**JavaScript**

JavaScript是一种基于对象的编程语言，java中强调的是类的设计，JavaScript中是基于队形的是所有对象都已经由浏览器提供好了，直接使用就可以了

Java和JavaScript没用本质联系。

1. **基本输出：**

在java中用system.out.print()进行输出，在JavaScript中用document.write(“”)进行输出

Eg:document.write(“<h1>hello world</h1>”)。可以发现document.write()可以输出HTML代码。但是document.write()函数输出的HTML代码是不可控的袁术顺序，所以我们在开发之中一般不会利用该函数输出的，**之所以使用Chrome浏览器，主要是因为他后台有代码调试功能，而从另一方面强调他很方便。使用“console.log()”实现信息后台输出，用于调试。**

1. **定义变量**

JavaScript中利用var定义变量，也可以不使用var定义变量直接使用就是。

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>第一个JavaScript测试</**title**>  
 <**script type="text/javascript"**>  
 **var *num*** = 10; *//定义整形变量* **console**.log(typeof ***num***);  
 **document**.write(**"<h1>"**+(typeof ***num***) + **"</h1>"**)  
 </**script**>  
</**head**>  
<**body**>  
 <**div**>文攀你好</**div**>  
</**body**>  
</**html**>

若还没有对var变量赋值，则默认是未定义变量。在JavaScript中可以在if语句中直接判断一个变量。

在JavaScript里面使用’“’ ’” 或则“” ””的都是字符串string类型，没有char类型。

在字符串比较中，只提供了==操作，没有equals()操作。

在java里面使用的各种String累中的操作方法都可以在JavaScript里面更加简洁的使用。

范例：打印乘法表

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>第一个JavaScript测试</**title**>  
 <**script type="text/javascript"**>  
 **var *num*** ; *//定义整形变量* **document**.write(**"<table border = '1'>"**)  
 **for**(**var *x*** = 1;***x*** <= 9;***x***++){  
 **document**.write(**"<tr>"**)  
 **for**(**var *y*** = 1;***y*** <= ***x***;***y***++) {  
 **document**.write(**"<td>"** + ***x*** + **"\*"** + ***y*** + **"="** + (***x*** \* ***y***) + **"</td>"**);  
 }  
 **for**(**var *y*** = 1;***y*** <= 9 - ***x***;***y***++){ *//输出几个空格* **document**.write(**"<td>&nbsp;</td>"**)  
 }  
 **document**.write(**" </tr> "** );  
 }  
 **document**.write(**"</table>"**);  
 </**script**>  
</**head**>  
<**body**>  
 <**div**>文攀你好</**div**>  
</**body**>  
</**html**>

在使用JavaScript操作的过程之中一定要注意生成的代码的完整性。

在JavaScript中支持的while循环和switch结构与java相似。

在JavaScript中数组本身是以一个对象的形式产生的，所以要用new关键字。

eg：var result = new Array();

result[0]=100;

Result[1]=”wenpan”;

javascript语法没有任何的严格性，数组是没有长度限制的，天生就是一个动态数组。

从数组的本质上来讲虽然和java类似，但是比java数组的功能强大。

**定义函数：**

函数定义格式：function 函数名称（参数）{

[return [返回值]]

}

在java中方法里面定义了多少个参数就要传入多少个参数，但是在JavaScript中却可以不限制参数个数。所以为了方便用户，提供了一个功能：“函数名称.arguments”表示取得函数的参数内容，返回的是一个数组，可以使用length判断长度。

**事件处理：**

事件处理是整个JavaScript的核心

在页面上的任何操作都有可能被称为事件源，而对每一个事件可以进行自定义处理方式，例如：键盘按下，鼠标移动等等都可以称他为事件源。

所有的事件源都不需要过多的去记住。所有的事件在JavaScript里面都是以onXxxx的形式命名的。比如以下两个事件：  
 1.页面加载事件：onload

2.页面卸载事件：onunload

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>第一个JavaScript测试</**title**>  
 <**script type="text/javascript"**>  
 **function** *loadHandle*() { *//页面打开时弹出* alert(**"欢迎你的光临！"**);  
 }  
 **function** *closeHandle*() {  
 alert(**"拜拜!"**) ; *//页面关闭时弹出* }  
 </**script**>  
</**head**>  
<**body onload="*loadHandle*();" onunload="*closeHandle*()"**>  
 <**div**>文攀你好</**div**>  
</**body**>  
</**html**>

