第 4 章

对网页添加交互性

网页可能要求用户作出输入并生成输出。例如，您可能需要创建购物网站网页以使用户能够从列表中选择要购买的产品并在文本框中输入数量。当用户指定这些值时，应该在另一文本框中生成应付总金额。此类功能不能由 HTML 独立执行。因此，需要脚本语言以向网页添加此类功能。

本章讨论脚本语言的基础知识。并解释在网页中实现 JavaScript 的过程。.此外，还讨论表达式和控制结构的用法。而且还解释函数的必要性和用法。

目标

在本章中，您将学习：

- 了解脚本
- 在网页中实现 JavaScript
- 使用变量、运算符和控制结构
- 实现函数


NIIT
training
china
.com

了解脚本

思考以下场景：您需要为网站 OnlineShop.com 创建网页。该网站使用户能够通过在线注册表单进行注册。注册时将要求用户选择电子邮件和邮件这两个选项之一作为通信模式。选择电子邮件选项时，将向用户提供文本字段以输入电子邮件地址。然而，如果用户选择邮件选项，将提供文本区域以输入邮寄地址。将按照指定地址向用户提供其交易报告、确认收据或促销邮件。此类网页可使用脚本语言开发。

注释

文本框、单选按钮和文本区域之类的控件将在下一章进行讨论。

脚本的类型

要创建动态交互网页，需要在网页中加入一个称为脚本的代码块。脚本可由 Web 浏览器或 Web 服务器执行。

当用户通过 Web 浏览器请求网页时，请求被发送到万维网 (WWW) 其他位置的计算机。运行浏览器的计算机称为客户机，接收请求的计算机称为 Web 服务器。

当 Web 服务器接收请求时，它处理请求并将请求的网页发送给客户机，然后网页显示在客户机的浏览器窗口中。

思考著名的科学半月刊网站的示例。它使客户能够在线预订该刊物。预订时，用户需要以永久地址和通信地址的形式提供联系详细信息。用户的通信地址可以与永久地址相同。填写永久地址详细信息后，如果用户选择标题为“与通信地址相同”复选框，那么“通信地址”字段中应填入同一地址。此功能可通过在客户端自身进行解释的脚本来实现。

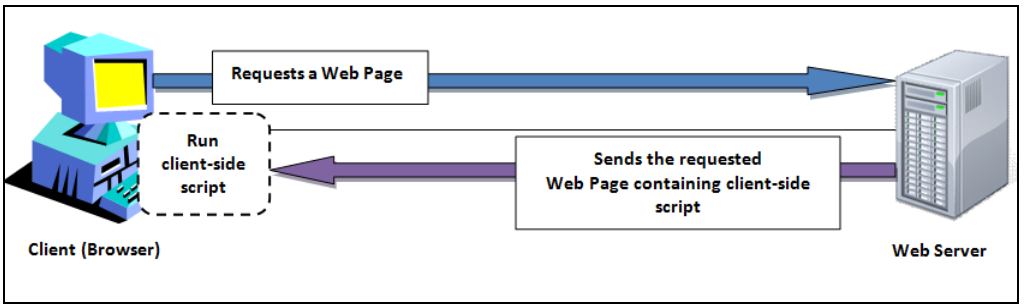
脚本可以有两种类型。具体如下：

- 客户端脚本
- 服务器端脚本

客户端脚本

客户端脚本是指运行在用户计算机上的 Web 浏览器在客户端执行的脚本。用于创建客户端脚本的语言包括客户端 JavaScript (CSJS) 和 Visual Basic Script (VBScript)。当 Web 浏览器请求网页时，服务器通过网络发送请求的网页，其中包含 HTML 语句和脚本语句。Web 浏览器读取网页，并显示通过解释 HTML 语句而生成的结果。并且，Web 浏览器在显示网页时遇到脚本语句时将执行这些语句。

客户端脚本情况下客户机和服务器之间的通信如下图所示。



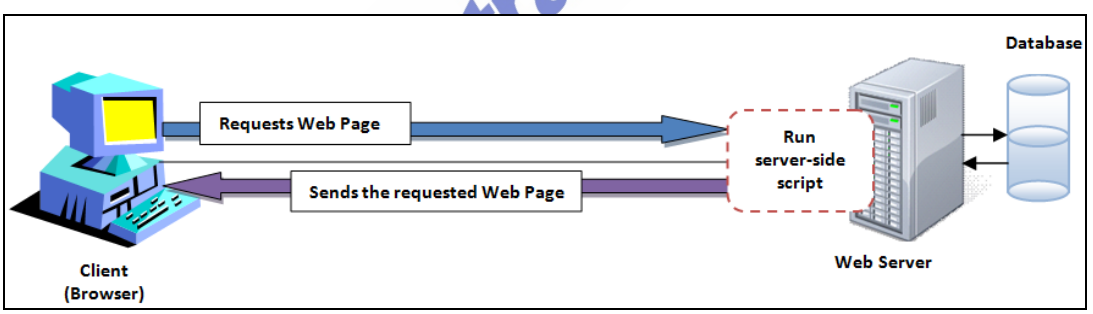
客户端脚本情况下的通信

服务器端脚本

服务器端脚本是指 Web 服务器根据用户请求执行的脚本。用于创建服务器端脚本的语言包括服务器端 JavaScript (SSJS)、Perl、PHP 和 Visual Basic Script (VBScript)。

服务器端脚本是在 Web 服务器上执行的。在此情况下，需要从客户端 Web 浏览器上收集信息并传递到 Web 服务器上执行的程序或脚本。Web 服务器上执行的脚本执行某些任务，例如建立数据库连接和验证客户机 Web 浏览器发送的数据。

服务器端脚本实现数据库交互，并且可用于处理客户端查询和将客户端数据存储在数据库中。此功能使用户能够与应用程序或服务器的其他用户共享信息并进行访问。 服务器端脚本情况下客户机和服务器之间的通信如下图所示。



