<mvc:****

**SpringMVC4**

**课程讲义**

**北京动力节点教育培训公司**

**制作人：老郭**

**JAVASCRIPT技术**

**2015年12月**

# 前言

20世纪90年代中期，当网上冲浪越来越流行时，对于开发客户端脚本的需求也逐渐增大。此时，大部分因特网用户还仅仅通过 28.8 kbit/s 的调制解调器连接到网络，即便这时网页已经不断地变得更大和更复杂。而更加加剧用户痛苦的是，仅仅为了简单的表单有效性验证，就要与服务器进行多次地往返交互。设想一下，用户填完一个表单，点击提交按钮，等待了 30 秒的处理后，看到的却是一条告诉你忘记填写一个必要的字段。

那时正处于技术革新最前沿的 Netscape，开始认真考虑开发一种客户端脚本语言来解决简单的处理问题。当时工作于 Netscape 的 Brendan Eich，开始着手为即将在 1995 年发行的 Netscape Navigator 2.0 开发一个称之为 LiveScript 的脚本语言。

当时Netscape和SUN公司有合作关系，Netscape 与 Sun 及时完成 LiveScript 实现。就在 Netscape Navigator 2.0 即将正式发布前，Netscape 将其更名为 JavaScript，目的是为了利用 Java 这个因特网时髦词汇。Netscape 的赌注最终得到回报，JavaScript 从此变成了因特网的必备组件。

html/css/javascript之间的关系是怎样的呢？首先它们都可以采用普通的文本编辑器开发，然后使用浏览器软件打开并解释执行；html是一种超文本标记语言，主要是做页面展示，W3C制定了html规范与标准；css是一种层叠样式表语言，主要用来辅助修饰html，让html网页看起来更加的丰富与漂亮；javascript是一种编程语言，它可以用来操作html中的节点以及css样式，进而达到网页的动态效果，增强网页与用户的交互度。

。

## JavaScript和Java

JavaScript是网景公司的布兰登艾奇开发的，Java是SUN公司詹姆斯高斯林带领团队开发的，JavaScript程序被内置在浏览器软件当中的JavaScript解释器解释执行，而Java语言必须运行在Java虚拟机当中，所以JavaScript和Java没有任何关系。只是名字当中都带有Java字眼，这也许只是一种市场营销策略。

## JS和JSP

JavaScript被程序员简称为JS，运行在浏览器客户端当中；JavaServer Pages简称为JSP，是一种基于Java语言实现的服务器端页面，JSP属于Java语言，运行在Java虚拟机当中。

## JavaScript语言特点

1.基于对象的编程语言

JavaScript是一种基于对象的编程语言而不能说是面向对象的编程语言，因为对象性的特征在JavaScript中并不像Java语言中那样纯正。在JavaScript中有内置的对象，同时用户也可以创建并使用自己的对象。

弱类型语言

JavaScript编程语言没有编译阶段，文本编辑器开发完毕之后，直接使用浏览器打开即可解释执行，所以JavaScript中的变量在定义的时候不需要指定数据类型，并且变量赋值的时候可以随意赋值，具体赋值的时候，值决定变量的数据类型。JavaScript在变量约束方面非常弱，所以称为弱类型语言。Java语言存在编译阶段，在Java语言当中声明变量i为int类型，那么变量i只能接收int类型的字面值，如果字面值的数据类型和变量的数据类型不同，程序是无法编译通过的，并且从int类型的变量i开始声明到最终程序执行结束，i变量的数据类型永远都是int类型，占用4个字节，像Java这种对变量约束较强的编程语言被称为强类型编程语言。

2.解释执行的脚本语言

JavaScript是一种脚本语言，脚本语言是一种解释性的语言，以普通文本形式保存，不需要编译生成目标程序，可以直接用文本编辑器打开查看。它不象c\c++等可以编译成二进制代码以可执行文件的形式存在。JavaScript脚本语言可直接使用JavaScript解释器解释执行，可能在解释的过程当中进行编译。

3.简单性

JavaScript基于Java的基本语法和语句流程，而Java是从C++语言发展而来，因此有过C语言开发经验的人员学习JavaScript十分容易。此外，JavaScript是一种弱类型语言，其变量并没有严格的数据类型，免去了许多麻烦。

4.动态性

JavaScript是基于事件驱动的，所谓事件驱动就是触发一定的操作而引起某些动作。例如，鼠标单击按钮，页面加载完毕等等这些都是事件。可以根据不同的事件创建相应的响应代码，这样就可以实现和用户的动态交互。

5.平台无关性

前面提到JavaScript代码在浏览器中解释执行，并没有利用具体平台的特性，所以只要有支持JavaScript的浏览器，无论在什么平台上代码都能得到执行。开发人员在编写JavaScript脚本过程中就无需考虑具体平台的限制。只需要考虑浏览器兼容问题即可。

6.安全性

JavaScript是安全的，其不允许访问本地硬盘，也不能将数据存入到服务器上，不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互。从而有效地防止数据的丢失和破坏。

## JavaScript组成

1. ECMAScript

Netscape（网景）公司推出JavaScript不久，不甘落后的微软也推出了微软版的JavaScript-JScript，微软主要将其用于自己的浏览器Internet Explorer。最初两家公司各自为政，分别执行自己的标准。那个时候的web程序员是很痛苦的，通常需要将相同的功能编写两份不同语法的程序。

1999年，欧洲计算机协会（ECMA）依据JavaScript制定了ECMAScript的ECMA-262规范，ECMAScript成为JavaScript的标准。之后两家公司都遵循了该规范，所以JavaScript和Jscript大部分是相同的，但也有小部分不同之处。

