添加代码，判断所单击的按钮：如果单击的是 “=” 按钮，则进行计算并显示结果；否则将按钮上的文本拼接显示到文本框中。

代码如下所示：

```
//简单计算器
function calculater(e) {
    //得到激发事件的节点对象
    var node = e.srcElement || e.target;
    //只处理按钮
    if (node.nodeName == "INPUT" && node.type == "button") {

        if (node.value == "=") {
            var expression = document.getElementById("txtData").value;
            document.getElementById("txtData").value = eval(expression);
        }
        else
            document.getElementById("txtData").value += node.value;
    }
}
```

• 完整代码

js_demo4.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Javascript Day04</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <script language="javascript" src="js_demo4.js"
type="text/javascript">
        </script>
        <link href="myStyle.css" type="text/css" rel="stylesheet" />
    </head>
    <body>
        <form>
            <!--其他部分代码，略-->

            <h2>6. 简单计算器</h2>
            <div onclick="calculater(event);">
                <input type="button" value="1" />
```

```
        <input type="button" value="2" />
        <input type="button" value="3" />
        <input type="button" value="4" />
        <input type="button" value="+" />
        <input type="button" value="-" />
        <input type="button" value="=" /><br />
        <input type="text" id="txtData"
style="font-size:15pt;border:1px solid gray;" />
    </div>
</form>
</body>
</html>
```

js_demo4.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//简单计算器
function calculator(e) {
    //得到激发事件的节点对象
    var node = e.srcElement || e.target;
    //只处理按钮
    if (node.nodeName == "INPUT" && node.type == "button") {
        if (node.value == "=") {
            var expression = document.getElementById("txtData").value;
            document.getElementById("txtData").value = eval(expression);
        }
        else
            document.getElementById("txtData").value += node.value;
    }
}
```

7. 使用 Object

• 问题

使用 Object 定义对象实例，并为其定义属性和方法，详细要求如下：

- 1、name 属性：记载姓名；
- 2、age 属性：记载年龄；
- 3、introduceSelf 方法：调用该方法，将弹出显示信息如 “i am mary,i am 18 years old.”，其中，mary 为 name 属性的值，18 为 age 属性的值。

然后在页面上按钮的单击事件中添加代码以测试所封装的对象。页面效果如图 - 14 所示：



图 - 14

单击图 - 14 上的按钮时，先读取两个属性的数据，并弹出显示，如图 - 15 所示：

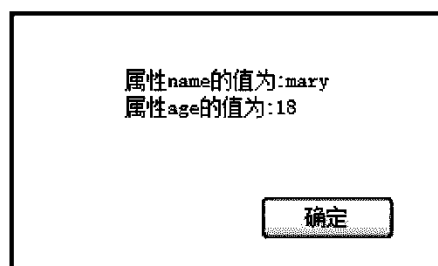


图 - 15

再调用对象的方法，界面效果如图 - 16 所示：

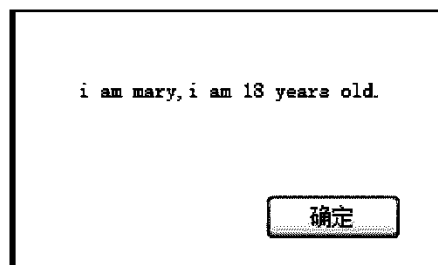


图 - 16

• 方案

此案例中，首先需要使用 Object 定义对象，并为其定义 name 和 age 属性以封装相应数据，代码如下：

```
var person = new Object();
person.name = "mary";
person.age = 18;
```

然后为对象定义名为 introduceSelf 的方法，代码如下：

```
person.introduceSelf = function () {
    var info = "i am " + this.name + ", i am " + this.age + " years old.";
    alert(info);
};
```


对象定义完成后，测试其数据和行为。测试的代码如下：

```
alert("属性 name 的值为:" + person.name + "\n 属性 age 的值为:" + person.age);  
person.introduceSelf();
```

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加元素

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加一个按钮，修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">  
  <head>  
    <title>Javascript Day04</title>  
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />  
    <script language="javascript" src="js_demo4.js"  
type="text/javascript">  
    </script>  
    <link href="myStyle.css" type="text/css" rel="stylesheet" />  
  </head>  
  <body>  
    <form>  
      <!--其他部分代码，略-->  
  
      <h2>7.面向对象：使用 Object</h2>  
      <input type="button" value=" 测试 对 象 （ 使 用 Object ） "  
onclick="testObject();" />  
  