服务器端脚本情况下的通信

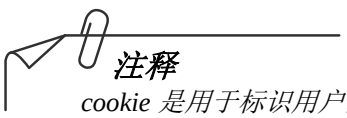
注释

JavaScript 和 VBScript 可用于客户端脚本和服务器端脚本。

了解 JavaScript 的好处

JavaScript 提供以下好处：

- **处理事件：**可用于在触发事件时执行函数。例如，当用户将鼠标滚动到任何图像上时，它的背景将发生变化。
- **收集浏览器信息：**JavaScript 可用于收集浏览器信息，例如浏览器名称和版本。服务器可使用此信息响应客户端请求。
- **操作 Cookie：**JavaScript 可用于以 cookie 的形式访问和存储用户信息，例如客户机的客户端首选项和验证信息。



注释

cookie 是用于标识用户的一段数据。它存储在用户的 Web 浏览器中，在用户浏览网站时从 Web 服务器进行发送。

在网页中实现 JavaScript

思考 BookYourHotel 网站的场景。预订酒店时，将要求客户对某些选项（例如冒险运动、电影、SPA 和观光）单击单选按钮来指定他们感兴趣的活动。客户可免费获得所选的设施。如果客户选择其中任何单选按钮，关于该设施的详细信息应该作为列表打开。例如，如果客户选择冒险运动单选按钮，那么诸如滑翔伞、蹦极或漂流的此类运动应该显示为复选框列表，客户可在其中选择想要的活动。此类功能可使用 JavaScript 实现。

JavaScript 是最受欢迎的脚本语言之一。它可用于向网页提供功能，例如在用户选择列表框中的选项时填充文本框。这些功能可由用户操作触发。

脚本可直接嵌入到网页中，方法是在 <SCRIPT> 标记中编写 JavaScript 代码或者在外部 JavaScript (.js) 文件中编写整个 JavaScript 代码。对 JavaScript 代码使用外部文件时，需要在网页上引用该文件。

将脚本嵌入到网页

可使用 <SCRIPT> 标记将 JavaScript 代码嵌入到网页。以下语法用于将脚本嵌入到网页：

```
<SCRIPT type="text/javascript"> JavaScript statements
</SCRIPT>
```

在以上语法中，<SCRIPT> 和 </SCRIPT> 标记分别指示脚本的开始和结束。<SCRIPT> 标记的 type 属性指示脚本语言的类型。<SCRIPT> 和 </SCRIPT> 标记之间的一条或多条 JavaScript 语句构成脚本块。脚本块是由浏览器的内置 JavaScript 解释器执行的。

<SCRIPT> 标记可插入到网页的主体部分和/或头部分。如果脚本是为了响应用户所执行操作而执行，那么它通常放在头部分。这有助于将所有脚本放在一个地方而不干扰页面内容。然而，如果脚本需要在加载页面时执行，那么将它放在网页的主体部分。

思考以下代码：

```
<!DOCTYPE HTML><HTML>
<BODY>
<SCRIPT type="text/javascript">
alert('Welcome to JavaScript');
</SCRIPT>
</BODY>
</HTML>
```

上述代码说明了 <SCRIPT> 标记在网页主体部分中的用法。它将在加载网页时在消息框中显示消息 **Welcome to JavaScript**。

创建和使用外部文件

在 HTML 页面中嵌入脚本时，将很难同时管理 HTML 代码和 JavaScript 代码。为了避免此问题，可在外部文件中编写 JavaScript 代码并在 HTML 文件中引用它。外部 JavaScript 文件是以 .js 扩展名保存的。以下语法用于引用外部 JavaScript 文件：

```
<SCRIPT type="text/javascript" src="URL">
```

在以上语法中，URL 引用外部 JavaScript 文件的路径。

思考 ShopForYou.com 网站的示例。当用户访问网站主页时，需要在消息框中显示可供销售的产品的列表。可在外部 JavaScript 文件中编写显示产品列表的代码，并在网站主页中添加该文件的引用。

要创建和使用外部 JavaScript 文件，需要执行以下步骤：

1. 打开 Notepad 并编写以下代码：

```
alert( " PRODUCTS ON SALE :\n" +" 1.LEO Mobile \n" +" 2.LEO Camera\n" + "
3.RED shoes \n"+" 4.KP Watch \n");
```

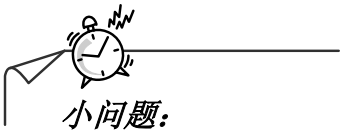
2. 将文件另存为 **sale.js**。
3. 创建希望引用外部 JavaScript 文件的网站主页。为此，在 Notepad 文件中编写以下代码：

```
<!DOCTYPE HTML><HTML>
<BODY>
<H1>Buy Products</H1>
</BODY>
</HTML>
```

4. 将文件另存为 **home.html**。
5. 在 **home.html** 文件中添加突出显示的代码段，如以下代码所示：

```
<!DOCTYPE HTML><HTML>
<BODY>
<H1>Buy Products </H1>
<SCRIPT type="text/javascript" src="sale.js">
</SCRIPT>
</BODY>
</HTML>
```

`<SCRIPT>` 标记的 `src` 属性用于指定外部 JavaScript 文件 **sale.js** 的路径。要在 `srctype` 属性中指定的路径可以是绝对路径或相对路径。



小问题：

以下哪个属性用于指定外部 JavaScript 文件的路径？

1. `src`
2. `type`
3. `url`
4. `text`

答案：

1. `src`

注释

绝对路径是文件的完整地址。例如，如果文件 sale.js 存储在 D: 驱动器上，它的绝对路径是 D:\sale.js。相对路径是文件相对于当前工作目录的路径。例如，如果包含外部 JavaScript 文件引用的 HTML 文件存储在 sale.js 文件所在目录中，那么文件的相对路径是 sale.js。