2.DOM

Document Object Model（文档对象模型），在JavaScript当中，把html文档当做一棵树，这棵树有一个根节点<html></html>，根节点下有很多子节点，每一个节点都可以看做是一个DOM对象，JavaScript通过操作DOM对象完成重构整个HTML文档、添加、移除、改变或重排页面上的项目。从而达到网页动态效果，提高与用户的交互度。

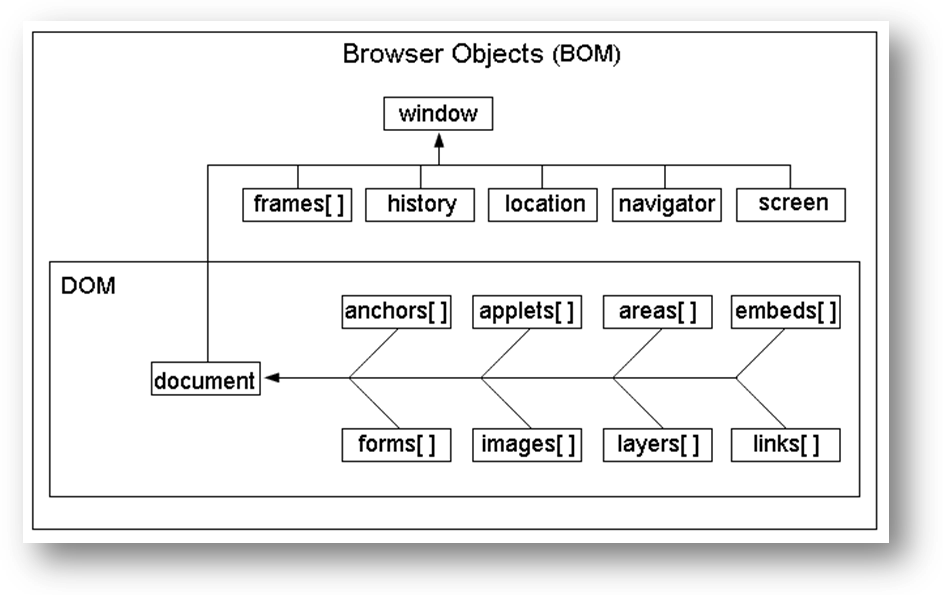
JavaScript对DOM元素的操作又可以称为“html dom编程”。HTML DOM是HTML Document Object Model(文档对象模型)的缩写，HTML DOM则是专门适用于HTML/XHTML的文档对象模型。熟悉软件开发的人员可以将HTML DOM理解为网页的API。它将网页中的各个元素都看作一个个对象，从而使网页中的元素也可以被计算机语言获取或者编辑。 例如Javascript就可以利用HTML DOM动态地修改网页。

根据W3C DOM规范,DOM是一种与浏览器，平台，语言无关的接口，使得你可以访问页面其他的标准组件。简单理解，DOM解决了Netscape的JavaScript和 Microsoft的JavaScript之间的冲突，给予web设计师和开发者一个标准的方法，让他们来访问他们站点中的数据、脚本和表现层对象。

3.BOM

Browser Object Model（浏览器对象模型），使JavaScript能够与浏览器进行“对话”，获取浏览器信息，操作浏览器。虽然W3C并没有对BOM作出规范，但是所有浏览器都支持BOM，有一些事实上的标准。

## BOM和DOM之间的关系



## 主流的浏览器

JavaScript脚本语言最终被浏览器当中的JS解释器解释执行，所以不同的浏览器执行相同的JS代码可能结果不尽相同，所以对于WEB前端开发人员来说，解决浏览器兼容问题是一件很头疼的事情。通常一个WEB前端开发人员需要在自己的电脑中安装不同厂商的浏览器，有可能相同厂商的浏览器也需要安装不同的版本。下面我们来看一看世界主流的5大浏览器都有谁？



左上角位置：谷歌公司的Chrome浏览器。

左下角位置：safari浏览器，苹果公司旗下浏览器。

右上角位置：微软公司的IE浏览器。

右下角位置：FireFox浏览器，简称FF浏览器，mozilla公司旗下浏览器

中间位置：opera浏览器，挪威厂商opera旗下浏览器。

我们在讲课的过程中安装的浏览器有：FF/Chrome/IE。

## JavaScript开发工具

1. 文本编辑器EditPlus

我们讲课的过程中，第一天的知识点采用普通的文本编辑器开发，这种开发方式还是比较适合JavaScript初学者的。建议初学者开始的时候使用文本编辑器。

1. 集成开发工具WebStorm

在实际的Web前端开发过程中，为了提高开发效率，可以借助某些集成开发环境，例如Web前端开发利器：WebStorm，这个名字就够气派的吧：Web风暴。该集成开发工具是Intellij IDEA开发工具中的一个插件，Intellij IDEA开发工具是一个非常棒的Java开发环境。我们在后期的课程中将使用WebStorm进行开发。

使用WebStorm将会体验到丰富的JS代码提示功能。它不仅仅对JS代码提供了丰富了提示，包括HTML、CSS提示也是非常全面的。

1. JavaScript帮助文档

<http://www.w3school.com.cn/> 全球最大的中文 Web 技术教程。我们给大家提供的帮助文档是w3shool的离线帮助文档

# JavaScript基础语法

## HTML嵌入JavaScript脚本

在<head>标签中使用<script>脚本标签

|  |
| --- |
| <html>  <head>  <title>HTML页面嵌入JS\_01</title>  <script language="javascript">  window.alert("Say Hello");  alert("Hello World!");  </script>  </head>  <body>  </body>  </html> |

