**WEB编程基础**

**HTML语言**

**CSS语言**

**JAVASCRIPT语言**

**正则表达式**

第一篇 HTML语言基础

[HTML简介 5](#_Toc125595136)

[HTML起源和历史 6](#_Toc125595137)

[为什么需要学习HTML? 7](#_Toc125595138)

[第一个HTML程序 7](#_Toc125595139)

[HTML基本规范 8](#_Toc125595140)

[HTML标签 9](#_Toc125595141)

[常用标签 9](#_Toc125595142)

[<head>中常见子元素的含义 10](#_Toc125595143)

[标题标记<hx> 10](#_Toc125595144)

[文本标记<b><i><u><del> 10](#_Toc125595145)

[预格式文本<pre> 10](#_Toc125595146)

[字体标签<font> 11](#_Toc125595147)

[换行<br/>或者<br> 11](#_Toc125595148)

[水平线<hr> 11](#_Toc125595149)

[超链接<a> 11](#_Toc125595150)

[列表标签<ul><ol><li> 12](#_Toc125595151)

[图像标签<img> 12](#_Toc125595152)

[内嵌<embed> 12](#_Toc125595153)

[表格标签<table><tr><td><th> 13](#_Toc125595154)

[块元素<div> 14](#_Toc125595155)

[行内元素<span> 14](#_Toc125595156)

[框架标签<frameset><frame><iframe> 14](#_Toc125595157)

[表单标签 15](#_Toc125595158)

[HTML字符实体 17](#_Toc125595159)

[本章课下作业 17](#_Toc125595160)

[自学课题 17](#_Toc125595161)

第二篇 CSS语言

[什么是CSS? 18](#_Toc125595162)

[CSS的主要优势? 18](#_Toc125595163)

[java程序员需要了解css中哪些内容? 19](#_Toc125595164)

[样式规则 19](#_Toc125595165)

[样式规则的位置 19](#_Toc125595166)

[选择器 20](#_Toc125595167)

[元素选择器 20](#_Toc125595168)

[类选择器 20](#_Toc125595169)

[ID选择器 21](#_Toc125595170)

[包含选择器 21](#_Toc125595171)

[CSS属性单位 21](#_Toc125595172)

[文本属性 22](#_Toc125595173)

[背景属性 22](#_Toc125595174)

[定位属性 23](#_Toc125595175)

[CSS盒子模型（box model） 23](#_Toc125595176)

[布局属性 24](#_Toc125595177)

[列表属性 24](#_Toc125595178)

[鼠标样式 24](#_Toc125595179)

[样式就近原则 25](#_Toc125595180)

[CSS中的继承 25](#_Toc125595181)

[使用JavaScript控制css样式 25](#_Toc125595182)

[CSS工具集 25](#_Toc125595183)

[自学内容 25](#_Toc125595184)

第三篇 JAVASCRIPT语言

[客户端编程 26](#_Toc125595185)

[JavaScript简介 26](#_Toc125595186)

[Javascript历史和版本 26](#_Toc125595187)

[Javascript和java的区别联系 27](#_Toc125595188)

[如何在网页中使用脚本语言 28](#_Toc125595189)

[变量的声明及赋值 29](#_Toc125595190)

[表达式与运算符 33](#_Toc125595191)

[分支语句 35](#_Toc125595192)

[循环语句 35](#_Toc125595193)

[数组 36](#_Toc125595194)

[函数及深化 37](#_Toc125595195)

[内部函数 38](#_Toc125595196)

[事件机制 40](#_Toc125595197)

[Event对象 44](#_Toc125595198)

[对话框 50](#_Toc125595199)

[常见内置对象 51](#_Toc125595200)

[时间对象 51](#_Toc125595201)

[Math对象 52](#_Toc125595202)

[String对象 53](#_Toc125595203)

[对象机制 54](#_Toc125595204)

[JS中类的典型定义方式 54](#_Toc125595205)

[继承的实现 54](#_Toc125595206)

[JSON 56](#_Toc125595207)

[Aptana开发环境和firebug 57](#_Toc125595208)

[浏览器对象(BOM)的层次结构 57](#_Toc125595209)

[window对象 58](#_Toc125595210)

[History对象 59](#_Toc125595211)

[Navigator对象 59](#_Toc125595212)

[location 对象 60](#_Toc125595213)

[JAVASCRIPT操作DOM模型 61](#_Toc125595214)

[DOM模型介绍 61](#_Toc125595215)

[结点 61](#_Toc125595216)

[元素结点的引用 61](#_Toc125595217)

[浏览器差异问题 62](#_Toc125595218)

[结点的属性 62](#_Toc125595219)

[处理属性节点 62](#_Toc125595220)

[处理文本结点 63](#_Toc125595221)

[改变文档的层次结构 63](#_Toc125595222)

[Javascript操作CSS 64](#_Toc125595223)

[表单操作 65](#_Toc125595224)

[获取表单对象的方法 65](#_Toc125595225)

[Form对象的常用属性 65](#_Toc125595226)

[Form对象的方法 65](#_Toc125595227)

[Form对象的事件 66](#_Toc125595228)

[表单域通用方法 66](#_Toc125595229)

[文本域 (text, password, textarea ) 66](#_Toc125595230)

[单选按钮组和复选框常用操作 66](#_Toc125595231)

[下拉列表的使用 68](#_Toc125595232)

[表单验证操作 69](#_Toc125595233)

第四篇 正则表达式语言

[正则表达式基础(Regular Expression) 72](#_Toc125595234)

[正则表达式简介 72](#_Toc125595235)

[工具软件RegexBuddy的使用 72](#_Toc125595236)

[正则表达式规则 73](#_Toc125595237)

[正则表达式的匹配模式 75](#_Toc125595238)

[开发中使用正则表达式的流程 75](#_Toc125595239)

[课堂练习 76](#_Toc125595240)

[开发软件中正则表达式的使用 76](#_Toc125595241)

[JAVA程序中使用正则表达式 77](#_Toc125595242)

[JAVASCRIPT中使用正则表达式 80](#_Toc125595243)

[针对表单域的验证，封装一个通用的函数 81](#_Toc125595244)

# HTML简介

HTML（HyperText Mark-up Language）即超文本标记语言，是目前网络上应用最为广泛的语言，也是构成网页文档的主要语言。HTML文本是由HTML命令组成的描述性文本，HTML命令可以说明文字、图 形、动画、声音、表格、链接等。HTML的结构包括头部（Head）、主体（Body）两大部分，其中头部描述浏览器所需的信息，而主体则包含所要说明的 具体内容。