了解 JavaScript 中使用的规则和约定

每个编程/脚本语言都有其自己的一组规则和约定。JavaScript 的一些规则和约定是：

- **分号：** JavaScript 不强制要求用分号来指示语句结束。但是，在脚本语句后插入分号是一中较好的做法，因为它提高了代码的可读性。例如，在同一行中编写两条语句时，可使用分号分隔它们。以下代码段演示分号的用法：

```
alert ("Product on sale is:"); alert("1.LEO Mobile");
```

- **引号：** JavaScript 允许您使用双引号 (" ") 或单引号 (' ') 将字符串括起。以下代码段演示双引号的用法：

```
alert(" PRODUCTS ON SALE :\n" +" 1.LEO Mobile \n" +" 2.LEO Camera\n" + "3.RED
shoes \n"+" 4.KP Watch \n");
```

以下代码段演示单引号的用法：

```
alert(' PRODUCTS ON SALE :\n' +' 1.LEO Mobile \n' +' 2.LEO Camera\n' + '
3.RED shoes \n'+' 4.KP Watch \n');
```

- **是否区分大小写：**JavaScript 是区分大小写的脚本语言。这表示语句 a=b 和 A=B 在 JavaScript 中的处理方式将不同。
- **注释：**注释是不会由解释器执行但是用于提高代码可读性和可理解性的语句。JavaScript 允许使用单行和多行注释。单行注释通过 // 指示，多行注释通过符号 /* 和 */ 指示。以下代码段演示单行注释的用法：

  ```
  //Comments are necessary for readability
  ```

  以下代码段演示多行注释的用法：

  ```
  /* Comments are necessary for readability */
  ```

活动 4.1：了解脚本

使用变量、运算符和控制结构

思考以下场景：您需要为 ShopForYou.com 创建网页以允许用户指定是以价格的升序还是降序显示产品列表。

要在网页上实现以上功能，需要比较这些产品的价格并且按用户要求以升序或降序显示它们。要执行此类任务，需要编写脚本以使用 JavaScript 中提供的变量、运算符、条件构造和循环构造。

定义变量

思考以下情境：您创建了网页以接受用户希望购买的产品的数量和单价并显示产品的总价。读取用户提供的数量时，需要存储此值以便可用于计算总价。此外，还需要存储产品的价格。

变量是内存中用于存储值的命名位置。在以上示例中，如果用户提供的需要购买的产品数量是 5，产品的价格是 $250，那么您需要两个变量，一个存储数量，另一个存储价格。以下代码段显示变量的赋值：

```
quantity = 5
price = 250
```

声明变量

在程序中使用变量前，应该先声明变量。JavaScript 允许您使用 var 关键字声明变量。声明变量的语法是：

```
var var_name;
```

在以上语法中，`var_name` 指的是变量名称，变量是使用 `var` 关键字声明的。

以下代码段声明变量 `employeeName`：

```
var employeeName;
```

JavaScript 是松散类型的语言，允许您在初始化变量时不指定数据类型。Numeric、String 和 Boolean 是 JavaScript 中常用的数据类型。

注释

变量分为两类：局部和全局。局部变量是在代码块中声明的，例如在函数或循环构造的主体内。这些变量只能在特定块内访问。全局变量是在任何代码块外（但是在脚本内）声明的。它们可以在脚本内的任何位置访问。

向变量赋值

可通过以下方式向变量赋值：

- 声明后向变量赋值
- 在声明时初始化变量

- 初始化变量时不显式声明

声明后向变量赋值

思考在声明后初始化变量的以下代码段：

```
var employeeName;
employeeName="Peter";
```

以上代码段声明变量 employeeName，然后向它赋值 Peter。

在声明时初始化变量

思考在声明时初始化变量的以下代码段：

```
var employeeName="Peter";
```

以上代码段声明变量 employeeName，然后向它赋值 Peter。

初始化变量时不显式声明

JavaScript 向您提供了使用变量时不进行声明的灵活性。如果在脚本中使用变量时不进行显式声明，那么将在首次使用它时自动声明。然而，在脚本中使用变量前进行声明是好的做法，因为它提高了程序的可读性。

思考初始化变量时不进行声明的以下代码段：：

```
employeeName="Peter";
```

在以上代码段中，向变量 employeeName 赋值但是未显式声明。因此，语句 employeeName="Peter" 隐式声明变量 employeeName 并向它赋值 Peter。

注释

数组是用于存储以确定顺序排列的多个值的特殊类型变量。这些值存储在数组内编入了索引的位置中。使用数组前，需要进行声明。以下语法用于声明数组：

```
var category = new Array(3)
```

在以上语法中，创建了大小为 3 的数组 category。

思考在每个编了索引的位置赋值的以下代码段：

```
category[0] = "Soaps"
category[1] = "Oils"
category[2] = "Food Products"
```

存储在数组中的值可通过存储了它们的索引位置进行访问。以下语法用于访问数组中的第一个元素：

```
category[0];
```

使用运算符

运算符是用于计算或比较的包含一个或多个字符的字符组。运算符可用于修改变量中存储的值。运算符作用于的这些值或变量称为操作数。操作数可以是字面值或变量。字面值是任何数据类型（例如数字、字符串或布尔值）的常量值。

以下代码演示运算符在脚本中的用法：

```
<!DOCTYPE HTML><HTML>
<BODY>
<SCRIPT type="text/javascript">
var employeeName;
var basicSalary;
var othersAllowance; var TotalSalary; employeeName="Peter"; basicSalary=20000;
othersAllowance=1000;
TotalSalary=basicSalary+othersAllowance;
</SCRIPT>
</BODY>
</HTML>
```

在以上代码中，使用的操作数是变量 `employeeName`、`basicSalary`、`TotalSalary` 和 `othersAllowance` 以及字符串字面值 `Peter` 和数字字面值 `20000` 和 `1000`。`+` 运算符对包含在 `basicSalary` 和 `othersAllowance` 中的值进行加法运算，并且将值赋给变量 `TotalSalary`。

