# JavaScript概述

## JavaScript概述

### JavaScript是一门独立的语言

JavaScript是基于对象和事件驱动的脚本语言，主要应用在客户端。

JavaScript是一门独立的语言。

JavaScript是一种脚本语言，不需要编译，是解释运行的语言。

JavaScript运行需要环境，最常见的运行环境就是浏览器，现代主流的浏览器都内置了JS引擎，可以运行JavaScript代码。JS也有服务器端运行环境，我们不做讨论（Node.js、GoogleV8引擎…）。

### JS的特点

解释运行，没有编译过程

基于对象

弱类型

### JavaScript不得不说的历史

和JAVA的关系：语法十分相似，但其实毫无关系。

1995年5月，Netscape，LiveScript

1995年12月，改名为JavaScript

1996年8月，微软，Jscript

1997-1999年，ECMA，ECMAScript，基于已有的JavaScript和Jscript，提出了标准的Script语法规则，JavaScript和Jscript都遵循这套标准。

1999-以后，ECMAScript不断的更新

### JS的优点：

交互性（它可以做的就是信息的动态交互）

安全性（不允许直接访问本地硬盘）

跨平台性（只要是可以解释Js的浏览器都可以执行，和平台无关）

# 和HTML的结合

## 在HTML中引入JS

### 在HTML中直接编写JS

在<head>标签内包含<script>标签，在其中编写JS代码，如图-1所示：

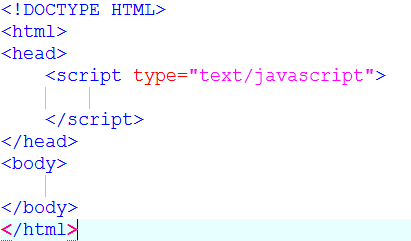


图-1

### 引入外部JS

大量的JS代码写在HTML中会导致页面复杂，难于维护。此时可以将JS代码提取到单独的js文件中。再在HTML中页面中通过标签引入，如图-2所示：

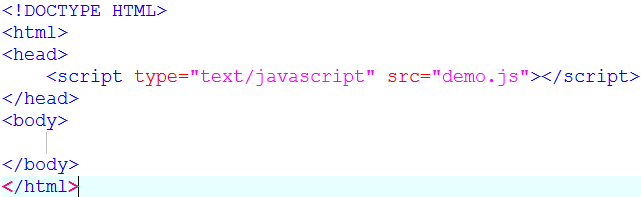


图-2

要注意，这种方式引入JS，标签不可以写成自闭标签。

# JavaScript语法

## 数据类型

### Js中的数据类型

JS中的数据类型分为基本数据类型和复杂数据类型。

## 基本数据类型

### 基本数据类型

JS基本类型一共有五种

Number、String、Boolean、undefined、null

### Number

代表数字的基本类型。

JS不区分整形和浮点型，JS中的所有数值类型底层实现都是浮点型。

数值类型中有如下几个特殊值

Infinity 无穷大

-Infinity 负无穷大

NaN 非数字，非数字非常特殊，和任何值都不行等，包括本身，即NaN==NaN的值为false。可以通过isNaN()判断某值是否为NaN。

数值类型是基本数据类型，但JS本身提供了对应的包装对象Number，具有和数值处理相关的属性和方法。

Number提供的属性：

Number.MAX\_VALUE 可表示的最大数字

Number.MIN\_VALUE 可表示的最小数字

Number.NaN 非数字值

Number.POSITIVE\_INFINITY 正无穷大

Number.NEGATIVE\_INFINITY 负无穷大

### String

Js中的字符串是基本数据类型，字符串常量必须用双引号引起来。

JS中提供了字符串的包装对象String，提供了字符串处理相关的属性和方法。

String对象提供的属性：

length：字符串中字符数

String 对象提供的方法：