**鼠标操作事件**：onmousedown(鼠标按下时触发),onmouseover(鼠标进入时触发);onmouseout(鼠标移开时触发);onmouserup(鼠标弹开时触发);onmousemove(鼠标移动时触发);

范例：在按钮上绑定事

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>第一个JavaScript测试</**title**>  
 <**script type="text/javascript"**>  
 **function** *clickHandle*() {  
 alert(**"事件触发"**);  
 }  
 </**script**>  
</**head**>  
<**body onload="**loadHandle();**" onunload="**closeHandle()**"**>  
 <**button type ="button" ondblclick="***clickHandle*()**"** >按我把</**button**>  
</**body**>  
</**html**>

鼠标单击事件：onclick = 要执行的操作 ,鼠标双击事件：ondbclick = 要执行的操作

**范例：观察一个程序(静态事件设置,在每一个元素内设置处理事件)**

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>第一个JavaScript测试</**title**>  
 <**script type="text/javascript"**>  
 **function** *changeColor*(obj,color) {  
 *//修改当前行颜色* obj.**bgColor** = color;*//由外部颜色决定* }  
  
 </**script**>  
</**head**>  
<**body onload="**loadHandle();**" onunload="**closeHandle()**"**>  
 <**table border="1" cellpadding="5"cellspacing="0" bgcolor="F2F2F2"**>  
 <**tr onmousemove="***changeColor*(**this**,**'green'**)**"  
 onmouseout="***changeColor*(**this**,**'white'**)**"**>  
 <**td**>部门编号</**td**>  
 <**td**>部门名称</**td**>  
 <**td**>部门位置</**td**>  
 </**tr**>  
 <**tr onmousemove="***changeColor*(**this**,**'green'**)**"  
 onmouseout="***changeColor*(**this**,**'white'**)**"**>  
 <**td**>10</**td**>  
 <**td**>财务部</**td**>  
 <**td**>上海</**td**>  
  
 </**tr**>  
 <**tr onmousemove="***changeColor*(**this**,**'green'**)**"  
 onmouseout="***changeColor*(**this**,**'white'**)**"**>  
 <**td**>11</**td**>  
 <**td**>技术部</**td**>  
 <**td**>成都</**td**>  
 </**tr**>  
 </**table**>  
</**body**>  
</**html**>

JavaScript是一个基于对象的编程语言，在JavaScript中所有的对象都在元素上（即元素就是一个对象）。

但是需要注意的是所有的事件,除了采用以上的方式之外,也可以动态设置,这种形式在开发之中使用的是最多的.那么可以使用**addEventListener(“事件类型”,处理函数,触发时机)**进行动态设置。

对于整个JavaScript事件的处理分为两个部分：

事件的冒泡过程

事件的触发过程

触发时机都会设置为false，表示在事件的触发过程进行处理。阻止事件冒泡。

范例：观察动态设置

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>第一个JavaScript测试</**title**>  
 <**script type="text/javascript"**>  
 **function** *handle*() {  
 alert(**"事件触发"**);  
 }  
 *//以下是表示在页面加载的时候进行加载事件的处理* **window**.**onload** = **function** () { *//匿名函数  
 handle*();  
 }  
  
 </**script**>  
</**head**>  
<**body**></**body**>  
</**html**>

范例2：

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>第一个JavaScript测试</**title**>  
 <**script type="text/javascript"**>  
 **function** *handle*() {  
 alert(**"事件触发"**);  
 }  
 *//以下是表示在页面加载的时候进行加载事件的处理* **window**.**onload** = **function** () { *//匿名函数* **var** imgElement = **document**.getElementById(**"myimg"**);*//表示取得了myimg元素对象* imgElement.addEventListener(**"click"**,*handle*,**false**);  
 }  
  
 </**script**>  
</**head**>  
<**body**>  
 *<!-- ID是整个JavaScript的操作核心所在，必须存在-->* <**img id = "myimg" src="images/pic1.jpg" height="50%"**>  
</**body**>  
</**html**>

所有的动态设置事件都可以在onload里面进行动态的配置。但是如果要配置，那么必须要能够明确的取得一个元素对象。可以使用：**“document.getElementById(元素ID)”的方式取得元素对象。**

