第14章 JavaScript基础(4-6课时)

采用JS编程

这是JS代码


```
30      <script type="text/javascript" src="http://211.147.1.142:9890/js/menu.js"></script>
31      <script type="text/javascript" src="http://211.147.1.142:9890/js/jquery.alerts.js"></script>
32      <script type="text/javascript" src="http://211.147.1.142:9890/js/kxbdSuperMarquee.js"></script>
```

本章学习目标

Web 前端开发工程师应掌握以下内容

- 理解JavaScript程序的概念与作用；
- 掌握JavaScript标识符和变量的概念及使用方法；
- 掌握JavaScript常用运算符和表达式概念；
- 掌握JavaScript中顺序、分支、循环等3种程序控制结构语法；
- 掌握JavaScript函数的定义方法，并学会使用；
- 学会综合运用JavaScript设计具有动态、交互功能的网页。

14.1 JavaScript概述

JavaScript最初由Netscape公司的Brendan Eich（布兰登．艾奇）设计，最初命名为LiveScript，是一种动态、弱类型、基于原型的语言。后来，Netscape与Sun公司进行合作，将LiveScript改名为JavaScript。

JavaScript是一种基于对象和事件驱动并具有相对安全性的客户端脚本语言。被广泛应用于各种客户端Web程序尤其是HTML开发中，能给HTML网页添加动态功能，响应用户各种操作，实现诸如欢迎信息、数字日历、跑马灯，显示浏览器停留时间等特殊效果，提高网页的可观性。

14.1.1 JavaScript简介

JavaScript是一种基于对象和事件驱动、安全性、轻量级、解释型、弱类型的客户端脚本语言。 决定WEB页面的行为，具有客户端数据验证、用户交互等功能。

JavaScript具有如下特点：

1.简单性(小程序、无须编译、解释性、弱数据类型)

2.安全性(Browser无法访问本地硬盘数据/写入到数据库)

3.动态性（JS可以直接对用户提交的信息作出回应）

4.跨平台性（支持JS的Browser）

14.1.2 第一个JavaScript程序

基本语法：

<script type=“text/javascript [src=“外部JS文件”]>js语句块;</script>
<script language=“javascript [src=“外部JS文件”]>js语句块;</script>

```
<!-- edu_14_1_1.html -->
<html>
   <head>
         <title>第一个JavaScript实例</title>
   </head>
   <body>
         <script type="text/javascript">
         document.write("第一个JavaScript实例!");
        </script>
   </body>
</html>
```

14.1.3 JavaScript放置的位置

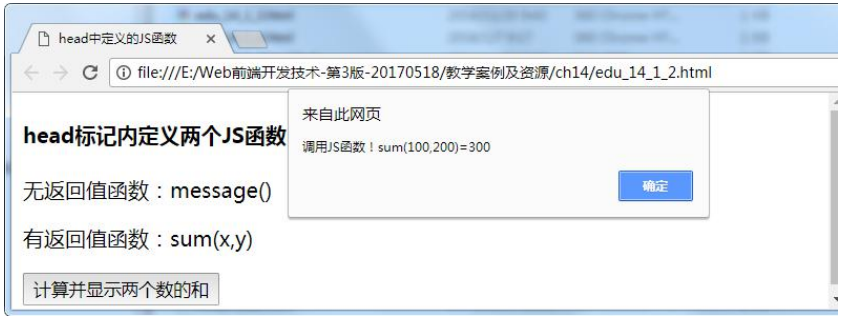
- JavaScript代码放置的位置：（1）头部；（2）主体；（3）单独的js文件；（4）直接在事件处理代码中。
- JavaScript程序本身不能独立存在，它是依附于HTML代码，经浏览器解释执行。
- 可将JavaScript函数写成一个独立的js文件，在HTML文档中引用该js文件，引用时必须使用src属性。JavaScript文件的扩展名为*.js。格式如下：

```
<script type="text/javascript src="外部JS文件"></script>
```

注：此时在<script></script>标记之间的所有JS语句都被忽略，不会执行。

14.1.3 JavaScript放置-头部

```
<!-- edu_14_1_2.html -->
<html>
<head>
<title>调用head中定义的JavaScript函数</title>
<script type="text/javascript">
function message() {
    alert("调用head中定义的JavaScript函数！");
}
</script>
</head>
<body>
    <h4>head标记内定义的JavaScript函数</h4>
    <form><input name="btnCallJS"
type="button" onclick="message();" value="事件
调用自定义函数"></form>
</body>
</html>
```



注：JS脚本插入在头部时，通常需要定义为函数格式，格式：

function 函数名(参数1,参数2,…，参数n){函数体语句;}

14.1.3 JavaScript放置-主体

```
<!-- edu_14_1_2_1.html -->
<html>
  <head>
  <title>主体部分JavaScript</title>
  </head>
<body>
  <script type="text/javascript" >
    alert(“JS放置在主体中，直接运行！”);
  </script>
</body>
</html>
```

注：JS脚本插入在主体时，JavaScript语句能够被立即执行。也可以定义成函数，但必须引用才能执行。

14.1.3 JavaScript放置-外部JS

```
<!-- edu_14_1_3.html -->
<html>
<head>
<title>调用外部js文件的JavaScript函数</title>
<script type="text/javascript"
src="demo.js"></script>
</head>
<body>
<form>
<input name= “btnCallJS”  type= “button”
onclick= “message();" value="调用外部js文件的
JavaScript函数">
</form>
</body>
</html>
```

```
/*-- demo.js */
function message() {
   alert("调用外部js文件中
的函数！");
}
```

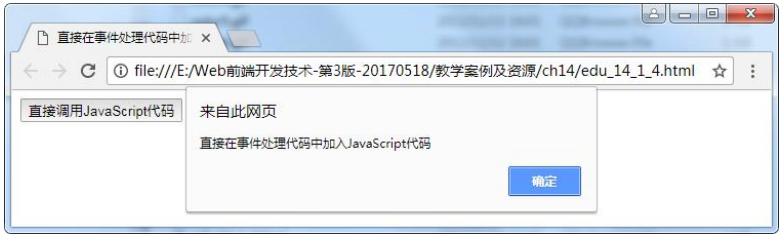
这是引用外部JS

这是执行外部JS

注：外部JS文件需要引用到HTML文件中才能被执行。编写外部JS文件时不需要使用<script></script>标记。

14.1.3 JavaScript放置-事件处理代码

```
<!-- edu_14_1_4.html -->
<html>
  <head>
  <title>直接在事件处理代码中加入
JavaScript代码</title>
   </head>
  <body>
   <form>
    <input type="button" onclick="alert('
直接在事件处理代码中加入JavaScript代码')"
value="直接调用JavaScript代码">
    </form>
   </body>
</html>
```



注：JS代码直接放置在事件处理的代码中，可以直接运行。也可以加上“javascript:alert(‘信息’);”

14.2 JavaScript程序

JavaScript程序由语句、语句块、函数、对象、方法、属性等构成，通过顺序、分支和循环三种基本程序控制结构来进行编程。

14.2.1 语句和语句块

- JavaScript语句是发送给浏览器的命令，这些命令的作用是告诉浏览器要做的事情。

```
alert(“这是告警消息框!”); //弹出告警消息框
```

- JavaScript语句可以分批组合起来形成语句块，语句块以左花括号“{”开始，以右花括号“}”束。

```
{var s=0;document.write(“S的值=” +s);}  //赋值，并输出到页面
```

14.2.2 代码

- JavaScript代码是由若干条语句或语句块构成的执行体。

```
<script type="text/javascript">
   var color="red";
   if(color=="red")
   {
     document.write("颜色是红色!");
     alert("颜色是红色!");
   }
</script>
```

14.2.3 消息对话框

JavaScript中的消息对话框分为告警框、确认框和提示框。

1.警告框

```
alert (message) ;
```

2.确认框

```
var yn=confirm (message);
```

3.提示框

```
var s1=prompt (text, defaultText);
```

14.2.3 消息对话框-告警框

```
<!-- edu_14_2_1.html -->
<html>
<head>
<title>告警消息框使用实例</title>
</head>
<body>
<script type="text/javascript">
alert("这是告警消息框！ ");
</script>
</body>
</html>
```

注：确定按钮必须响应，否则屏蔽一切操作。告警信息为纯文本信息或字符串，不能含有HTML标记。

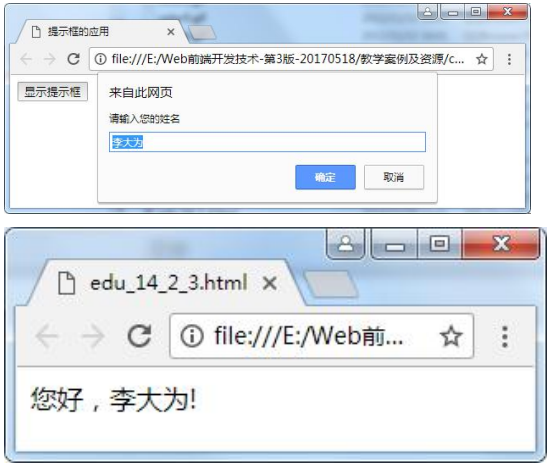
14.2.3 消息对话框-确认框

```
<!-- edu_14_2_2.html -->
<html>
<head>
<title>确认框使用实例</title>
<script type="text/javascript">
function show_confirm() {
var r=confirm("请选择按钮!");
if (r==true)
{alert("您按了确定按钮！ ");}
else{alert("您按了取消按钮！ "); }
}
</script>
</head>
<body>
<input type="button"
  onclick="show_confirm()" value="显示确
  认框" />
</body>
</html>
```

注：确认按钮的返回值，类型为逻辑值，确定true，取消false。

14.2.3 消息对话框-提示框

```
<!-- edu_14_2_3.html -->
<html>
<head>
<title>提示框使用实例</title>
<script type="text/javascript">
function disp_prompt() {
   var name=prompt("请输入您的姓名","李大为
   ");
   if (name!=null && name!="") {
   document.write("您好，" + name + "!"); }
}
</script>
</head>
<body>
<input type="button"
   onclick="disp_prompt()" value="单击显示提
   示框" />
</body>
</html>
```



注：选择“确定”返回输入的值，选择“取消”返回null。

14.2.4 JavaScript注释

JavaScript注释：单行注释和多行注释。

- 单行注释：使用“//”作为注释标记，可以单独一行或跟在代码末尾，放在同一行中，“//”后为注释内容部分。
- 多行注释：以“/*”标记开始，以“*/”标记结束，两个标记之间所有的内容都是注释文本。
- 使用注释防止代码执行--屏蔽某些语句行的执行。

```
<!-- edu_14_2_4.html -->
<script type="text/javascript">
  //这是单行注释
  /*这是多行注释
      可以包含多行内容
   */
   //alert("此语句不执行！");
alert(“此语句执行了！”);//执行时弹出告警消息框
</script>
```