    </form>  
  </body>  
</html>
```

步骤二：在 js 文件中定义方法 testObject

打开文件 js_demo4.js，在其中添加代码，定义名为 testObject 的方法。js 文件中添加的代码如下所示：

```
//使用 Object 定义对象实例  
function testObject() {  
}
```

步骤三：定义对象

为 testObject 方法添加代码。首先使用 Object 定义对象，并为其声明属性和方法。代码如下所示：

```
//使用 Object 定义对象实例  
function testObject() {
```

```
//定义对象封装数据和行为
var person = new Object();
person.name = "mary";
person.age = 18;
person.introduceSelf = function () {
    var info = "i am " + this.name + ",i am " + this.age + " years old.";
    alert(info);
};

}
```

步骤四：测试对象

继续为 testObject 方法添加代码，测试所创建的对象：读取属性并调用方法。代码如下所示：

```
//使用 Object 定义对象实例
function testObject() {
    //定义对象封装数据和行为
    var person = new Object();
    person.name = "mary";
    person.age = 18;
    person.introduceSelf = function () {
        var info = "i am " + this.name + ",i am " + this.age + " years old.";
        alert(info);
    };

    //测试对象的数据和行为
    alert("属性 name 的值为:" + person.name + "\n 属性 age 的值为:" + person.age);
    person.introduceSelf();
}

}
```

• 完整代码

js_demo4.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo4.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>7.面向对象：使用 Object</h2>
      <input type="button" value="测试对象（使用 Object）"
onclick="testObject();" />
    </form>
  </body>
</html>
```

js_demo4.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//使用 Object 定义对象实例
function testObject() {
    //定义对象封装数据和行为
    var person = new Object();
    person.name = "mary";
    person.age = 18;
    person.introduceSelf = function () {
        var info = "i am " + this.name + ",i am " + this.age + " years old.";
        alert(info);
    };

    //测试对象的数据和行为
    alert("属性 name 的值为:" + person.name + "\n 属性 age 的值为:" + person.age);
    person.introduceSelf();
}
```

8. 自定义对象

• 问题

创建对象的模板，并为其定义属性和方法，详细要求如下：

- 1、name 属性：记载姓名；
- 2、age 属性：记载年龄；
- 3、introduceSelf 方法：调用该方法，将弹出显示信息如 “i am jerry,i am 20 years old.”，其中，jerry 为 name 属性的值，20 为 age 属性的值。

然后在页面上按钮的单击事件中添加代码进行测试。页面效果如图 - 17 所示：



图 - 17

单击图 - 17 上的按钮时，先读取两个属性的数据，并弹出显示，如图 - 18 所示：

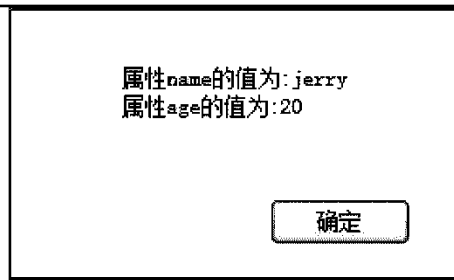


图 - 18

再调用对象的方法，界面效果如图 - 19 所示：

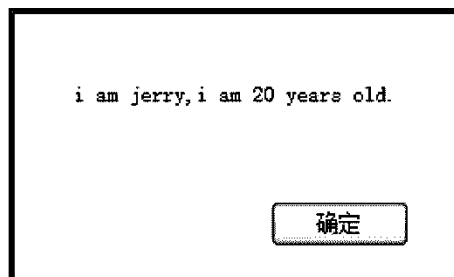


图 - 19

• 方案

此案例中,为了使用自定义对象,在 javascript 代码中,首先需要创建对象的模板。因此,先定义名为 Person 的对象模板,并为其定义属性和方法。代码如下所示：

```
//定义自定义对象
function Person(n, a) {
    this.name = n;
    this.age = a;
    this.introduceSelf = introFunc;
}
```

然后为 Person 定义方法的实现，代码如下：

```
function introFunc() {
    var info = "i am " + this.name + ",i am " + this.age + " years old.";
    alert(info);
}
```

定义完毕后，创建对象，并测试其属性和方法。代码如下：

```
var p1 = new Person("jerry", 20);
alert("属性 name 的值为:" + p1.name + "\n属性 age 的值为:" + p1.age);
p1.introduceSelf();
```

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加元素

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加一个按钮，修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo4.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>8.面向对象：自定义对象</h2>
      <input type="button" value="测试对象(自定义对象)" onclick="testClass();"
/>

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义 Person

打开文件 js_demo4.js，在其中添加代码，定义名为 Person 的对象模板，并为其定义属性和方法，以及方法的实现。js 文件中添加的代码如下所示：

