第十一章 JavaScript语言基础

Web程序设计基础

学习目标

了解JavaScript的数据类型

了解变量的声明和作用范围

了解JavaScript运算符及运算顺序

掌握流程控制语句及其编程

掌握的函数的定义及调用方法

目录

语法概述

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。

- 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。例如：

```
alert("这是告警消息框!");
```

- 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。例如：

```
{var s=0;document.write("S的值=" +s);}
```

语法概述

JavaScript代码是由若干条语句或语句块构成的执行体。例如：

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

语法概述

- JavaScript程序按照在HTML文件中出现的顺序逐行执行。

- JavaScript严格区分字母大小写，例如，变量username与变量userName是两个不同的变量。

- JavaScript每行结尾的分号可有可无。

- JavaScript中有两种注释：单行注释和多行注释。

  - 单行注释使用双斜线“//”作为注释标记，它后面的内容就是注释部分。
  - 多行注释可以包含任意行数的注释文本。多行注释是以“/*”标记开始，以“*/”标记结束，中间的所有内容都为注释文本。

11.1 数据类型与变量

双引：　“你好！”
单引：　‘你好！’
嵌套：　“学习‘语言’”
错误用法：交叉
“学习不是一件‘容易”的事件’

数据类型是每一种计算机语言中的[illegible]t中的数据类型可分为**三大类：**

简单数据类型

- **数值数据number**

JavaScript中没有整数和浮点数之分。 例如

- **文本数据string**

由双引号或者单引号括起来的0个或多个字符组成的序列。 例如

- **布尔数boolean**

是一种只含有true和false这两个值的数据类型,还可以用0表示flase，非0整数表示true。

复合数据类型

- **数组Array**

数组主要用来保存一组相同或不同数据类型的数据。

- **函数function**

函数来保存一段程序，这段程序可以在JavaScript中反复被调用

- **对象Objects**

对象用来保存一组不同类型的数据和函数等。

特殊数据类型

- **无定义数据类型undefined**
  - 在引用一个定义过但没有赋值的变量时的返回值。
  - 在引用一个不存在的数组时的返回值。
  - 在引用一个不存在的对象属性时的返回值。
- **空数据null**

表示没有值。用于定义空的或不存在的引用。

11.1 数据类型与变量

在程序运行期间，随时可能产生一些临时数据，应用程序会将这些数据保存在一些内存单元中。变量就是指程序中一个已经命名的存储单元，它的主要作用就是为数据操作提供存放信息的容器。

变量的命名

1. 必须以字母或下划线开头，中间可以是数字、字母或下划线。
2. 变量名不能包含空格、加、减等符号。
3. 不能使用JavaScript中的关键字和保留字作为变量名，如表11-3和表11-4所列。
4. JavaScript的变量名严格区分大小写。

11.1 数据类型与变量

声明变量基本语法格式

```
var 变量名;
```

声明变量时，需要遵循的规则如下：

声明变量遵循规则

1. 可以使用一个关键字var同时声明多个变量。

例如：

```
var a,b,c        //同时声明a、b和c三个变量
```

11.1 数据类型与变量

声明变量基本语法格式

```
var 变量名;
```

声明变量时，需要遵循的规则如下：

声明变量遵循规则

1. 可以使用一个关键字var同时声明多个变量
2. 可以在声明变量的同时对其赋值，即初始化。

例如：

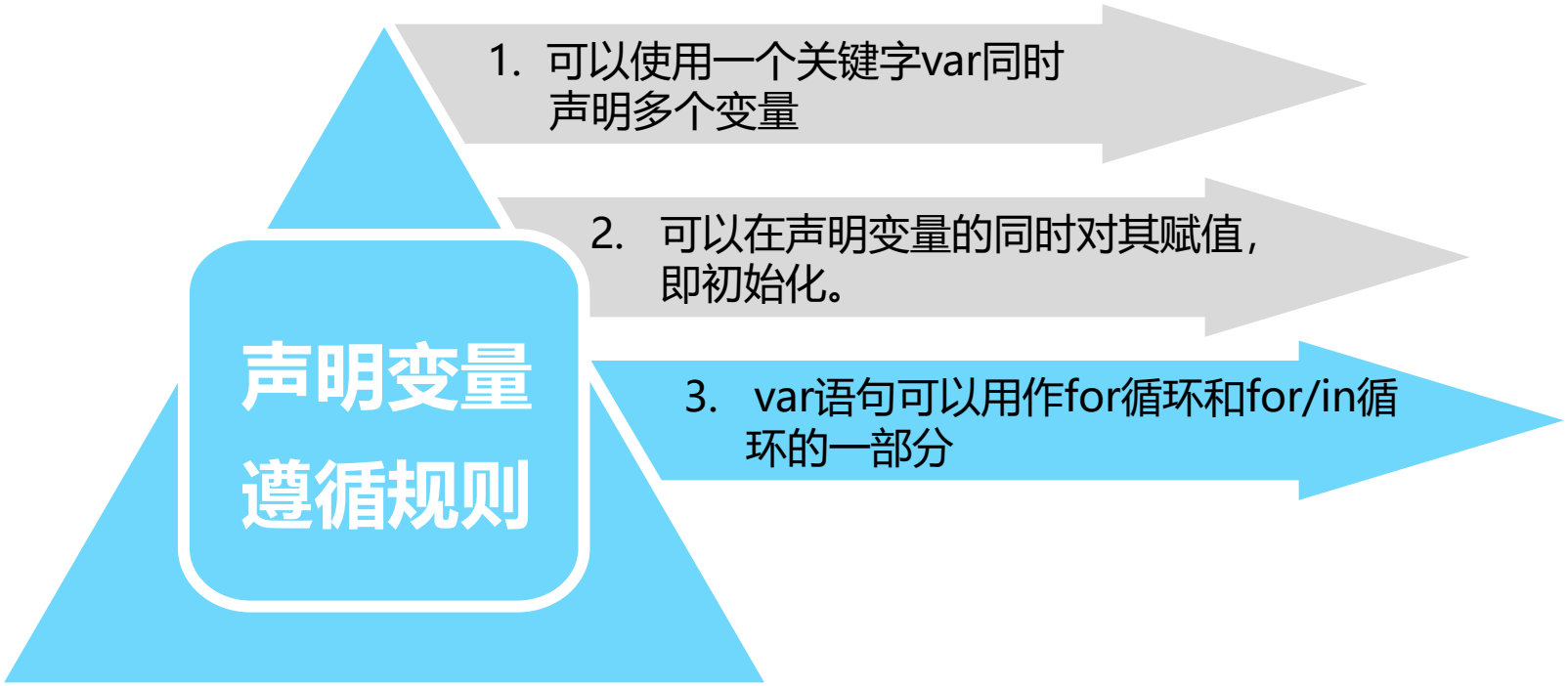
```
var a=1,b=2,c=3;  //同时声明a、b和c三个变量，并分别对其进行初始化
```