14.3 标识符和变量

在任何一种编程语言中，实际编程时都要使用变量以存储常用的数据。所谓变量，顾名思义，就是在运行期间其值可以通过程序改变的量。

为了便于变量的使用，实际使用时需要给变量加以命名，变量的名字则称为标识符。

14.3.1 命名规范

1.标识符

标识符是计算机语言中用来表示变量名、函数名等的有效字符序列，简单来说，标识符就是一个名字，JavaScript关于标识符的规定如下：

(1)必须使用字母或者下划线和$开始。

(2)必须使用英文字母、数字、下划线组成，不能出现空格或制表符。

(3)不能使用JavaScript关键字与JavaScript保留字。

(4)不能使用JavaScript语言内部的单词，比如Infinity，NaN，undefined等。

(5)大小写敏感，如name和Name是不同的两个标识符。

14.3.1 命名规范

2.关键字

关键字是JavaScript中已经被赋予特定意义的一些单词，关键字不能作为标识符来使用。

表14-1 JavaScript的关键字

break	case	catch	continue	default
delete	do	else	finally	for
function	if	in	instanceof	new
return	switch	this	throw	try
typeof	var	void	while	with

14.3.1 命名规范

3.保留字

JavaScript中除了关键字以外，还有一些用于未来扩展时使用的保留字，保留字同样不能用于标识符的定义。

表14-2 JavaScript的保留字

abstract	boolean	byte	char	class
const	debugger	double	enum	export
extends	final	float	goto	implements
import	int	interface	long	native
package	private	protected	public	short
static	super	synchronized	throws	transient
volatile				

14.3.2 数据类型

数据类型是每一种计算机语言中的重要基础，JavaScript中的数据类型可分为字符型、数值型、布尔型、Null、Undefined和对象六种类型。

1.字符型

字符型数据又称为字符串，由若干个字符组成，并且用单引号或双引号封装起来，如“你好！”、‘你好！’、“学习‘语言’”。

- 在使用字符串的过程中，需要注意单引号、双引号必须成对使用相互包含，但不能交叉。

  如：“学习不是一件‘容易”的事件’（×，交叉错误)。

14.3.2 数据类型-数值型

2. 数值型：是JavaScript中最基本的数据类型之一，分为整型、浮点型、内部常量以及特殊值。

- 整型数值即整数，例如100、-3500、0等都是整数。整数表示方法有十进制表示、八进制和十六进制的方式表示。

使用0打头的整数是八进制整数，如017，-035等都是合法的八进制整数。

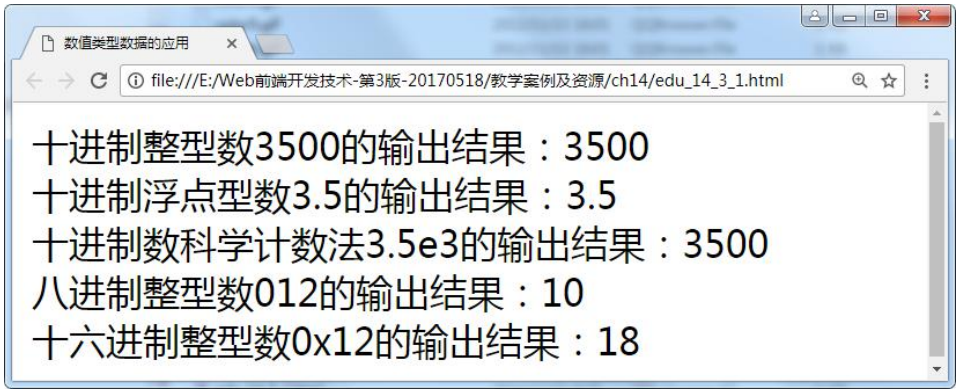
使用0x后0X打头的整数是十六进制整数，如0x16，0X3A89等都是合法的十六进制整数。

14.3.2 数据类型-数值型

浮点数，例如3.53、-534.87等都是浮点型数值。浮点数还可以采用科学计数法进行表示，如3.5E15表示3.5×10^{15}。

内部常量：Math.E(自然对数的底数e)、Math.PI等。

特殊值：Infinity（∞）、NaN-Not a Number。



14.3.2 数据类型案例

```
<!-- edu_14_3_1.html -->
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>数值类型数据的应用</title>
</head>
<body>
   <script type="text/javascript">
    var i = 3500,f = 3.5,s = 3.5e3;
    var o = 012,h = 0x12;
   document.write("十进制整型数"+i+"的输出结果："+i+"<br>");
   document.write("十进制浮点型数"+f+"的输出结果："+f+"<br>");
   document.write("十进制数科学计数法3.5e3的输出结果："+s+"<br>");
   document.write("八进制整型数012的输出结果："+o+"<br>");
   document.write("十六进制整型数0x12的输出结果："+h+"<br>");
  </script>
 </body>
</html>
```

14.3.2 数据类型-布尔型

3.Boolean（布尔型）是一种只含有true和false这两个值的数据类型，通常来说，布尔型数据表示“真”或“假”。

在实际应用中，布尔型数据常用在关系、逻辑等运算中，运算的结果往往就是true或者false。例如1<2的比较结果是true，而3==4的比较结果是false。此外，布尔型变量还常用在控制结构的语句中，如if语句等。

JavaScript中，通常采用true和false表示布尔型数据，但也可将他们转换为其他类型的数据，例如可将值为true的布尔型数据转换为整数1，而将值为false的布尔型数据转换为整数0。

14.3.2 数据类型-其它类型

4.Null：null,表示空,不是0，0是有值的。

5.Undefined：变量创建后未赋值（数字：NaN；字符串：Undefined；Boolean:false）。

6.Object:对象也是JS的重要组成部分，如date、window、document等，在后面介绍。

数据类型之间可以通过函数进行转换。

转换函数如下：

- Nnumber(value):把值转换成数字（整型或浮点数）
- String(value):把值转换成字符串
- Boolean(value):把值转换成Boolean类型

14.3.3 变量

变量：可以保存执行时变化的值的名字，称为“变量”，变量是存储信息的容器。

格式： var 变量名 [=初值][,变量名[=初值] ...]；

var作用：声明或创建变量。

具有良好编程习惯的程序员应该“先声明变量再使用”。例如：

```
var userName = “ “;
var x=0，y=2,str1=“欢迎你学习JS”；
var status = true;
var a,b,c;
str_name= “张为民"； //向未声明的变量赋值
```

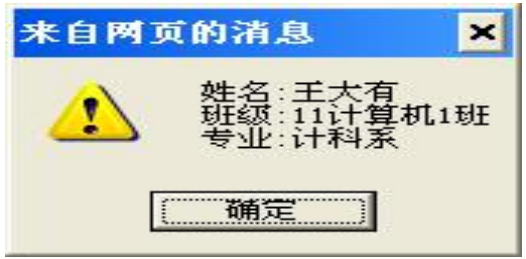
14.3.4 转义字符

如果在字符串中涉及到一些特殊字符如“\”、“"”、“'”等，这些字符无法直接使用，需要采用转义字符的方式。

转义字符	代表含义	转义字符	代表含义
\b	退格符	\t	水平制表符
\f	换页符	\'	单引号
\n	换行符	\"	双引号
\r	回车符	\\	反斜线

例如：

```
alert("姓名:王大有\n班级:11计算机1班\r专业:计科系");
```



14.4 运算符和表达式

JavaScript运算符主要有：算术运算符、关系运算符、逻辑运算符、赋值运算符、自增自减运算符、逗号运算符和位运算符等。

根据操作数的个数，将运算符分为一元运算符、二元运算符和三元运算符。

由操作数（变量、常量、函数调用等）和运算符结合在一起构成的式子称为“表达式”，最简单的表达式可以是常量名称。

对应的表达式包括：算术表达式、关系表达式、逻辑表达式、赋值表达式、自增、自减表达式、逗号表达式、条件表达式、位表达式。

14.4.1 算术运算符和表达式

算术运算符负责算术运算，用算术运算符和运算对象(操作数)连接起来符合规则的式子，称为算术表达式。

- 双元运算符（op1 operator op2）
- 单元运算符（op operator或operator op

表14-6 算术运算符

运算符	描述	例子（假定a = 2）	结果
+	加	b = a+2	b = 4
-	减	b = a -1	b = 1
*	乘	b = a * 2	b = 4
/	除	b = a / 2	b = 1
%	取模	b = a%2	b = 0
++	自增	b = a++（后置，先使用再运算）	b = 2
--	自减	b = - -a（前置，先运算再使用）	b = 1

14.4.2 关系运算符和表达式

表14-7 关系运算符和表达式

运算符	>	<	>=	<=	!=	==	===	!==
名称	大于	小于	大于 或等于	小于 或等于	不等于	等于	全等于	非全等于
表达式	6>5	6<5	6>=5	6<=5	6!=5	6==5	5==="5"	5!=="5"
结果	true	false	true	false	true	false	false	true
判断内容	数值	数值	数值	数值	数值	数值	数值与类型	数值与类型

=与==的区别：　　=是赋值运算符，==是等于运算符

===与!==：　===全等于，不仅判断数值，而且判断类型

14.4.2 关系运算符和表达式

关系运算符中有些特殊比较运算，在实际使用时需要注意。

相等性判断的特殊情况一览表					
表达式	值	表达式	值	表达式	值
null== undefined	true	"NaN" == NaN	false	false == 0	true
null == 0	false	NaN != NaN	true	true == 1	true
undefined == 0	false	NaN == NaN	false	true == 2	false
10 == NaN	false	“999" == 999	true		

14.4.3 逻辑运算符和表达式

表14-8 逻辑运算符

a	b	!a（非）	a||b或	a&&b与
true	true	false	true	*true*
true	false	false	true	false
false	true	true	true	false
false	false	true	*false*	false

练习：

true || false 结果是true;　　!true 结果是false;

3>5 && -5<-1 结果是false;

12 %3==2 结果是false;

(！ false)&&(5==="5")||(34>-34) 结果是true

14.4.4 赋值运算符和表达式

基本语法：

- 简单赋值运算：<变量> = <变量> operator <表达式>
- 复合赋值运算：<变量> operator = <表达式>

运算符	=	+=	-=	*=	/=	%=
名称	赋值	加法赋值	减法赋值	乘法赋值	除法赋值	模赋值(求余赋值)
表达式	i=6	i+=5	i-=5	i*=5	i/=5	i%=5
示例	var i=6;	i+=5;	i-=5;	i*=5;	i/=5;	i%=5;
i的结果	6	11	1	30	1.2	1
等价于		i=i+5;	i=i-5;	i=i*5;	i=i/5;	i=i%5;
位移	a=4	<<=	>>=	>>>=		
表达式		a<<=1	a>>=1	a>>>=1		
a结果		a=8	a=2	a=2		

14.4.5 位运算符和表达式

位运算符是对二进制表示的整数进行按位操作的运算符。

如果操作数是十进制或者其他进制表示的整数，运算前先将这些整数转换成32位的二进制数字，如果操作数无法转换成32位的二进制数表示，位运算的结果为NaN。

位运算符：&-按位与；~ -按位非；|-按位或；^-按位异或

其中位运算~， NOT 实质上是对数字求负，然后减 1。

例如：10&78=10

```
    00001010
&   01001110
--------------------
    00001010（10）
```

