1. **JavaScript**

## 章节目标

* 掌握脚本的基本结构
* 掌握JavaScript的执行原理
* 掌握JavaScript的核心语法
* 掌握JavaScript函数

## 为什么要学习JavaScript

　JavaScript 通常被称为 JS，他发明的目的，就是作为浏览器的内置脚本语言，为网页开发者提供操控浏览器的能力，他可以让网页呈现出各种特殊效果，为用户提供友好的互动体验。

场景一：

在实际开发中，表单验证的应用场合比较常见，例如，网站中常见的会员注册，我们填写注册信息时，如果某项信息格式输入错误（例如用户名为空，密码长度位数不够等），表单页面将及时给出错误提示，这些错误在没有提交到服务器前，在客户端提前进行验证。这样，用户得到了即时的交互，同时呢，也减轻了网站服务器的压力，这是JavaScript最常用的场合。

场景二:

在实际开发中，会经常使用到层的切换特性，比如编写响应鼠标点击事件，创建动态页面特性，从而高效的控制页面的内容。它们可以在有限的页面空间展现更多的内容，从而增加客户端的体检，也可以使我们的网站更加有动感，有魅力，更加吸引更多的浏览者。

## 什么是JavaScript

JavaScript是一种**基于对象(Object)**和**事件驱动(Event Driven)**并具有**安全性**能的脚本语言。使用它的目的是与HTML超文本标记语言、Java 脚本语言（Java小程序）一起实现在一个Web页面中链接多个对象，与Web客户交互作用。从而可以开发客户端的应用程序等。它是通过嵌入或调入在标准的HTML语言中实现的。它的出现弥补了HTML语言的缺陷，它是Java与HTML折衷的选择。

**什么是脚本语言？**

* 脚本语言介于HTML和C,C++,Java,C#等编程语言之间
* 脚本语言与编程语言有相似地方，其函数与编程语言类似，也有变量。与编程语言之间最大的区别是编程语言的语法和规则更为严格和复杂一些。
* 脚本语言是一种解释性语言，例如Python、VBScript、JavaScript等等，它不像c\c++等可以编译成二进制代码，以可执行文件的形式存在。
* 脚本语言一般都有相应的脚本引擎来解释执行，JavaScript的脚本引擎就是浏览器。

### JavaScript的组成

JS 由三部分组成：

* ECMAScript：欧洲计算机协会，制定JS的语法、语句等，这是 JS 的核心部分。
* DOM：文档对象模型（Document Object Model）。DOM 赋予了 JS 操作 HTML 的能力，即 document 操作。
* BOM：浏览器对象模型（Browser Object Model）。BOM 赋予了 JS 操作浏览器的能力，即 window 操作。

### 脚本的基本结构

通常JavaScript代码是用<script>标签嵌入HTML文档中，可以将多个脚本嵌入到一个文档中，只需将每个脚本都封装在<script>标签中即可。浏览器在遇到<script>标签时，将逐行读取内容，直到遇到</script>结束标签为止。然后，浏览器将检查JavaScript语句的语法，如果有任何错误，就会在警告框中显示；如果没有错误，浏览器将编译并执行语句。

* + - 1. **脚本的基本结构**

脚本的基本结构如下：

|  |
| --- |
| <script type=*"text/javascript"*>  JavaScript语句;  </script> |

* + - 1. **使用JavaScript脚本在页面上显示输出**

示例代码1：

|  |
| --- |
| <script type=*"text/javascript"*>  document.write("hello word");  </script>  <p>加油</p> |

其中 document.write()字段是标准的JavaScript命令，用来向页面输出内容，把 document.write()命令输入到<script>与</script>之间后，浏览器就会把它当做一条JavaScript命令来执行，这样浏览器就会向页面写入hello word。

### 在网页中引用JavaScript的方式

JavaScript作为客户端程序，嵌入网页有以下三种方式：

* 使用<script>标签
* 使用外部JavaScript文件
* 直接在HTML标签中

1. **使用<script>标签**

直接使用<script>标签把JavaScript代码加入到HTML文档中，这是最常用的方法，大部分含有JavaScript的网页都采用这种方法，例如：

|  |
| --- |
| <script type=*"text/javascript"*>  document.write("你好");  </script> |