了解运算符类别

可在 JavaScript 中使用以下类别的运算符：

- 算术运算符
- 赋值运算符
- 算术赋值运算符
- 比较运算符
- 逻辑运算符

算术运算符

算术运算符用于对变量和字面值执行算术运算。下表描述常用的算术运算符。

运算符	*描述*	*示例*
+	*用于将两个数字相加。*	*X=Y+Z;* *如果 Y 等于 20，Z 等于 2，X 的值将是 22。*

-	*用于将两个数字相减。*	X=Y-Z; *如果 Y 等于 20，Z 等于 2，X 的值将是 18。*
*	*用于将两个数字相乘。*	X=Y*Z; *如果 Y 等于 20，Z 等于 2，X 的值将是 40。*
/	*用于将一个数字除以另一个数字。返回除法的商。*	X=Y/Z; *如果 Y 等于 21，Z 等于 2，X 的值将是 10.5。.*
%	*用于将两个数字相除并返回余数。该运算符称为取模运算符。*	X=Y%Z; *如果 Y 等于 21，Z 等于 2，X 的值将是 1。*

算术运算符

赋值运算符

赋值运算符 (=) 用于将值或者表达式的结果赋给变量。例如，表达式 x=5 将值 5 存储在变量 x 中。

算术赋值运算符

算术赋值运算符用于执行算术运算并将值赋给运算符左边的变量。

下表描述常用的算术赋值运算符及其用法。

运算符	***用法***	***描述***
+=	X+=Y;	*相同于：* X = X + Y;
-=	X-=Y;	*相同于：：* X = X - Y;
*=	X*=Y;	*相同于：* X = X * Y;

/=	X/=Y;	相同于: X = X / Y;
%=	X%=Y;	相同于: X = X % Y;

算术赋值运算符

比较运算符

比较运算符用于比较两个值并根据比较结果执行操作。使用比较运算符时，生成的表达式将保存布尔值。

下表描述比较运算符的用法。

运算符	***用法***	***描述***	***示例*** ***（假设：X 值为 20，Y 值为25）***
<	*expression1 < expression2*	*用于检查 expression1 是否小于 expression2。*	*var Result;* *Result = X < Y;* *Result 的值是 true*
>	*expression1 > expression2*	*用于检查 expression1 是否大于 expression2。*	*var Result;* *Result = X > Y;* Result 的值是 false
<=	*expression1 <= expression2*	*用于检查 expression1 是否小于等于 expression2。*	*var Result;* *Result = X <= Y;* *Result 的值是* true
>=	*expression1 >= expression2*	*用于检查 expression1 是否大于等于 expression2。*	*var Result;* *Result = X >= Y;* *Result 的值是 false*

运算符	用法	描述	示例 （假设：X 值为 20，Y 值为25）
==	expression1 == expression2	用于检查 expression1 是否等于 expression2。	var Result; Result = X == Y; Result 的值是 false
!=	expression1 != expression2	用于检查 expression1 是否不等于 expression2。	var Result; Result = X != Y; Result 的值是 true
===	expression1===expression2	用于检查 expression1 是否等于 expression2 并且类型相同。	var Result; Result = X === Y; Result 的值是 false

比较运算符

逻辑运算符

逻辑运算符用于对复杂表达式求值，在复杂表达式中需要对一个或多个表达式求值以求出结果。它们返回布尔值。

下表描述逻辑运算符的用法。

运算符	用法	描述	示例 （假设：X 值是 20，Y 值是 25）
&&	expression1 && expression2	如果 expression1 和 expression2 都为 true，则返回 true。	(X >10 && Y> 20) 返回 true
!	! expression	如果 expression 为 false，则返回 true。	!(X == Y) 返回 true
||	expression1 ||	如果 expression1 和	(X ==5 || Y==5)

	expression2	*/或 expression2 为 true，则返回 true。.*	*返回 false*

逻辑运算符

使用条件构造

思考以下情境：您需要在脚本中根据某个条件执行特定语句。例如，您希望检查给定的数字是奇数还是偶数并相应显示消息。为此，需要在脚本代码中执行决策，并根据所作的决策执行一组不同的语句。

可使用条件结构来实现。这些构造允许根据所评估表达式的结果执行选择性语句或语句块。JavaScript 中的两个条件构造是：

- `if...else` 构造
- `switch...case` 构造

if...else 构造

`if...else` 条件构造中的 `if` 语句后跟括在圆括号中的逻辑表达式。将对此条件进行评估，并根据此结果作出决策。以下语句描述 `if...else` 构造的语法：

```
if (exp)
{
// Statements;
}
else
{
// Statements;
}
```

在以上语法中，对表达式 `exp` 求值。如果结果为 true，将执行 `if` 构造内的语句。如果结果为 false，将执行 `else` 构造内的语句。

思考一个游戏的示例：其中您需要验证玩家的年龄。如果年龄大于 12，允许玩家玩游戏。否则，需要显示相应的消息。以下代码段显示 `if...else` 构造的用法：

```
var age=5;
if (age<12)
{
alert('Sorry! This game is for children above 12 Years');
}
else
{
alert('Play the Game');
}
```

以上代码段检查玩家的年龄是否小于 12。`if` 语句中的条件检查玩家的年龄。如果条件评估为 true，将显示消息 **Sorry! This game is for children above 12 Years**。如果条件评估为 false，将显示消息 **Play the Game**。

switch...case 构造

JavaScript 中提供的另一条件构造是 switch...case 构造。它用于需要评估变量的多个值的情况。

以下代码描述 switch...case 构造的语法：

```
switch(VariableName)
{
case ConstantExpression_1:
// statements;
break;
case ConstantExpression_2:
// statements;
break;
case ConstantExpression_n:
// statements;
break;
default:
// statements;
break;
}
```

执行 switch 语句后，将对作为参数传递给 switch 语句的变量进行评估，并且分别与每个 case 语句指定的每个常量表达式进行比较。如果其中一个常量表达式等于 switch 语句中给出的变量的值，那么控制将传递给所匹配 case 语句后面的语句。break 语句用于退出 switch 语句。它通过结束 switch...case 构造的执行，防止执行其余 case 构造。如果没有任何 case 是匹配的，那么将执行 default 语句下的语句。