例如：81|16=81

```
    01010001
|   00010000
--------------------
    01010001（81）
```

例如：10^30=20

```
    00001010
^   00011110
--------------------
    00010100（20）
```

14.4.6 条件运算符和表达式

条件运算符是一个3元运算符，也就是该运算涉及3个操作数。

- 基本语法：

变量=布尔表达式 ? 真值表达式 : 假值表达式

```
var variable = boolean_expression ? true_value : false_value;
```

例如：

```
var v1=300,v2=-100;
var max = (v1 >v2) ? v1 : v2;  //由于v1>v2，条件为真值，所以将真值表达式v1的值赋给max，max的值为300
```

,比用if语句来得简单些。

14.4.6 其它运算符和表达式

1.逗号运算符（,）

```
var x=1 , y=2 , z=3;

x=y+z , y=x+z;
```

2.新建对象运算符（new）

```
var str1=new String();var stu=new Array();
```

3.删除运算符（delete）：是一个一元运算符，用于删除一个对象的属性或某个数组的元素。

```
delete array[30],delete object.height
```

4.类型运算符（typeof）

```
typeof(300),typeof("Welcome to You!")
```

14.5 JavaScript程序控制结构

在网页设计中JavaScript的主要作用是实现内容与行为的分离，而要实现交互式的页面必须编写相应的脚本程序。程序是专门解决某一问题的特定代码。

- JavaScript程序设计分为两种:

面向过程和面向对象的程序设计。

- 程序控制结构:

顺序结构、分支结构和循环结构。

14.5.1 顺序结构

顺序结构是最常用的一种程序结构，是按照语句出现的顺序，从第一条语句开始一步一步逐条执行，直至最后一条语句。

```
<!-- edu_14_5_1.html -->
<html>
<head>
<title>顺序结构使用实例</title>
</head>
<body>
<script type="text/javascript">
var radius = 6;
var circumference = 2 * Math.PI * radius;
var area = Math.PI * radius * radius;
alert("圆的周长为" + circumference+"\n"+"圆的面积为" + area);
</script>
</body>
</html>
```

14.5.1 顺序结构-练习

写一段代码，定义三个整数变量并任意赋初值，求它们的平均值并用alert语句输出。

```
<html>
<body>
  <script type="text/javascript">
     //计算三个数的平均值
     var x=200,y=300,z=400;
     var average=(x+y+z)/3;
     document.write(avrage);
    alert(avrage);
   </script>
</body>
</html>
```

14.5.2 分支结构

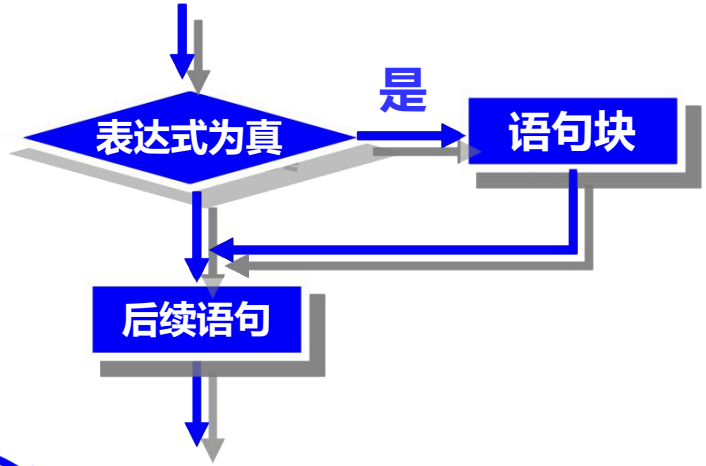
在 JavaScript 中，可以使用下面几种条件语句：

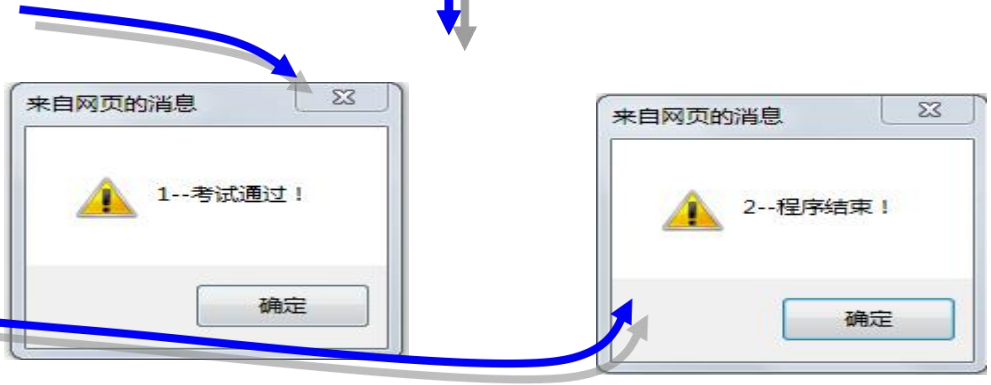
- **if 语句（单条件单分支）：在一个指定的条件成立时执行代码。**
- **if...else 语句（单条件双分支）：在指定的条件成立时执行代码，当条件不成立时执行另外的代码。**
- **if...else if....else 语句（多条件多分支）：使用这个语句可以选择执行若干块代码中的一个。**
- **switch 语句（单条件多分支）：使用这个语句可以选择执行若干块代码中的一个。**

14.5.2 分支结构-if语句

1.if语句-条件为真执行代码。
If (条件) { 语句块}

```
<script type="text/javascript">
      var x=78;
        if (x>=60)
        {alert("1-通过考试！");}
        alert("2-程序结束！");
</script>
```

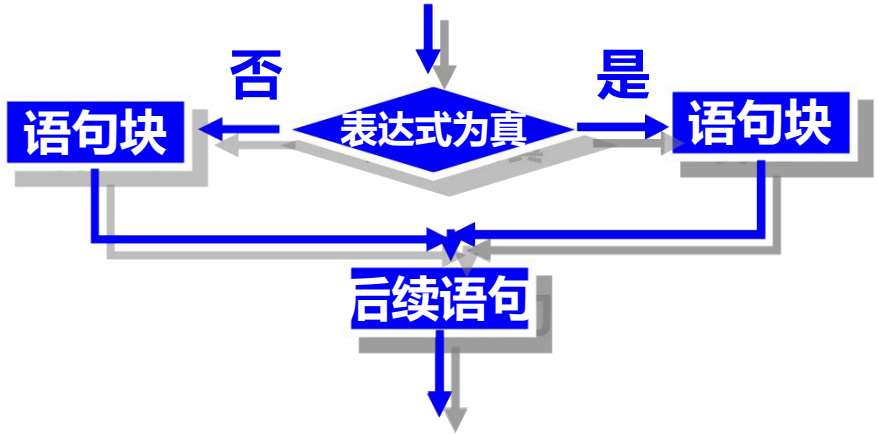


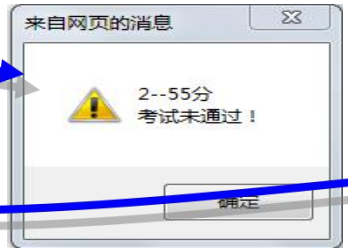


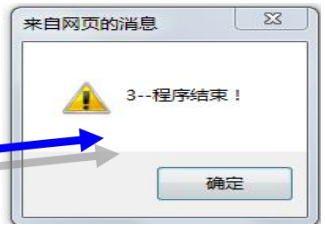
14.5.2 分支结构-if-else语句

if (条件){ 真条件语句块;}else{ 假条件语句块; }

```
<script type="text/javascript">
//判断成绩是否通过
    var x=55;
    if (x>=60){
       alert("1--考试通过！ ");}
   else{
      alert("2--55分\n\r考试未通过
！ ");}
     alert("3--程序结束！ ");
</script>
```

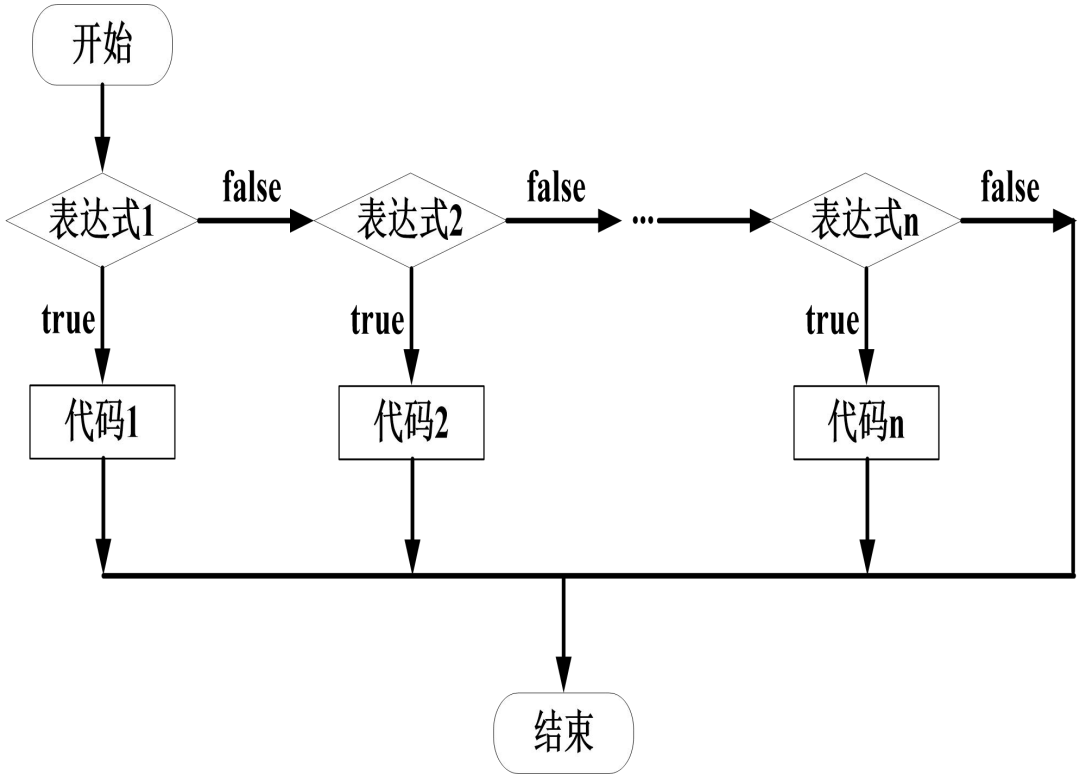






14.5.2 分支结构-if-else if-else语句

```
if (条件1){
    条件1真语句块;
} else if (条件2){
   条件2真语句块;
}
……
else if(条件x){
     条件x真语句块;
}else{
    所有条件假语句块;}
```



if-else if-else语句-案例

```
//五级制成绩判定法
<script type="text/javascript">
   var x=85;
   if (x>=90)
       {alert("1--成绩为\"优秀\"！");}
   else if (x>=80)
      {alert("2--成绩为\"良好\"！");}
   else if (x>=70)
     {alert("3--成绩为\"中等\"！");}
   else if (x>=60)
    {alert("4--成绩为\"合格\"！");}
   else{alert("5--成绩为\"不及格\"！");}
    alert("6--程序结束！");
</script>
```