## 在HTML页面中引入外部js文件

|  |
| --- |
| 新建1.js文件，在该文件中编写如下程序：  alert("hello world!"); |
| <html>  <head>  <title>HTML页面嵌入JS\_02</title>  **<script language="javascript" src="1.js"></script>**  **<!--错误的写法：-->**  **<script language="javascript" src="1.js" />**  </head>  <body>  </body>  </html> |

## 直接嵌入到HTML标签中

|  |
| --- |
| <html>  <head>  <title>HTML页面嵌入JS\_03</title>  </head>  <body>  **<input type="button" value="SayHello" onclick="javascript:alert('Hello1');alert('Hello2');"/>**  </body>  </html> |

## JavaScript注释

* + **单行注释**

|  |
| --- |
| <html>  <head>  <title>JS\_注释\_01</title>  **<script>**  **//alert方法是内置对象window的方法，通常通过该方法测试JS程序**  **alert("警告信息"); //弹出警告框**  **</script>**  </head>  <body>  </body>  </html> |

* **多行注释**

|  |
| --- |
| <html>  <head>  <title>JS\_注释\_02</title>  **<script>**  **/\***  **alert方法是内置对象window的方法，通常通过该方法测试JS程序**  **弹出警告框**  **\*/**  **alert("警告信息");**  **</script>**  </head>  <body>  </body>  </html> |

## JavaScript标识符

* + 标识符可以标识变量名、属性名、数组名、函数名等。
  + 第一个字符必须是字母、下划线(\_)或美元符号($)，其他字符可以使用字母、数字、下划线或美元符号。
  + 标识符不能和JavaScript的关键字同名。
  + JavaScript中标识符是区分大小写的。

## JavaScript关键字

abstract, boolean, break, byte, case, catch, char, class, const, continue, debugger, default, delete, do, double, else, enum, export, extends, false, final, finally, float, for, function, goto, if, implements, import, in, instanceof, int , interface, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, transient, true, typeof, var, volatile, void, while, with

这些不必刻意去记，先知道这么个情况。因为后面很多关键字你会用到，你会明白它所代表的实际意义，Javascript不允许使用关键字来命名变量或参数，也不能使用关键字来作为对象的属性名称。因为这些关键字有特定的含义，属于专有用语。另外，还有一些字符也不要使用，虽然没有明确作为关键字，但它们也是有特定的含义， undefined，NaN，Infinity这三个会经常见到。

## JavaScript变量

变量在程序中是存储数据的容器，因为javascript是一种脚本语言，不需要编译，直接使用浏览器软件中的JS解释器解释执行。JavaScript是一种弱类型的语言，不像java语言在程序编译阶段就确定变量的数据类型，例如int age = 10;在程序编译阶段age变量的类型就被确定为int类型。但是javascript是不需要编译的，直接解释执行，所以在编写javascript程序的时候不需要指定变量的类型，程序运行期确定变量类型。在javascript中定义变量使用如下语法：

|  |
| --- |
| **var** age = 10;  var name = “jack”; //字符串双引号可以  var ename = ‘KING’; //字符串单引号也可以  var empno=7369,sal=800,deptno=10; //一行上可以同时定义多个变量  var job; //先声明  job = “CEO”; //再赋值  var deptname; //如果只声明没有赋值则是undefined |

每一个变量都是有作用域的，所谓作用域就是变量的作用范围，在哪个范围是有效的，根据变量出现的位置可以分为全局变量和局部变量。

|  |
| --- |
| <html>  <head>  <title>var\_01</title>  <script language="javascript">  var ename1 = "SMITH";//全局变量  function testVar1(){//这是一个函数  var ename2 = "KING";//局部变量  alert(ename1); //SMITH  alert(ename2); //KING  }  testVar1();//调用函数  alert(ename1);  //alert(ename2); //无法访问  function testVar2(){  var ename1 = "FORD";  alert(ename1); //就近原则  dname = "ACCOUNT"; //不带有var关键字的变量一定是全局变量  }  testVar2();  alert(dname);  </script>  </head>  </html> |

## JavaScript函数初步

函数function类似于java语言中的方法Method，也是一段可以完成特定功能的代码片段，并且这段代码片段是可以重复利用的。在javascript中定义函数的语法：

|  |
| --- |
| 定义函数：  **第一种方式：**  function 函数名(形式参数列表){ //形参是局部变量  函数体由js语句构成;  }  调用函数：  函数名(实参);  **第二种方式：**  函数名 = function(形式参数列表){  函数体由js语句构成;  } |

.JS语法是弱类型的，所以不需要指定函数的返回值类型。但是JS的函数仍然有返回值。是否返回值取决于程序员的代码决定。包括函数的形参类型也不需要指定。

JS中没有函数重载的概念，在同一个script标签中，函数名不能重名。

示例：

|  |
| --- |
| <html>  <head>  <title>function\_01</title>  <script language="javascript">  function sayHello(ename){ //定义函数  window.document.write("<h1>Hello&nbsp;&nbsp;"+ename+"</h1>");  }  sayHello("JACK"); //调用函数  function getUser(id,name){//定义函数  return id + "-->" + name;  }  document.write("<center>"+getUser(77,"JACK")+"</center>");  alert(getUser(45)); **//参数个数不匹配也可以调用,参数多或少都可以**  </script>  </head>  </html> |