HTML语言被称为互联网的三大基石之一(其余两大基石分别为：HTTP协议、URL)。它解决了如何以丰富的效果展示数据内容的问题。互联网中，数据是在服务器和浏览器之间互相传送和执行。三大基石分别解决了如下问题：

* HTTP协议：解决了服务器和浏览器之间数据如何传送、传送格式的问题！实现了分布式的信息共享。
* URL协议：解决了众多服务器中资源定位问题。从而让浏览器可以访问不同的服务器资源，实现了全球信息的精确定位。
* HTML语言：解决了数据在浏览器中如何丰富多彩的展示，及如何合理标示信息的问题。

我们访问一个互联网资源：<http://www.baidu.com>，流程如下：

第一步：地址栏输入：



第二步：服务器收到请求，将HTML源代码传给客户端：

|  |
| --- |
| <!doctype html><html><head><meta http-equiv="Content-Type" content="text/html;charset=gb2312"><title>百度一下，你就知道 </title> </head>  <body><p id="u"><a href="[/gaoji/preferences.html](view-source:http://www.baidu.com/gaoji/preferences.html)">搜索设置</a>&nbsp;|&nbsp;<a href="[http://passport.baidu.com/?login&tpl=mn](view-source:http://passport.baidu.com/?login&tpl=mn)">登录</a></p><div id="m"><p id="lg"><img src="[http://www.baidu.com/img/baidu\_logo.gif](view-source:http://www.baidu.com/img/baidu_logo.gif)" width="270" height="129" usemap="#mp"></p><p id="nv"><a href="[http://news.baidu.com](view-source:http://news.baidu.com/)">新&nbsp;闻</a>　<b>网&nbsp;页</b>　<a href="[http://tieba.baidu.com](view-source:http://tieba.baidu.com/)">贴&nbsp;吧</a>　<a href="[http://zhidao.baidu.com](view-source:http://zhidao.baidu.com/)">知&nbsp;道</a>　<a href="[http://mp3.baidu.com](view-source:http://mp3.baidu.com/)">MP3</a>　<a href="[http://image.baidu.com](view-source:http://image.baidu.com/)">图&nbsp;片</a>　<a href="[http://video.baidu.com](view-source:http://video.baidu.com/)">视&nbsp;频</a>　<a href="[http://map.baidu.com](view-source:http://map.baidu.com/)">地&nbsp;图</a></p><div id="fm"><form name="f" action="s"><input type="text" name="wd" id="kw" maxlength="100"><input type="submit" value="百度一下" id="su"></form><div id="mCon"><span>输入法</span></div><ul id="mMenu"><li><a href="[#](view-source:http://www.baidu.com/)" name="ime\_hw">手写</a></li><li><a href="[#](view-source:http://www.baidu.com/)" name="ime\_py">拼音</a></li><li class="ln"></li><li><a href="[#](view-source:http://www.baidu.com/)" name="ime\_cl">关闭</a></li></ul></div>  <p id="lk"><a href="[http://hi.baidu.com](view-source:http://hi.baidu.com/)">空间</a>　<a href="[http://baike.baidu.com](view-source:http://baike.baidu.com/)">百科</a>　<a href="[http://www.hao123.com](view-source:http://www.hao123.com/)">hao123</a><span> | <a href="[/more/](view-source:http://www.baidu.com/more/)">更多&gt;&gt;</a></span></p><p id="lm"></p><p><a id="st" onClick="this.style.behavior='url(#default#homepage)';this.setHomePage('http://www.baidu.com')" href="[http://utility.baidu.com/traf/click.php?id=215&url=http://www.baidu.com](view-source:http://utility.baidu.com/traf/click.php?id=215&url=http://www.baidu.com)">把百度设为主页</a></p><p id="lh"><a href="[http://e.baidu.com/?refer=888](view-source:http://e.baidu.com/?refer=888)">加入百度推广</a> | <a href="[http://top.baidu.com](view-source:http://top.baidu.com/)">搜索风云榜</a> | <a href="[http://home.baidu.com](view-source:http://home.baidu.com/)">关于百度</a> | <a href="[http://ir.baidu.com](view-source:http://ir.baidu.com/)">About Baidu</a></p><p id="cp">&copy;2011 Baidu <a href="[/duty/](view-source:http://www.baidu.com/duty/)">使用百度前必读</a> <a href="[http://www.miibeian.gov.cn](view-source:http://www.miibeian.gov.cn/)" target="\_blank">京ICP证030173号</a> <img src="[http://gimg.baidu.com/img/gs.gif](view-source:http://gimg.baidu.com/img/gs.gif)"></p></div><map name="mp"><area shape="rect" coords="43,22,227,91" href="[http://hi.baidu.com/baidu/](view-source:http://hi.baidu.com/baidu/)" target="\_blank" title="点此进入 百度的空间"></map></body> |

注：限于篇幅原因，部分源代码被删减!

第三步：浏览器解释执行上面的代码，出现如下丰富多彩的效果：



# HTML起源和历史

HTML作为定义万维网的基本规则之一，最初由蒂姆·本尼斯李（Tim Berners-Lee）于1989年在CERN（Conseil Europeen pour la Recherche Nucleaire）研制出来。HTML的设计者是这样考虑的：HTML格式将允许科学家们透明地共享网络上的信息，即使这些科学家使用的计算机差别很大。因此，这种格式必须具备如下几个特点：

* 独立于平台，即独立于计算机硬件和操作系统。这个特性对各种受从是至关重要的，因为在这个特性中，文档可以在具有不同性能（即字体、图形和颜色差异）的计算机上以相似的形式显示文档内容。
* 超文本。允许文档中的任何文字或词组参照另一文档，这个特性将允许用户在不同计算机中的文档之间及文档内部漫游。
* ·精确的结构化文档。该特性将允许某些高级应用，如HTML文档和其他格式文档间互相转换以及搜索文本数据库。