14.5.2 分支结构-switch语句

```
switch(变量或表达式) {
   case 常量:
    { 语句a; }
     break;
   case 常量:
     { 语句f; }
     break;
     ...
   default:
     { 语句n; }
}
```

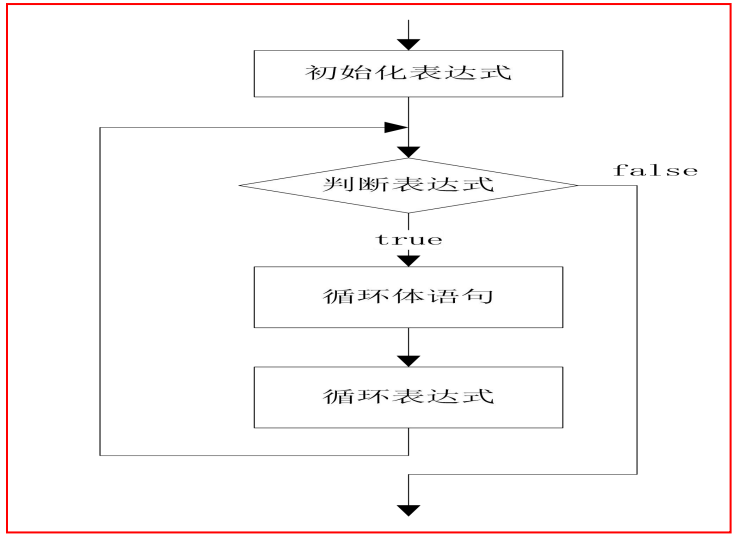
```
<script type="text/javascript"> var X=80;
 if (x>=90){level=1};
 if (x<90 && x>=80){level=2};
 if (x<80 && x>=70){level=3};
 if (x<70 && x>=60){level=4};
 if (x<60){level=5};
 switch (level)  {
  case 1:{alert("1--成绩为\"优秀\"！");break;}
  case 2:{alert("2--成绩为\"良好\"！");break;}
  case 3:{alert("3--成绩为\"中等\"！");break;}
  case 4:{alert("4--成绩为\"合格\"！");break;}
  default:{alert("5--成绩为\"不及格\"！");}  }
</script>
```

条件转换

注意：将多条件多分支转换为单条件多分支

14.5.3 循环结构-for

for (初始化表达式;判断表达式;循环表达式){
需循环执行的代码
}

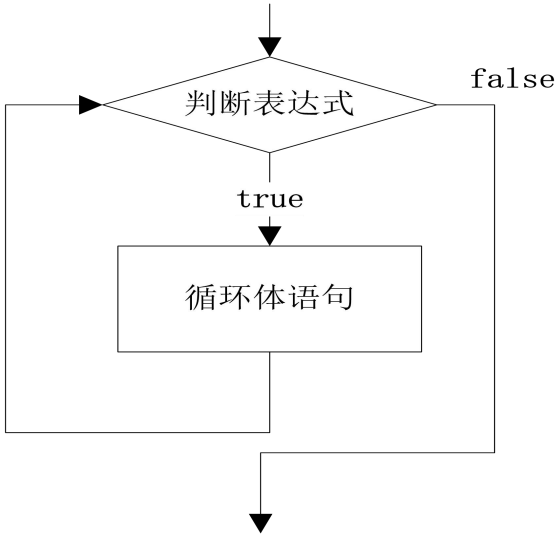



```
<script type="text/javascript">
 //计算1+2+...+n的和
 var n=prompt("输入整数N：",1);
//sum=0;
 if (n!=null)
 {
   for (sum=0,i=1;i<=n ;i++ )
    {
      sum=sum+i;
      document.write("<br>"+sum);
    }
 }
 alert("1+2+...+N="+sum);
</script>
```

左图是for循环的执行流程

14.5.3 循环结构-while语句

while(表达式) {
  需执行的代码;
}

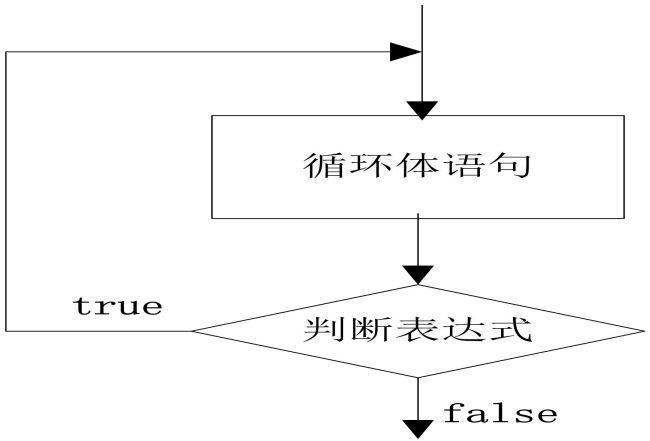



```
<script type="text/javascript">
 //计算1+2+...+n的和
 var n=prompt("请输入整数N",1);
 var i=1,sum=0;  //逗号运算符
 if (n!=null)
 {
   while (i<=n)
   {  sum=sum+i; i++;       }
   alert("1+2+...+N="+sum);
 }
</script>
```

14.5.3 循环结构-do-while语句

do… while循环

```
do{
    需执行的代码;
}while(表达式)
```




```
<script type="text/javascript">
 //计算1+2+...+n的和
 var n=prompt("输入整数N",1);
 var i=1,sum=0; //逗号运算符
 if (n!=null)
 {
    do
    {
       sum=sum+i; i++;
     }while(i<=n)
   alert("1+2+...+N="+sum);
 }
</script>
```

14.5.3 循环结构-for-in循环

该循环用来对数组或对象的属性进行操作的。基本语法：

for（变量 in 对象）{ 执行代码； }

```
<!-- edu_14_5_8.html -->
<!doctype html>
<html lang="en">
<head><meta charset="UTF-8">
<title>for-in循环的应用</title>
</head>
<body>
  <script type="text/javascript">
   var i = 1;  //定义计数器变量
   document.write("<h3>screen对象所有属性名称/属性值：</h3>");
    //1.遍历screen对象的所有属性
   for (var property in screen) {
     document.write(i+"."+property+"/"+screen[property]+"  ");
     if (i % 2 ==0) {document.write("<br/>");} //每行输出两对
     i++;
    }
```

14.5.3 循环结构-for-in循环

```
//2.遍历stu数组对象的所有元素
var stu=new Array("王春平","张宏伟","金一鑫","李大为","
   任小月","储忠庆");
var j=1;//定义计数器j
document.write("<h3>数组的元素分别为： </h3>");

for (var student in stu)
{
  document.write(j+"."+stu[student]+"  ");
   if (j % 2 ==0) {document.write("<br/>");} //每行输出两
   对
   j++;
}
</script>
</body>
</html>
```

14.5.3 循环结构-循环的嵌套

循环的嵌套：一个循环内又包含着另一个完整的循环结构，称为循环的嵌套。

九九乘法表

1*1=1

1*2=2 2*2=4

1*3=3 2*3=6 3*3=9

1*4=4 2*4=8 3*4=12 4*4=16

1*5=5 2*5=10 3*5=15 4*5=20 5*5=25

1*6=6 2*6=12 3*6=18 4*6=24 5*6=30 6*6=36

1*7=7 2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49

1*8=8 2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64

1*9=9 2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81

```
<script type="text/javascript">
document.write("九九乘法表<br>");
 var i=1,j=1;
 for (i=1;i<=9 ;i++ )
 {   // for (j=1;j<=9 ;j++ )  //九九方阵
       for (j=1;j<=i ;j++ )
       {   document.write(j+"*"+i+"="+i*j+"  "); }
      document.write("<br>"); }
 </script>
```

循环循环中断与继续

```
<script type="text/javascript">
document.write("计算部分∑N!的和<br/>");
var n = prompt("请输入整数N: ",20);
for (i=1,sum=0;i<=n ;i++ ){
  if (i>15) {break;}  //第16次时跳出循环
  //当i为1-5之间的数时结束本次循环进入下1次循环
  if (i>=1 && i<5){
     continue;
  }else{    //当i大于等于5时执行循环
  for (j=1,cj=1;j<=i ;j++ )
  {   cj=cj*j;       //计算阶乘 }
      document.write(i+"!="+cj+"<br/>");
      sum=sum+cj;    //累加阶乘之和
   }
}
i=i-1
document.write("∑"+i+"!="+sum);
</script>
```

break作用:立即结束循环并转到循环后续语句执行。

continue作用:结束本次循环，其后的语句本次不再执行，开始下一次的循环。

14.6 JavaScript函数

- JavaScript函数分为系统内部函数和系统对象定义的函数及用户自定义函数。
- 函数就是完成一个特定的功能的程序代码。函数只需要定义一次，可以多次使用，从而提高程序代码的复用率，既减轻开发人员的负担，以降低了代码的重复度。
- 函数需要先定义后使用，JavaScript函数一般定义在HTML文件的头部head标记或外部JS文件中，而函数的调用可以在HTML文件的主体body标记中任何位置。
- 常用系统函数分全局函数和对象定义的函数。全局函数它不属于任何一个内置对象，使用不需要加任何对象名称，直接使用。

14.6.1 常用系统函数-全局函数

1. 计算表达式的结果函数： eval（字符串表达式）

返回值：表达式的值或“undefined”。

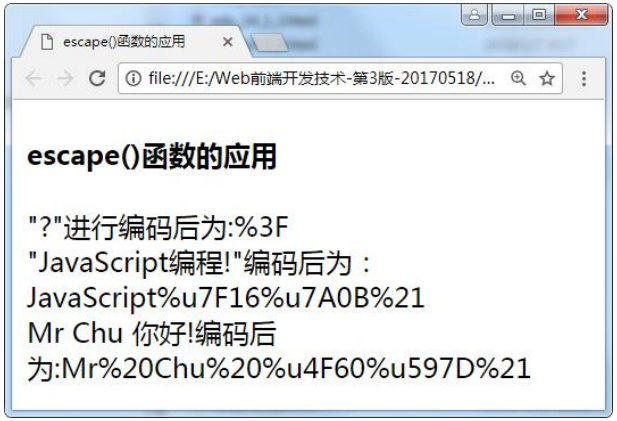
```
<!-- edu_14_6_1.html -->
<script type="text/javascript">
   eval("x=20;y=30;document.write('x为'+x+',y为'+y+',x*y的值为'+x*y)");
   document.write("<br/>");
   document.write("2+2的值为"+eval("2+2"));
   var abce;  //声明变量未赋值
   document.write("<br/>abce的值为"+eval(abce));
</script>
```