参考JS文档

### Boolean

Boolean类型只有两个值，true、false。

JS中提供了布尔类型的包装对象Boolean，提供了布尔类型处理的相关属性和方法。

参考JS文档

### Undefined

Undefined类型只有一个值就是undefined，表示变量未定义，当一个变量未初始化时，值为undefined。

### Null

Null只有一个值就是null。null用来表示尚未存在的对象，常用来表示函数企图返回一个不存在的对象。

## 复杂数据类型

### 复杂数据类型

Js中的复杂数据类型指的是对象。关于对象我们后边再详细讲解。

### JS数据类型的转换

Js数据类型转换

Js中的数据类型会在需要时自动进行转换。转换的规则如下。

数字：可以转为对应字符串 0转为false其他数字转为true 转为对应的Number包装对象

字符串：空字符串转换为0，数字字符串转换为对应数字，其他字符串转换为NaN，空字符串转为false，其他字符串为true，转为对应的String包装对象。

布尔类型：转为对应字符串，ture转为1false转为0，转为对应的Boolean包装对象。

对象：转为字符串，如果为null则为“null”，如果对象为null则转为false，否则转为true，如图-3所示：

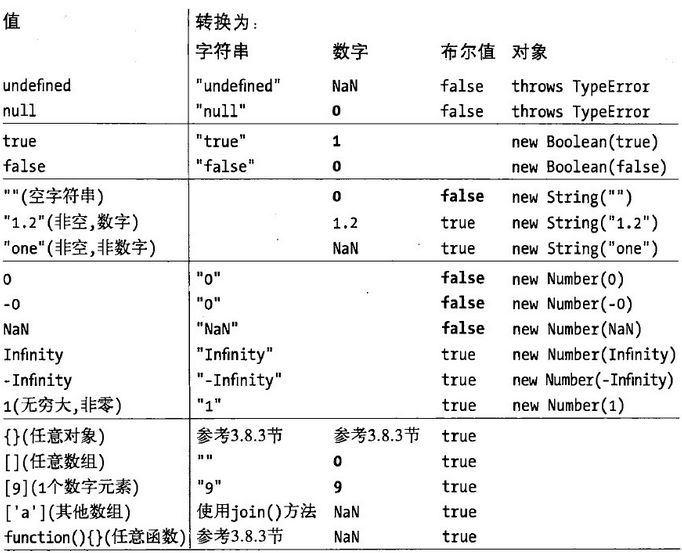


图-3

## 定义变量

### 变量的定义

JS中有数据类型，但是JS的引用是不区分类型的，所以称JS为弱类型，即，一个引用可以先指向数字类型，后再指向字符类型，还可以再指向对象类型。如下 的代码是没有问题的。

var x = 99；

x = “aaa”;

x = true;

x = new Object();

## 运算符

### Js中的运算符

Javascript中的运算符和Java大致相同。

只是运算过程中需要注意几点：

var x = 3120/1000\*1000; x = 3120;而不是3000。

var x = 2.4+3.6 ; x = 6;而不是6.0

var x = “12” + 1; x = “121”; x = “12” – 1 ; x = 11;

加号对于字符串是连接符

&& || 是逻辑运算符 & | 是位运算符。

也支持三元运算符 ? :

特殊运算符 typeof : 返回一个操作表达式的数据类型的字符串。

var x = 3;

var y = “123”;

var z = false;

typeof(x); //number

typeof(y); //string

typeof(z); //boolean

## 语句

### 判断结构(if语句)

var x = 3;

if(x==4)//可以进行比较运算。

if(x=4)//可以进行赋值运算，而且可以同样进行判断。不报错。

因为在Js中0或者null就是false，

非0或者非null就是true（通常用1表示）。

所以if(x=4)结果是true；

可以通过if(4==y)来解决该问题。因为4=y不会进行判断，而是会报错。

### 选择结构(switch语句)

和java基本相同，不同的是：因为弱类型，也可以对字符串进行选择。

### 循环结构(while语句，do…while语句，for语句)。

注：和java基本相同，不同的是，没有了具体数据类型的限制，使用时要注意。

## 函数

### 普通方法定义函数

function 函数名(形式参数...)