11.1 数据类型与变量

声明变量基本语法格式

```
var 变量名;
```

声明变量时，需要遵循的规则如下：

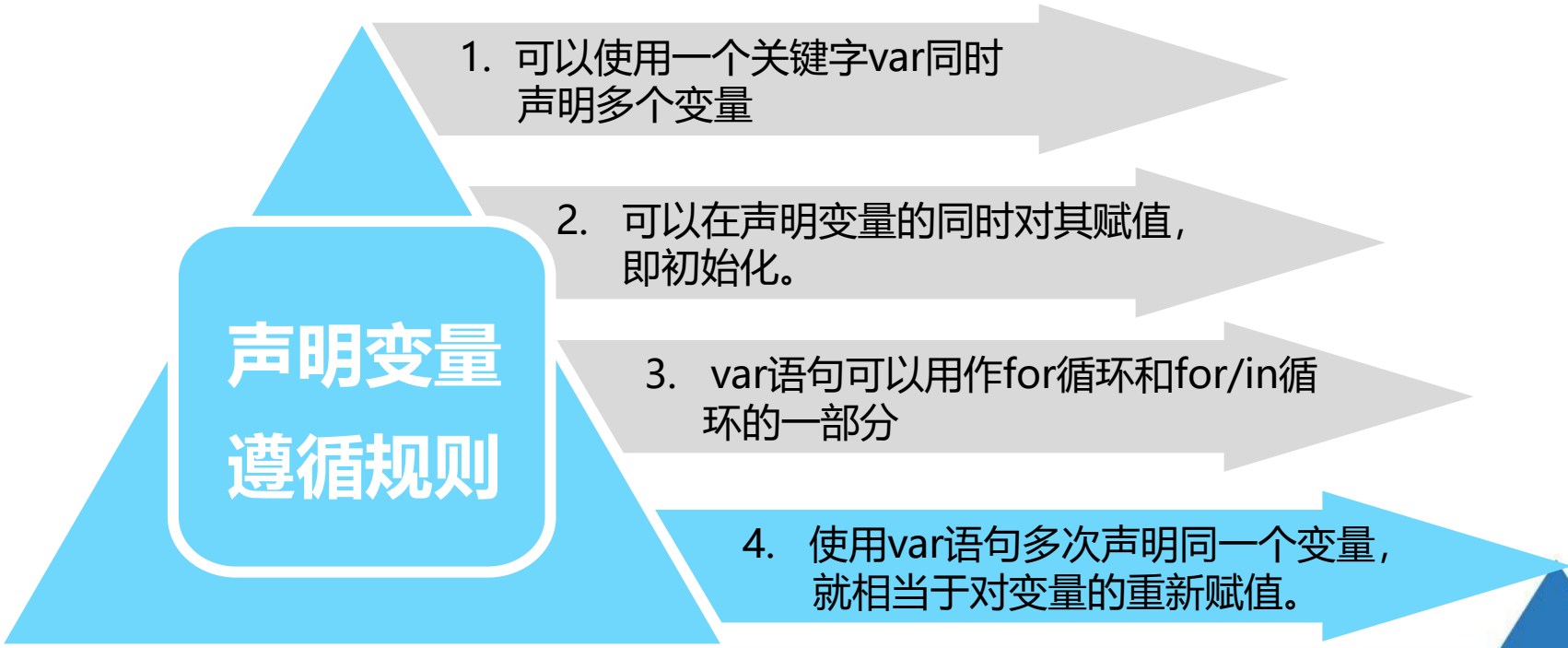


11.1 数据类型与变量

声明变量基本语法格式

```
var 变量名;
```

声明变量时，需要遵循的规则如下：



由于JavaScript采用弱类型的形式，因此可以不理会变量的数据类型，即可把任意类型的数据赋值给变量。

例如：

```
var a=100;                                  //数值类型
var str=“太原理工大学软件学院";    //字符串类型
var bue=true;                               //布尔类型
```