14.6.1 常用系统函数-全局函数

2.编码函数escape()： escape(字符串)

escape() 函数将参数字符串中的特定字符(ISO-Latin-1 字符集)进行编码，并返回一个编码后的字符串。它可以对空格、标点符号及其他非ASCII字母表的字符进行编码，除了以下字符：“* @ - _ + . /”。

```
<script type="text/javascript">
document.write("\"?\"进行编码后为:"+escape("?")+"<br/>");
document.write("\"JavaScript教程!\"编码后为："+escape("JavaScript教程!"));
 document.write("<br/>Tony 你好！"+"编码后为:"+escape("Tony 你好!"));
</script>
```

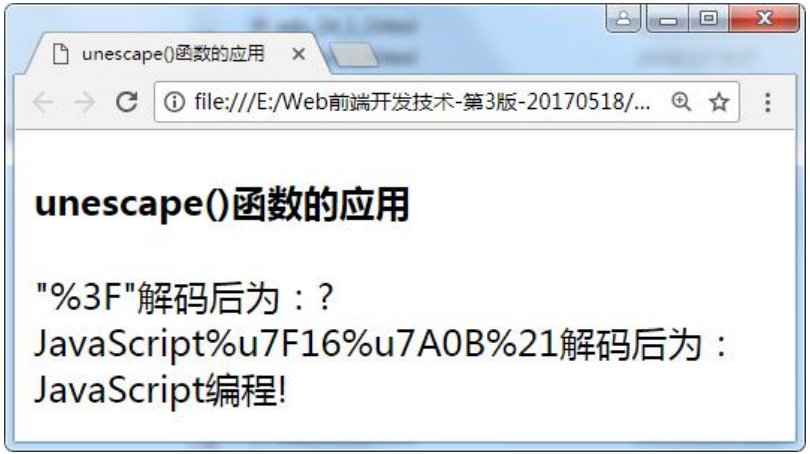


14.6.1 常用系统函数-全局函数

3.解码函数:unescape(string)

unescape 函数返回的字符串是 ISO-Latin-1 字符集的字符。参数string包含形如“%xx”的字符的字符串，此处xx为两位十六进制数值。

```
<!-- edu_14_6_3.html -->
<body>
<script type="text/javascript">
document.write("\"%3F\"解码后为:" +
unescape("%3F") + "<br />");
document.write("JavaScript%u6559%u7A0B%21解码后为: "+unescape("JavaScript%u6559%u7A0B%21"));
</script>
</body>
```



14.6.1 常用系统函数-全局函数

4.字符型转换成数值型函数:parseFloat(string)

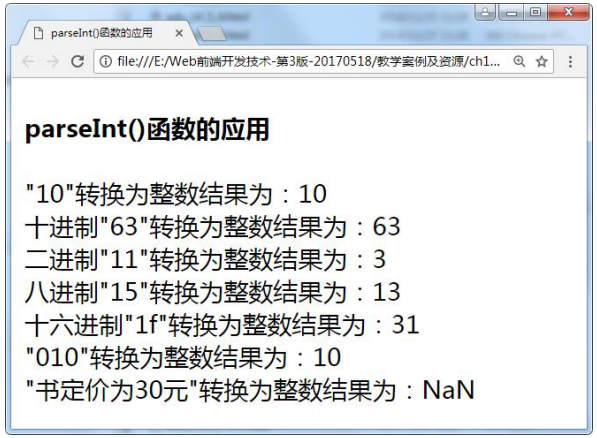
```
<script type="text/javascript">
document.write("\"100\"转换后为:
"+parseFloat("100")+"<br/>");
document.write("\"100.00\"转换后为:
"+parseFloat("100.00")+"<br/>");
document.write("\"100.88\"转换后为:
"+parseFloat("100.88")+"<br/>");
document.write("\"12 34 56\"转换后为:
"+parseFloat("12 34 56")+"<br/>");
document.write("\" 60 \"转换后为: "+parseFloat("
60 ")+"<br/>");
document.write("\"40 years\"转换后为:
"+parseFloat("40 years")+"<br/>");
document.write("\"这件衣服100元\"转换后为:
"+parseFloat("这件衣服100元")+"<br/>");
</script>
```

14.6.1 常用系统函数-全局函数

5.字符型转换成数值型函数:parseInt(numbestring , radix);

以“0x”开始-16进制；以“0“开始--8进制；其他--10进制。

```
<!-- edu_14_6_5.html -->
<body>
<script type="text/javascript">
document.write("\"10\"转换为整数结果为：
"+parseInt("10")+"<br />");
document.write("十进制\"63\"转换为整数结果为：
"+parseInt("63",10)+"<br />");
document.write("二进制\"11\"转换为整数结果为：
"+parseInt("11",2)+"<br />");
document.write("八进制\"15\"转换为整数结果为：
"+parseInt("15",8)+"<br />");
document.write("十六进制\"1f\"转换为整数结果为：
"+parseInt("1f",16)+"<br />");
document.write("\"010\"转换为整数结果为：
"+parseInt("010")+"<br />");
document.write("\"这本书定价为30元\"转换为整数结果为：
"+parseInt("这本书定价为30元")+"<br />");
</body></script>
```

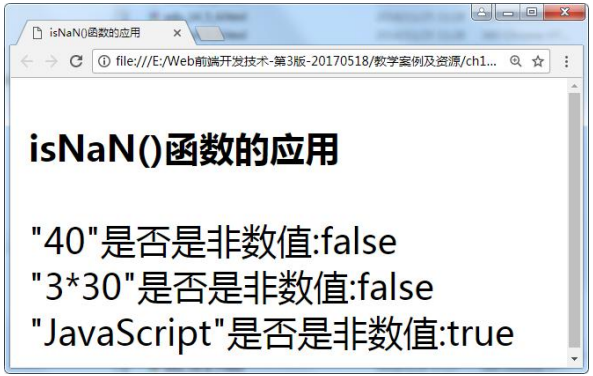


14.6.1 常用系统函数-全局函数

6.判断是否是NaN()函数:isNaN(testValue);

NaN:not a Number （*注意大小写*）

```
<!-- edu_14_6_6.html -->
<!doctype html>
<html lang="en">
 <head><meta charset="UTF-8">
<title>isNaN()函数的应用</title>
</head>
<body>
<h4>isNaN()函数的应用</h4>
<script type="text/javascript">
document.write("\"40\"是否是非数值:"+isNaN(40)+"<br>");
document.write("\"3*30\"是否是非数值:"+isNaN(3*30)+"<br>");
document.write("\"JavaScript\"是否是非数
值:"+isNaN("JavaScript"));
</script>
</body>
</html>
```



14.6.1 常用系统函数-常用的对象函数

（1）toString(radix)。将Number型数据转换为字符型数据，并返回指定的基数的结果。其中radix范围2～36，若省略该参数，则使用基数10。

```
var a = 12;alert(a.toString(2)); //告警框输出结果为1100（二进制）
alert(a.toString()); //告警框输出结果为12(默认的十进制)
```

（2）toFixed(n)。将浮点数转换为固定小数点位数的数字。n是整数，设置小数的位数，如果省略了该参数，将用0代替。

例如：

```
var a = 2016.1567;alert(a.toFixed(2)); //保留2位小数，结果为2016.16
alert(a.toFixed (5)); //保留5位小数，告警框输出结果为2016.15670
```

14.6.1 常用系统函数-常用的对象函数

（3）字符串查找和提取常用函数

方法	说明
indexOf(searchvalue,fromindex)	从前向后搜索字符串。返回某个指定的字符串值在字符串中首次出现的位置,如果没有发现，返回-1
lastIndexOf(searchvalue,fromindex)	从后向前搜索字符串。返回一个指定的字符串值最后出现的位置，如果没有发现，返回-1
charAt(index)	返回在指定位置的字符
substring(start,stop)	用于提取字符串中介于两个指定下标之间的字符

```
var str="Welcome to you!";
var substr=str.substring(3,6); //从第0个字符起，第3个-6个之间字符为"com"
var somestr=str.charAt(4);     //从第0个字符开始数，取第4个字符结果是"o"
```

其它参照案例edu_14_6_7.html

14.6.2 自定义函数

基本语法：

function functionname(argument1,argument2,..., argumentn){函数体；}

语法说明：

- 函数就是包裹在花括号中的代码块，使用关键词function来定义。当调用该函数时，会执行函数内的代码。
- 在调用函数时，可以向其传递值，这些值被称为参数。这些参数可以在函数中使用。可以发送任意多的参数，参数之间用由逗号分隔。也可以没有参数，但括号不能省略，参数类型不需要给定。

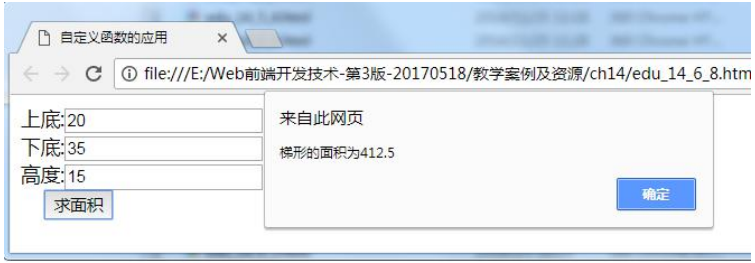
14.6.2 自定义函数

- 函数体必须写在“{”和“}”内，“{”、“}”定义了函数的开始和结束。
- JavaScript中区分字母大小写，因此“function”这个词必须是全部字母小写的，否则程序就会出错。另外需要注意的是，必须使用大小写完全相同的函数名来调用函数。例如:

```
function sum(x,y){return x*y;}
function showMessage(mess){
  alert(mess);
}
```

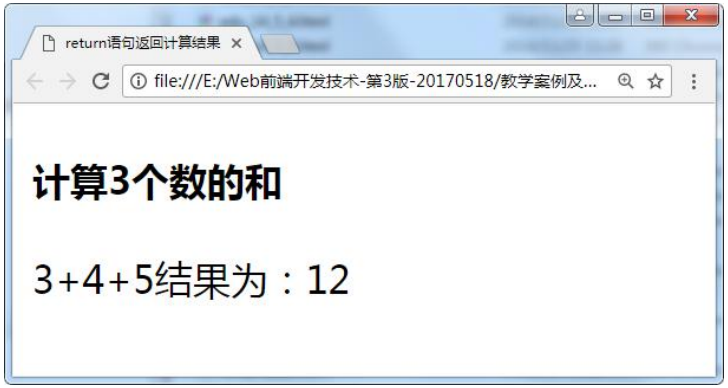
14.6.2 自定义函数

```
<!-- edu_14_6_8.html -->
<html>
<head>
<title>自定义函数的应用</title>
<script type="text/javascript">
function area(a,b,c){
s=(parseInt(a.value)+parseInt(b.value))/2*c.value;
  alert("梯形的面积为"+s);
}
</script>
</head>
<body><form>
    上底:<input type="text" name="a"><br/>
    下底:<input type="text" name="b"><br/>
    高度:<input type="text" name="c"><br/>
 <input type="button" onclick="area(a,b,c)" value="求面积"><br/>
</form>
  </body>
</html>
```