{

执行语句；

return 返回值；

}

函数是多条执行语句的封装体，只有被调用才会被运行。

注意：调用有参数的函数，但没有给其传值，函数一样可以运行，或者调用没有参数的函数，给其传值，该函数也一样运行。

说的简单点：只要写了函数名后面跟了一对小括号，该函数就会运行。那么传递的参数呢？

其实，在函数中有一个参数数组对象（arguments），该对象将传递的参数都封装在一个数组中。

例：

function demo()//定义函数。

{

alert(arguments.length);

}

demo(“hello”,123,true);//调用函数。

那么弹出的对话框结果是3，如果想得到所有的参数值，可以通过for循环遍历该数组。

for(var x=0; x<arguments.length; x++)

{

alert(arguments[x]);

}

为了增强阅读性，最好按照规范，按定义好的形式参数传递实际参数。

函数在调用时的其他写法：

var show = demo();//show变量接收demo函数的返回值。

var show = demo;//这种写法是可以的，意为show和demo代表同一个函数。

//那么该函数也可以通过show()的方式运行。

### 动态函数

通过Js的内置对象Function实现。

例：var demo = new Function(“x”,”y”,”alert(x+y);”);

demo(4,6);

如同：

function demo(x,y)

{

alert(x+y);

}

demo(4,6);

不同的是，动态函数，参数以及函数体都可以通过参数进行传递，可以动态指定。

### 匿名函数

格式：function(){...}

例：var demo = function(){...}

demo();

通常在定义事件属性的行为时较为常用。

例：

function test()

{

alert(“load ok”);

}

window.onload = test;

可以写成匿名函数的形式：

window.onload = function()

{

alert(“load ok”);

}

匿名函数就是一种简写格式。

## 数组

### 定义js数组

var arrayObj = new Array();　//创建一个数组

var arrayObj = new Array([size]);　//创建一个数组并指定长度，注意不是上限，是长度

var arrayObj = new Array([element0[, element1[, ...[, elementN]]]]);　创建一个数组并赋值

var arrayObj = [1,2,5,6,3];//数组直接量定义数组

### Js中的数组特点

Js中数组中的特点：可以存任意元素，长度是可变的。

格式：

var arr = new Array();

arr[0] = “hello”;

arr[1] = 123;

var arr = [‘hello’,123,true,”abc”];

### JS数组的包装对象

JS中提供了数组的包装对象Array，提供了数组处理相关的属性和方法。

Array对象提供的属性：

length：数组长度

Array 对象提供的方法：

参考JS文档

### JS中遍历数组

通过遍历即可对数组进行基本操作。

for(var x=0; x<arr.length; x++)

{

alert(arr[x]);

}

## 对象

### JS内置对象

String对象

strVariable.length

返回 String 对象的长度。

strObj.charAt(index)

返回指定索引位置处的字符。

string1.concat([string2[, string3[, . . . [, stringN]]]])

返回字符串值，该值包含了两个或更多个提供的字符串的连接。

strObj.indexOf(subString[, startIndex])

返回 String 对象内第一次出现子字符串的字符位置。

strObj.lastIndexOf(substring[, startindex])

返回 String 对象中子字符串最后出现的位置。

stringObj.match(rgExp)

使用正则表达式模式对字符串执行查找，并将包含查找的结果作为数组返回。

stringObj.replace(rgExp, replaceText)

返回根据正则表达式进行文字替换后的字符串的复制。

stringObj.search(rgExp)

返回与正则表达式查找内容匹配的第一个子字符串的位置。

stringObj.slice(start, [end])

返回字符串的片段。

stringObj.split([separator[, limit]])

将一个字符串分割为子字符串，然后将结果作为字符串数组返回。

stringvar.substr(start [, length ])