思考以下代码：您希望根据变量值显示星期几的名称：

```
<!DOCTYPE HTML><HTML>
<BODY>
<SCRIPT type="text/javascript">
var day="5";
switch(day)
{
case "1":
alert("Day is Monday");
break;
case "2":alert("Day is Tuesday");
break;
case "3":
alert("Day is Wednesday");
break;
case "4":
alert("Day is Thursday");
break;
case "5":
alert("Day is Friday");
break;
case "6":
alert("Day is Saturday");
break;
case "7":
```

```
alert("Day is Sunday");
break;
default:
alert("Not a valid number");
break;
}
</SCRIPT>
</BODY>
</HTML>
```

在以上代码中，switch 语句用于通过 case 常量评估和比较变量 day 的值，并且显示星期几的名称。

使用循环构造

循环构造用于反复执行一行或多行代码。在 JavaScript 中，可使用以下循环结构：

- while 循环
- do...while 循环
- for 循环

while 循环

while 循环用于在条件评估为 true 前反复执行语句块。while 语句在执行循环中的语句前总是检查条件。while 循环构造的语法是：

```
while (expression)
{
statements;
}
```

在以上语法中，对表达式进行评估。如果结果为 true，将执行循环主体内的语句。一旦执行了块中的所有语句，控制将传回循环，并且将重新评估表达式。循环将在表达式评估为 false 时退出。

以下代码演示 while 循环的用法：

```
<!DOCTYPE HTML><HTML>
<BODY>
<SCRIPT type="text/javascript">
var num=0;
while(num<20)
{ num=num+1; alert(num);
}
</SCRIPT>
</BODY>
</HTML>
```

以上代码定义变量 num 并用值 0 初始化。while 语句检查 num 的值是否小于 20。如果条件评估为 true，将执行 while 循环内的语句。此过程将继续，直到 var 的值小于 20。

do...while 循环

do...while 循环构造类似于 while 循环构造。然而，do...while 循环内的语句至少执行一次，对比之下 while 循环块中的语句在条件评估为 false 时将不执行。此外，do...while 循环内的语句将在检查条件前执行，对比之下 while 循环块中的语句将在检查条件之后执行。以下语法用于声明 do...while 循环：

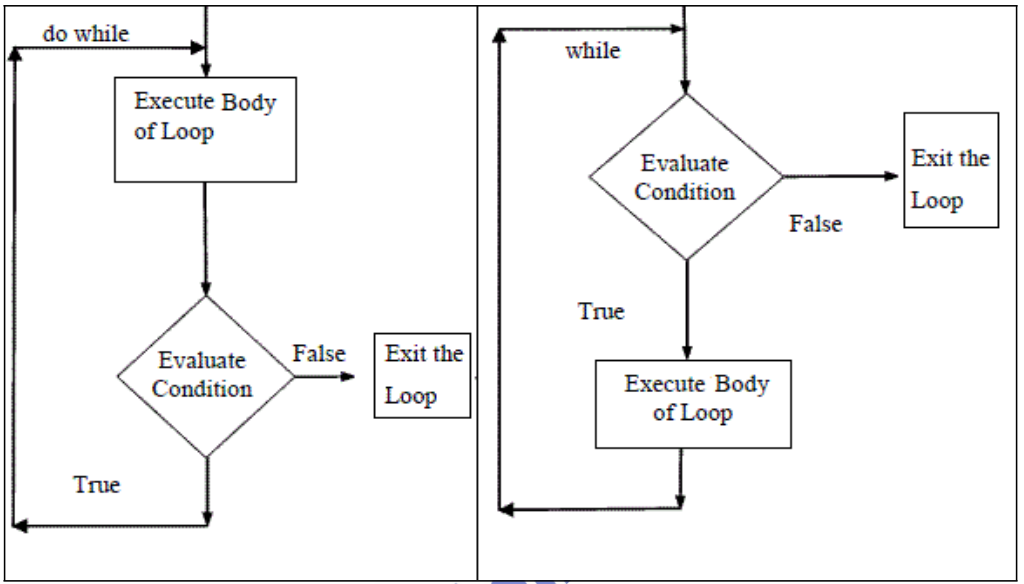
```
do
{ Statements;
}
while(condition)
```

思考演示 do...while 循环用法的以下代码：

```
<!DOCTYPE HTML><HTML>
<BODY>
<SCRIPT type="text/javascript">
var num=0;
do
{
num=num+1;
alert(num);

}
while(num<20)
</SCRIPT>
</BODY>
</HTML>
```

下图显示了 do...while 循环构造和 while 循环构造在功能上的区别。



do…while 和 while 循环的功能

for 循环

for 循环允许根据测试条件的评估结果执行代码块。以下语法用于声明 for 循环：

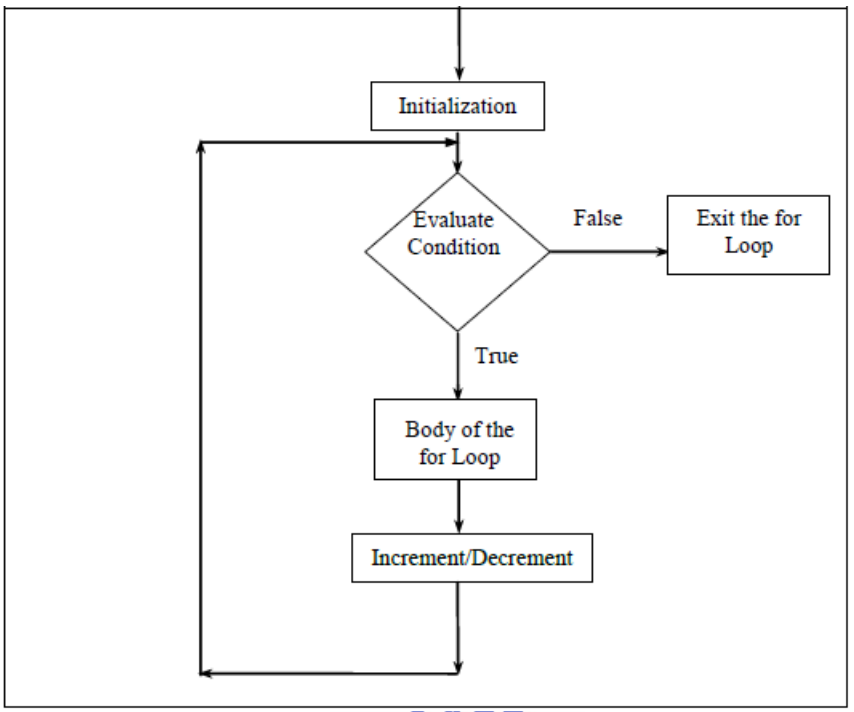
```
for(initialize variable; test condition; step value)
{
// code block
}
```

