# 函数的定义和调用

## 函数简介

代码设计的一个原则：可重复利用，**即执行相同功能的代码应该只定义一次**。

JS中的alert()、parseInt()、console.log()、document.write()等。

函数：**完成特定功能的一段代码**（主要要实现**可重用性**，因此对于函数中的代码越多，那么所完成的功能就越多，重用率也就越低，此外要实现**任务分解**，将一个大的行为，分解成不同的事件函数）

## 函数的三要素：（函数名，函数参数，函数返回值）

– 函数名：如alert、 parseInt 、……

– 函数的参数：传递给函数名的值，代表将被函数处理的数据，如alert ( ‘hello’ )

– 函数的返回值：函数执行的返回结果，如confirm()，其返回值为true或false

注意：

**（1）函数可以有return，也可以没有return，其中有return返回代表着返回后面的值，没有return代表返回的是一个undefined值。**

**（2）函数的参数可以有多个，也可以没有。**

**JS中函数的分类**

* **JS内置函数**：如parseInt( )、Boolean( )、String( )、alert( )、……
* **自定义函数**

## 函数的定义

* 使用**function关键字**定义函数

**function** funName([arg1, arg2,……])

{

**functionBody;**

return returnValue(可选);

}

Demo：

**function sum(a,b){**

**return a+b;**

**}**

**alert(sum(2,3));**

* 将函数**赋值给一个变量**，使用**函数直接量**来定义函数

var **funName** = function ([arg1, arg2,……]) {

**functionBody;**

return returnValue(可选);

}

Demo：

**var sum = function (a,b){//在这里赋值给一个变量，相当于**

**sum是这个函数；**

**return a+b;**

**}**

**alert(sum(2,3));**

* 匿名函数

函数定义时，函数名是可选的，即可以定义没有函数名的函数，但该函数必须马上执行或赋值给一个变量（或事件）

(function (name) {

alert('hello ,' + name + "!");

})('Mike');//**定义时，没有函数名，因此需要马上执行**

window.onload = function() {

//页面加载完成以后，执行的一系列动作

};//**该函数赋值给一个变量，因此引用这个变量就可以**

## 函数的调用

* 直接调用函数

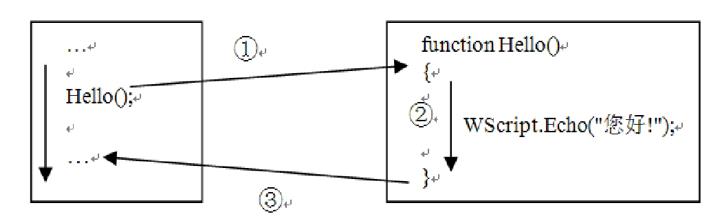
– 使用**( )**运算符，调用一个函数

– 可以向函数**传递参数**

– 函数可能含有返回值，该返回值可做为普通数据进行处理

* 在事件中调用函数

– 当事件产生时，JS可以调用函数来**响应事件**



在函数调用过程中，如果函数调用在函数定义的前面，程序也是可以执行的，虽然程序是一行一行执行，但是在JavaScript程序之前，要先将函数进行域解析，函数事先放到内存中，因此这样是可以出结果的。

## 函数式编程

函数式编程：（JS中）函数像普通变量一样可以赋值给其他变量，可以作为参数传递，也可以作为函数的返回值返回。



使用函数的注意事项

* 定义函数时，函数名必须是**合法的标识符**，不能使用保留字当函数名。函数名要通俗易懂，最好可以通过函数名就能看出函数的功能。
* 设计函数时，**最好每个函数只能实现一种功能**，有利于函数的扩展、引用和维护。
* 为了便于引用，**常用的或者先用的函数**应该放在整个JavaScript代码的前面。

# 函数的参数和返回值

## 函数的参数

形式参数，在函数定义的时候，要在括号里面有形式参数，

实际参数，在函数调用的时候，要在括号里面传上真正的实参。



上述代码主要实现的是在body标签内写入一条span语句，完成的页面效果是Javascript是一门比较容易入门的编程语言！

注意事项：

（1）在定义函数时使用了多少个形参，在该函数调用的时候应该给出**相同数目的实参**。

（2）多个参数之间用“**,**”分隔。

（3）在函数体内，形参其实就是一个变量。

（4）**使用多个参数时，调用所给出的各个实参按照其排列的先后顺序依次传递给形参。（比如给了2个形参，在调用时写了3个实参，也会按照前两个计算，自动的忽略后面的。）**

function sum(x,y)

{

return x + y;

}

document.write(sum(2) + "<br/>");//相当于2+undefined 得到的是NaN；

document.write(sum(2,5) + "<br/>");

document.write(sum(2,5,9) + "<br/>");//9会视而不见，返回7

## 函数的返回值

函数调用时，一方面可以通过参数向函数传递数据，另一方面也可以从函数获取数据。

注意：

1、返回值可以直接赋予变量或用于表达式中

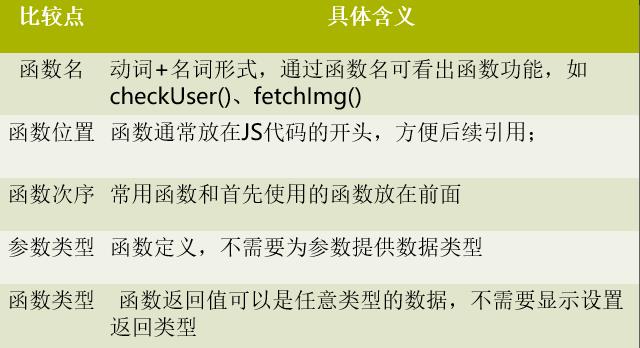
2、return语句表示结束当前函数的执行

3、return语句**可以不带表达式**（例如：return;）

4、return语句不带表达式时仍会返回值，该值为**undefined**

5、函数中可以不出现return语句，**仍会返回值，该值为undefined**

## 函数定义的注意事项



# 函数的嵌套

## 函数的嵌套

* 在一个函数定义的函数体语句中出现对另一个函数的调用，这就是函数的嵌套调用。
* 当一个函数调用另一个函数时，应该提前定义好被调用函数

function first()