11.1 数据类型与变量

变量的作用范围 **指可以访问该变量的代码区域。**按变量的作用范围分为全局变量和局部变量。

全局变量

可以在整个HTML文档范围中使用的变量，这种变量通常都是在函数体外定义的变量。

局部变量

只能在局部范围内使用的变量，这种变量通常都是在函数体内定义的变量，所以只在函数体内部有效。

全局变量和局部变量的应用案例

[点击查看案例]

```
<!DOCTYPE html>
 <html>  <head>
  <title>变量的作用范围</title>
  <script type="text/javascript">
     var name="此处为全局变量的信息"; //定义全局变量
     //函数的定义
     function test(){
          var name="此处为局部变量的信息";       //定义局部变量
          alert(name);       //弹出信息：“此处为局部变量的信息”
         }
  //调用函数
  test();
  alert(name);                  //弹出信息：“此处为全局变量的信息”
</script> </head>
<body></body></html>
```

当test()函数中的局部变量name与全局变量name重名时，**函数中变量将覆盖全局变量**，弹出信息为“此处为局部变量的信息”。

11.2 运算符与表达式

运算符 是程序执行特定算术或逻辑操作的符号，用于执行程序代码运算。JavaScript中的运算符主要包括：

◆ 算术运算符

[点击查看案例]

```
<!DOCTYPE html>
<html>
 <head>
  <title>算术运算符</title></head>
 <body>
  <script type="text/javascript">
    var num1=100,num2=25;
    document.write("100+25="+
                   (num1+num2)+"<br>");
    document.write("100-25="+
                   (num1-num2)+"<br>");
    document.write("100*25="+
                   (num1*num2)+"<br>");
    document.write("100/25="+
                   (num1/num2)+"<br>");
    document.write("(100++)="+
                   (num1++)+"<br>");
    document.write("(++100)="+
                   (++num1)+"<br>");
  </script>
 </body>
</html>
```

算术运算符

运算符	描述
+	加，用于计算两个数之和
-	减，用于计算两个数之差
*	乘，用于计算两个数之积
/	除，用于计算两个数之商
%	取余，除法运算中的取余数
++	自加，在原来的基础上加1
--	自减，在原来的基础上减1

◆ 赋值运算符

常用赋值运算符

运算符	描述	运算符	描述
+=	x+=y，即对应于x=x+y	&=	x&=y，即对应于x=x&y
-=	x-=y，即对应于x=x-y	|=	x|=y，即对应于x=x|y
*=	x*=y，即对应于x=x*y	^=	x^=y，即对应于x=x^y
/=	x/=y，即对应于x=x/y	<<=	x<<=y，即对应于x=x<<y
%=	x%=y，即对应于x=x%y	>>=	x>>=y，即对应于x=x>>y
		>>>=	x>>>=y，即对应于x=x>>>y

赋值运算的应用案例

[点击查看案例]

```
<!DOCTYPE html>
<HTML>
  <HEAD>
       <title>赋值运算</title></head>
 <body>
  <pre>
<script type="text/javascript">
    var a=3,b=2;
    document.writeln("a=3,b=2");
    document.write("a+=b=");a+=b;document.writeln(a);
    document.write("a-=b=");a-=b;document.writeln(a);
    document.write("a*=b=");a*=b;document.writeln(a);
    document.write("a/=b=");a/=b;document.writeln(a);
    document.write("a%=b=");a%=b;document.writeln(a);
  </script>   </pre>
 </body></html>
```

◆ 赋值运算符

- 在书写**复合赋值运算符**时**必须连续书写**，符号之间不允许使用空格，否则将会出错。
- 赋值运算符的**左操作数必须是一个变量**，JavaScript中可以对变量进行连续赋值，这时为右关联，从右向左运算符被分组。例如：

  a = b = c等价于a = (b = c)

- 如果赋值运算符两边的**操作数类型不一致**，如果存在隐式转换，系统会自动将赋值运算符右边的类型转换为左边的类型再赋值；如果不存在隐式转换，则无法赋值程序会报错，为避免此类情况，用户可以**先进行类型转换，然后再赋值**。

◆ 关系运算符

运算符	描 述
>	大于，左侧的值大于右侧的值时，则返回true；否则返回false
>=	大于等于，左侧的值大于等于右侧的值时，则返回true；否则返回false
<	小于，左侧的值小于右侧的值时，则返回true；否则返回false
<=	小于等于，左侧的值小于等于右侧的值时，则返回true；否则返回false
!=	不等于，左侧与右侧的值不相等时，则返回true；否则返回false
==	等于，左侧与右侧的值相等时，则返回true；否则返回false
!==	严格不等于，左侧与右侧的值不相等或数据类型不同时，返回true；否则返回false
===	严格等于，左侧与右侧的值相等，并且数据类型相同时，返回true；否则返回false