经过二十多年的发展，HTML已经成为IT时代最重要的技术！HTML标准经历了如下版本更换：

1. 超文本置标语言(第一版)——在1993年6月发为[互联网](http://baike.baidu.com/view/6825.htm)工程工作小组 (IETF)工作草案发布（并非标准）.
2. HTML 2.0——1995年11月作为RFC 1866发布，在RFC 2854于2000年6月发布之后被宣布已经过时
3. HTML 3.2——1996年1月14日，W3C推荐标准
4. HTML 4.0——1997年12月18日，W3C推荐标准
5. HTML 4.01（微小改进）——1999年12月24日，W3C推荐标准
6. HTML5草案的前身名为Web Applications 1.0。於2004年被WHATWG提出，於2007年被W3C接纳，并成立了新的HTML工作团队。在2008年1月22日，第一份正式草案已公布,预 计将在2010年9月正式向公众推荐。WHATWG表示该规范是目前正在进行的工作，仍须多年的努力。

# 为什么需要学习HTML?

目前软件的架构多数采用B/S架构，此处的B即为Browser浏览器。那么，要想在浏览器中展示数据，必然涉及到HTML。因此，HTML也成为目前软件工程师必须要掌握的基本技能！

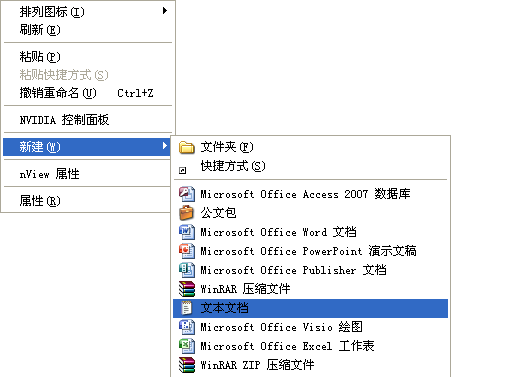
对于我们JAVA工程师来讲，我们将更加关注HTML中的一些常用标签的使用，尤其是表单标签是我们需要掌握的重点。

我们并不需要特别关注，HTML标签显示效果，页面的美观度。术业有专攻，这些将会由公司美工来负责。🡪UI工程师

# 第一个HTML程序

HTML文件是普通的文本文件，只是文件扩展名为：.html或者.htm。下面我们开始创建我们的第一个HTML程序：

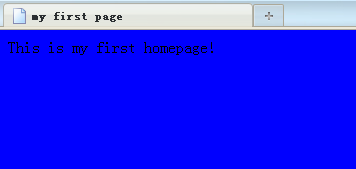
1. 新建文本文件：1.html

1. 使用记事本打开，输入代码：

|  |
| --- |
| <html>  <head>  <title>my first page</title>  </head>  <body bgcolor=blue>  This is my first homepage!  </body>  </html> |

1. 使用浏览器打开该文件，即可看到展示效果：



蓝色背景

# HTML基本规范

基本概念总结：

* HTML 指的是超文本标记语言 (**H**yper **T**ext **M**arkup **L**anguage)
* HTML 不是一种编程语言，而是一种*标记语言* (markup language)
* 标记语言是一套*标记标签* (markup tag)
* HTML 使用*标记标签*来描述网页
* HTML语言不区分大小写！

HTML标记标签通常被称为 HTML 标签 (HTML tag)。

* HTML 标签是由*尖括号*包围的关键词，比如 <html>
* HTML 标签通常是*成对出现*的，比如 <b> 和 </b>
* 标签对中的第一个标签是*开始标签*，第二个标签是*结束标签*
* 某些 HTML 元素没有结束标签，比如 <br />

注：开始标签的英文翻译是 start tag 或 opening tag，结束标签的英文翻译是 end tag 或 closing tag

HTML元素：标记+中间内容

<b>bjsxt</b> 称之为一个元素！

HTML属性：

属性率属于标签元素，属性之间使用空格隔开！属性值可以使用双引号、单引号、或不使用引号。

<p align=”center” >bjsxt</p>

建议：在表明资源路径的属性上一定要使用引号！

# HTML标签

## 常用标签

|  |  |  |
| --- | --- | --- |
| 标签名称 | 使用说明 | 常用属性 |
| <html></html> | 创建一个HTML文档 |  |
| <head></head> | 设置文档标题以及其他不在WEB网页上显示的信息 |  |
| <body></body> | 设置文档的可见部分 | bgcolor, background |
| <title></title> | 将文档的题目放在标题栏中 |  |
| <pre></pre> | 创建预格式化文本 |  |
| <h1>,<h2><h3>,  <h4>,<h5>,<h6> | 标题元素 |  |
| <b></b> | 黑体 |  |
| <i></i> | 斜体字 |  |
| <strong></strong> | 加重一个单词（通常是斜体加黑体） |  |
| <font> | 字体 | Color,size |
| <a> | 创建超链接、锚点 | href：链接地址  name：锚点名字 |
| <p> | 创建新的段落 | Align |
| <br>或者<br/> | 换行 |  |
| <ul><li><ol> | 创建列表 |  |
| <div> | 一个用来排版大块HTML段落的标签 |  |
| <img> | 图像标签 | src:源文件地址  width:宽度  height:高度  alt:说明文字 |
| <hr> | 水平线 |  |
| <table></table> | 创建一个表格 |  |
| <tr></tr> | 开始表格中的每一行 |  |
| <td></td> | 开始一行中的每一个格子 |  |
| <th></th> | 设置表格头：一个通常使用黑体居中文字的格子 |  |
| <frameset></frameset> | 放在一个窗框文档的<body>标签之前，也可以嵌在其他窗框文档中 |  |
| <frame> | 定义一个窗框内的单一窗或窗区域 |  |
| <iframe> | 内帧、内框 |  |
|  |  |  |

### <head>中常见子元素的含义

* 1. 说明HTML文档一些要素,告诉浏览器如何解析。
* 2. 告诉搜索引擎本文档相关的内容。**(seo)**

|  |
| --- |
| <head>  <title>典型的head</title>  <meta http-equiv="Content-Type" content="text/html**;** charset=**gb2312**" />  <meta name="author" content="**zhangsan**">  <meta http-equiv="pragma" content="no-cache">  <meta http-equiv="cache-control" content="no-cache">  <meta http-equiv="expires" content="0">  <meta name="keywords" content=“**JAVA,学习,培训** ">  <meta name="description " content=“关于北京地区java培训的介绍">  <meta http-equiv="refresh" content="**5;URL=http://www.163.com**">=======  **<link rel="stylesheet" type="text/css" href="css/pagination.css" />**  **<script type="text/javascript" src="js/search.js"></script>**  </head> |

### 标题标记<hx>

<h1>、<h2>、<h3>、<h4>、<h5>、<h6>,作为标题使用，并且依据重要性递减。<h1>是最高的等级。例如:

　　<h1>文档标题</h1>

　　<h2>次级标题</h2>

搜索引擎比较重视<hx>标记中的内容。

### 文本标记<b><i><u><del>

<b>粗体显示

<i>斜体显示

<u>下划线显示

<del>中划线显示(删除效果)

### 预格式文本<pre>

<pre> 标签的一个常见应用就是用来表示计算机的源代码。

### 字体标签<font>

<font> 规定文本的字体、字体尺寸、字体颜色。比如：

<font size="3" color="red">This is some text!</font>

<font size="2" color="blue">This is some text!</font>

<font face="verdana" color="green">This is some text!</font>

### 换行<br/>或者<br>

<br> 标签是空标签（意味着它没有结束标签，因此这是错误的：<br></br>）。在 XHTML 中，把结束标签放在开始标签中，也就是 <br />。

### 水平线<hr>

<hr> 标签在 HTML 页面中创建一条水平线。

水平分隔线（horizontal rule）可以在视觉上将文档分隔成各个部分。

注意：[width](http://www.w3school.com.cn/tags/att_hr_width.asp) 属性可以使用*pixels像素值，也可以使用百分比。请注意它们的区别。如果是百分比，会随着窗口的变化，宽度也发生变化。像素值的话，窗口大小发生变化，水平线宽度也不发生变化。*

### 超链接<a>

* <a href=“5.htm”>5.htm</a> 超链接
* 图片作为链接。

<a href="lastpage.htm">  
 <img border="0" src=".\images\next.gif">

</a>

* target属性(定义从什么地方打开链接地址 )

<a href="http://www.163.com/" target="**\_blank**">163!</a>

* 锚点签和name属性
  + **命名一个锚点：**

<a name="**label**">Text to be displayed</a>

* + **链接到锚点：**

<a name="http://www.w3schools.com/html\_links.asp**#label**">Jump to the label</a>

* 创建邮件链接  
  <a href="mailto:**test@163.com**?subject=AboutHTML">

发邮件给我</a>

### 列表标签<ul><ol><li> 网页导航

<ul> 标签定义无序列表。

|  |  |
| --- | --- |
| <ul>  <li>Coffee</li>  <li>Tea</li>  <li>Milk</li>  </ul> |  |

<ol> 标签定义有序列表。

|  |  |
| --- | --- |
| <ol>  <li>Coffee</li>  <li>Tea</li>  <li>Milk</li>  </ol> |  |

注：列表标签经常跟CSS结合，制作菜单导航！

### 图像标签<img>

src:源文件地址 width:宽度 height:高度 alt:说明文字

注：

* 1. 一般建议都增加alt属性，这样有利于搜索引擎的搜索。
  2. 虽然可以使用width、height调整图像的大小，但是实际编程时，最好还是通过后台程序将图片进行缩放处理。

### 内嵌<embed>

H5标签

<embed>标记用于在页面中嵌入多媒体文件，但是用户计算机上需要事先安装相应的处理程序。

* 常用嵌入式文档的格式：mp3, mid, wma, asf, swf, flv, rm, ra, ram, avi
* 典型用法：

<embed src="ee.wmv" autostart="true" loop='true' hidden='false' width="100px" height="100px" />

autostart:是否自动播放嵌入的文件。

loop:是否循环播放。也可以取数字，表明循环多少次。

hidden:是否显示播放器。

-----------------------------------------------------------------------

### 表格标签<table><tr><td><th>

<table> 标签定义 HTML 表格。

简单的 HTML 表格由 table 元素以及一个或多个 tr、th 或 td 元素组成。

tr 元素定义表格行，th 元素定义表头，td 元素定义表格单元。

|  |  |
| --- | --- |
| 基本示例：  <table border="1">  <tr>  <th>Month</th>  <th>Savings</th>  </tr>  <tr>  <td>January</td>  <td>$100</td>  </tr>  </table> |  |
| 跨行(rowspan)、跨列(colspan)操作：  <table border="1">  <tr>  <th>Month</th>  <th>Savings</th>  </tr>  <tr>  <td>January</td>  <td>$100.00</td>  <td rowspan="2">$50</td>  </tr>  <tr>  <td>February</td>  <td>$10.00</td>  </tr>  </table> |  |
| <table border="1">  <tr>  <th>Month</th>  <th>Savings</th>  </tr>  <tr>  <td colspan="2">January</td>  </tr>  <tr>  <td colspan="2">February</td>  </tr>  </table> |  |
| **单线表格的实现：**  <table width=400 border=1 bordercolor="#FF0000" style="border-collapse:collapse">  <tr>  <td>  cell1  </td>  <td>  cell2  </td>  <td rowspan=2>  cell4  </td>  </tr>  <tr>  <td colspan=2>  cell3  </td>  </tr>  </table> |  |

**注：**

表格标签以前经常用于网页的布局排版，不过现在已经基本上被CSS布局替代！

### 块元素<div>

Div + CSS

<div> 可定义文档中的分区或节（division/section）。

<div> 标签可以把文档分割为独立的、不同的部分。它可以用作严格的组织工具，并且不使用任何格式与其关联。

<div> 是一个块级元素。这意味着它的内容自动地开始一个新行。

在CSS的学习中，我们会详细讲述<div>的用法。

### 行内元素<span>

<span> 标签被用来组合文档中的行内元素。在CSS的学习中，我们会详细讲述它的用法。

### 框架标签<frameset><frame><iframe>

frameset 元素可定义一个框架集。它被用来组织多个窗口（框架）。每个框架存有独立的文档。在其最简单的应用中，frameset 元素仅仅会规定在框架集中存在多少列或多少行。您必须使用 cols 或 rows 属性。

<frame> 标签定义 frameset 中的一个特定的窗口（框架）。

frameset 中的每个框架都可以设置不同的属性，比如 border、scrolling、noresize 等等。

|  |
| --- |
| <html>  <frameset cols="25%,50%,25%">  <frame src="frame\_a.htm" />  <frame src="frame\_b.htm" />  <frame src="frame\_c.htm" />  </frameset>  </html> |
|  |