返回一个从指定位置开始的指定长度的子字符串。

strVariable.toLowerCase( )

返回一个字符串，该字符串中的字母被转换为小写字母。

strVariable.toUpperCase( )

返回一个字符串，该字符串中的所有字母都被转化为大写字母。

Array对象

numVar = arrayObj.length

返回一个整数值，这个整数比数组中所定义的最高位元素的下标大 1。

array1.concat([item1[, item2[, . . . [, itemN]]]])

返回一个新数组，这个新数组是由两个或更多数组组合而成的。

arrayObj.join(separator)

返回字符串值，其中包含了连接到一起的数组的所有元素，元素由指定的分隔符分隔开来。

arrayObj.pop( )

移除数组中的最后一个元素并返回该元素。

arrayObj.push([item1 [item2 [. . . [[itemN]( ) ]]]])

将新元素添加到一个数组中，并返回数组的新长度值。

arrayObj.reverse( )

返回一个元素顺序被反转的 Array 对象。

arrayObj.shift( )

移除数组中的第一个元素并返回该元素。

arrayObj.slice(start, [end])

返回一个数组的一段。

arrayobj.sort(sortfunction)

返回一个元素已经进行了排序的 Array 对象。

arrayObj.splice(start, deleteCount, [item1[, item2[, . . . [,[itemN]( )]]]])

从一个数组中移除一个或多个元素，如果必要，在所移除元素的位置上插入新元素，返回所移除的元素。

arrayObj.unshift([item1[, item2 [, . . . [, itemN]]]])

将指定的元素插入数组开始位置并返回该数组。

Date对象

dateObj = new Date()

dateObj = new Date([dateVal]( ))

dateObj = new Date(year, month, date[, hours[, minutes[, seconds[,[ms]( )]]]])

**参看文档**

Function 对象

function.arguments

为当前执行的 function 对象返回一个arguments 对象。

functionName.caller

返回一个对函数的引用，该函数调用了当前函数。

Global对象

是一个内部对象，目的是把所有全局方法集中在一个对象中。

Global 对象没有语法。直接调用其方法。

eval(codeString)

检查 JScript 代码并执行.

isNaN(numValue)

返回一个 Boolean 值，指明提供的值是否是保留值 NaN （不是数字）。

parseFloat(numString)

返回由字符串转换得到的浮点数。

parseInt(numString, [radix])

返回由字符串转换得到的整数。

Math对象

Var num = Math.PI

返回圆的周长与其直径的比值，约等于 3.141592653589793。

Math.ceil(number)

返回大于等于其数字参数的最小整数。

Math.floor(number)

返回小于等于其数值参数的最大整数。

Math.abs(number)

返回数字的绝对值。

Math.exp(number)

返回 e（自然对数的底）的幂。

**Math.random( )**

返回介于 0 和 1 之间的伪随机数。

Math.round(*number*)

返回与给出的数值表达式最接近的整数。

RegExp 对象

re = /pattern/[flags]

re = new RegExp("pattern",["flags"])

rgExp.exec(str)

用正则表达式模式在字符串中运行查找，并返回包含该查找结果的一个数组。

rgexp.test(str)

返回一个 Boolean 值，它指出在被查找的字符串中是否存在模式。

### 自定义对象

除了js内置的对象，也可以自己定义对象。

（1）构造函数定义对象1

function Person()//很象java中的构造函数。P不一定要大写。

{

}

var p = new Person();

p.name = “zhangsan”;//定义成员变量。

p.age = 20;

//定义成员函数。

p.run = function()

{

alert(“run”);

}

p.run();

（2）构造函数定义对象2

function Person(name,age)

{

this.name = name;

this.age = age;

}

var p = new Person(“zhangsan”,20);

（3）对象直接量：

var obj = {name:”张”,age:19,run:function(){alert(“run”)}};

### JS对象操作

（1）with语句。

格式：

with(对象)