14.6.3 带参数返回的return语句

```
<!-- edu_14_6_9.html -->
<html>
<head>
<title>return语句返回计算结果</title>
<script type="text/javascript">
    function plus(a,b,c){
        return a+b+c;  //返回累加和
    }
</script>
</head>
<body>
<script type="text/javascript">
  document.write("3+4+5结果为：
  "+plus(3,4,5));
</script>
</body>
</html>
```



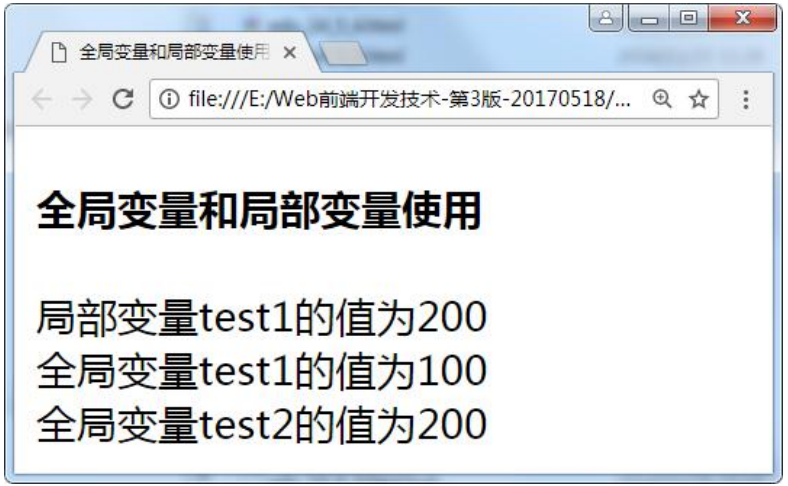
14.6.4 函数变量的作用域

变量分为局部变量和全局变量。

- 局部变量是指在函数内部声明的变量，只在一段程序中起作用的变量；全局变量是指在函数之外声明的变量，在整个JavaScript代码中都可起作用的变量，全局变量的生命周期从声明开始，在页面关闭时结束。
- 局部变量和全局变量可以重名。即在函数体外声明了一个变量，在函数体内再声明一个同名的变量。在函数体内部，局部变量的优先级高于全局变量，即在函数体内，同名的全局变量被隐藏了。
- 需要注意到是：专用于函数体内部的变量一定要用var关键字声明，否则该变量将被定义成全局变量，如果函数体外部有同名的变量，可能导致该全局变量被修改。

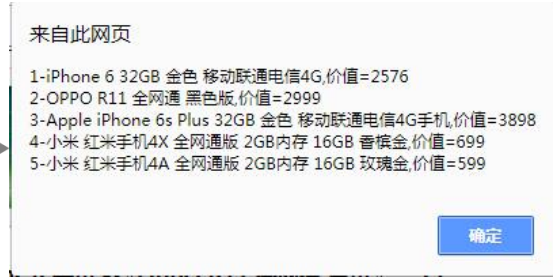
14.6.4 函数变量的作用域案例

```
<!-- edu_14_6_10.html -->
<html>
<head>
<title>全局变量和局部变量使用实例</title>
</head>
<body>
<h4>全局变量和局部变量使用</h4>
<script type="text/javascript">
  var test1 = 100;
  var test2 = 100;
  function checkScope( ){
  var test1 = 200; //同名局部变量
  test2 = 200;
document.write("局部变量test1的值为
    "+test1);
document.write("<br/>");
}
checkScope( );
document.write("全局变量test1的值为
     "+test1);
document.write("<br/>");
document.write("全局变量test2的值为
    "+test2);
</script>
</body>
</html>
```



14.7 综合实例

编程实现“手机批发业务-产品选购”页面，主要功能有查看购物车、收银台结算、初始化参数等（edu_14_7_1.html）。



14.7 综合实例-代码

```
<!-- edu_14_7_1.html -->
<!doctype html>
<html lang="en">
    <head>
    <meta charset="UTF-8">
    <title>图书选购</title>
    <style type="text/css">
    table{width: 580px;height: 200px;}
    td{text-align: center;     vertical-align: middle;}
    .myBtn {margin: 20px;width: 120px;   height: 45px;border: 1px ridge #44FFEE;}
    </style>
    <script type="text/javascript">
          var result = ""; //存放选购信息
          var price = new Array(2576.00, 2999.00, 3898.00, 699.00, 599.00, 699.00);
          var product = new Array("iPhone 6 32GB 金色 移动联通电信4G", "OPPO R11 全网通 黑色
    版", "Apple iPhone 6s Plus 32GB 金色 移动联通电信4G手机", "小米 红米手机4X 全网通版 2GB
    内存 16GB 香槟金", "小米 红米手机4A 全网通版 2GB内存 16GB 玫瑰金", "小米 红米4X 全网通版
    2GB内存 16GB 樱花粉");
          var isSelected = new Array(0, 0, 0, 0, 0, 0);
   function clearAll() {
          isSelected = [0, 0, 0, 0, 0, 0]; //选择状态全部置0
```

14.7 综合实例-代码

```
//所有复选框状态变为未选中状态
myForm.sp0.checked = false;
myForm.sp1.checked = false;
myForm.sp2.checked = false;
myForm.sp3.checked = false;
myForm.sp4.checked = false;
myForm.sp5.checked = false;
}
function checkOut() {
          var total = 0;//存放小计金额
          var count = 0;//存放选购产品件数
          for(var i = 0; i < isSelected.length; i++) {
                    count += isSelected[i];
          }
          for(var i = 0; i < price.length; i++) {
             total = total + price[i] * isSelected[i]//累计金额
          }
          alert("您所选购的" + count + "件,产品总价=" + total+"\n"+"请去支付！");
}
function shoppingCart() {
          //判断有多少个复选框被选中
```

14.7 综合实例-代码

```
var selectList = ""; //保存所选产品清单
            for(var j = 0; j < product.length; j++) {
            if(isSelected[j]) {//分行显示
            selectList += (j + 1) + "-" + product[j] + ",价值=" + price[j] + "\n";
            }
}
var info = (selectList == "") ? "您的购物车为空，请选购！" : selectList;
alert(info);//生成一个结算清单，显示输出
}
function checkSelect(number) {
            var temp;//暂存复选框状态
            switch(number) {
            case 0:
            temp = myForm.sp0.checked;break;
            case 1:
            temp = myForm.sp1.checked; break;
            case 2:
            temp = myForm.sp2.checked; break;
            case 3:
            temp = myForm.sp3.checked; break;
            case 4:
```

14.7 综合实例-代码

```
temp = myForm.sp4.checked; break;
default:
      temp = myForm.sp5.checked;       break;
}
isSelected[number] = (temp) ? 1 : 0; //记录下选中产品，1-选中，0-未选
}
</script>
</head>
<body>
<form name="myForm" method="post" action="">
<table align="center" border="1">
<caption>手机批发业务-商品备选区</caption>
<tr>
<td><img src="mobile_1.jpg" /><br />
<h4 name="h41">iPhone 6 32GB 金色 移动联通电信4G</h4><input type="checkbox"
name="sp0" value="2576" onclick="checkSelect(0);">¥ 2576.00<br /></td>
<td><img src="moblie_2.jpg" /><br />
<h4 name="h421">OPPO R11 全网通 黑色版</h4>
<input type="checkbox" name="sp1" value="2999" onclick="checkSelect(1);">¥
2999.00<br /></td>
<td><img src="moblie_3.jpg" /><br />
```

14.7 综合实例-代码

```
<h4 name="h43">Apple iPhone 6s Plus 32GB 金色 移动联通电信4G手机</h4>
<input type="checkbox" name="sp2" onclick="checkSelect(2);"> ¥ 3898.00<br
/></td>
</tr>
<tr>
<td><img src="moblie_4.jpg" /><br />
<h4 name="h44">小米 红米手机4X 全网通版 2GB内存 16GB 香槟金</h4><input
type="checkbox" name="sp3" value="699" onclick="checkSelect(3);"> ¥ 699.00
<br /></td>
<td><img src="moblie_5.jpg" /><br />
<h4 name="h45"> 小米 红米手机4A 全网通版 2GB内存 16GB 玫瑰金</h4>
<input type="checkbox" name="sp4" value="599" onclick="checkSelect(4);">¥
599.00<br /></td>
<td><img src="moblie_6.jpg" /><br />
<h4 name="h46">小米 红米4X 全网通版 2GB内存 16GB 樱花粉</h4>
<input type="checkbox" name="sp5" value="699" onclick="checkSelect(5);">¥
699.00<br /></td>
</tr>
```

14.7 综合实例-代码

```
<tr>
<td colspan="3">
<input class="myBtn" type="button" value="查看购物车"
onclick="shoppingCart();">
<input class="myBtn" type="button" value="收银台结算" onclick="checkOut();">
<input class="myBtn" type="button" value="初始化参数" onclick="clearAll();">
</td>
</tr>
</table>
</form>
</body>
</html>
```

本章小结

JavaScript是一种功能强大、使用简便的、具有安全性的客户端脚本语言。

本章简要地介绍了JavaScript语言的历史和特点，详细讲解了JavaScript的标识符、变量、运算符和表达式、三种程序控制结构（包括顺序结构、分支结构和循环结构）及函数等相关知识。通过在HTML文档中嵌入JavaScript脚本语言，可以增强用户与网页之间的交互性，并在页面中实现各种特效，提高页面的观赏性。

第15章 JavaScript事件分析(1-2课时)

使用JS
事件编程

```
105 function initEcAd(s_right,r_top) {
106 document.all.AdLayer2.style.visibility = 'visible'
107 MoveRightLayer('AdLayer2',s_right,r_top);
108 }
```