◆ **==与===的区别在于：**

- == 支持自动类型转换，只要前后两个变量的值相同就返回true，而忽略数据类型的比较
- === 需要两个变量的值相同并且数据类型一致时才返回true

关系运算符的应用案例

[点击查看案例]

```
<!DOCTYPE html>
<html>
 <head>
  <title>关系运算符</title>
 </head>
 <body>
  <script type="text/javascript">
        document.write("8>5的结果是： "+(8>5)+"<br/>");
        document.write("8>=10的结果是： "+(8>=10)+"<br/>");
        document.write("8!=8结果是： "+(8!=8)+"<br/>");
        document.write("8!='8'结果是： "+(8!='8')+"<br/>");
        document.write("8=='8'结果是： "+(8=='8')+"<br/>");
        document.write("8!=='8'结果是： "+(8!=='8')+"<br/>");
        document.write("8==='8'结果是： "+(8==='8')+"<br/>");
  </script>
 </body>
</html>
```

◆ 位运算符

位运算符 是对二进制数进行运算的运算符。JavaScript中位运算符有：&（与）、|（或）、^（异或）、~（非）、<<（左移）、>>（右移）。其中，非运算符为一元运算符，其他的位运算符都是二元运算符。

运算符	描 述
&	位与运算。 操作数中两个位都为1，结果为1，两个位中有一个为0，结果为0
|	位或运算。 操作数中两个位都为0，结果为0，否则，结果为1
^	位异或运算。 两个操作位相同时，结果为0，不同时，结果为1
~	位非运算。 操作数各个位取反，1变为0，0变为1
<<	左位移。 操作数按位左移，高位被丢弃，低位依次补0
>>	右位移。 操作数按位右移，低位被丢弃，高位依照原有符号位填充空位

位运算符的应用案例

[点击查看案例]

```
<!DOCTYPE html>
<html>  <head>
   <title>位运算符</title></head>
 <body>
  <script type="text/javascript">
     document.write(" 8&3="+ (8&3)+"<br>");
     document.write (" 8|3="+ (8|3)+"<br>");
     document.write (" 8^3="+ (8^3)+"<br>");
     document.write (" ~8="+ (~8)+"<br>");
     document.write (" 8 << 2="+ (8 << 2)+"<br>");
     document.write (" 8 >> 2="+ (8 >> 2)+"<br>");
   </script>
 </body>
</html>
```

◆ 逻辑运算符

[点击查看案例]

逻辑运算符 JavaScript语言提供了&&、||、!三种逻辑运算符，分别是逻辑与、逻辑或、逻辑非。

```
<!DOCTYPE html>
<html>
 <head>
     <title>逻辑运算符</title>  </head>
 <body>
   <script type="text/javascript">
document.write("8>5 && '8'>10的结果是： "+(8>5 && '8'>10)+"<br/>");
document.write("8>5 || '8'>10的结果是： "+(8>5 || '8'>10)+"<br/>");
document.write("!true结果是： "+(!true)+"<br/>");
document.write("!false结果是： "+(!false)+"<br/>");
   </script>
 </body>
</html>
```

◆ 其他运算符

条件运算符 是JavaScript中唯一的一个三元运算符，其符号为“?:”。由条件运算符组成的表达式称为条件表达式 。

基本语法格式

条件表达式? 表达式1：表达式2

执行过程： 先计算条件表达式，若结果为true，计算表达式1，若结果为false ，计算表达式2。

- ?:的第一个操作数必须是一个可以隐式转换成布尔型的常量、变量或表达式，如果不满足上述条件，则发生运行错误。
- ?:的第二个和第三个操作数控制了条件表达式的类型。它们可以是JavaScript中任意类型的表达式。

条件表达式的应用案例

[点击查看案例]

```
<!DOCTYPE html>
<html>
<head>
   <title>条件表达式</title>
</head>
 <body>
  <script type="text/javascript">
     var  a=3,b=4
     document.write("a和b中最大的是"+(a>b?a:b));
</script>
 </body>
</html>
```

◆ 运算符的优先级

优先级（1最高）	说明	运算符
1	括号	( )
2	自加、自减运算符	++ --
3	乘法、除法、求余运算符	* / %
4	加法、减法运算符	+ -
5	小于、小于等于、大于、大于等于	< <= > >=
6	等于、不等于	== !=
7	逻辑与	&&
8	逻辑或	||
9	赋值运算符和快捷运算符	=、+=、-=、*=、/=、%=、

11.3 流程控制语句

无论传统的编程语言，还是脚本语言，构成**程序的基本结构**都是**顺序结构**、**选择结构**和**循环结构**三种。

1. 选择语句

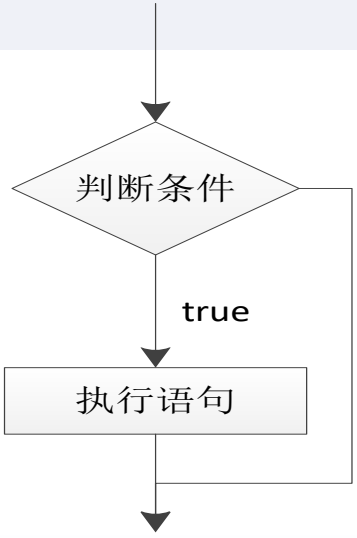
所谓选择语句就是对语句中不同条件的值进行判断，进而根据不同的条件执行不同的语句。选择语句主要有两类：一类是if判断语句；另一类是switch多分支语句。

