Web开发(二)

---1-3 函数与事件处理

河北师范大学软件学院
Software College of Hebei Normal University

内容提纲

- 函数的定义和调用
- 函数的参数和返回值
- 函数的嵌套
- 事件及事件处理

内容提纲

- **函数的定义和调用**
- **函数的参数和返回值**
- **函数的嵌套**
- **事件及事件处理**

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
  - 数学中的函数：y = f(x)，传递一个x，即返回x所对应的函数值
  - C语言中的sqrt()函数、scanf()函数、……
  - JS中的alert()、parseInt()、……
- 函数：完成特定功能的一段代码
  - 可重用性
  - 任务分解

函数简介

- 函数的要素
  - 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

函数简介

- JS中函数分类：
  - 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);
}
```

demo1-3-1

函数的定义

- 使用函数直接量定义函数

```
var funName = function ([arg1, arg2,.....]) {
        functionBody;
        return returnValue(可选);
}
```

函数的定义

- 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');

window.onload = function {
        alert('hello ,' + name + "!");
};
```

函数的定义

- 使用函数直接量定义函数（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');
```

```
var add = function(a, b) {
    return a + b;
};
var sum = add(1, 2);
alert(add(1, 2));
```

```
window.onload = function() {
    //页面加载完成以后，执行的一系列动作
};
```


demo1-3-2

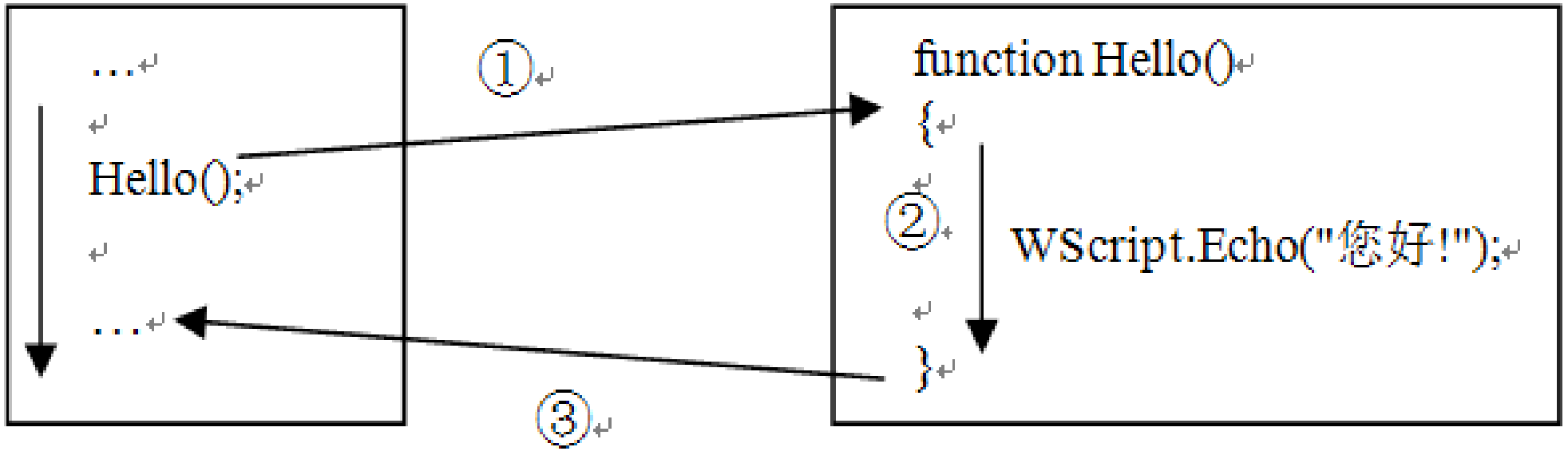
函数调用

- 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
- 在表达式中调用函数