另外需要注意：<script>...</script>的位置并不是固定的，可以包含在<head>...</head>或<body></body>中任何地方。

但是使用这种方式适用于JavaScript代码较少，并且网站中的每个页面使用的JavaScript代码均不相同的情况。在实际工作中，有时候会希望在若干个页面中运行JavaScript实现相同的页面效果，并且在每个页面中写不同的脚本，对于这样的问题，通常使用外部JavaScript文件。

**2.使用外部JavaScript外部文件**

外部JavaScript文件是将JavaScript写入一个外部文件中，以\*.js为后缀保存这个文件，然后把\*.js文件指定给<script>标签中的“src”属性，这样就可以使用这个外部文件了，如下代码：

|  |
| --- |
| <script type=*"text/javascript"* src=*"js/hello.js"*></script>  <p>加油</p> |

Hello.js就是JavaScript外部文件，src属性表示指定外部JavaScript文件路径。

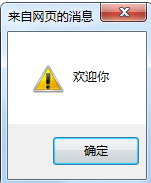
**3.直接在HTML标签中**

有时需要在页面加入很简单的JavaScript代码实现一个简单的页面效果，例如单击按钮时弹出一个对话框，这样通常会在按钮事件中加入JavaScript处理程序，如下面例子：

关键代码：

|  |
| --- |
| <input name=*"btn"* type=*"button"* value=*"弹出消息"* onclick="javascript:alert('欢迎你');" /> |

当点击“弹出消息”按钮时，弹出提示对话框，如下图：



## JavaScript核心语法

### 变量的声明和赋值

JavaScript是一种弱类型语言，没有明确的数据类型，也就是说，在声明变量时，不需要指定变量的类型，变量的类型由赋给变量的值决定。在JavaScript中，变量是使用关键字var声明的。具体语法格式如下：

|  |
| --- |
| **var** 合法的变量名; |

可以在声明变量的同时为变量赋值，也可以在稍后的脚本中为变量赋值。

|  |
| --- |
| **var** a=20; //声明变量a的同时，将数值20赋给变量a  **var** x,y,z=10; //同时声明多个变量,各个变量之间用逗号隔开 |

特别注意：**JavaScript区分大小写，所以大小写不同的变量名表示不同的变量。**另外，由于JavaScript是一种弱类型语言，所以允许不声明变量而直接使用，系统将会自动声明该变量，例如：

|  |
| --- |
| b=66;//没有声明变量b，直接使 |

**但是这种方法很容易出错，也很难查找排除，不建议使用。**

### 数据类型

虽然JavaScript是一种弱类型语言，在声明变量时不需要声明变量的数据类型，而是由赋给变量的值决定，但有时也需要检测变量的具体数据类型，在JavaScript中，提供了常用的基本数据类型，如下：

* undefined（未定义类型）
* null（空类型）
* number（数值类型）
* string(字符串类型)
* boolean（布尔类型）

**1.typeof运算符**

ECMAScript提供了typeof运算符来判断一个值或变量究竟属于哪种数据类型。语法为：

|  |
| --- |
| **typeof**（变量或者值）; |

其返回结果有以下几种：

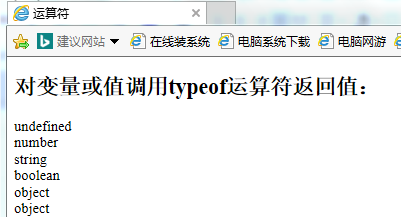
* Undefined：如果变量是undefined型的。
* Number：如果变量是number型的。
* string：如果变量是string型的。
* boolean：如果变量是boolean型的。
* Object:如果变量是null型，或者变量是一种引用类型，例如对象，函数。

如下例子。

实例2：

|  |
| --- |
| <script type=*"text/javascript"*>  document.write("<h2>对变量或值调用typeof运算符返回值：</h2>");  **var** a,b=10,name="zhansgan";  **var** date=**new** Date();  document.write(**typeof**(a)+"<br>");  document.write(**typeof**(b)+"<br>");  document.write(**typeof**(name)+"<br>");  document.write(**typeof**(**true**)+"<br>");  document.write(**typeof**(**null**)+"<br>");  document.write(**typeof**(date)+"<br>"); |