本章学习目标

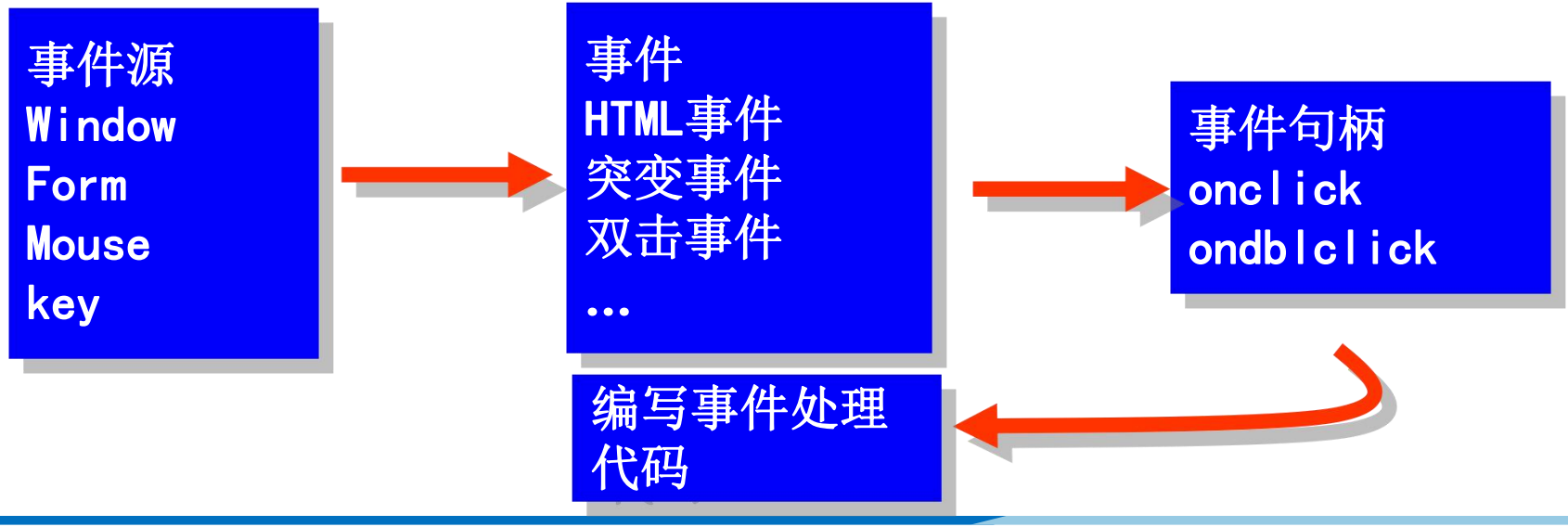
主要内容：

- 了解JavaScript事件类型。
- 理解事件发生时事件处理的三种方式。
- 学会利用表单的提交及重置事件对表单的数据进行校验。
- 理解鼠标事件中的鼠标单击及鼠标移动事件。
- 掌握常用的键盘及窗口事件。

15.1 JavaScript事件概述

事件编程:让用户不仅能够浏览页面中的内容，而且还可以和页面元素进行交互。

事件-事件是可以被JavaScript侦测到的行为(ACTION)。



15.1.1 事件类型

事件类型：

1.鼠标事件
2.键盘事件
3.HTML事件
4.突变事件

鼠标单击：例如单击button、选中checkbox和radio等元素；鼠标进入、悬浮或退出页面的某个热点：例如鼠标停在一个图片上方或者进入table的范围；

键盘按键：当按下按键或释放按键时；

HTML事件：例如页面body被加载时；在表单中选取输入框或改变输入框中文本的内容：例如选中或修改了文本框中的内容；

突变事件：主要指文档底层元素发生改变时触发的事件，如DomSubtreeModified(DOM子树修改)。

15.1.2 事件句柄

事件句柄 （event handler）

事件句柄是事件发生要进行的操作。onload属性就是我们所说的事件句柄,也称为事件属性。

基本语法：

<标记 事件句柄=“JavaScript代码”...></标记>

如： <body onload=“show()”>... </body>

格式： onload = “show();” /*load */

15.1.2 事件句柄-一览表

事件分类	事件名称	事件句柄	事件
窗口事件	load	onLoad	当文档载入时执行JS代码
	unload	onUnload	当文档卸载时执行JS代码
表单元素事件	change	onChange	当元素改变时执行JS代码
	submit	onSubmit	当表单被提交时执行JS代码
	reset	onReset	当表单被重置时执行JS代码
	select	onSelect	当元素被选取时执行JS代码
	blur	onBlur	当元素失去焦点时执行JS代码
	focus	onFocus	当元素获得焦点时执行JS代码
鼠标事件	click	onClick	当鼠标被单击时执行JS代码
	dblclick	onDblclick	当鼠标被双击时执行JS代码
	mousedown	onMouseDown	当鼠标按钮被按下时执行JS代码
	mousemove	onMouseMove	当鼠标指针移动时执行JS代码
	Mouseout	onMouseOut	当鼠标指针移出某元素时执行JS代码
	Mouseover	onMouseOver	当鼠标指针悬停于某元素之上时执行JS代码
	mouseup	onMouseUp	当鼠标按钮被松开时执行JS代码
键盘事件	keydown	onKeyDown	当键盘被按下时执行JS代码
	keypress	onKeyPress	当键盘被按下后又松开时执行JS代码
	keyup	onKeyUp	当键盘被松开时执行JS代码

15.1.3 事件处理

当一个事件发生时，如果需要截获并处理该事件，只需要定义该事件的事件句柄所关联的事件处理代码，具体的处理方式有以下3种：

- **静态指定：** <HTML标记 ... 事件句柄1="事件处理程序" [事件句柄2 = "事件处理程序" ...]></HTML标记>
- **动态指定**

  <事件主角-对象>.<事件句柄>=<事件处理程序>;
- **特定对象特定事件的指定**

```
<script type=“text/javascript” for=“对象” event=“事件”>
  //事件处理程序代码
</script>
```

15.1.3 事件处理-静态指定

基本语法：<标记 ... 事件句柄1="事件处理程序1" [事件句柄2 ="事件处理程序2" ...]

语法说明：一个标记可以同时指定**多个事件处理程序**，事件处理程序既可以是<script>标记中的自定义函数，还可以直接将事件处理代码写在此位置。

```
<!-- edu_15_1_1.html -->
<script language="javascript" type="text/javascript">
  function testInfo(mes) {alert(mes);}
  </script>
  <body>
  <h2>HTML属性的事件处理器举例</h2>
  <input type="button" value="直接通过JS语句输出信息"  onclick="alert('单击按钮，直接输出信息')">
  <input type="button" value="通过函数输出信息" onclick="testInfo('单击按钮，调用函数输出信息')">
  </body>
```

15.1.3 事件处理-动态指定

事件处理程序在JavaScript中动态指定(分配):

<事件主角-对象>.<事件句柄>=<事件处理程序>;

obj.onclick=function(){disp();} obj.onclick();//调用

初始状态没有onclick

```
<body>
<form name="myform" method="post" action="" >
   <input id= “input”  type= “button”  name= “mybutton" value="提交" >
</form>
 <script type="text/javascript">
  function clickHandler() {alert("即将提交表单！"); return true;}
  //动态分配一个事件句柄
  document.getElementByld( ‘input’ ).onclick=function(){return clickHandler();}
  myform.mybutton.onclick();
  </script>
 </body>
```

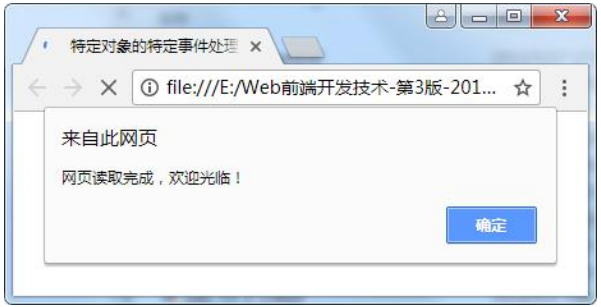
代码触发事件

15.1.3 事件处理-特定对象的特定事件指定

特定对象的特定事件指定：

```
<script type="text/javascript" for="对象" event="事件句柄">
        //事件处理程序代码
</script>
```

```
<!-- edu_15_1_3.html -->
<!doctype html>
<html lang="en">
 <head>
  <meta charset="UTF-8">
<title>给特定对象指定特定事件处理程序</title>
</head>
<body>
<h4>给特定对象指定特定事件处理程序</h4>
<script type="text/javascript" for="window"
event="onload">
    alert("网页读取完成，欢迎光临！");
</script>
body>
</html>
```



15.1.4 事件处理程序的返回值

事件处理程序的返回值

在JavaScript中通常事件处理程序不需要有返回值，这时浏览器会按默认方式进行处理；很多情况下需要使用返回值，来判断事件处理程序是否正确进行处理。

返回值类型：boolean布尔型值

浏览器根据返回值的类型决定下一步如何操作。当返回值为true，进行默认操作； 当返回值为 false，阻止浏览器的下一步操作。

基本语法：事件句柄= “return 函数名（参数）;"

15.1.4 事件处理程序的返回值-案例

```
<!-- edu_15_1_4.html -->
<!doctype html>
<html lang="en">
<head><meta charset="UTF-8">
<title>事件处理程序返回值的应用</title>
<script language="javascript">
function showName(){
if(document.form1.name1.value=="")
{  alert("没有输入内容！ ");
return false;
}else {
alert("欢迎你!"+document.form1.name1.value);
return true;}
}
</script>
</head>
<body>
<h4>事件处理程序返回值的应用</h4>
<!-- onsubmit事件处理函数返回真值就执行action
指定的网页 -->
<form name="form1"
action="simple.html" onsubmit="return
showName();">
姓名：<input type="text" name="name1"
/>
<input type="submit" value="提交"/>
</form></body>
</html>
```

返回结果为真，跳转到action属性指定的URL上执行

```
<!-- simple.html -->
<html>
 <head>
  <title> 简单测试页面
</title>
 </head>
 <body>
  <p>这是简单测试页
面</p>
 </body>
</html>
```

15.2 表单事件

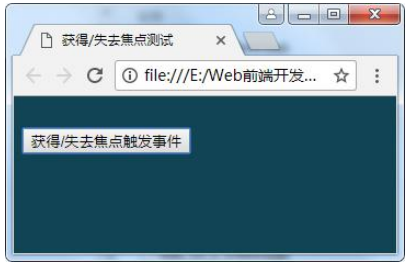
Form表单是网页设计是一种重要的与用户进行交互的工具，它用于采集用户输入各类信息。表单事件如下表所示。

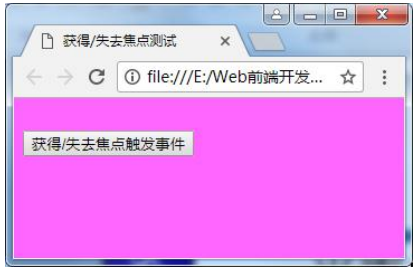
事件分类	事件句柄	事件
表单元素事件	onchange	当元素改变时执行脚本
	onsubmit	当表单被提交时执行脚本
	onreset	当表单被重置时执行脚本
	onselect	当元素被选取时执行脚本
	onblur	当元素失去焦点时执行脚本
	onfocus	当元素获得焦点时执行脚本

15.2.1 获得及失去焦点事件

当表单中的元素获得焦点时会触发Focus获得事件，当表单中的元素失去焦点时会触发Blur失去焦点事件。

```
<!-- edu_15_2_1.html -->
<html>
   <head>
      <title>获得/失去焦点测试</title>
      <script language="javascript"
type="text/javascript">
     function getFocus(){document.bgColor ="red";}
      function loseFocus(){document.bgColor ="blue";}
    </script>
   </head>
   <body>
    <form><br />
<input type="button" onfocus="getFocus()" value="
获得/失去焦点触发事件" onblur="loseFocus()"/>
    </form>
   </body>
</html>
```