{

alert('this is the first function ');

}

function second()

{

first();

}

second();//由这个函数继续向上进行查找，查找到了函数体内又有一个函数，继续进行查找，知道执行了命令即可。

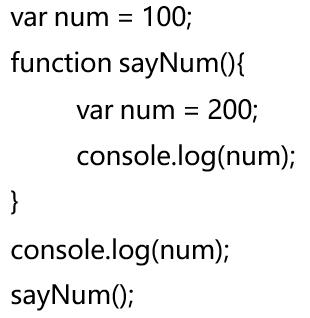
## 变量作用范围

全局变量：

在**所有函数之外**定义，或者是**在函数中没有通过var声明的变量**。其作用范围是同一个页面文件中的所有脚本。

局部变量：

通过var声明且定义在函数体之内的变量。只作用于该函数体。

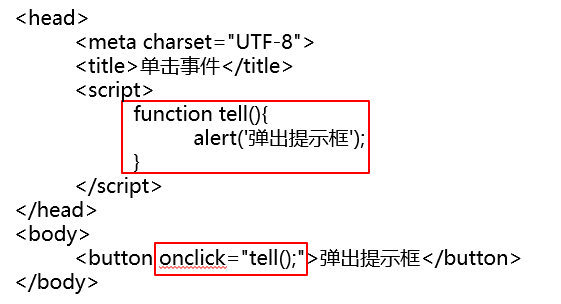


上述代码中，第一个值是100，表示的全局变量的值，第二个时200，表示的是函数局部内部的num的值

# 事件及事件处理

## 事件基本定义

* 事件：能被JavaScript检测到的活动
  + 用户动作（鼠标或键盘操作等）
  + 状态变化（加载、改变文本框内容等）
* 事件处理函数：当该活动发生时（称之为触发事件时），所执行的响应该活动的函数



**事件的三要素**

* 在哪个HTML元素上？
* 发生什么事件？
* 程序作何处理（事件处理函数）？

**添加事件方法**

* 在JavaScript中，为元素添加一个事件，一般有两种方法
  + 在HTML元素中，添加**HTML动作属性**，绑定一个事件处理函数

<button onclick="tell();">弹出提示框</button>

* + 在JavaScript中，为HTML元素动态添加事件处理函数

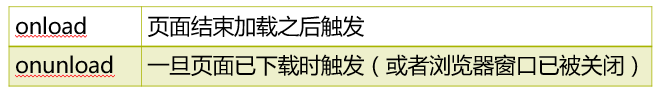
document.getElementById('btn').onclick = function(){ }

* 事件处理机制：当某一个事件触发时，会执行操作以响应该事件；当该事件再次发生时，响应操作会再次执行。
* 响应事件的操作是**一段代码（如函数）**，会捕获每一次事件触发的动作，然后执行该段代码。即事件处理机制中，函数的执行是由事件所触发的。

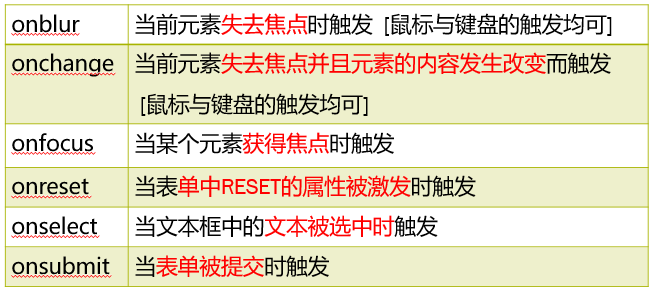
## 常用事件类型

页面加载退出：

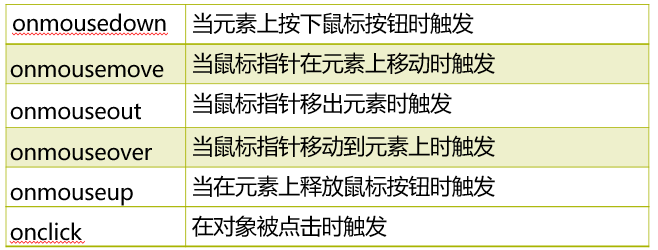
* onload 页面加载事件（文档元素）—— 在页面或图像加载完成后立即发生。
* onunload 页面退出事件 —— 在用户退出页面时发生。



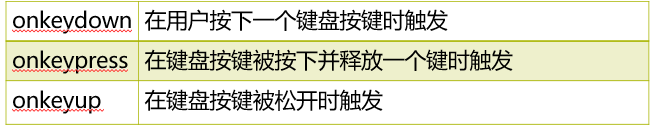
表单事件（表单及表单控件元素）



鼠标事件（所有元素）



键盘事件



* JavaScript程序是“**基于事件驱动**”
  + 通过事件同用户产生交互
  + 初始化代码通常在文档加载事件中执行