示例：

|  |
| --- |
| <html>  <head>  <title>JS中函数初步</title>  <script language="javascript">  function sayHello(username){  //alert(username + "，你好！");  //document是JS中的DOM的顶层对象  //document内置对象中有一个write方法，可以向浏览器输出HTML代码。  //document.write(username + "，你好！");  document.write("<center><font color='blue' size='7'>" + username + "，你好！</font></center>");  }  //函数还可以这样定义  /\*  sayHello = function(username){  alert(username + "，你好！");  }  \*/  //调用函数  //sayHello("李四");  </script>  </head>  <body>  <form name="form1">  <input type="text" name="username"/>  <!--  要获取form表单中的一个标签的value？ form的名字.标签的名字.value  -->  <inputtype="button"value="sayHello"onclick="sayHello(form1.username.value);"/>  </form>  </body>  </html> |

示例：

|  |
| --- |
| **<**html>  <head>  <title>JS中函数初步</title>  <script language="javascript">    //JS中的函数也可以有返回值，也可以没有。  //JS函数最后可以编写return语句，也可以不写。  function validateName(username){  //字符串length属性用来获取字符串的长度。  if(username.length<6){  return false;  }  return true;  }  //alert(validateName("jack123123")?"合法":"不合法");    /\*  var flag = validateName("jack");  if(flag){  alert("合法");  }else{  alert("不合法");  }  \*/  </script>  </head>  <body>  <!--  失去焦点事件：blur  失去焦点事件句柄：onblur  以下程序中的this表示当前对象，当前对象是input框。  this.value是input框的值。  -->  <input  type="text"  name="username"  **onblur="alert(validateName(this.value)?'合法':'不合法');"** />  </body>  </html> |

## JavaScript数据类型

JavaScript中有5种数据类型：4种原始类型（基本数据类型）Undefined、Boolean、Number、String。1种引用数据类型—Object，自定义的函数也是一种引用数据类型，可以看做Object类型的子类型。ES6规范中又添加了一种新的类型：Symbol

　　JavaScript是松散类型的，因此需要有一种手段来检测给定变量的数据类型，typeof运算符就是负责提供这方面信息的操作符。对一个值使用typeof操作符可能返回下列某个字符串：

* "undefined"——如果这个值未定义；
* "boolean"——如果这个值是布尔值；
* "string"——如果这个值是字符串；
* "number"——如果这个值是数值；
* "object"——如果这个值是对象或null；
* "function"——如果这个值是函数；

语法有三种写法：

1.typeof 变量名

2.typeof(变量名)

3.typeof 函数名

示例：

|  |
| --- |
| <html>  <head>  <title>js中的数据类型</title>  <script language="javascript">    /\*  回顾java语言中的数据类型：  基本数据类型：byte,short,int,long,float,double,boolean,char  引用数据类型：Object...    Javascript中的数据类型：  原始数据类型：Undefined,Boolean,Number,String  引用数据类型：Object  注意：自定义的函数，其实也是一种类型，  这个类默认继承Object类型。所以Function  也属于引用数据类型。    JS中的数据类型的类型名就是一个摆设，没用：  String s = "abc"; //Error  //js中变量的声明不需要指定数据类型，是一种弱类型的语言  //JS程序因为是不需要编译的，直接运行，所有变量的数据类型  //在程序运行阶段确定。  var s = "abc";  s = false;    关于 typeof 运算符：  语法有三种写法：  1.typeof 变量名  2.typeof(变量名)  3.typeof 函数名    typeof运算符的运算结果一定是“String”类型，并且只有以下6种结果：  "undefined"  "boolean"  "string"  "number"  "object"  "function"  typeof运算符主要存在的意义是判断变量的数据类型。  \*/  var i; //系统会默认赋值 undefined, undefined是一个值，属于Undefined类型。  alert(typeof(i) == "undefined"); //true  alert(typeof(i) == undefined); //false  if(typeof i == "undefined"){  i = 100;  alert(i);  }  var v = '中国';  if(typeof v == "number"){  alert(v + 10);  }else{  alert("它不是一个数字！");  }    //Object类是一种引用数据类型  //Object是JS中内置的。  //Object既是一个函数的名字，也是一个类名。  var o = new Object();  alert(typeof(o)); //"object"    //User类中有一个属性username  User = function(a){  this.username = a;  }  o = new User("SMITH");  alert(typeof(o)); //"object"  alert(typeof(User)); //"function"  </script>  </head>  <body>  </body>  </html> |

1. Undefined类型

　　Undefined类型只有一个值，即特殊的undefined。在使用var声明变量但未对其加以初始化时，这个变量的值就是undefined，例如：

var message;

alert(message == undefined) //true

示例：

|  |
| --- |
| **<**html>  <head>  <title>Undefined类型</title>  <script language="javascript">    /\*  Undefined类型只有1个值，这个值就是undefined  如果一个变量只声明，没有手动赋值，则系统默认赋值undefined  \*/  var i;  var i1 = undefined;  alert(i); //alert方法会使用()中的非字符串类型转换成字符串类型。  alert(i1);//"undefined"  alert( i == i1); //true  </script>  </head>  </html> |

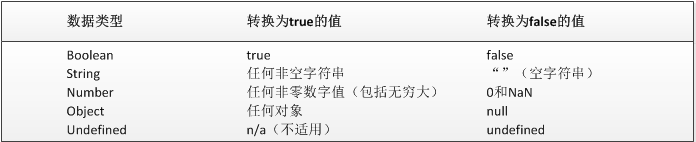
1. Boolean类型

　　该类型只有两个字面值：true和false。这两个值与数字值不是一回事，因此true不一定等于1，而false也不一定等于0。

　　虽然Boolean类型的字面值只有两个，但JavaScript中所有类型的值都有与这两个Boolean值等价的值。要将一个值转换为其对应的Boolean值，可以调用类型转换函数Boolean()，例如：

var message = 'Hello World';

var messageAsBoolean = Boolean(message);