1. 选择语句

if语句

（1）单向判断语句

```
if（执行条件）{
        执行语句
}
```



单向判断语句执行流程

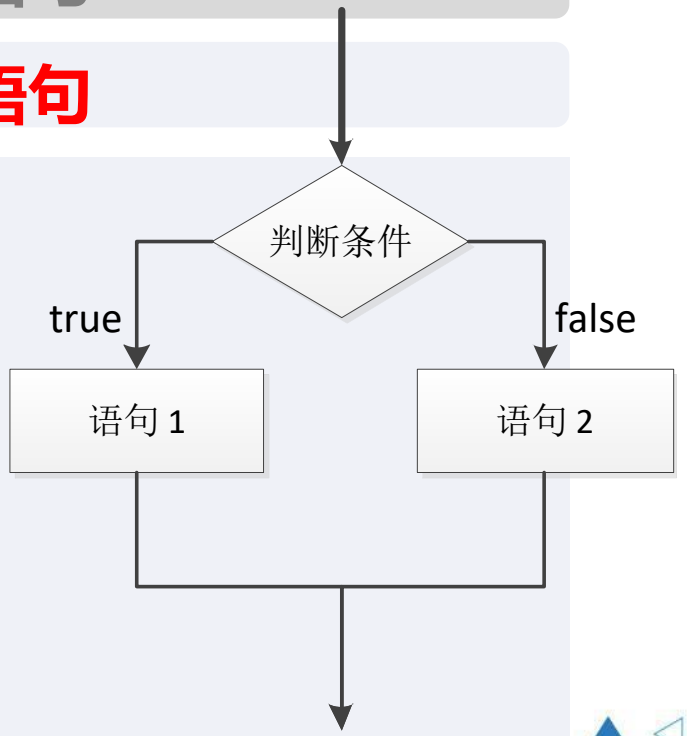
1. 选择语句

if语句

（1）单向判断语句

（2）双向判断语句

```
if（执行条件）{
执行语句1
}
else{
执行语句2
}
```



【例11-1】设计一个程序，判断考试成绩是否合格。以60分为标准，60分一下为不合格，60分以上为合格。在第一个文本框文本框输入分数，点击【确定】按钮，在第二个文本框内显示成绩是否合格。

1 单向判断实现

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>判断成绩是否合格</title>
<script>
function score()
{var userScore=document.scoreForm.txtScore.value;
  var result="合格";
   if (userScore<60)
      {result="不合格";}
   document.scoreForm.txtResult.value = result;
}
</script> </head>
```

【例11-1】设计一个程序，判断考试成绩是否合格。以60分为标准，60分一下为不合格，60分以上为合格。在第一个文本框文本框输入分数，点击【确定】按钮，在第二个文本框内显示成绩是否合格。

[点击查看案例]

```
<body>
<form id="scoreForm" name="scoreForm" method="get">
 <p><label>输入分数：</label>
 <input type="text" id="txtScore" name="txtScore"></p>
 <p><label>判断结果：</label><input type="text" id="txtResult"
name="txtResult"></p>
 <p><input type="button" value="确 定" onClick="score()"></p>
</form>
</body>
</html>
```

【例11-1】设计一个程序，判断考试成绩是否合格。以60分为标准，60分一下为不合格，60分以上为合格。在第一个文本框文本框输入分数，点击【确定】按钮，在第二个文本框内显示成绩是否合格。

2 双向判断实现

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>判断成绩是否合格</title>
<script>
function score()
{var userScore=document.scoreForm.txtScore.value;
  var result
   if (userScore<60)
      {result="不合格";}
else {result="合格";}
   document.scoreForm.txtResult.value = result;
}
</script> </head>
```

【例11-1】设计一个程序，判断考试成绩是否合格。以60分为标准，60分一下为不合格，60分以上为合格。在第一个文本框文本框输入分数，点击【确定】按钮，在第二个文本框内显示成绩是否合格。

[点击查看案例]

```
<body>
<form id="scoreForm" name="scoreForm" method="get">
 <p><label>输入分数：</label>
 <input type="text" id="txtScore" name="txtScore"></p>
 <p><label>判断结果：</label><input type="text" id="txtResult"
name="txtResult"></p>
 <p><input type="button" value="确 定" onClick="score()"></p>
</form>
</body>
</html>
```

1. 选择语句

switch语句

```
switch (表达式){
        case 目标值1:
                执行语句1
                break;
        case 目标值2:
                执行语句2
                break;
        . . . . . .
        case 目标值n:
                执行语句n
                break;
        default:
                执行语句n+1
                break;
}
```

