Java

head 或body 中使用<script>标签嵌入java

```
<script type="text/javascript">
    window.alert("Hello"); //note also /*note for lines here*/

    gday();
    funciton gday(){
        var dd=new Date();
        var d=dd.getDay();  //星期几
        alert(d);
        document.write("Today is "+d.toString());
    }

    function xxx(){
        var shf="441900292929292929";
        if(shf.length!=18)
            alert("Wrong length");
        else
            alert("Success!");
    }
</script>
```

> 把脚本置于 元素的底部，可改善显示速度，因为脚本编译会拖慢显示

添加多个脚本：

```
<script src="myScript1.js"></script>
<script src="myScript2.js"></script>
```

- ECMAScript的变量类型是松散类型的，可以用来保存任何类型的数据，每个变量只是一个用于保存数值的占位符
- 省略var的是全局变量
- 简单数据类型：Undefined Null Boolean Number String
- 复杂类型Object

```
var id, sex, age="ma";
```

- 检查变量类型：typeof var 或者typeof (var)
- undefined类型只有一个值，一般不需要显式定义某个变量为undefined
- null为一个空对象指针，undefined==null结果为true（前者）
- number表示整数和浮点数
- NaN：非数值，与任何数都不相等（包括自身）

    - isNaN(n)检测n是否是非数值，返回一个boolean
    - Number() 把非数值转化为任何类型
    - parseInt() 字符串转化为整数
    - parseFloat() 字符串转化为浮点
- string表示字符串
    - toString() 任何类型转化为字符串
- boolean
    - 除了0，‘’，null和undefined，返回值都为true

- = = = ，!==判断数值的同时判断数据类型是否相同
- && | |
    - 短路原则
- ! 返回一个boolean

显示

JavaScript 能够以不同方式“显示”数据：

- 使用 window.alert() 写入警告框
- 使用 document.write() 写入 HTML 输出
- 使用 innerHTML 写入 HTML 元素
- 使用 console.log() 写入浏览器控制台

如需访问 HTML 元素，JavaScript 可使用 document.getElementById(id) 方法。

id 属性定义 HTML 元素。innerHTML 属性定义 HTML 内容

> 更改 HTML 元素的 innerHTML 属性是在 HTML 中显示数据的常用方法

在 HTML 文档完全加载后使用 **document.write()** 将*删除所有已有的 HTML*， 只用于测试

在浏览器中，您可使用 console.log() 方法来显示数据。

请通过 F12 来激活浏览器控制台，并在菜单中选择“控制台”

语法

在 JavaScript 中，首字符必须是字母、下划线（_）或美元符号（$）

JavaScript 程序员倾向于使用以小写字母开头的驼峰大小写

如果再次声明某个 JavaScript 变量，将不会丢它的值

相加两个数字，将返回和，但对一个数字和一个字符串相加将返回一个字符串

该运算中的任何数值运算数都会被转换为 32 位的数。结果会被转换回 JavaScript 数

<<	**零填充左位移**
>>	有符号右位移
>>>	零填充右位移

```
var x = 911 + 7 + "Porsche"; // x='918Porsche'
var x = "Porsche" + 911 + 7; // x='Prosche9117'

var cars = ["Porsche", "Volvo", "BMW"];
```

任何变量均可通过设置值为 undefined 进行清空。其类型也将是 undefined

```
person = undefined;          // 值是 undefined，类型是 undefined
```

undefined 与 null 的值相等，但类型不相等，后者的类型为object

typeof只会返回function或者object

typeof 运算符把对象、数组或 null 返回 object。

typeof 运算符不会把函数返回 object，返回function

对长字符串换行的最安全做法（但是有点慢）是使用字符串加法：

```
document.getElementById("demo").innerHTML = "Hello" +
"Kitty!";
```

字符串是不可变的：字符串不能更改，只能替换

可以通过 split() 将字符串转换为数组：

```
var txt = "a,b,c,d,e";   // 字符串
txt.split(",");          // 用逗号分隔
txt.split(" ");          // 用空格分隔
txt.split("|");          // 用竖线分隔
txt.split("");           // 分隔为字符
```

> JavaScript 对象无法进行对比，比较两个 对象的话JavaScript 将始终返回 false

函数

访问没有 () 的函数将返回函数定义

```
function toCelsius(fahrenheit) {
    return (5/9) * (fahrenheit-32);
}

document.getElementById("demo").innerHTML = toCelsius;
// display 'function toCelsius(f) { return (5/9) * (f-32); }'
```

对象

创建对象

```
var car = {type:"porsche", model:"911", color:"white"};
```

对象也可以有*方法*。方法是在对象上执行的*动作*。方法以*函数定义*被存储在属性中。方法是作为属性来存储的函数.

```
var person = {
  firstName: "Bill",
  lastName : "Gates",
  id       : 678,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

在函数定义中，this 引用该函数的“拥有者”。在上面的例子中，this 指的是“拥有” fullName 函数的 *person 对象*。

访问属性:

```
person.lastName;
person["lastName"];
```

事件

事件	描述
onchange	HTML 元素已被改变
onclick	用户点击了 HTML 元素

事件	描述
onmouseover	用户把鼠标移动到 HTML 元素上
onmouseout	用户把鼠标移开 HTML 元素
onkeydown	用户按下键盘按键
onload	浏览器已经完成页面加载