运行结果如下图：



**2.undefined类型**

Undefined类型只有一个值，即undefined。当声明的变量未初始化时，该变量的默认值是undefined。

|  |
| --- |
| **var** a; |

这行代码声明了变量a，且变量没有初始值，显示为undefined。

1. **null类型**

只有一个值的类型是null，是一个表示“什么都没有”的占位符，可以用来检测某个变量是否被赋值。值undefined实际上是值null派生来的，因此ECMAScript把它们定义为相等的。

|  |
| --- |
| alert(**null**==undefined);//返回值为true |

尽管这两个值相等，但它们的含义不同，undefined是声明了变量但未对该量赋值，null则表示对该变量赋予了一个空值。

**4.number类型**

ECMA-262中定义的最特殊的类型是number类型，这种类型即可以表示32位的整数，还可以表示64位的浮点数。如下列代码：

|  |
| --- |
| **var** num=23;//定义了一个整数  **var** num1=23.0;//定义了一个浮点数 |

1. **string类型**

在JavaScript中字符串是一组被引号括起来的文本，例如：

|  |
| --- |
| **var** str ="好好学习";//定义了一个字符串 |

与Java不同的是，JavaScript不对“字符”和“字符串”加以区别，如以下例子，也是定义了一个字符串。

|  |
| --- |
| **var** str1 ="a";//定义了一个字符的字符串 |

**6.boolean类型**

Boolean类型是JavaScript最常用的类型之一，它只有两个值true和false.

### 运算符

与Java中运算符一样，在JavaScript中常用的运算符分为算数运算符、比较运算符、逻辑运算符和赋值运算符。

算术运算符：+一\*/%++--  
比较运算符：==, != ,>=,<=

逻辑运算符：&&, || ,!(非)

字符串运算符：

|  |
| --- |
| **var** x = "yellow";  **var** y = "green";  **var** z = x + y + "white"; //则z为yellowgreenwhite  **var** w = y + 9; //则w为green9 |

### 逻辑控制语句

在JavaScript中，逻辑控制语句用于控制程序的执行顺序。同Java一样，逻辑语句也分为两类，条件结构和循环结构。

* + - 1. **条件结构**

与Java一样，条件结构分为if结构和switch结构。

* + - 1. **循环语句**

同Java一样，JavaScript中也分为for循环，while循环，do-while循环

### JavaScript语法规则

* + - 1. **区分大小写**

几乎所有JavaScript的内容都是区分大小写的：小写字母和大写字母是不能互相替换的。几个基本的规则如下：

* JavaScript的关键字，例如for和if，永远是小写。
* 内置对象，例如Math和Date是以大写字母开头的。
* DOM对象的名称通常是小写，但其方法经常是大小写混合的。通常第一个字母不是大写，例如getElementById。
  + - 1. **变量、对象和函数名称**

当定义自己使用的变量、对象或函数时，可以选择它们的名称。**名称可以包括大写字母、小写字母、数字和下划线。名字必须以一个字母或下划线开头**。

可以选择在变量名中使用大写字母还是小写字母，**但必须记住JavaScript是区分大小**写的：score、Score和SCORE将被认为是三个不同的变量。要保证每次使用同一个变量时都是同样的名称。

* + - 1. **保留字**

变量名称的另一个规则是它们不能使用保留字。保留字包括JavaScript语言的组成部分，（例如if和for）、DOM对象的名称（例如window和document）和内置对象的名称（例如Math和Date）。

* + - 1. **空格**

JavaScript是忽略空格的。程序中可以在一行中包括空格和制表符，也可以包括空白行，这些不会造成错误。空格通常是为了使JavaScript程序更具可读性。

### 注释

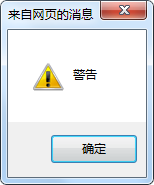
JavaScript语言注释与Java语言的注释一样，分为单行注释和多行注释两种。

### 常用的输入、输出

在网页上经常会弹出一些信息框，例如注册时弹出某些文本框中必须输入值的信息框或弹出一个等待用户输入数据的对话框，这样的输入或输出在JavaScript中称为警告对话框和提示对话框。