在以上语法中：

- initialize variable： 用值初始化循环变量。
- test condition： 指定执行代码块语句前要检查的条件。
- step value：指示每次迭代要执行的增量或减量。

for 循环执行开始时将初始化循环变量。然后它将评估测试条件。如果测试条件评估为 true，将执行循环主体中的代码块。如果评估为 false，将退出 for 循环。一旦执行了代码块中的所有语句，将按步长值所指定来增大或减小变量。迭代了值以后，将重新评估测试条件。在测试条件评估为 true 之前此过程将一直继续。

下图显示 for 循环的执行顺序。



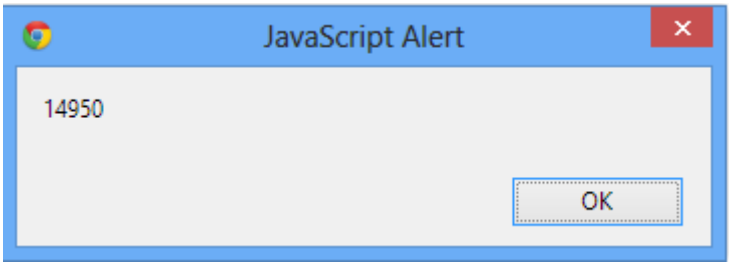
for 循环的执行顺序

以下代码演示 for 循环的用法：

```
<!DOCTYPE HTML><HTML>
<BODY>
<SCRIPT type="text/javascript">
var num;
var sum=0;
for(num=100;num<200;num=num+1)
{
sum=sum+num;
}
alert(sum);
</SCRIPT>
</BODY>
</HTML>
```

以上代码创建变量 num 和 sum。在 for 循环中，向 num 变量分配值 100。for 循环内的条件检查 num 的值是否小于 200。如果条件评估为 true，将执行 for 循环内的语句。此过程将在 num 变量的值小于 200 之前一直继续。

将显示以上代码的输出，如下图所示。



代码输出

注释

`while` 循环的工作原理类似于 `for` 循环。主要区别在于 `while` 循环变量是代码块内递增，而 `for` 循环变量是在执行代码块所有语句后递增。

Break 和 Continue 语句

在某些情况下，可能需要在迭代后检查循环条件前退出循环。`break` 语句用于退出循环。它防止执行循环的其余语句。`break` 语句通常放在循环的 `if` 构造内。`continue` 语句用于跳过所有后续指令，并使控制转回循环开头。

以下语法用于显示 `break` 语句在 `while` 循环内的用法：

```
while(test condition)
{
//Statement1;
if (test condition1)
{
break;

}
//Statement2;
//Statement3;
}
```

在以上语法中，如果 `test condition` 为 true，控制将进入循环主体。它执行 `statement1`。接着，它检查 test condition1。如果它评估为 true，控制将转到循环外。因此，不会执行 `Statement2` 和 `Statement3`。如果 `test condition1` 评估为 false，循环将继续正常的执行过程。因此，将执行 `Statement2` 和 `Statement3`。

思考演示 `break` 和 `continue` 语句用法的以下代码：

```
<!DOCTYPE HTML><HTML>
<HEAD></HEAD>
<BODY>
<SCRIPT type="text/javascript">
var iNum = 0;
```

```
for (var i=1; i < 10; i++)
{
if (i % 3 == 0)
{
break;
}
iNum++;
}
alert(iNum);
</SCRIPT>
</BODY>
</HTML>
```

以上代码创建变量 iNum 并向它赋值 0。for 循环从 1 到 10 迭代变量 i。for 循环中的 if 语句验证 i 值是否能够被 3 整除。如果可被 3 整除，将执行 break 语句，并且控制离开 for 循环。以上代码执行后将显示值 2。

然而，如果以上代码执行时将 break 替换为 continue 语句，将显示不同输出。例如，思考以下代码：

```
<!DOCTYPE HTML><HTML>
<HEAD></HEAD>
<BODY>
<SCRIPT type="text/javascript">
var iNum = 0;
for (var i=1; i < 10; i++) {
if (i % 3 == 0) {
continue;
}
iNum++;
}
alert(iNum);
</SCRIPT>
</BODY>
</HTML>
```

在以上代码中，for 循环的 if 语句验证 i 值是否可被 3 整除。如果 i 可被 3 整除，将执行 continue 语句，并且控制将回到 for 循环的开头。以上代码执行后，将显示值 6。

实现函数

思考 ShopForYou.com 网站。该公司必须计算客户购买的产品上所征收的税。税额因产品而异。例如，某些产品（例如药品）的赋税是 0%，而移动产品赋税则是 12.5%。需要在页面脚本中的各个位置使用计算各类产品赋税的代码。如果代码在所有这些位置重复，代码中所需的更改将需要在多个位置执行。例如，如果希望将税率从 12.5% 更改为 10%，需要在使用此代码的所有位置进行此更改。此外，如果任何一条重用的代码包含错误，需要在多个位置更正错误。