```
//定义自定义对象
function Person(n, a) {
  this.name = n;
  this.age = a;
  this.introduceSelf = introFunc;
}

function introFunc() {
  var info = "i am " + this.name + ",i am " + this.age + " years old.";
  alert(info);
}
```

步骤三：在 js 文件中定义方法 testClass

继续在文件 js_demo4.js 中添加代码，定义名为 testClass 的方法。代码如下所示：

```
//使用自定义对象
function testClass() {
}
```

步骤四：创建对象并测试

继续为 testClass 方法添加代码，创建对象并测试：读取属性并调用方法。代码如下所示：

```
//使用自定义对象
function testClass() {

    var p1 = new Person("jerry", 20);
    alert("属性 name 的值为:" + p1.name + "\n 属性 age 的值为:" + p1.age);
    p1.introduceSelf();

}
```

• 完整代码

js_demo4.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo4.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>8. 面向对象：自定义对象</h2>
      <input type="button" value="测试对象（自定义对象）"
onclick="testClass();" />
    </form>
  </body>
</html>
```

js_demo4.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//使用自定义对象
function testClass() {
    var p1 = new Person("jerry", 20);
    alert("属性 name 的值为:" + p1.name + "\n 属性 age 的值为:" + p1.age);
    p1.introduceSelf();
}
```

9. 使用 JSON

• 问题

使用 JSON 定义对象，并为其定义属性和方法，详细要求如下：

1、name 属性：记载姓名；

2、age 属性：记载年龄。

然后在页面上按钮的单击事件中添加代码进行测试。页面效果如图 - 20 所示：

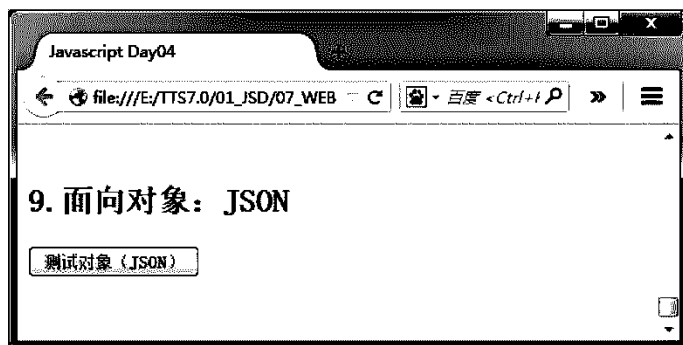


图 - 20

单击图 - 20 上的按钮时，读取两个属性的数据，并弹出显示，如图 - 21 所示：

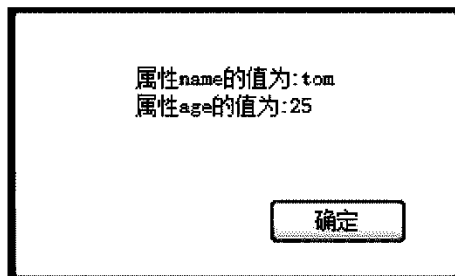


图 - 21

• 方案

此案例中，首先需要使用 JSON 格式创建对象的实例，并封装数据。代码如下：

```
var o = {  
    "name": "tom",  
    "age": 25 };
```

然后测试此对象的属性以及方法，代码如下：

```
alert("属性name的值为:" + o.name + "\n 属性age的值为:" + o.age);
```

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：为页面添加元素

在上一个案例的 html 页面上继续添加此案例的内容。

首先，在 <form> 标记中添加一个按钮，修改后的 html 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo4.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>9.面向对象：JSON</h2>
      <input type="button" value="测试对象 (JSON)" onclick="testJSON();" />

    </form>
  </body>
</html>
```

步骤二：在 js 文件中定义方法 testJSON

继续在文件 js_demo4.js 中添加代码，定义名为 testJSON 的方法。代码如下所示：

```
//使用 JSON
function testJSON() {
}
```

步骤三：创建对象

继续为 testJSON 方法添加代码，创建对象。代码如下所示：

```
//使用 JSON
function testJSON() {

    var o = {
        "name": "tom",
        "age": 25 };

}
```

步骤四：测试对象

继续为 testJSON 方法添加代码测试上个步骤中所创建的对象 读取属性并调用方法。
代码如下所示：

```
//使用 JSON
function testJSON() {
    var o = {
        "name": "tom",
        "age": 25 };
}
```



```
alert("属性 name 的值为:" + o.name + "\n 属性 age 的值为:" + o.age);
```

```
}
```