在这个例子中，字符串message被转换成了一个Boolean值，该值被保存在messageAsBoolean变量中。可以对任何数据类型的值调用Boolean()函数，而且总会返回一个Boolean值。至于返回的这个值是true还是false，取决于要转换值的数据类型及其实际值。下表给出了各种数据类型及其对象的转换规则。

  这些转换规则对理解流控制语句（如if语句）自动执行相应的Boolean转换非常重要，例如：

var message = 'Hello World';

if(message)

{

alert("Value is true");

}

运行这个示例，就会显示一个警告框，因为字符串message被自动转换成了对应的Boolean值（true）。由于存在这种自动执行的Boolean转换，因此确切地知道在流控制语句中使用的是什么变量至关重要。

示例：

|  |
| --- |
| **<**html>  <head>  <title>Boolean类型</title>  <script language="javascript">  /\*  Boolean类型只有两个值：true、false  Boolean类型提供一个Boolean(数据)函数.  Boolean(数据) 函数：  1.该函数的作用是：将非Boolean类型转换成Boolean类型。  2.该函数的运算结果一定是：true、false  3.转换规则：  3.1 所有的“非""字符串”都被转换成true  所有的 "" 被转换成 false  3.2 任何非0的数字包括无穷大(Infinity)都被转换成true  0和 NaN 都被转换成false  3.3 任何“非null”对象都被转换成 true  null 被转换成false  3.4 undefined 是一个值，被转换成 false  \*/  var v1; //undefined  alert(Boolean(v1)); //false  v1 = "abc"; //非"" 字符串  alert(Boolean(v1)); //true  v1 = ""; //""字符串  alert(Boolean(v1)); //false  v1 = 0; //0  alert(Boolean(v1)); //false  v1 = NaN;  alert(Boolean(v1)); //false  v1 = Infinity; //无穷大  alert(Boolean(v1)); //true  v1 = null;  alert(Boolean(v1)); //false  v1 = new Object();  alert(Boolean(v1)); //true  //关于if语句 和java语法结果相同。区别在于if(这里的数据如果不是Boolean  //类型，系统会自动调用Boolean函数进行类型的转换)  var a = "ABCDEF";  if(a){  alert(a);  }else{  alert("空字符串");  }  </script>  </head>  </html> |

1. Number类型

　　这种类型用来表示整数和浮点数值，还有一种特殊的数值，即NaN（非数值 Not a Number）。这个数值用于表示一个本来要返回数值的操作数未返回数值的情况。

　　NaN本身有两个非同寻常的特点。首先，任何涉及NaN的操作（例如NaN/10）都会返回NaN，这个特点在多步计算中有可能导致问题。其次，NaN与任何值都不相等，包括NaN本身。例如，下面的代码会返回false。

alert(NaN == NaN); //false

　　JavaScript中有一个isNaN()函数，这个函数接受一个参数，该参数可以使任何类型，而函数会帮我们确定这个参数是否“不是数值”。isNaN()在接收一个值之后，会尝试将这个值转换为数值。某些不是数值的值会直接转换为数值，例如字符串”10“或Boolean值。而任何不能被转换为数值的值都会导致这个函数返回true。例如：

alert(isNaN(NaN)); //true

alert(isNaN(10)); //false(10是一个数值)

alert(isNaN("10")); //false(可能被转换为数值10)

alert(isNaN("blue")); //true(不能被转换为数值)

alert(isNaN(true)); //false(可能被转换为数值1)

示例：

|  |
| --- |
| **<**html>  <head>  <title>Number类型</title>  <script language="javascript">  /\*  Number类型包括以下值：  整数(0,1,2,3.....)  浮点数(1.0,2.0,3.0.....)  NaN (这就是一个值，属于Number类型，表示“不是一个数字”，Not a Number)  Infinity (这就是一个值，属于Number类型，表示无穷大)  \*/  //整数  var v1 = 0;  var v2 = 100;  alert(v2);  alert(typeof v2);  //浮点数  var v3 = 3.1;  alert(v3);  alert(typeof v3);  //Not a Number  var v4 = NaN;  alert(v4);  alert(typeof v4);    var v5 = 100/"中国"; //运算结果本该是数字却没有返回数字的，则返回NaN  alert(v5);  alert(typeof v5);  //Infinity  var v6 = 10/0; //除数是0的时候运算结果是无穷大Infinity。  alert(v6);  alert(typeof v6);  //js中是：3.333333  alert(10/3); //在java中是 3    **/\***  **重点：isNaN函数**  **语法：isNaN(变量) 判断该变量是否是一个数字。**  **“is not a number”**  **isNaN的运算结果是true、false**  **true：不是一个数字**  **false：是一个数字**  **\*/**  var ename = "JACK";  if(isNaN(ename)){  alert(ename + "不是一个数字");  }  var x = NaN;  var y = NaN;  //NaN和任何数据都不相等，和自己也不相等。  alert(NaN==NaN); //false  alert(isNaN(NaN)); //true  </script>  </head>  </html> |