为了克服此问题，引入了函数。可编写需要在函数内重用的代码。现在，您只需调用包含所需代码的函数，而无需重复使用代码。这使您可以只编写一次代码并在需要时使用。因此，函数可用于通过实现可重用性的概念优化性能。

介绍函数

函数是具有名称的自包含语句块。如果需要在脚本不同位置重复执行相同代码，可执行函数。

函数是作为附加了名称的独立模块创建的。每个函数中可以包含若干语句。使用函数时，可方便地检测到脚本中的错误，方法是将每个函数执行的期望结果与该函数执行的实际结果进行比较。如果结果不同，则表示代码有错误。因此更容易调试代码。

换句话说，函数使代码模块化、简单、可重用。在 JavaScript 中，函数具有以下类型：

- 内置函数
- 用户定义函数

内置函数

*内置函数*是现成可用的，因为它们的代码已经写好。JavaScript 支持的一些内置函数包括：

- `isNaN()`
- `parseInt()`
- `parseFloat()`
- `eval()`
- `prompt()`
- `confirm()`

isNaN()

`isNaN()` 函数确定参数是否非数字。如果参数非数字，该函数返回 true。

以下代码演示 `isNaN()` 函数的用法：

```
<!DOCTYPE HTML><HTML>
<BODY>
<SCRIPT type="text/javascript">
num1="Two thousand"; num2=8000; Result=isNaN(num1); alert(Result);
Res=isNaN(num2); alert(Res);
</SCRIPT>
</BODY>
```

```
</HTML>
```

以上代码创建变量 num1 并向它赋值 Two thousand。它还创建变量 num2 并向它赋值 8000。isNaN() 函数评估变量 num1 和 num2 是否包含数值数据。由于 num1 变量中的值是字符串，isNaN(num1) 函数将返回 true。然而，num2 中的值是数值。因此，isNaN(num2) 函数将返回 false。

注释

NaN 表示 Not a Number（不是数字）。

parseInt()

parseInt() 函数解析字符串参数并返回相应整数。

思考以下代码段：

```
x="5";
y=parseInt(x);
```

在上述代码段中，变量 x 存储字符串 5。 parseInt() 函数取参数 x，对它进行解析并向变量 y 赋整数值 5。

parseFloat()

parseFloat() 函数取字符串参数并返回浮点数字。思考以下代码段：

```
x="6.2";
y=parseFloat(x);
```

在上述代码段中，变量 x 存储字符串 6.2。parseFloat() 函数取参数 x，对它进行解析并向变量 y 赋浮点数 6.2。

eval()

eval() 函数用于评估或者执行参数。如果参数是表达式，将对表达式进行评估。然而，如果参数是 JavaScript 语句，将执行语句。

思考以下代码：

```
<!DOCTYPE HTML><HTML>
<BODY>
<SCRIPT type="text/javascript">
var res1=0; res1=eval(5+10); alert(res1);
</SCRIPT>
</BODY>
</HTML>
```

以上代码对表达式 5+10 进行评估并显示结果。

思考以下代码：

```
<!DOCTYPE HTML><HTML>
<BODY>
<SCRIPT type="text/javascript">
eval("num1=5;num2=10;res1=num1+num2;alert(res1);");
</SCRIPT>
</BODY>
</HTML>
```

以上代码对 `eval()` 函数内的 JavaScript 语句进行评估，并在消息框内显示结果。

prompt()

`prompt()` 函数用于显示提示对话框以允许用户输入值。提示对话框包含两个按钮 **OK** 和 **Cancel**。如果用户单击 **OK** 按钮，`prompt()` 函数将返回用户输入的值。然而，如果用户单击 **Cancel** 按钮，将返回空值。

以下代码演示 `prompt()` 函数的用法：

```
<!DOCTYPE HTML><HTML>
<BODY>
<SCRIPT type="text/javascript">
var name=prompt("Please Enter Your Name","John");
alert(name);
</SCRIPT>
</BODY>
</HTML>
```

以上代码通过使用 `prompt()` 函数提示用户输入名称，并在消息框中显示该名称。`prompt()` 函数包含两个参数。第一个参数 **Please Enter Your Name** 要求用户提供名称，第二个参数将 **John** 指定为默认名。第一个参数在对话框中显示消息，是必需参数。另一方面，第二个参数指定默认文本，是可选参数。

`prompt()` 函数总是返回字符串值。因此，如果您需要处理字符串以外的数据，则需要对 `prompt()` 函数返回的字符串值进行类型转换。

以下代码显示如何转换 `prompt()` 返回的字符串值：

```
<!DOCTYPE HTML><HTML>
<BODY>
<SCRIPT>
var a,b,c;
a=parseInt(prompt("Enter number 1:"));
b=parseInt(prompt("Enter number 2:"));
c=a+b;
document.write(c);
</SCRIPT>
</BODY>
</HTML>
```

上述代码提示用户输入两个数字，然后通过 `parseInt()` 函数来转换它们。

注释

`document.write()` 函数在 JavaScript 中用于在网页上直接显示字符串输出。您将在后续章节中更详细地了解其内容。

confirm()

`confirm()` 函数用于显示对话框以使用户能够确认或接受任务。此对话框包含两个按钮 **OK** 和 **Cancel**。如果用户单击 **OK** 按钮，`confirm()` 函数返回 true。如果用户单击 **Cancel** 按钮，`confirm()` 函数返回 false。

以下代码演示 `confirm()` 函数的用法：

```
<!DOCTYPE HTML><HTML>
<BODY>
<SCRIPT type="text/javascript">
var response;
response=confirm("Do You Wish to Continue");
if(response==true){
alert("You can proceed further");
}
else
{
alert("You cannot proceed further");
}
</SCRIPT>
</BODY>
</HTML>
```

如果用户单击 **OK** 按钮，以上代码将显示消息 **You can proceed further**。然而，如果用户单击 **Cancel** 按钮，将显示消息 **You cannot proceed further**。