【例11-2】 设计一个分数等级转换程序。90分以上为优秀，80~89分为良好，70~79分为中等，60~69分为及格，60分一下为不及格。要求在第一个文本框内输入分数，单击【确定】按钮，在第二个文本框内显示成绩等级。

```
<!doctype html>
<html><head><meta charset="utf-8">
<title>判断成绩等级</title>
<script>
function score()
{var userScore=document.scoreForm.txtScore.value;
  var result
  switch(true){
        case userScore>=0&&userScore<60:result="不合格";break;
        case userScore>=60&&userScore<70:result="及格";break;
        case userScore>=70&&userScore<80:result="中等";break;
        case userScore>=80&&userScore<90:result="良好";break;
        case userScore>=90&&userScore<100:result="优秀";break;
        default:result="成绩不合法";break;}
     document.scoreForm.txtResult.value = result;
}
</script> </head>
```

【例11-2】 设计一个分数等级转换程序。90分以上为优秀，80~89分为良好，70~79分为中等，60~69分为及格，60分一下为不及格。要求在第一个文本框内输入分数，单击【确定】按钮，在第二个文本框内显示成绩等级。

[点击查看案例]

```
<body>
<form id="scoreForm" name="scoreForm" method="get">
 <p><label>输入分数：</label>
 <input type="text" id="txtScore" name="txtScore"></p>
 <p><label>判断结果：</label><input type="text" id="txtResult"
name="txtResult"></p>
 <p><input type="button" value="确 定" onClick="score()"></p>
</form>
</body>
</html>
```

2. 循环语句

一个完整的循环结构，必须有以下四个基本要素：循环变量初始化、循环条件、循环体和改变循环变量的值。

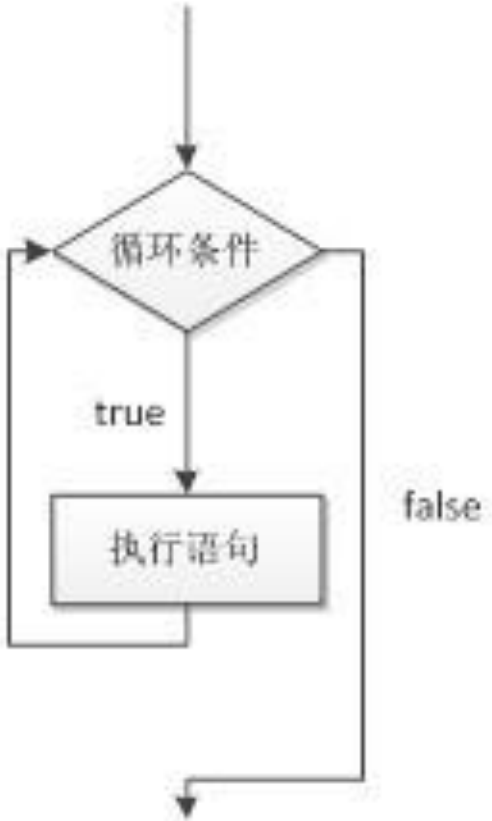
JavaScript语言提供了while、do...while、for三种循环语句。