示例：

|  |
| --- |
| **<**html>  <head>  <title>Number类型</title>  <script language="javascript">  //isNaN  function checkAge(age){    var ageError = document.getElementById("ageError");  **// 通过isNaN()判断用户输入的数据是否是一个数值**  if(isNaN(age)){  //alert("年龄必须是数字");  ageError.innerHTML = "<font color='red'>年龄必须是数字</font>";  return;  }  if(age<18 || age>50){  //alert("年龄不在指定范围中");  ageError.innerHTML = "年龄不在指定范围中";  return;  }  ageError.innerHTML = "";  }  function clearError(){  var ageError = document.getElementById("ageError");  ageError.innerHTML = "";  }  </script>  </head>  <body>  <!--  blur是失去焦点 onblur是失去焦点句柄  -->  <input  type="text"  name="userage"  onblur="checkAge(this.value);"  onfocus="clearError();"/>  <!--该图层负责显示错误信息,span不是独占行的-->  <span id="ageError"></span>  </body>  </html> |

　　有3个函数可以把非数值转换为数值：Number()、parseInt()和parseFloat()。**第一个函数，即转型函数Number()可以用于任何数据类型**，而另外两个函数则专门用于把字符串转换成数值。这3个函数对于同样的输入会返回不同的结果。

　　Number()函数的转换规则如下：

* 如果是Boolean值，true和false将分别被替换为1和0
* 如果是数字值，只是简单的传入和返回
* 如果是null值，返回0
* 如果是undefined，返回NaN
* 如果是字符串，遵循下列规则：
  + - 如果字符串中只包含数字，则将其转换为十进制数值，即”1“会变成1，”123“会变成123，而”011“会变成11（前导的0被忽略）
    - 如果字符串中包含有效的浮点格式，如”1.1“，则将其转换为对应的浮点数（同样，也会忽略前导0）
    - 如果字符串中包含有效的十六进制格式，例如”0xf“，则将其转换为相同大小的十进制整数值
    - 如果字符串是空的，则将其转换为0
    - 如果字符串中包含除了上述格式之外的字符，则将其转换为NaN
* 如果是对象，则调用对象的valueOf()方法，然后依照前面的规则转换返回的值。如果转换的结果是NaN，则调用对象的toString()方法，然后再依次按照前面的规则转换返回的字符串值。

var num1 = Number("Hello World"); //NaN

var num2 = Number(""); //0

var num3 = Number("000011"); //11

var num4 = Number(true); //1

　　由于Number()函数在转换字符串时比较复杂而且不够合理，**因此在处理整数的时候更常用的是parseInt()函数**。**parseInt()函数在转换字符串时，更多的是看其是否符合数值模式。**它会忽略字符串前面的空格，直至找到第一个非空格字符。如果第一个字符串不是数字字符或者负号，parseInt()会返回NaN；也就是说，用parseInt()转换空字符串会返回NaN。如果第一个字符是数字字符，praseInt()会继续解析第二个字符，直到解析完所有后续字符或者遇到了一个非数字字符。例如，"1234blue"会被转换为1234，”22.5“会被转换为22，因为小数点并不是有效的数字字符。

　　如果字符串中的第一个字符是数字字符，parseInt()也能够识别出各种整数格式（即十进制、八进制、十六进制）。为了更好的理解parseInt()函数的转换规则，下面给出一些例子

var num1 = parseInt("1234blue"); //1234

var num2 = parseInt(""); //NaN

var num3 = parseInt("0xA"); //10（十六进制）

var num4 = parseInt("22.5"); //22

var num5 = parseInt(""070); //56（八进制）

var num6 = parseInt("70"); //70

parsInt()可以把一个string按照同进制进行解析

var num7 = parseInt("10",”2”); //2（按二进制解析） 2也可以使用”2”

var num8 = parseInt("10",8); //8(按八进制解析)

var num9 = parseInt("10",10); //10（按十进制解析）

var num10 = parseInt("10",16); //16（按十六进制解析）

var num12 = parseInt("AF",16); //175

　　与parseInt()函数类似，parseFloat()也是从第一个字符（位置0）开始解析每个字符。而且也是一直解析到字符串末尾，或者解析到遇见一个无效的浮点数字字符为止。也就是说，字符串中的第一个小数点是有效的，而第二个小数点就是无效的了，因此它后面的字符串将被忽略。例如，”22.34.5“将会被转换成22.34。

**parseFloat()和parseInt()的第二个区别在于parseFloat始终都会忽略前导的零。由于parseFloat()只能解析十进制值，因此它没有用第二个参数指定基数的用法。**

var num1 = parseFloat("1234blue"); //1234

var num2 = parseFloat("0xA"); //0

var num3 = parseFloat("22.5"); //22.5

var num4 = parseFloat("22.34.5"); //22.34

var num5 = parseFloat("0908.5"); //908.5

1. String类型

　　String类型用于表示由零或多个16位Unicode字符组成的字符序列，即字符串。字符串可以由单引号(')或双引号(")表示。

var str1 = "Hello";

var str2 = 'Hello';

　　任何字符串的长度都可以通过访问其length属性取得

　　alert(str1.length); //输出5

　　要把一个值转换为一个字符串有两种方式。第一种是使用几乎每个值都有的toString()方法。

var age = 11;

var ageAsString = age.toString(); //字符串"11"

var found = true;

var foundAsString = found.toString(); //字符串"true"

　　数值、布尔值、对象和字符串值都有toString()方法。但null和undefined值没有这个方法。

　　多数情况下，调用toString()方法不必传递参数。但是，在调用数值的toString()方法时，可以传递一个参数：输出数值的基数。

var num = 10;

alert(num.toString()); //"10"

alert(num.toString(2)); //"1010" ; 2 可以是数值,也可以是sring格式的”2”

alert(num.toString(8)); //"12"

alert(num.toString(10)); //"10"

alert(num.toString(16)); //"a"