内嵌框架：

<iframe src="http://www.baidu.com" width="300px" height="300px"></iframe>

<noframes>的作用：可为那些不支持框架的浏览器显示文本。

### 表单标签

|  |  |  |
| --- | --- | --- |
| 表单元素 | 标签 | 描述 |
| 文本框 | <INPUT TYPE=‘text’ …> | 输入一行文本 |
| 密码 | <INPUT TYPE=‘password’ …> | 输入一行文本，但不可见，只以星号显示 |
| 单元按钮 | <INPUT TYPE=‘radio’ …> | 只能选择一个选项 |
| 复选框 | <INPUT TYPE=‘checkbox’ …> | 可以选择一个或多个选项 |
| 隐藏域 | <INPUT TYPE=‘hidden’ …> | 该域的值对用户不可见，但会提交给服务器 |
| 下拉列表 | <SELECT …>  <OPTION …> … </OPTION> (可以放多个option)  </SELECT> | |
| 文本域 | <TEXTAREA …>…</TEXTAREA> | 可以输入多行文本 |
| 提交按钮 | <INPUT TYPE=‘submit’> | 提交表单 |
| 普通按钮 | <INPUT TYPE=‘button’> |  |
| 重置按钮 | <INPUT TYPE=‘reset’> | 重置表单信息 |

基本功能：收集用户填写的数据，然后提交到服务器上。

1. 要提交数据的表单域必须加name属性。不然，该表单域的数据不会提交到服务器上。

文本框：

<input type="text" name="uname" value="333" size="30" maxlength="10" />

密码框：**（跟文本框的用法基本一致，除了页面上以星号显示）**

<input type="password" name="pwd" />

单选按钮(**用于多选一的情况**)：

性别：<input type="radio" name="gender" value="1" />男

<input type="radio" name="gender" value="0" />女

1. 单选按钮是分组的， **name相同即为一组**。 一组只能选中一个按钮。
2. **value：如果不增加value属性，选中该按钮提交时默认为：on，**而不是空字符串。所以，建议大家使用单选按钮一定要指定value.

复选框(**用于多选多的情况**)：

喜欢编程语言：<br />

<input type="checkbox" name="favorite"/> JAVA

<input type="checkbox" name="favorite" value="2"/> C++

<input type="checkbox" name="favorite" value="3"/> RUBY

<input type="checkbox" name="favorite" value="4"/> C

1. **复选框是分组的， name相同即为一组**。 可同时多个复选框
2. **value：如果不增加value属性，选中该按钮提交时默认为：on**，而不是空字符串。
3. 选中的选项会被传递成多个同名不同值的参数。

**下拉列表（实现多选一 或多选多的效果。）：**

国籍：<br />

<select name="country" multiple>

<option>China</option>

<option value="2">America</option>

<option value="3">France</option>

</select>

1. 当<option>没有指定value属性。如果被选中，则提交<option>中的提示文本，而不是On或空字符串。

**注：单选按钮、复选框默认选中用：checked属性。 下拉列表使用：selected属性。**

隐藏域：

隐藏域：<input type="hidden" name="hid" value="55"/> <br />

多行文本域：

<textarea name="introduce" rows="4" cols="40"> </textarea>

重置按钮：将各表单域的值恢复为默认值。

提交按钮：提交表单数据。

图片按钮：提交表单数据，通过单击指定的图片提交表单。例如：

<input type=image src=”1.jpg” />

# HTML字符实体

* 有些字符，比如说“<”字符，在HTML中有特殊的含义，因此不能在文本中使用。想要在HTML中显示一个小于号“<”，需要用到字符实体：&lt;或者&#60;
* **实体名是大小写敏感的**。
* 字符实体拥有三个部分：一个and符号（**&**），一个**实体名**或者一个**实体号**，最后是一个分号（**;**）

|  |  |  |  |
| --- | --- | --- | --- |
| 显示结果 | 描述 | 实体名 | 实体号 |
|  | 不可拆分的空格 | &nbsp; | &#160; |
| < | 小于 | &lt; | &#60; |
| > | 大于 | &gt; | &#62; |
| & | and符号 | &amp; | &#38; |
| " | 引号 | &quot; | &#34; |
| £ | 英镑 | &pounds; | &#163; |

# 本章课下作业

1. 完成163的注册表单(样式可以不一致)

2. 阅读老师发的模板页面中的页面，进一步熟悉HTML标签的用法

# 自学课题

下面的内容跟我们以后的工作学习关系不大，但是希望大家通过资料查询、自学技术锻炼大家自学的能力。并且，要将相关资料整理成规范的文档。选题包含下面的(选取一个即可)

HTML5技术展望

Dreamweaver使用技巧

Firework使用技巧

网页模板如何使用(结合Fireworks软件)

**第二篇 CSS语言**

# 什么是CSS?

* CSS是层叠样式表（Cascading Style Sheets）的缩写，它用于定义HTML元素的显示形式，是一种格式化网页内容的技术。CSS现在已经被大多数浏览器所支持，成为网页设计者必须掌握的技术之一。
* W3C自1996年12月发布第一个CSS正式推荐版CSS 1.0以来，一直在对CSS标准进行修订、升级。1999年1月，CSS 2.0 正式推荐版发布，增加了对其它媒体（打印机、视觉设备）、可下载字体、元素定位和表格的支持。最新版本的CSS标准CSS 3.0正在开发中。

# CSS的主要优势?

1. 专注于显示。使显示和数据本身分离。
2. 优势：
   1. CSS将从基础开始建设直到全面替代传统Web设计方法。W3C组织创建的CSS技术将替代HTML中用于表现的HTML元素。
   2. 提高页面浏览速度。使用CSS，比传统的Web设计方法至少节约50%以上的文件尺寸。
   3. 缩短改版时间，降低维护费用。只要简单修改几个CSS文件就可以重新设计一个有成百上千页面的站点。
   4. 强大的字体控制和排版能力。有了CSS，我们不再需要用font标记或者透明的1 px GIF图片来控制标题，改变字体颜色、字体样式等等。
   5. CSS非常容易编写。我们可以象写HTML代码一样轻松地编写CSS。
   6. 可以一次设计，随处发布。你的设计不仅仅用于Web浏览器，也可以发布在其他设备上。
   7. 更好的控制页面布局。结合CSS和div元素，可以比传统的使用table元素更好地控制页面布局。
   8. 实现表现和结构、内容相分离。将网页的表现形式部分剥离出来放在一个独立样式文件中，可以减少未来网页无效的可能。
   9. 更方便搜索引擎的搜索。用只包含结构化内容的HTML代替嵌套的标签，搜索引擎将更有效地搜索到网页的内容，并可能给网页一个较高的评价。