• 完整代码

js_demo4.html 文件的代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Javascript Day04</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <script language="javascript" src="js_demo4.js"
type="text/javascript">
    </script>
    <link href="myStyle.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <form>
      <!--其他部分代码，略-->

      <h2>9.面向对象：JSON</h2>
      <input type="button" value="测试对象（JSON）" onclick="testJSON();"
/>
    </form>
  </body>
</html>
```

js_demo4.js 文件中的代码如下所示：

```
//其他案例的代码部分，略

//使用 JSON
function testJSON() {
  var o = {
    "name": "tom",
    "age": 25 };

  alert("属性 name 的值为:" + o.name + "\n 属性 age 的值为:" + o.age);
}
```

课后作业

1. 简述 window 对象除 document 以外的一些常用子对象 ,并描述其作用。
2. 如何在 js 代码中 , 获取 event 对象并读取相关信息 ?
3. 简述三种创建对象的方式。
4. 创建页面 , 模拟数据的批量提交。

创建页面 , 效果如图 - 1 所示 :



图 - 1

单击页面上的 “增加产品” 按钮 , 则在表格中增加一行 , 行中的各单元格显示文本框 , 用于录入相应的数据 , 界面效果如图 - 2 所示 :

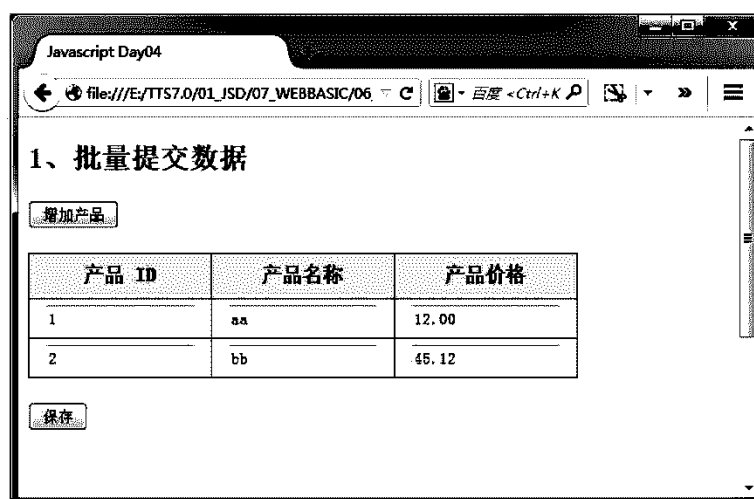


图 - 2

录入完毕后,单击页面上的“保存”按钮,则使用自定义对象(三种方式任选一种)封装每个产品数据,最后形成对象的数组以备提交。其中,每个产品对象需要有 id 属性、name 属性和 price 属性。

另外,此案例中只能模拟提交的效果,因此,在将数据封装完毕并记载成为数组之后,需要循环对象的数组,逐一读取对象中的数据,并弹出显示进行测试。效果如图 - 3 和图 - 4 所示:

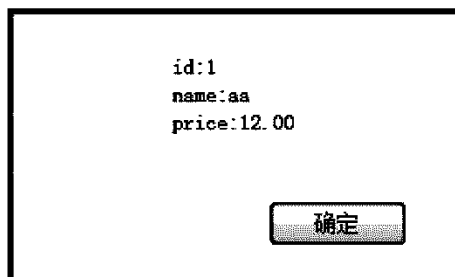


图 - 3

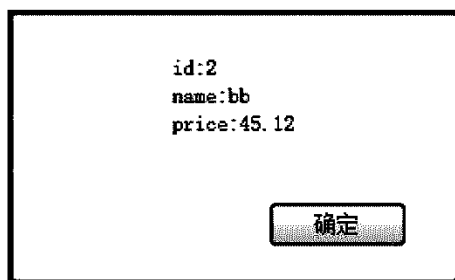


图 - 4

5. 实现选择框功能。

有页面效果如图 - 5 所示:



图 - 5

图 - 5 所示的页面中,显示多个用户供选择,如果用户选择了某收件人(如张三),则

信息会显示在页面下方的框中，如果选择了多个收件人，则下方框中会显示滚动条。页面效果如图 - 6 所示：



图 - 6

如果取消某收件人的选择，则该用户在下方框中消失。

另外，如果单击收件人右上角的“x”图片（如图 - 7 中的圈出部分所示），将取消该用户的选择，且该用户从下方框中消失。



图 - 7