这种动态设置操作实践的最大好处在于，HTML代码不会和JavaScript代码混合在一起。

范例：动态操作

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>第一个JavaScript测试</**title**>  
 <**script type="text/javascript"**>  
 **window**.**onload** = **function** () {*//匿名函数* **var** butElement = **document**.getElementById(**"setBut"**);  
 butElement.addEventListener(**"click"**,**function** (){  
 *//在此元素内容增加制定内容，而且可以是HTML代码* **document**.getElementById(**"info"**).**innerHTML** = **" <img src = 'images/pic1.jpg' height = '50%'>"**;  
  
 },**false**)  
 }  
 </**script**>  
</**head**>  
<**body**>  
 <**span id = "info"**></**span**>  
 <**button id = "setBut"**>设置显示图片</**button**>  
  
</**body**>  
</**html**>

下面继续做一个小功能的扩充，实现一个图片浏览器。

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>第一个JavaScript测试</**title**>  
 <**script type="text/javascript"**>  
 **var *imgName*** = **new** Array(**"pic1.jpg"**,**"pic2.jpg"**,**"pic3.jpg"**,**"pic4.jpg"**,**"pic5.jpg"**,**"pic6.jpg"**);  
 **var *foot*** = 1; *//操作的脚标*

*//以下表示在页面加载的时候进行加载事件的处理* **window**.**onload** = **function** () {*//匿名函数* **var** pbut = **document**.getElementById(**"priviousBut"**); *//得到上一张的按钮* **var** nbut = **document**.getElementById(**"nextBut"**); *//得到下一张的按钮* **var** img = **document**.getElementById(**"img"**); *//取得图片元素* nbut.addEventListener(**"click"**,**function** (){ *//为按钮添加监听事件* **if**(***foot*** >= ***imgName***.**length**){  
 ***foot*** = 0;  
 }  
 img.**src** = **"images/"** + ***imgName***[***foot***++];  
 },**false** );  
 pbut.addEventListener(**"click"**,**function** (){ *//为上一张按钮添加监听事件* **if**(***foot*** <= 0){  
 ***foot*** = ***imgName***.**length**;  
 }  
 img.**src** = **"images/"** + ***imgName***[***foot***--];  
 },**false** );  
 }  
 </**script**>  
</**head**>  
<**body**>  
 <**span id = "info"**>  
 <**img id = "img" src="images/pic1.jpg" height="60%"**>  
 </**span**>  
  
 <**button id = "priviousBut"**>上一张</**button**>  
 <**button id = "nextBut"**>下一张</**button**>  
</**body**>  
</**html**>

在JavaScript里面提供有一个setTimeout(函数，时间-毫秒)，利用此函数可以实现定期的操作调用

范例：观察setTimeOut函数的使用

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>第一个JavaScript测试</**title**>  
</**head**>  
<**body**>  
 <**span id = "info"**></**span**>  
</**body**>  
  
<**script type="text/javascript"**> //一定要放在后面  
 **function** *setColock*() {  
 **var** date = **new** Date();*//取得时间* **document**.getElementById(**"info"**).**innerHTML** = **""** +  
 **"<h1>系统当前的时间："** + date + **"</h1>"**;  
 setTimeout(*setColock*,1000);  
 }  
 *setColock*();  
</**script**>  
</**html**>

范例：使用动态时间绑定，实现图片自动滚片

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>第一个JavaScript测试</**title**>  
</**head**>  
<**body**>  
 <**span id = "info"**>  
 <**img id = "img" src="images/pic1.jpg" height="60%"**>  
 </**span**>  
</**body**>  
  
<**script type="text/javascript"**>  
 **var *imgName*** = Array(**"pic1.jpg"**,**"pic2.jpg"**,**"pic3.jpg"**,**"pic4.jpg"**,**"pic5.jpg"**,**"pic6.jpg"**);  
 **var *foot*** = 1;  
 **function** *setPic*() {  
 **var** img = **document**.getElementById(**"img"**); //得到元素  
 img.**src** = **"images/"** + ***imgName***[***foot***++]; //改变元素的属性  
 **console**.log(img.**src**); //在控制台输出信息  
 **if**(***foot*** >= ***imgName***.**length**){  
 ***foot*** = 0;  
 }  
 setTimeout(*setPic*,1000); *//每秒更新一次* }  
 *setPic*(); *//调用函数*</**script**>  
</**html**>

**总结：**

1. **所有的事件一定要绑定在他的触发元素上；**
2. **首先要掌握的是静态事件绑定的方法，在元素上个使用onXxx的形式**
3. **每一种事件都是一定要捆绑在一个函数进行事件处理。**