　　通过这个例子可以看出，通过指定基数，toString()方法会改变输出的值。而数值10根据基数的不同，可以在输出时被转换为不同的数值格式。

　　在不知道要转换的值是不是null或undefined的情况下，还可以使用转型函数String()，这个函数能够将任何类型的值转换为字符串。String()函数遵循下列转换规则：

* 如果值有toString()方法，则调用该方法（没有参数）并返回相应的结果
* 如果值是null，则返回"null"
* 如果值是undefined，则返回”undefined“

var value1 = 10;

var value2 = true;

var value3 = null;

var value4;

alert(String(value1)); //"10"

alert(String(value2)); //"true"

alert(String(value3)); //"null"

alert(String(value4)); //"undefined"

示例：

|  |
| --- |
| **<**html>  <head>  <title>String类型</title>  <script language="javascript">  **/\***  **1.js中的字符串可以使用**  **'' ; "" 都可以。**  **2. js中的字符串有length属性用来获取字符串的长度**  **3. js 中的字符串有 prototype属性，用来给字符串类型动态扩展属性和方法**  **\*/**  var s1 = 'ABC';  alert(s1.length);  s1 = "AAAA";  alert(s1.length);    **//Error:这里不能执行。**  **//s1.trim();**  **//给字符串类多态扩展属性和方法，该属性和方法可以作用到所有字符串对象上。**  String.prototype.trim = function(){  return this.replace(/^\s+/,"").replace(/\s+$/,"");  }  //调用  var s1 = " hello "  s1 =s1.trim();  alert(s1 + " ,长度:" + s1.length);  </script>  </head>  <body>  </body>  </html> |

1. Object类型

　　对象其实就是一组数据和功能的集合。对象可以通过执行new操作符后跟要创建的对象类型的名称来创建。而创建Object类型的实例并为其添加属性和（或）方法，就可以创建自定义对象。

　　var o = new Object();

　　Object的每个实例都具有下列属性和方法：

* + - constructor——保存着用于创建当前对象的函数
    - **prototype--------给类型动态扩展属性（类级别），将作用到所有对象上。（类名.）方式访问**
    - hasOwnProperty(propertyName)——用于检查给定的属性在当前对象实例中（而不是在实例的原型中）是否存在。其中，作为参数的属性名(propertyName)必须以字符串形式指定（例如：o.hasOwnProperty("name")）
    - isPrototypeOf(object)——用于检查传入的对象是否是另一个对象的原型
    - propertyIsEnumerable(propertyName)——用于检查给定的属性是否能够使用for-in语句来枚举
    - toString()——返回对象的字符串表示
    - valueOf()——返回对象的字符串、数值或布尔值表示。通常与toString()方法的返回值相同。

示例：

|  |
| --- |
| **<**html>  <head>  <title>Object类型</title>  <script language="javascript">  **/\***  **Object类型是JS中引用数据类型的顶层类型（根类型）。**  **自定义的函数也是一种类型，默认继承Object类。**  **Object类是JS中内置的。**  **\*/**  **//创建Object对象**  var o1 = new Object();    **//JS中的对象都有toString方法**  alert(o1);  alert(o1.toString());  **//关于constructor属性，用来获取创建该对象的构造函数**  alert(o1.constructor);    **//关于prototype属性，用来给类型动态扩展属性以及方法**  Emp = function(ename,sal){  this.ename = ename;  this.sal = sal;  /\*\*/  this.toString = function(){  return this.ename + "--->" + this.sal;  }  }  var e1 = new Emp("KING",5000);    //Error  //e1.printInfo();    //动态扩展  Emp.prototype.printInfo = function(){  document.write(this.ename + "," + this.sal);  }  //调用方法  e1.printInfo();  alert(e1);  alert(e1.toString());  alert(e1.constructor);  </script>  </head>  <body>  </body>  </html> |

## void让超链接失效

语法：

javascript:void(表达式);

javascript:void 表达式 //括号可以省略

|  |
| --- |
| <a href=”javascript:void(0);” >我虽然是一个超级链接，但是点击我没有任何效果</呢个a> |
| <a href=”javascript:void(document.form.submit())”>点击我可以提交网页中的表单</a> |

javascript:void(0);的实际应用

说白了，href="#"这种形式会整体刷新页面，而 href="javascript:void(0)" 则不会

|  |
| --- |
| <a href="javascript:void(0);"  onclick="alert('Hello World!');">点击我执行一段程序</a>  <a href="#">点我看看有什么效果<a> |

## null、NaN和undefined的区别

|  |
| --- |
| var a1; var a2 = true; var a3 = 1; var a4 = "Hello"; var a5 = new Object(); var a6 = null; var a7 = Na2N; var a8 = undefined;  alert(typeof a);   //显示"undefined" alert(typeof a1); //显示"undefined" alert(typeof a2); //显示"boolean" alert(typeof a3); //显示"number" alert(typeof a4); //显示"string" alert(typeof a5); //显示"object" alert(typeof a6); //显示"object" alert(typeof a7); //显示"number" alert(typeof a8); //显示"undefined" |

从上面的代码中可以看出未定义的值和定义未赋值的为undefined，null是一种特殊的object, NaN是一种特殊的number。

比较运算

|  |
| --- |
| var a1;        //a1的值为undefined var a2 = null; var a3 = NaN;  alert(a1 == a2);**//显示"true" null和undefined 可以被认为相等** alert(a1 != a2); //显示"false"  **//null和undefined类型不同。**  alert(null === undefined); //false  alert(a1 == a3); //显示"false" alert(a1 != a3); //显示"true"  alert(a2 == a3); //显示"false" alert(a2 != a3); //显示"true"  alert(a3 == a3); //显示"false" alert(a3 != a3); //显示"true" |