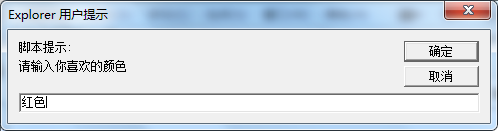
* + - 1. **警告（alert）**

alert()方法会创建一个特殊的小窗口，该窗口带有一个字符串和一个“确定”按钮，如下图：



* + - 1. **提示（prompt）**

prompt()方法会弹出一个提示对话框，等待用户输入一行数据，如下图：



prompt()方法的基本格式语法为：

|  |
| --- |
| prompt('提示信息','请输入框的默认值'); |

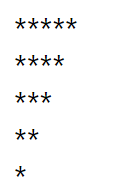
该方法的返回值可以被引用或存储到变量中，例如：

|  |
| --- |
| **var** color=prompt("请输入你喜欢的颜色","红色"); |

prompt()方法的第一个参数显示在对话框上，通常是一些提示信息，第二参数出现在用户输入的文本框中，且被选中，作为默认使用，第二个参数也可以省略。

* + - 1. **课上小练习**

1. 根据prompt（）方法输入的次数，连续在页面上输入hello word。
2. 根据用户输入的整数在页面打印倒三角。



### 程序调试

## JavaScript常用语法—函数

在JavaScript中函数类似于Java中的方法，是执行特定任务的语句块。但是**JavaScript中的函数使用更简单，不用定义函数属于哪个类，更不需要像Java中使用“对象名.方法名()”的方式调用函数，可以直接调用函数名称来使用函数。**在JavaScript中函数有如下特点：

* JavaScript中的函数是完成特定功能并可以随时随地地运行的语句，比如简易的计算器、表单验证等。
* 函数不仅能在一个或多个HTML页面中被多次调用，而且能够在不同网站中应用，从而提高代码的复用率。
* 函数为程序设计人员带来了很多方便，通常在进行一个复杂的程序设计时，总是根据所要完成的功能，将程序划分为一些相对独立的部分，每一部分编写一个函数，从而使各部分充分独立，任务单一，使程序结构清晰，易读，易懂，易重用，易维护。
* JavaScript函数可以封装那些在程序中可能要多次用到的模块，并可以作为事件驱动的结果而被调用，从而实现一个函数与事件驱动相关联。
* 可以将值传递给函数，函数也可以返回一个值。

JavaScript中函数有两种，一种是JavaScript自带的系统函数，一种是用户自行创建的自定义函数。

### 常用系统函数

ECMScript提供了两种把非数字的原始值转成数字的函数，即parseInt()和parseFloat()，另外还提供了一个检查是否是非数字的函数isNaN().

* + - 1. **parseInt()**

parseInt()函数可解析一个字符串，并返回一个整数，语法格式为：

|  |
| --- |
| parseInt("字符串"); |

在判断字符串是否是数字前，parseInt()和parseFloat()都会分析该字符串。

parseInt()函数首先查看位置0处的字符，判断它是否是个有效数字，如果不是则返回NaN,不在继续执行其他操作。但如果该字符是有效数字，该函数将查看位置1处的字符，进行同样的测试，这一过程将持续到发现非有效数字的字符为止，此时parseInt将该字符串之前的字符转换成数字，例如：

|  |
| --- |
| **var** num1=parseInt("78.89");//返回值为78  **var** num2=parseInt("4567cc");//返回值为4567  **var** num3=parseInt("this36");//返回值为NaN |

下面通过一个例子来掌握parseInt()函数的用法。使用prompt()方法获取两个数，使用parseInt()函数把这两个数转换为整数，然后相加并输出在页面，实例代码如下：

|  |
| --- |
| <script type=*"text/javascript"*>  **var** op1=prompt("请输入第一个数：",",");  **var** op2=prompt("请输入第二个数：",",");  **var** p1=parseInt(op1);  **var** p2=parseInt(op2);  **var** result=p1+p2;  document.write(p1+"+"+p2+"="+result);  </script> |