用户定义函数

JavaScript 使您能够根据需求定义自己的函数。例如，您需要通过接受学生分数来计算其平均分。在此情况下，可创建用户定义的函数 `Average()`，并在需要计算平均分时调用此函数。它提高了代码的模块性和效率。

创建函数

使用关键字 `function` 后跟函数名称和圆括号 () 创建函数。函数通常是在网页的头部分定义的。

以下语法用于创建函数：

```
function [functionName] (Variable1, Variable2)
{
//function statements
}
```

用户定义函数可选择性地接受一系列参数。函数所接受的参数是在圆括号中提供的，并由逗号分隔。在上述代码中，Variable1 和 Variable2 表示传递到函数的参数。函数可使用这些参数中传递的值来执行某些操作。

思考演示创建函数的以下代码段：

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript">
function tax(product_category,product_name,price)
{
if(product_category=="Mobile")
{
total_price=(12.5/100)*price+price;
alert("Total price of " +product_name +" is:$" +total_price);
}
if(product_category=="Medicine")
{
total_price=price;
alert("Total price of " +product_name +" is:$" +total_price);
}
}
</SCRIPT>
</HEAD>
<BODY>
. ...
</BODY>
</HTML>
```

在以上代码段中，创建了 tax() 函数以接受三个参数 product_category、product_name 和 price。tax() 函数用于计算在将税金加到产品价格以后的产品总价。

访问函数

一旦创建了函数，可从代码中需要使用该函数所提供功能的任何部分调用它。函数可向调用代码返回任何数据类型的值。让我们来了解如何调用函数，如何从函数返回值，如何检索函数返回的值。

调用函数

函数是通过函数名调用的。以下语法用于访问函数：

```
functionName ();
```

在以上语法中，functionName 是函数的名称。

调用函数时，还可向它传递变量或值。传递给函数的变量或值称为参数。如果需要提供参数，调用函数时在圆括号中进行指定。以下语法用于在函数中指定参数：

```
functionName (parameter1, parameter 2...);
```

以下代码演示如何访问函数：

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript">
. ..
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT type="text/javascript">
product_category="Medicine"; tax(product_category,"Paracetamol",8000);
product_category="Mobile"; tax(product_category,"GV3",3000);
tax(product_category,"XV5",5000);
</SCRIPT>
</BODY>
</HTML>
```

在以上代码中，调用 tax() 函数并将 product_category、product_name 和 price 作为参数传递。tax() 函数显示将税金加到产品价格之后的总价。如果 product_category 是 Medicine，不会将税金加到产品价格。然而，如果 product_category 是 Mobile，会将产品价格的 12.5 % 作为税金进行加价。

从函数返回值

函数可通过 return 语句返回值。以下代码演示如何从函数返回值：

```
function functionName()
{
var variable=10;
return variable;
}
```

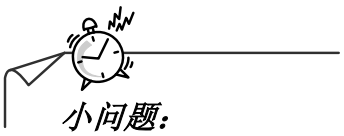
思考以下代码：

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript">
function sum(a,b)
{
return a+b;
}
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT type="text/javascript"> total = sum (3, 6); document.write(total);
</SCRIPT>
</BODY>
</HTML>
```

在以上代码中，sum() 函数接受两个数字 3 和 6 作为参数，并且返回它们的和 9。

注释

函数可以接受多个参数，但是只能返回一个值。



小问题：

以下哪个函数用于接受用户输入？

1. `alert()`
2. `prompt()`
3. `isNan()`
4. `confirm()`

答案：

2. `prompt()`

活动 4.2：实现函数

练习问题

1. 以下哪个选项是逻辑运算符？

   a. `>=`

   b. `+`

   c. `?`

   d. `&&`

2. 请思考以下代码：

```
<HTML>
<BODY>
<SCRIPT type="text/javascript">
var i=6;
if(i%2==0)
{
alert(i+ " is an even number ");
}
else
{
alert(i+ " is an odd number ");
}
</SCRIPT>
</BODY>
</HTML>
```

   指出将在警报框中显示的消息。

   a. i is an even number.

   b. 6 is an odd number.

   c. 6 is an even number.

   d. 它将生成错误。

3. 以下哪个选项是 `if...else` 构造的替代？

   a. `switch…case`

   b. `break`

   c. `continue`

   d. `for`

4. 以下哪个函数用于显示对话框以使用户能够确认或接受任务？

   a. `confirm()`

   b. `prompt()`

   c. `eval()`

   d. `parseint()`

5. 以下哪个运算符用于将两数相除并返回余数？

   a. /

   b. %

   c. *

   d. +

小结

在本章中，您学习了：

- 要创建动态交互网页，需要在网页中加入一个称为脚本的代码块。
- 客户端脚本是指运行在用户计算机上的 Web 浏览器在客户端执行的脚本。
- 服务器端脚本是指 Web 服务器根据用户请求执行的脚本。
- 脚本可直接嵌入到网页中，方法是在 `<SCRIPT>` 标记中编写 JavaScript 代码或者在外部 JavaScript (.js) 文件中编写整个 JavaScript 代码。
- 外部 JavaScript 文件是以 .js 扩展名保存的。
- 可通过以下方式向变量赋值：
  - 在声明后向变量赋值
  - 在声明时初始化变量
  - 初始化变量时不显式声明
- 运算符是用于计算或比较的包含一个或多个字符的字符组。
- 可在 JavaScript 中使用以下类别的运算符：
  - 算术运算符
  - 赋值运算符
  - 算术赋值运算符
  - 比较运算符
  - 逻辑运算符
- JavaScript 中的两个条件构造是：
  - `if...else` 构造
  - `switch...case` 构造
- 在 JavaScript 中，可使用以下循环结构：
  - `while` 循环
  - `do...while` 循环
  - `for` 循环
- 函数是具有名称的自包含语句块。
- JavaScript 支持的一些内置函数是：
  - `isNaN()`
  - `parseInt()`
  - `parseFloat()`
  - `eval()`
  - `prompt()`
  - `confirm()`
- 使用关键字 `function` 后跟函数名称和圆括号 () 创建函数。