从上面的代码可以得出结论：（1）undefined与null是相等；（2）NaN与任何值都不相等，与自己也不相等。

## JavaScript控制语句

* if 语句
* switch 语句
* do...while 语句
* while 语句
* for 语句
* for...in 语句
* break 和continue 语句
* **for…in 语句**

for...in 语句是一种精准的迭代语句，可以用来枚举对象的属性

|  |
| --- |
| function Product(pno,pname,price){  this.pno = pno;  this.pname = pname;  this.price = price;  }  var p = new Product(100,"西瓜",2.5);  for(var propertyName in p){  alert(propertyName); // 取出属性的名称  alert(p[propertyName]); 取出该属性对应的值  } |

使用for…in语句迭代数组

|  |
| --- |
| <script language="javascript">  var colors = ["红色","绿色","蓝色"];  // index是数组的索引  for(var index in colors){  alert(colors[index]);  }  </script> |

## JavaScript常用内置类

1. Array

创建数组对象：

var a1 = new Array(3);

var a2 = new Array(“A”,”B”,100);

var a3 = [“D”,”E”,90];

常用属性：

length 获取数组的长度

prototype 动态给数组对象添加属性或者方法

常用方法：

concat(另一个数组) 连接两个数组成一个新数组.

join(特殊字符) 将数组中所有的元素以特殊字符连接起来生成一个字符串.

pop() 弹出数组中最后一个元素.

push(ele1,ele2….) 向数组末尾添加多个元素

reverse() 反转数组.

slice(从哪个下标开始) 从指定下标开始向右截取子数组.

splice(从哪个下标开始,元素个数,newEle1,newEle2…). 添加或者删除数组中某些元素

sort 对数组升序排.

toString 将数组转换成字符串.

1. Date

创建日期对象：

var t1 = new Date();

var t2 = new Date(y,m,d);

var t3 = new Date(y,m,d,h,mi,s);

属性：

prototype 日期对象动态扩展属性和方法

常用方法：

getDate 获取日期的“几号”.

getDay 获取日期是星期几.

getHours 获取日期的小时

getMinutes 获取日期的分钟.

getMonth 获取日期的月份.

getSeconds 获取日期的秒数.

getTime 获取自1970年1月1日到指定时间的毫秒数.

|  |
| --- |
| var nowTime = new Date();  var timeStamp = nowTime.getTime();  获取时间戳，在AJAX中解决浏览器缓存问题。 |

*getYear 获取日期的年*. （很少使用）

getFullYear 获取日期的完整年份（推荐使用，避免千年虫问题）

setDate 修改日期的“几号”.

setHours 修改日期的“小时”.

setMinutes 修改日期的“分钟”.

setMonth 修改日期的“月份”.

setSeconds 修改日期的“秒”.

setTime 修改时间，该方法参数是毫秒数（自1970年1月1日00:00:00的毫秒数）.

*setYear 修改日期的年份*.

toGMTString 将日期转换成字符串（以GMT（格林威治时间）格式转换）.

toLocaleString 将日期转换成字符串，以本地计算机环境为基础。.

UTC 获取自1970年1月1日到指定时间的毫秒数（国际标准时间）.

|  |
| --- |
| Date.UTC(2012,1,1,0,0,0); |

# JavaScript事件及事件句柄

## 常见事件及事件句柄

1. click事件

鼠标单击事件，事件句柄：onclick

1. dblclick事件

鼠标双击事件，事件句柄：ondblclick

1. blur事件

失去焦点事件，事件句柄：onblur

1. change事件

当文本框、文本域中的文本内容发生变化以后并失去焦点会触发或者下拉列表选中项发生改变时，该事件发生，事件句柄：onchange

1. focus事件

获得焦点事件，事件句柄：onfocus

1. load事件

网页加载完毕后发生，事件句柄：onload，通常编写在body标签当中。

1. keydown事件

键盘按键被按下时发生，可以捕获所有键，除“prt sc sysrq”键之外。事件句柄：onkeydown

1. keypress事件

键盘按键被按下时发生，主要用来捕获数字、字母、小键盘，其它键无法捕获，但是在FF浏览器当中可以捕获所有键。事件句柄：onkeypress

1. keyup事件

键盘按键弹起时发生，事件句柄：onkeyup

1. mousedown事件

鼠标按下事件，事件句柄：onmousedown

1. mousemove事件

鼠标在某控件上移动时发生，事件句柄：onmousemove

1. mouseover事件

鼠标经过某控件时发生，事件句柄：onmouseover

1. mouseout事件

鼠标离开某控件时发生，事件句柄：onmouseout

1. mouseup事件

鼠标按键释放时发生，事件句柄：onmouseup

1. reset事件

表单数据被重置的时候发生，事件句柄：onreset

1. submit事件

表单数据被提交时发生，事件句柄：onsubmit

1. select事件

当文本框、文本域中的内容被选中时发生，事件句柄：onselect

1. abort事件

图像的加载被中断时发生，事件句柄：onabort

1. error事件

加载图像或文档出错时发生，事件句柄：onerror

1. resize事件

窗口或框架被重新调整大小时发生，事件句柄：onresize

1. unload事件

用户退出当前页面时发生，事件句柄：onunload

## 注册事件的另一种方式

文件名：事件注册.html



## 键盘事件控制只能输入数字

# 