{

}

应用：当调用一个对象中多个成员时，为了简化调用，避免”对象.”这种格式的重复书写。

var p = new Person(“zhangsan”,20);

alert(p.name+”,”+p.age);

可以写成：

var p = new Person(“zhangsan”,20);

with(p)

{

alert(name+”,”+age);

}

with语句定义了某个对象的作用域，在该域中可以直接调用该对象的成员。

（2）for...in语句

用于遍历对象属性。

例：

var p = new Person(“zhangsan”,20);

for(x in p)

{

alert(x);//结果是两个对话框，一个是name，一个是age。

alert(x+”:”+p[x]);

//可以得到属性与属性的值。p[x]:p对象就是个数组，要通 //过指定的元素名获取元素的值。

}

# DOM编程

## DHTML概述

### DHTML概述

我们学习JS技术时，提到过，JS本身是一门和运行环境无关的独立的语言，浏览器是其最常见的运行环境。

既然JS是一门和运行环境无关的独立的语言，我们又如何通过这么语言去操作浏览器及其中展示的页面呢？

为了解决这个问题W3C提出来了DHTML即动态HTML技术。

所谓的DHTML就是将浏览器中常用的对象和HTML中的常用标签用JS对象表示，并在这些对象上提供属性和方法，从而实现通过操作这些JS对象来操作浏览器及浏览器中展示的页面内容的技术，从而将原来静态的HTML变为了可以通过JS操作的能动的页面，即动态HTML，DHTML。

现代的主流浏览器都支持DHTML，所以我们只要学习DHTML相关的技术，就可以在任意主流浏览器中通过JS技术操作HTML页面了。

### BOM和DOM

DHTML分为BOM和DOM。

其中BOM是Browse Object Modle 即浏览器对象模型，其中封装了浏览器操作相关对象。

DOM是Document Object Modle 即文档对象模型，将整个HTML文档按照文档结构组织成了树形结构。

## BOM

### Window 对象

代表当前浏览器窗口的对象。是这个浏览器运行环境，所以调用其属性和方法时，可以省略window.

子对象

document

history

navigator

location

方法

alert

confirm

[prompt](mk:@MSITStore:E:\template\WEB\day04_dom\DHTML手册.chm::/DHTMLref/methods/prompt.html)

close

[setInterval](mk:@MSITStore:E:\template\WEB\day04_dom\DHTML手册.chm::/DHTMLref/methods/setInterval.html)

clearInterval

事件

Onload

### history对象

代表浏览器的浏览历史记录。

属性

Length

方法

Back

Forward

Go

### location

代表当前浏览器地址栏

属性

Href

方法

Reload

### navigator

属性

appName

## DOM

### 获取对象

根据id获取页面中的元素

document.getElementById(“xxx”);

根据name获取页面中的元素

document.getElementsByName(“xxx”);

根据标签名获取页面中的元素

document.getElementsByTagName(“xxx”);

### 进行增删改查

在父节点上增加子节点

parentNode.appendChild(childNode);

在父节点上删除子节点

parentNode.removeChild(childNode)

直接移除自己,但是只是移除自己,子孙不移除,也可以在此方法中传入一个true表明在删除自己的同时,子孙也被移除

childNode.removeNode();

childNode.removeNode(true);

凭空创建节点

var tag = document.createElement("tagName");

var attr = document.createAttribute("href");

设置属性

tag.setAttributeNode(attr);

替换子节点

parentNode.replaceChild(newChild,oldChild);

克隆节点

var ele2 = ele.cloneNode();//只克隆当前元素

var ele2 = ele.cloneNode(true);//克隆当前元素包括子元素

### 修改样式

方式一：通过class属性修改样式

document.getElementById("tagName").className =”newClzName”;

方式二：直接修改css样式

tagEle.style.stylename = “stylevalue”

# EasyMall中的Dom编程

## 用户注册的表单校验

### 用户注册的表单校验

如图-1所示：



图-1

### 购物车页面JS实现

如图-2所示：



图-2