2. 循环语句

While循环语句 | do...While循环语句 | for循环语句

基本语法格式

```
while(循环条件){
循环体语句;
}
```

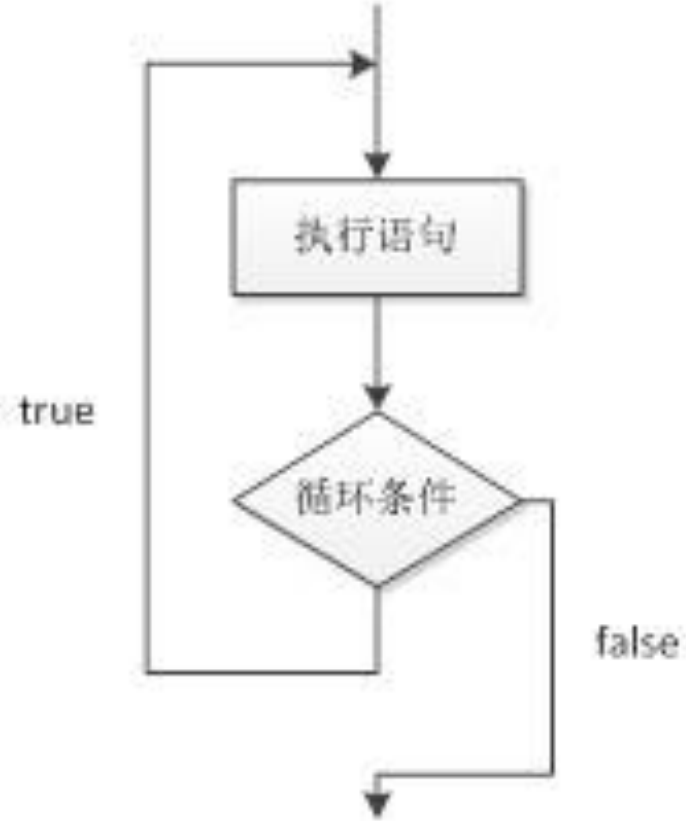


2. 循环语句

While循环语句 | do...While循环语句 | for循环语句

基本语法格式

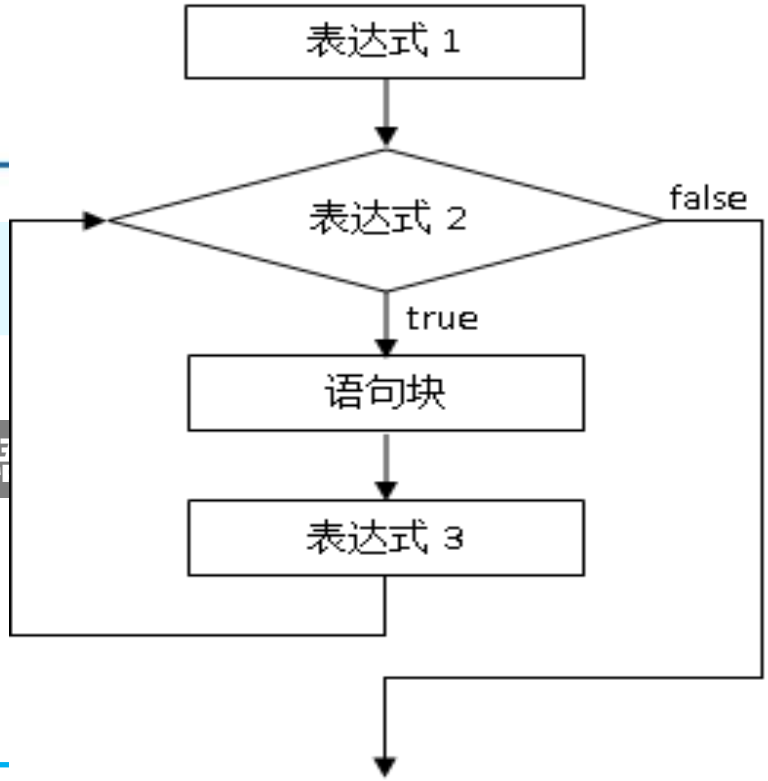
```
do {
循环体语句;
} while(循环条件);
```



11.3 流程控制语句

2. 循环语句

While循环语句　　do...While循环语



基本语法格式

```
for（初始化表达式; 循环条件; 操作表达式）{
        循环体语句;
}
```

【例11-4】计算100以内自然数之和，即1+2+3+...+100。并显示结果。

[点击查看案例]

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>while语句计算100以内自然数之和</title>
<script>
var i=1,sum=0;  //声明变量i和sum，并赋初值
while(i<=100)
{
        sum+=i;
        i++;
}
document.write("1+2+3+...+100 = "+sum);       //输出运算结果
</script>
</head>
<body></body>
</html>
```

【例11-5】计算100以内自然数之和，即1+2+3+...+100。并显示结果。(改用do……while)

[点击查看案例]

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>while语句计算100以内自然数之和</title>
<script>
var i=1,sum=0;  //声明变量i和sum，并赋初值
do
{
        sum+=i;
        i++;
}
while(i<=100);
document.write("1+2+3+...+100 = "+sum);        //输出运算结果
</script>
</head>
<body></body></html>
```

【例11-6】计算100以内自然数之和，即1+2+3+...+100。并显示结果。(改用for)

[点击查看案例]

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>while语句计算100以内自然数之和</title>
<script>
var sum=0;
for(var i=1;i<=100;i++)
{
        sum+=i;
}
document.write("1+2+3+...+100 = "+sum);        //输出运算结果
</script>
</head>
<body>
</body>
</html>
```

11.4 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。

JavaScript**函数**分为**用户自定义函数**、**系统内部函数**及**系统对象定义的函数**。

1. 自定义函数

使用函数前，先要**定义函数**。在JavaScript中定义函数常用的方法有两种：**声明式函数**、**匿名函数**。

□ 声明式函数

定义函

```
<script typ
        functi                    ]) {
            函数体
            [return 表达式]
        }
</script>
```

参数：外界传递给函数的值，它是可选的，当个参数时，各参数”分隔。

函数体：函数定义的主体，实

return：指定函数的返回值，可选参数，如果省略返回值（表达式），代表结束函数。

1. 自定义函数

- 例如：

这里定义了名为welcome的函数，无参数

```
function welcome(){
  alert("Welcome to JavaScript World");
}
```

```
function welcome(msg){
  alert(msg);
}
```

当弹出的对话框的显示内容根据需要发生改变时，可以通过形参msg的传递，动态改变。

1. 自定义函数

- 函数可以通过使用函数名称的方法进行调用。例如：

```
welcome();
```

- 如果该函数存在参数，则调用时必须在函数的小括号内传递对应的参数值（实参）。

```
welcome("Hello JavaScript!");
```

- 函数可以在JavaScript代码的**任意位置**进行调用，也可以在指定的**事件发生**时调用。例如在按钮的点击事件中调用函数：

```
<button onclick="welcome()">点击此处调用函数</button>
```

1. 自定义函数

关于形参和实参的理解注意：

- 在未调用函数时，形参不占用存储单元，只在调用时为形参分配空间，调用结束后自动释放。
- 形参必须是声明的变量，实参可以是常量、变量或表达式。
- 在函数调用中，实参列表中参数的数量、类型、和顺序与形参列表中的参数可以不匹配。若形参个数大于实参，多出来的形参值为undifined；反之，多出来的实参将被忽略。
- 实参对形参的数据传递是单向传递，即实参→形参

声明函数的应用案例

[点击查看案例]