运行结果如下图：



* + - 1. **parseFloat()**

parseFloat()函数可解析一个字符串，并返回一个浮点数，语法格式为：

|  |
| --- |
| parseFloat("字符串"); |

parseFloat()函数和parseInt()函数处理方式相同。从位置0开始查看每个字符串，直到找到第一个非有效的字符为止，然后把该字符之间的字符串转换为浮点数。

对于这个函数来说，第一个出现的小数和点是有效字符，如果两个小数点，那么第二个小数点被看作是无效的，例如：

|  |
| --- |
| **var** num1=parseFloat("4567cc");//返回值为4567  **var** num2=parseFloat("45.58");//返回值为45.58  **var** num3=parseFloat("45.58.25");//返回值为45.58  **var** num4=parseFloat("this36");//返回值为NaN  alert(num1);  alert(num2);  alert(num3);  alert(num4); |

* + - 1. **IsNaN()**

isNaN()函数用于检查其参数是否是非数字，语法格式如下：

|  |
| --- |
| isNaN(x); |

如果x是特殊的非数字值，返回值就是true，否则返回false，例如：

|  |
| --- |
| **var** num1=isNaN("12.5");//返回值为false  **var** num2=isNaN("12.5s");//返回值为true  **var** num3=isNaN("45.8");//返回值为false  alert(num1);  alert(num2);  alert(num3); |

### 自定义函数

同Java语言一样，我们需要先定义函数，然后 才能调用函数，下面来详细介绍。

* + - 1. **创建函数**

在JavaScript中，自定义函数由关键字function，函数名，一组参数以及置于括号中的待执行的JavaScript语句组成。语法格式为：

|  |
| --- |
| **function** 函数名(参数1，参数2,...){  JavaScript语句;  } |

* 参数1，参数2等是传入函数的变量或值。
* “{”和“}”定义了函数的开始和结束。

函数中的参数是可选的，当传入参数时通常把函数称为有参函数，当没有传入参数时，称为无参函数，无参函数必须在其函数名后加括号。例如：

|  |
| --- |
| **function** 函数名(){  JavaScript语句;  } |

在JavaScript中，return语句用来规定从函数返回的值，因此需要返回某个值的函数必须使用return语句。

* + - 1. **调用函数**

要执行一个函数，必须先调用这个函数，当调用函数时，必须指定函数名及其后面的参数（如果有参数）。函数的调用一般和表单元素的事件结合使用，调用格式如下：

|  |
| --- |
| 事件名=“函数名()”; |

下面通过示例1和示例2来学习如何创建函数和调用函数。

示例1：

|  |
| --- |
| <html xmlns=*"http://www.w3.org/1999/xhtml"*>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"* />  <title>示例1</title>  <script type=*"text/javascript"*>  **function** showHello(){  **for**(**var** i=0;i<5;i++){  document.write("<h2>Hello Word</h2>");  }  }  </script>  </head>  <body>  <input name=*"btn"* type=*"button"* value=*"显示5次Hello Word"* onclick="showHello()" />  </body>  </html> |

* showHello()是创建的无参函数。
* Onclick表示按钮单击事件，当点击按钮时调用函数showHello()。

在浏览器运行示例1如图1所示，单击按钮，调用showHello()无参函数，动态的在页面中循环输出5行“Hello Word”，如图2所示：

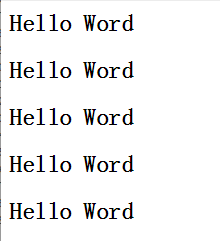
 

图1 图2

在示例1中使用的无参函数，运行一次页面只能输出5行“Hello Word”，如果需要根据用户的要求每次输入不同行数，该怎么办呢？有参函数可以实现这样的功能。

下面修改示例1，把函数showHello()修改成一个有参函数，使用prompt()提示用户每次输出“Hello Word”的行数，然后啊prompt()返回的值作为参数传递给函数showHello().