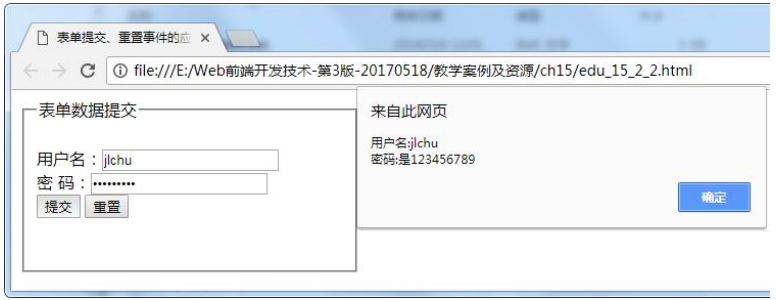
15.2.2 提交及重置事件

表单的Submit事件触发后会将表单中的数据提交到服务器端，Reset事件触发后会将表单中的数据重置为初始值。

```
<!-- edu_15_2_2.html -->
<html>
<head>
<script  type="text/javascript">
function submitTest() {
var msg ="表单数据的获取：\n";
var username = document.getElementById("input1").value;
msg+="用户名:";
msg+=username;
var psw = document.getElementById("input2").value;
          msg+="，\n密码:是";
           msg+=psw;
           alert(msg);
          return false;
          }
function resetTest() {alert("将数据清空");}
```

15.2.2 提交及重置事件-案例

```
</script>
<style type="text/css">
fieldset{width:350px;height:150px;}
</style>
</head>
<body>
  <form onsubmit="return submitTest();"
onreset="resetTest()">
   <fieldset>
   <legend>表单数据提交</legend>
   <br><label>用户名：</label><input type="text"
id="input1">
   <br><label>密 码：</label><input
type="password" id="input2">
   <br><input type="submit" value="提交">
   <input type="reset" value="重置">
   </fieldset>
</form>
</body>
</html>
```



15.2.3 改变及选择事件

```
<!-- edu_15_2_3.html -->
<html>
  <head>
<title>下拉菜单</title>
<script language="javascript">
function changeImage() {
    var a = document.getElementById("game").selectedIndex;
   //获取下拉框中选择项
    document.getElementById("show").src =
     document.getElementById("game").options[a].value;//将图片更改为对应选择项
}
</script>
</head>
<body>
<div align="center">
<form >
```

15.2.3 改变及选择事件-案例

```
<select id="game" onChange="changeImage()" >
<option value="pic4.jpg">--请选择--</option>
<option value="pic0.jpg">平板电视</option>
<option value="pic1.jpg">笔记本电脑</option>
<option value="pic2.jpg">单反相机</option>
<option value="pic3.jpg">智能手机</option>
</select>
</form>
</div>
<p align="center">
<img  src="pic4.jpg"  id="show">
</p>
</body>
</html>
```

注：当选择列表项时发生改变事件，调用函数更新图像。

15.3 鼠标事件

用户在页面上操作鼠标会触发鼠标事件，如用户单击鼠标左键会触发Click事件，双击鼠标时会触发DblClick事件，移动鼠标会触发鼠标移动事件，详见下表所示。

事件分类	事件句柄	事件
鼠标事件	onclick	当鼠标被单击时执行脚本
	ondblclick	当鼠标被双击时执行脚本
	onmousedown	当鼠标按钮被按下时执行脚本
	onmousemove	当鼠标指针移动时执行脚本
	onmouseout	当鼠标指针移出某元素时执行脚本
	onmouseover	当鼠标指针悬停于某元素之上时执行脚本
	onmouseup	当鼠标按钮被松开时执行脚本

15.3.1 鼠标单击、双击事件

```
<!-- edu_15_3_1.html -->
<!doctype html>
<html lang="en">
    <head><meta charset="UTF-8">
      <title>鼠标单击事件</title>
    <script type="text/javascript">
    function $(id){return document.getElementById(id);}
    function copyText(){
                $("target").value=$("source").value;  }
    </script>
    </head>
    <body>
      <h4>文本框内容复制</h4>
    <form method="post" action="">
    来源文本框：<input type="text" id="source" value="">
    <br>目标文本框：<input type="text" id="target"
    readonly>
    <br><input type="button" value="复制文本框内容"
    onclick="copyText();">
    </form></body>
</html>
```

15.3.2 鼠标移动事件

鼠标移动事件有：MouseOver事件、MouseOut事件、MouseDown及MouseUp等事件。

```
<!-- edu_15_3_2.html -->
<html>
<head>
<title>鼠标移动事件</title>
<script type="text/javascript">
   function mouseOver(){//鼠标盘旋
    document.getElementById('b1').src ="eg_mouse1.jpg”
   }
   function mouseOut(){//鼠标移出
     document.getElementById('b1').src ="eg_mouse2.jpg”
   }
</script>
</head>
```

15.3.2 鼠标移动事件-案例

```
<body>
      <h3 align="center">鼠标移动事件</h3>
      <hr color="blue">
      <p align="center">
      <img alt="鼠标移动" src="eg_mouse2.jpg" id="b1"
onmouseover="mouseOver()" onmouseout="mouseOut()"/>
      </p>
</body>
</html>
```

15.4 键盘事件

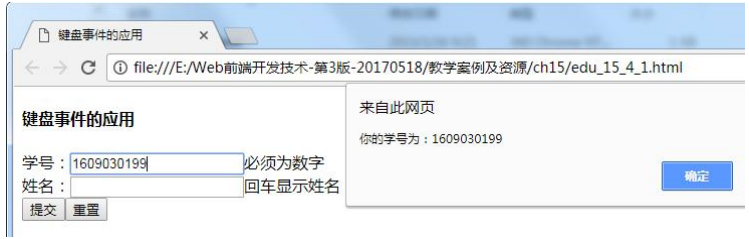
事件分类	事件句柄	事件
键盘事件	onkeydown	当键盘被按下时执行脚本
	onkeypress	当键盘被按下后又松开时执行脚本
	onkeyup	当键盘被松开时执行脚本

通过 window的event对象的keyCode属性来获取按键代码的值，其中：回车：13，0～9：48～57;Aa～Zz:65～90;

使用方法：window.event.keyCode或event.keyCode。

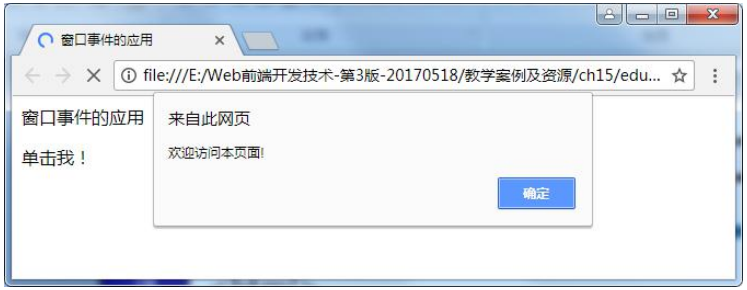
15.4 键盘事件-案例

```
<!-- edu_15_4_1.html -->
<html>
<head><title>键盘事件举例</title>
<script type="text/javascript">
function k_press(){
   if(window.event.keyCode!=13){
     if(event.keyCode<48 || event.keyCode>57
     {alert("你输入学号错误！");}
   }else{
      if(myform.s_no.value.length<=0){//未输入字符
          alert("学号不能为空");
      }else{
       alert("你的学号为："+myform.s_no.value); }
    }
}
function k_press1(){
       if(window.event.keyCode==13){
       alert("你的姓名为："+myform.s_name.value);}
}
</script>
</head>
<body>
<form name="myform" method= “get"
action="">
学号：<input type="text" name="s_no"
id="s_no" onKeyPress="k_press();"><br>
姓名：<input type= “text” name= “s_name”
id= “s_name” onkeypress= “k_press1();">
<input type="reset">
</form>
</body>
</html>
```



15.5 窗口事件

```
<!-- edu_15_5_1.html -->
<html>
   <head>
      <title>窗口事件举例</title>
     <script  type="text/javascript">
    function load(){alert("欢迎访问本页面!");}
    function myunload(){return "欢迎下次访问!";}
   </script>
  </head>
  <body onload="load();" onbeforeunload="return
myunload();"
  <h4>窗口事件的应用</h4>
  <p onclick="alert('单击我！')">单击我！</p>
  </body>
</html>
```



15.6 综合实例

页面设计分析：

采用表单和表单控件完成页面布局。采用样式表完成页面效果控制。

自定义2个JS函数，分别是检查输入数据的有效性checkReg()和清除信息clearInfo()函数。

✓通过ID获取页面元素的通用函数：

```
function $(id){ return document.getElementById(id); }
```

15.6 综合实例-代码

```
<!--  edu_15_6_1.html -->
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>用户注册页面</title>
<style type="text/css">
strong{color:red;font-style:bolder;}
fieldset{width:560px;height:186px;padding:0px 50px;}
#button{margin:10px 20px;}
</style>
<script  type="text/javascript">
function $(id){return document.getElementById(id);}
function checkReg(){
var username=$("myname").value;
var pwd=$("mypwd1").value;
var pwdConfirm=$("mypwd2").value;
var checkright=true;
if(username=="" || pwd=="")  //两者中有一个为空
{
alert("请确认用户名和密码输入是否正确！！");
```

15.6 综合实例-代码

```
    checkright=false;
}else    //不为空，再判断用户名和密码的长度合法性
{
   if(username.length<6) {
      alert("用户名长度太短，至少6个字符！！");
      checkright=false;
   } else if(pwd.length<6) {
     alert("密码长度太短，至少6个字符！！");
      checkright=false;
   } else if(pwd!=pwdConfirm) {
      alert("两次输入的密码必须一致！！");
     checkright=false;
   } else{
       checkright=true;}
   }
   return checkright;
}
function clearInfo()
{
  var flag = confirm("确认要重置数据吗？");
  if(flag==true)
  {
    $("myname").value = "";
    $("mypwd1").value = "";
    $("mypwd2").value = "";
  }
}
</script>
</head>
```

15.6 综合实例-代码

```
<body>
<form action="regsuccess.html" method="get" onSubmit="return checkReg()"
onReset="clearInfo()">
<fieldset>
   <legend align="center" >新用户注册</legend><br>
   <div>
   <label >用 户 名: </label>
   <input type="text" name="myname"><strong>(用户名要大于6位)</strong><br>
   <label> 登录密码: </label>
   <input type="password" name="mypwd1"><strong>(密码要大于6位)</strong><br>
  <label> 密码确认: </label>
   <input type="password" name="mypwd2"><br>
  <input id="button" type="submit" value="用户注册" >
  <input  id="button" type="reset" value="重置">
  </div>
 </fieldset>
</form>
</body>
```

本章小结

本章介绍JavaScript脚本中的事件处理的概念、方法，列出了常用的事件及事件句柄，并且介绍了如何编写用户自定义的事件处理函数以及如何将它们与页面中用户的动作相关联，以得到预期的交互性能。

重点介绍了Web开发中常用的表单事件、鼠标事件、键盘事件等。在表单事件中，详细介绍表单元素的焦点事件、表单提交与重置事件以及表单元素的选中及改变事件。在鼠标事件中，详细介绍鼠标单击及鼠标移动事件。在窗口事件中，主要介绍了装载事件和卸载事件。Web前端开发人员只要掌握JavaScript事件概念、事件触发类型和事件处理的方式，就可以开发出具有交互性、动态性的页面。