```
<!doctype html>
 <html><head>
  <meta charset="utf-8">
  <title>计算三次方函数</title>
  <script>
   function cubic(x){
          var y;                          //声明变量y，存储计算结果
          y = x*x*x;                      //计算x的三次方
          alert("计算结果 "+y);           //输出运算结果
   }
  </script></h
 <body>
   <input type="button" value="计 算" onClick="cubic(prompt('请输
入一个数值：'))">
 </body>
</html>
```

Prompt是系统内置的一个调入对话框的方法：弹出一个包含“确定”、“取消”和“文本框”的对话框，可接收用户输入信息，单击确定返回输入内容，单击取消，返回null。参看p227

1. 自定义函数

匿名函数

匿名函数定义格式

不指定函数名，只需要使用关键字和可选参数。

```
<script type="text/javascript">
        function ([参数1,参数2,.....]) {
            函数体
            [return 表达式]
        } ;
</script>
```

没有函数名，就需要通过变量来接收，以方便调用。

1. 自定义函数

[点击查看案例]

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>匿名函数赋值给变量</title>
<script>
var sayHi=function(toWhom)
{
        alert("Hi"+toWhom);
}
sayHi("World! ");
</script>
</head>
<body></body></html>
```

2. 系统内部函数

系统内部函数也叫系统内置函数，或全局函数。这种函数可在JavaScript程序中直接调用而无需定义。

□ eval（）函数

功能：将eval中的字符串参数作为脚本代码来执行。

代码格式

```
eval(字符串)
```

Eval( )的应用案例

[点击查看案例]

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>eval的应用</title>
<script>
    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>
</head>
<body></body></html>
```

2. 系统内部函数

□ isFinite（）函数

功能：检查某个值是否为有穷大的数。

代码格式

```
isFinite(数字)
```

- **参数为有限值，返回值为ture。**
- **参数为非数字或者正、负无穷大，返回值为false。**

isFinite( )的应用案例

[点击查看案例]

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>isFinite的应用</title>
 <script>
   document.write("123是无穷大吗:"+isFinite(123)+"<br/>");
   document.write("函数是无穷大吗:"+isFinite("函数"));
 </script>
</head>
<body></body>
</html>
```

2. 系统内部函数

□ isNaN（）函数

功能：检查某个值是否为数字。

代码格式

```
isNaN(任意参数)
```

- **参数为数字，返回值为false 。**
- **参数为非数字，返回值为ture 。**

isNaN( )的应用案例

[点击查看案例]

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>isNaN的应用</title>
 <script>
   document.write("123是数字吗:"+isNaN(123)+"<br/>");
   document.write("函数是数字吗:"+isNaN("函数"));
 </script>
</head>
<body></body>
</html>
```

2. 系统内部函数

□ parseInt（）函数

功能：解析一个字符串，并返回一个整数。

代码格式

```
parseInt(任意参数)
```

- **参数第一个字符为数字字符串，返回值为整数。**
- **参数为非数字，返回值为NaN。**

parseInt( )的应用案例

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>parseInt的应用</title>
 <script>
   document.write(parseInt("123")+"<br/>");
   document.write(parseInt("12.35")+"<br/>");
   document.write(parseInt("12 34")+"<br/>");
   document.write(parseInt("4M6")+"<br/>");
   document.write(parseInt("a2")+"<br/>");
   document.write(parseInt("05")+"<br/>");
   document.write(parseInt("10",16)+"<br/>");
 </script>
</head>
<body></body>
</html>
```

[点击查看案例]

第2个参数16代表被解析的数字基数为16。

2. 系统内部函数

□ parseFloat（）函数

功能：解析一个字符串，并返回一个浮点数。

代码格式

```
parseFloat(任意参数)
```

- **参数第一个字符为数字字符串，返回值为浮点数 。**
- **参数为非数字，返回值为NaN 。**

parseFloat( )的应用案例

[点击查看案例]

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>parseInt的应用</title>
 <script>
   document.write(parseFloat("123")+"<br/>");
   document.write(parseFloat ("12.35")+"<br/>");
   document.write(parseFloat ("12 34")+"<br/>");
   document.write(parseFloat ("4M6")+"<br/>");
   document.write(parseFloat ("a2")+"<br/>");
   document.write(parseFloat ("05")+"<br/>");
   document.write(parseFloat ("years 20")+"<br/>");
 </script>
</head>
<body></body>
</html>
```

2. 系统内部函数

- 其他内置函数

函数	描 述
Number(参数)	把参数转换为数字。
encodeURI(URI)	把字符串编码为 URI。
decodeURI(URI)	解码某个编码的 URI。
encodeURIComponent(URI组件)	把字符串编码为 URI 组件。
decodeURIComponent(URI组件)	解码一个编码的 URI 组件。
escape(字符串)	对字符串进行编码。
unescape(字符串)	对由escape( )编码的字符串进行解码。

2. 系统内部函数

- 其他内置函数

函数	描 述
Number(参数)	把参数转换为数字。
encodeURI(URI)	把字符串编码为 URI。
decodeURI(URI)	解码某个编码的 URI。
encodeURIComponent(URI组件)	把字符串编码为 URI 组件。
decodeURIComponent(URI组件)	解码一个编码的 URI 组件。
escape(字符串)	对字符串进行编码。
unescape(字符串)	对由escape( )编码的字符串进行解码。

本章结束