示例2：

|  |
| --- |
| <html xmlns=*"http://www.w3.org/1999/xhtml"*>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"* />  <title>示例2</title>  <script type=*"text/javascript"*>  **function** showHello(count){  **for**(**var** i=0;i<count;i++){  document.write("<h2>Hello Word</h2>");  }  }  </script>  </head>  <body>  <input name=*"btn"* type=*"button"* value=*"请输入显示Hello Word的次数"* onclick="showHello(prompt('请输入显示Hello Word的次数',','))" />  </body>  </html> |

showHello(count)是有参函数，count表示传递的参数。

使用prompt('请输入显示Hello Word的次数',',')弹出窗口，把用户输入的值作为参数传递给函数showHello(count)。

在for循环中，参数count的值作为循环输出“Hello Word”的次数。

在浏览器运行示例2如图1所示，单击按钮，弹出提示用户输入显示Hello Word次数的窗口；用户输入值后，根据用户输入的值在页面上输出Hello Word，如图2所示：

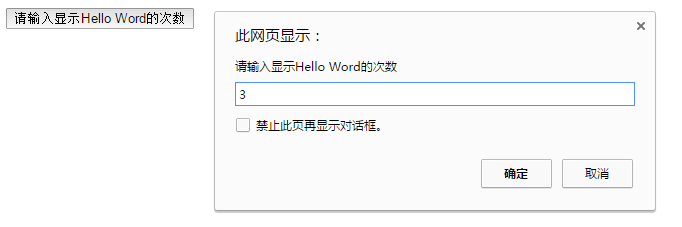
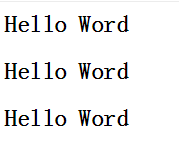
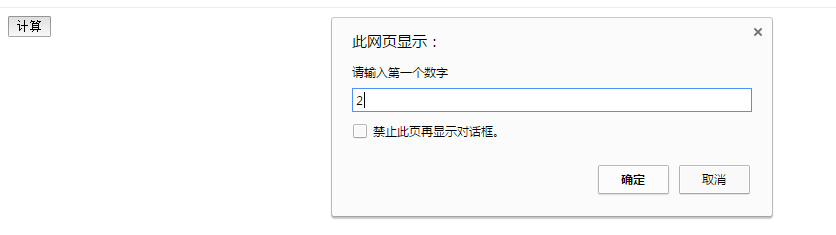
 

图1 图2

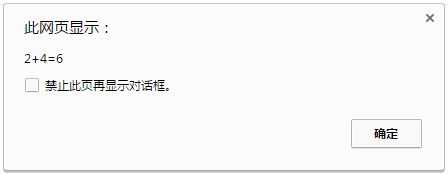
### 上机练习3

需求说明:

单击页面的按钮时，调用函数，使用prompt()方法获取两个变量的值和一个运算符，如下图所示：



使用两个运算的结果使用alert()方法显示出来，如下图：



使用switch判断获取运算符。

### 变量的作用域

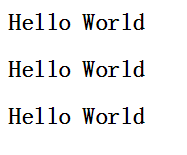
与Java中的变量一样，在JavaScript中，根据变量作用的范围不同，可分为全局变量和局部变量。

JavaScript中的全局变量与Java中的成员变量类似，是在所有函数之外的脚本中声明的变量，作用的范围是该变量定义后的所有语句，包括其后定义的函数中的代码，以及其后的<script>和</script>标签中的代码。

JavaScript中的局部变量与Java中局部变量类似，是在函数内声明的变量，只有在该函数中且位于该变量之后的代码可以使用这个变量，如果在之后的其他函数中声明了与这个局部变量同名的变量，则后声明的变量与这个局部变量毫无关系。

如以下例子

|  |
| --- |
| <html xmlns=*"http://www.w3.org/1999/xhtml"*>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"* />  <title>变量的作用域</title>  <script type=*"text/javascript"*>  **var** num;  **function** showHello(){  **for**(**var** i=0;i<num;i++)  {  document.write("<h2>Hello World</h2>");  }  }  **function** counts(){  num=prompt("请输入显示HelloWorld的次数：","");  showHello();  }  </script>  </head>  <body>  <input name=*"btn"* type=*"button"* value=*"请输入显示HelloWorld的次数"* onclick="counts()"/>  </body>  </html> |