# java程序员需要了解css中哪些内容?

1. **Css是什么？有什么优势？（css：cascade style sheet）**
2. **怎么样引用css文件、css样式？**
3. **Css选择器（如果对ajax感兴趣的话，可以了解）**

# 样式规则

* CSS与字处理程序中的样式的工作方式相似，我们可以先定义一个包含格式选项的“样式”，然后将其应用于文档元素。CSS样式由样式规则组成，所有的样式规则的语法遵循如下相同的基本格式：
  + 选择器 { 属性1: 值1; 属性2: 值2; ... 属性N: 值N; }
* 一条样式规则由一个选择器和一个或者多个声明组成；
* 选择器指定规则可作用于HTML文档中哪个或者哪些元素；
* 声明要用大括号（{}）括起来。每个声明要用分号（;）结束。
* 一个属性/值对组成一个声明，属性和值之间用冒号（:）分隔；
* 如果属性值中包含了空格，那么该值就必须用双引号（""）括起来。例如：font-family: "Times New Roman";
* 我们还可以指定多个选择器使用相同的样式定义，选择器之间用逗号分隔开。例如，如果我们想要把所有标题列标记（从1到6）的文本字体颜色都定义为红色，可以使用如下定义：

h1, h2, h3, h4, h5, h6 { color: red; }

# 样式规则的位置

* 外部样式表

不需要style标签

<link rel=“stylesheet” href=“” />

* 嵌入式样式表

<style type=“text/css”>

p{}

</style>

* 内联样式

属性名为style

举例:<p style=“”></p>

# 选择器

## 元素选择器

HTML元素是最典型的选择器类型，任何一个HTML元素都可以做为选择器。元素选择器的有效范围为页面中所有的、名称相同的HTML元素。

**格式： 元素 { 属性: 值; }**

**例如： h1 { color: red; } h2 { color: #FF0000; } 或者：h1, h2 { color: #ff0000; }**

## 类选择器

如果想要把某一个样式应用到不同的HTML元素上，可以采用类选择器来定义，然后在不同标签中应用它们。

格式1： **.类名 { 属性: 值; }**

**格式2：元素.类名 { 属性: 值; }**

代码实例：

|  |
| --- |
| **<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"**  **"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**  **<html xmlns="http://www.w3.org/1999/xhtml">**  **<head>**  **<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />**  **<title>类选择器示例1</title>**  **<style type="text/css">**  **.tt{ font-size:10pt; font-family:黑体; color:#800080; font-weight:bold; }**  **</style>**  **</head>**  **<body>**  **<p class="tt">示例一</p> /\*p标签应用名字为tt的类层叠样式表\*/**  **<h1 class="tt">示例二</h1> /\*h1标签也应用名字为tt的类层叠样式表\*/**  **</body>**  **</html>** |
| **<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"**  **"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**  **<html xmlns="http://www.w3.org/1999/xhtml">**  **<head>**  **<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />**  **<title>类选择器示例2</title>**  **<style type="text/css">**  **h1.left { color:blue; }**  **h1.right { color:red; }**  **</style>**  **</head>**  **<body>**  **<h1 class="left">第一行</h1> /\*h1应用名为h1.left的类样式规则/**  **<h1 class="right">第二行</h1> /\*h1也应用名为h1.right 的类样式规则\*/**  **</body>**  **</html>** |

## ID选择器

除了使用类选择器的方式定义一个样式以外，还可以使用ID定义样式。ID与类选择器的概念相似，只是ID选择器只能被引用一次，而类选择器可以被多次引用。

格式：**#id名 { 属性: 值; }**

示例：**#red { color: red; }**

**<p id="red">欢迎使用id选择器</p>**

## 包含选择器

如果需要为位于某个标记符内的元素设置特定的样式规则，则应将选择器指定为具有上下文关系的HTML标记

**h1 b{ color:blue; } /\*注意h1和b之间以空格分隔\*/**

# CSS属性单位

* 长度单位
  + 绝对长度值包括cm、mm、in、pt、pc等，绝对长度值最好用于打印输出设备，而在仅仅作为屏幕显示用时，绝对长度值并无多大意义。
  + 相对长度值包括px、em、ex等。相对长度是指元素尺寸相对于浏览器的系统默认值来相应的缩放。
* 颜色单位
  + 用百分比值来表示，如color:rgb(50%,0,50%) ；
  + 使用0-255之间的整数值来设置，如color:rgb(128,0,128) ；
  + 使用十六进制数组定义颜色，如#fc0eab；
  + 使用简化的十六进制数定义颜色，如#080；
  + 为颜色取名：aqua,black,blue,fuchsia,gray,green等。
* URL单位
* URL单位的具体格式是：在“url”后面紧跟一个括号，括号中是url的地址。如果在地址中使用了括号、逗号、空格、单引号和双引号，那么就必须在整个地址的外面加上一对单引号或者双引号。地址可以是相对地址或绝对地址。

例如：

p{ background-image : url(img/logo.gif) ; }

# 文本属性

|  |  |  |
| --- | --- | --- |
| 属性名 | 说明 | 参数说明 |
| font-family | 字体名称 |  |
| font-size | 字体大小 | 例：24px |
| font-style | 字体风格 | normal(缺省)，italic, oblique |
| font-weight | 字体粗细 | normal(缺省)，bold或数值。 |
| color | 字体颜色 |  |
| line-height | 行高 | normal | *length* |
| letter-spacing | 字符间距 | normal | *length* |
| text-decoration | 字体装饰 | none || underline || blink || overline || line-through |
| text-overflow | 文本溢出处理 | clip | ellipsis |
| text-align | 文本对齐属性 | left (居左，缺省值)  right (居右) center (居中) |

Blink属性：ie不支持，火狐支持。（浏览器差异的问题）

实现溢出文本eclipsis的解决：

overflow:hidden;text-overflow:ellipsis; white-space:nowrap;