```
alert( add(1,2) );
```

- 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

函数的调用



函数调用　　函数定义

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。

demo1-3-3

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。

内容提纲

- 函数的定义和调用
- 函数的参数和返回值
- 函数的嵌套
- 事件及事件处理

函数的参数

- **定义函数**

```
function 自定义函数名(参数1,参数2… )
{
  函数体;
  return 返回值;
}
```

形式参数

- **调用函数**

```
函数名(参数1,参数2… )
```

实际参数

函数的参数

```
<script type="text/javascript">
        function show(text,size){
                document.write("<span style='font-
size:"+size+"'>"+text+"</span>");
        }
        show("J","20px");
        show("avaScript 是一门比较容易入门的编程语言！","14px");
</script>
```

demo1-3-4

函数的参数

- 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。
- 多个参数之间用“,”分隔。
- 在函数体内，形参其实就是一个变量。
- 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。

函数的参数

```
<script type="text/javascript">
        function sum(x,y){
                return x + y;
        }
        document.write(sum(2) + "<br/>");
        document.write(sum(2,5) + "<br/>");
        document.write(sum(2,5,9) + "<br/>");
</script>
```

demo1-3-5

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。

```
function 自定义函数名(参数1,参数2... )
{
    函数体;
    return 返回值;
}
```

函数的返回值

```
function Max(x,y) {
        var max;
        if (x>y)
                max=x;
        else
                max=y;
        return max;
}
var m;
m = Max(100,200);
alert("Max(100,200)=" + m);
```

demo1-3-6

函数的返回值

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

函数定义注意事项

比较点	具体含义
函数名	动词+名词形式，通过函数名可看出函数功能，如checkUser( )、fetchImg( )
函数位置	函数通常放在JS代码的开头，方便后续引用；
函数次序	常用函数和首先使用的函数放在前面
参数类型	函数定义，不需要为参数提供数据类型
函数类型	函数返回值可以是任意类型的数据，不需要显示设置返回类型

内容提纲

- 函数的定义和调用
- 函数的参数和返回值
- **函数的嵌套**
- 事件及事件处理

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( );
```


demo1-3-7

变量作用范围

- 全局变量

在所有函数之外定义，或者是没有通过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。

变量作用范围

```
var num = 100;
function sayNum(){
        var num = 200;
        console.log(num);
}
console.log(num);
sayNum();
```

内容提纲

- 函数的定义和调用
- 函数的参数和返回值
- 函数的嵌套
- **事件及事件处理**

事件

- 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

单击事件

```
<head>
        <meta charset="UTF-8">
        <title>单击事件</title>
        <script>
                function tell(){
                        alert('弹出提示框');
                }
        </script>
</head>
<body>
        <button onclick="tell();">弹出提示框</button>
</body>
```

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

- 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

- 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。

onload	页面结束加载之后触发
onunload	一旦页面已下载时触发（或者浏览器窗口已被关闭）

demo1-3-8

常用事件类型

- 表单事件（表单及表单控件元素）

onblur	当前元素失去焦点时触发 ［鼠标与键盘的触发均可］
onchange	当前元素失去焦点并且元素的内容发生改变而触发 ［鼠标与键盘的触发均可］
onfocus	当某个元素获得焦点时触发
onreset	当表单中RESET的属性被激发时触发
onselect	当文本框中的文本被选中时触发
onsubmit	当表单被提交时触发

常用事件类型

- 鼠标事件（所有元素）

onmousedown	当元素上按下鼠标按钮时触发
onmousemove	当鼠标指针在元素上移动时触发
onmouseout	当鼠标指针移出元素时触发
onmouseover	当鼠标指针移动到元素上时触发
onmouseup	当在元素上释放鼠标按钮时触发
onclick	在对象被点击时触发

常用事件类型

- 键盘事件

onkeydown	在用户按下一个键盘按键时触发
onkeypress	在键盘按键被按下并释放一个键时触发
onkeyup	在键盘按键被松开时触发

事件在JS中的地位

- 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

Thank You !