运行上面的例子，在prompt()弹出的输入框中输入3，点击确定按钮，运行结果如下图：  


从运行结果可以看出，一开始声明的num为全局变量，但在showHello()中声明的num为局部变量，它只作用于函数showHello()中。

## 课后总结

* JS 由三部分组成：
* ECMAScript：也叫解释器，充当翻译角色，这是 JS 的核心部分。
* DOM：文档对象模型（Document Object Model）。DOM 赋予了 JS 操作 HTML 的能力，即 document 操作。
* BOM：浏览器对象模型（Browser Object Model）。BOM 赋予了 JS 操作浏览器的能力，即 window 操作。
* 在HTML页面中引用js的三种方式：直接把JavaScript代码写在标签<script>和</script>之间；使用外部JavaScript文件或直接把简短的JavaScript代码写在HTML标签中。
* JavaScript的核心语法有变量的声明和赋值、数据类型、逻辑控制语句，注释，在JavaScript中代码区分大小写。并且建议每一句的末尾使用分号结束。
* 在JavaScript中常用的输入、输出是prompt()方法和alert()方法。
* 使用alert()方法调试程序。
* 函数分为系统函数和自定义函数；自定义函数需要先创建，然后才能调用。
* 自定义函数分为有参函数和无参函数。
* 使用window对象可以实现弹出窗口、关闭当前窗口、弹出页面消息框等效果。
* Date对象可以获得当前系统日期，时间。
* 使用定时函数与Date对象可以制作时钟特效。

## 课后作业

1. **描述prompt()、alert()和confrim()三者的区别，举例说明。**

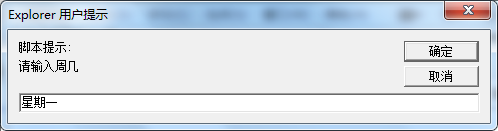
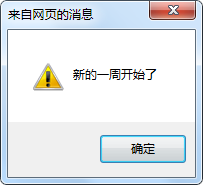
2.使用JavaScript脚本在页面上输出一个正方形，要求如下：

* 使用prompt()方法输入正方形的行数
* 在页面上输出正方形，如下图所示：

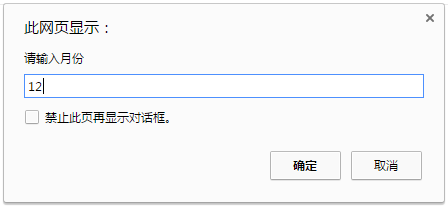
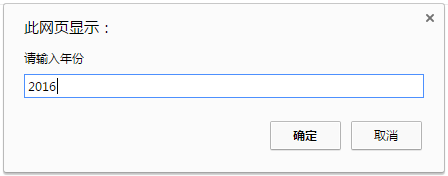


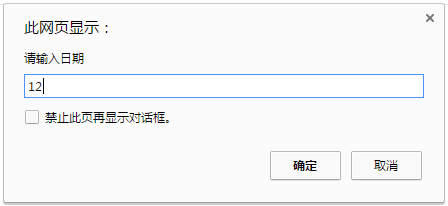
3.prompt()方法在页面中弹出提示，根据用户输入星期一至星期日的不同，弹出不同的信息提示框，要求如下：

* 输入“星期一”时，弹出“新的一周开始了”。
* 输入“星期二”，“星期三”，“星期四”时，弹出“努力工作”。
* 输入“星期五”时，弹出“明天就是周末了”。
* 输入其他内容弹出“放松的休息”。

4.prompt()方法在页面中弹出提示，根据用户输入年份，月份和天数的不同，计算出是输入年份的第几天（注意闰年和平年的计算）





5.制作显示年、月、日，显示星期几，显示时钟特效，时钟为12小时进制

实现思路

* 使用getFullYear()获得当前年份
* 使用getMonth()+1获得当前月份
* 使用getDate()获得当前日期
* 根据getHours()获得的小时，使用if语句判断当前时间是否大于12
* 使用getDay()获取当前表示星期几的数字，然后使用switch设置当前星期几

根据Date对象获取当前的日期和时间，根据不同时间显示不同的问候语

* 如果当前时间小于12点（含），显示上午好
* 如果当前时间大于12点，小于18点（含），显示下午好
* 如果当前时间大于18点，显示晚上好