# 背景属性

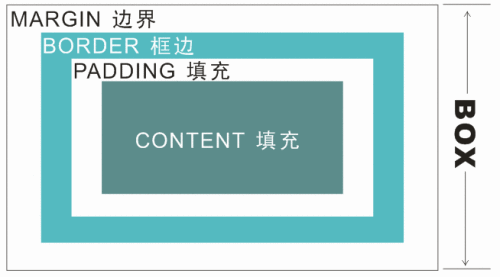
|  |  |  |
| --- | --- | --- |
| 属性名 | 说明 | 参数 |
| background-color | **背景颜色** | 颜色单词或数值 |
| background-image | **背景图片** |  |
| background-repeat | **背景重复** | 默认：同时向x, y方向重复  repeat-x 背景图片横向重复  repeat-y 背景图片竖向重复  no-repeat 背景图片不重复 |
| background-attachment | **背景附着** | scroll（缺省），fixed |

background-attachment:fixed:属性：火狐支持，ie6不支持；ie7支持。

# 定位属性

* **定位的定义**
* position: relative | absolute | static | fixed
  + **static** 没有特别的设定，遵循基本的定位规定，**不能通过z-index进行层次分级。**
  + **relative** 不脱离文档流，参考自身静态位置,通过 top,bottom,left,right 定位，并且可以通过z-index进行层次分级。
  + **absolute** 脱离文档流，通过 top,bottom,left,right 定位。选取其最近的父级定位元素，当父级 position 为 static 时，absolute元素将以body坐标原点进行定位，可以通过z-index进行层次分级。
  + **fixed** 固定定位，这里他所固定的对象是可视窗口而并非是body或是父级元素。可通过z-index进行层次分级。
* CSS中定位的层叠分级：z-index: auto | number ;
  + auto 遵从其父对象的定位  
    number  无单位的整数值。可为负数。z-index使用整数表示元素的前后位置，数值越大，就会显示在相对靠前的位置，并且CSS同意在z-index中使用负数

# CSS盒子模型（box model）



margin属性: margin-top, margin-bottom, margin-left, margin-right

border属性: border-color, border-style, border-width, border-top…

padding属性: padding-top, padding-bottom, padding-left, padding-right

content属性:

Box model：

**背景包括：padding+ content**

**宽度和高度指的是content的宽度和高度。**

# 布局属性

|  |  |  |
| --- | --- | --- |
| 属性名 | 说明 | 常用参数 |
| float | **漂浮** | **none | left |right** |
| clear | **是否允许漂浮** | **none | left |right | both** |
| display | **能否显示** | **none |block** |
| visibility | **可见性** | **inherit | visible | hidden** |

Display:为None时，其他元素可以占据该元素的位置。

Visibility: 为hidden时，其他元素不能占据该元素的位置。

# 列表属性

|  |  |  |
| --- | --- | --- |
| 属性名 | 说明 | 参数 |
| list-style-type | **列表样式**  **类型属性** | disc (缺省值，黑圆点)  circle (空心圆点) square (小黑方块)  decimal (数字排序)  lower-roman upper-roman 罗马数字  lower-alpha upper-alpha 英文字母  none (无列表项标记) |
| list-style-position | **列表样式位置属性** | outside (以列表项内容为准对齐)  inside (以列表项标记为准对齐) |
| list-style-image | **列表样式图片属性** |  |

# 鼠标样式

* a:hover 鼠标移上时的超链接  
  a:link 常规，非访问超链接  
  a:visited 访问过的超链接  
  a:active 鼠标点击时的超链接

# 样式就近原则

误解：外部文件<内部样式块对象<style属性

一句话：谁离得近，谁优先级高！

# CSS中的继承

CSS中，有一些样式是可以继承的，如字体、文本属性。有一些样式是不能继承的，如边框、间距、布局、定位等。

# 使用JavaScript控制css样式

可以通过HTML元素的style、cssText、className属性控制CSS样式的变化。这些内容，我们将在JAVASCRIPT中学习。

# CSS工具集

<http://www.devdude.com/tools/css/button_text/>， 在线编辑得到按钮

<http://cssmenumaker.com/> 各种CSS的菜单生成（相当不错!）

<http://www.buttonator.com/> 在线生成各种按钮图片（相当不错！）

<http://www.favikon.com/> 在线生成favikon,就是图标

<http://www.3d-pack.com/> 做3D包装效果

<http://www.reflectionmaker.com/> 生成含有倒影效果的图片

[http://www.fodey.com/generators/](http://www.fodey.com/generators/newspaper/snippet.asp) 生成各种动画效果文字

<http://genmicha.cn/befunkycom-funny-ways-to-express-yourself.htm> 把你的照片卡通化

<http://www.stripegenerator.com/> 在线生成各种条纹效果

<http://web2.0stylr.com/stylr.aspx> 在线生成LOGO

<http://www.web20badges.com/> 在线生成徽章

<http://www.roundedcornr.com/> 在线生成圆角效果

# 自学内容

1. CSS框架(960-Grid-System)自学
2. 制作CSS导航条

# 客户端编程

**编写在浏览器中运行的程序。**

javascript（最主要的）

jscript(微软开发的，现在语法基本和javascript相同。不过建议使用javascript，javascript是w3c的标准，所有浏览器都支持。Jsscript只是微软的产品，ie支持较好)

applet(java的第一次大显身手，使互联网生动起来。现在有的软件还是用applet，涂鸦软件)

activex组件

plug-in技术

* + flash技术的最新发展
  + jsdk包含的plug-in
  + yahoo 的工具条
  + 百度搜索伴侣
  + CNNIC网络实名等
  + 3721
  + **火狐的插件(火狐最重要的功能)**

# JavaScript简介

* JavaScript是一种可以与HTML标记语言混合使用的脚本语言，其编写的程序可以直接在浏览器中**解释**执行。
* 对于很多初学者而言，往往会将JavaScript脚本语言与JAVA编程语言混为一谈。实际上从本质上来说，这二者并没有什么必然的联系。
* Javascript的国际标准是ECMAScript。

JavaScript是一种基于对象和[事件驱动](http://baike.baidu.com/view/536048.htm)并 具有相对安全性的客户端脚本语言。同时也是一种广泛用于客户端Web开发的脚本语言，常用来给HTML网页添加动态功能，比如响应用户的各种操作。它最初 由网景公司的Brendan Eich设计，是一种动态、弱类型、基于原型的语言，内置支持类。JavaScript是Sun公司的注册商标。[1] Ecma国际以JavaScript为基础制定了ECMAScript标准。JavaScript也可以用于其他场合，如服务器端编程。完整的 JavaScript实现包含三个部分：ECMAScript，文档对象模型，字节顺序记号。

# Javascript历史和版本

ECMAScript是一种由[欧洲计算机制造商协会](http://baike.baidu.com/view/2233504.htm)（[ECMA](http://baike.baidu.com/view/786648.htm)）通过ECMA-262标准化的脚本程序设计语言。这种语言在万维网上应用广泛，它往往被称为[JavaScript](http://baike.baidu.com/view/16168.htm)或[JScript](http://baike.baidu.com/view/40829.htm)，但实际上后两者是ECMA-262标准的扩展。详解：

<http://baike.baidu.com/view/810176.htm>

**JavaScript/jscript / ECMAScript 关系？**

* + javaScript是由Netscape公司开发 .它的前身是Live Script
  + Microsoft发行jscript用于internet explorer
  + 最初的jscript和javascript差异过大，web程序员不得不痛苦的为两种浏览器编写两种脚本。于是诞生了ECMAScript,是一种国际标准化的javascript版本。现在的主流浏览器都支持这种版本。



|  |  |
| --- | --- |
| **JavaScript version** | **Relationship to ECMA version** |
| JavaScript 1.1 | ECMA-262, Edition 1 is based on JavaScript 1.1. |
| JavaScript 1.2 | ECMA-262 was not complete when JavaScript 1.2 was released. |
| JavaScript 1.3 | JavaScript 1.3 is fully compatible with ECMA-262, Edition 1. |
| JavaScript 1.4 | JavaScript 1.4 is fully compatible with ECMA-262, Edition 1. |
| JavaScript 1.5 | JavaScript 1.5 is fully compatible with ECMA-262, Edition 3. |

# Javascript和java的区别联系

Javascript和java除了名字和语法有点像，其他没有任何的关系。主要做:富客户端开发.

做个比较是为了让大家更好地理解javascript，事实上，两个语言根本没有可比性，是完全不同的。

|  |  |  |
| --- | --- | --- |
|  | Javascript | Java |
| 对象 | **基于对象**，不能说是面向对象。比如：javascript不支持直接继承，而要通过一个原型对象来间接实现。多态就更不支持了。它们使用一些封装好的对象，调用对象的方法，设置对象的属性，但是它们无法让程序员派生新对象类型，他们只能使用现有对象的方法和属性，所以当你判断一个新技术是否是面向对象的时候，通常可以使用后两个特性加以判断，“面向对象”和“基于对象”都实现了“封装”的概念，但“面向对象”实现了“继承”和“多态”，而“基于对象”可以不实现这些。 | 面向对象 |
| 解释和编译 | 解释 | 编译、解释 |
| 变量类型 | 弱变量类型。  js的弱类型是变量的弱类型 | 强变量类型 |

# 如何在网页中使用脚本语言

* 在HTML文档中插入脚本语言可以使用<script>标记

|  |
| --- |
| <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂JavaScript初级</title>  <script>  function test(){  alert("我的第一个js程序！");  }  </script>  </head>  <body>  <input type=button value=测试 onclick="test();" />  </body>  </html> |

* 此外，在实际应用中考虑到脚本语言书写的代码可能会比较长，可以将脚本语言单独写在扩展明为.js的脚本文件中，然后在<script>标记中设置src属性的值为脚本文件的位置和名称

|  |
| --- |
| <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂JavaScript初级</title>  <script src="2.js" type="text/javascript"></script>  </head>  <body>  <input type=button value=测试 onclick="test();" />  </body>  </html> |
| /\*\*  \* @author Administrator  \*/  function test(){  alert("我的第一个js程序！");  }    //不能再js文件中写：<script>标记，这个标记是html标记! |

**<script>标记的使用要点**

* <script>标记可以置于页面任意位置！
* <script>定义函数一般置于<head>中
* <script>可以直接写js语句。

|  |
| --- |
| <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂JavaScript初级</title>  <script>  function test(){  alert("我的第一个js程序！");  }  </script>  </head>  <body>  <input type=button value=测试 onclick="test();"/>  <script>  alert("script标记可以置于页面任意部分！甚至可以直接执行js语句！");  </script>  </body>  </html> |

# 变量的声明及赋值

* JavaScript 是一种弱类型的脚本语言。
* 变量的声明(变量使用之前必须加var声明，编程规范)
  + 可以通过var关键字来声明一个变量
  + 典型声明方式：
    - var a=1; var a, b=2, c; var date = new Date();
    - var array = new Array();
  + 不能使用未经声明的变量。
* 全局变量
  + 在方法外部声明的变量
  + 方法内部，没有加var关键字声明的变量
* 局部变量
  + 方法内部，使用var声明的变量
* 补充：
  + Javascript:void(0); 用于超链接
  + Javascript伪协议

|  |
| --- |
| <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂JavaScript初级</title>  <script>  var a = 1; //全局变量，页面被加载时就执行！  function test1(){  var b = 3; //局部变量!  c = 4; //执行完方法test1后就被解释成全局变量！  }  function test2(){  alert(a);  alert(c); //先执行test1，再执行test2才能看到c的值！  alert(b);  }  </script>  </head>  <body>  <input type=button value=测试 onclick=test1(); />  <a href="javascript:void(0);" onclick="test2();" >ceshi </a> <!-- 表示执行完test2()后，什么都被不做！不会跳转到其他地方！-->  <input type=button value=测试javascript伪协议 onclick="javascript:var a=3;alert(a);alert('a+3='+(a+3));" />  <input type=button value=测试javascript伪协议 onclick="var a=3;alert(a);alert('a+3='+(a+3));" /> <!--javascript:可以省略！ -->  </body>  </html> |

* 变量的命名规则
  + 1.变量命名必须以英文字母或是下标符号”\_”或者”$”为开头。
  + 2.变量名长度不能超过255个字符。
  + 3.变量名中不允许使用空格。
  + 4.不用使用脚本语言中保留的关键字及保留符号作为变量名。例如：var、\*等。
  + 5.**变量名区分大小写。(javascript是区分大小写的语言)**

**数据类型**

* 变量是没有类型的，数据是有类型的。常用的有如下6种类型：
  + undefined未定义 (声明了变量但从未赋值 )
  + null空 (赋值为null)
  + boolean布尔型 (true,false)
  + 字符串string (单引号、双引号均可)
  + 数值number
    - 包括整数和小数，NaN（Not a Number），Infinity, -Infinity
  + 对象object

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂JavaScript初级</title>  <script>  function testNullUndefined(){  var a;  var b = null;    document.write("a的数据类型："+a+"<br>");  document.write("b的数据类型："+b+"<br>");  }    function testBoolean(){  var a = true;  var b = false;  if(a){  alert("a的数据类型是boolean！");  }  }    function testString(){  var a = "aaa";  var b = 'bbbb';  var c = new String("cccc");  document.write(a+"<br/>"+b+"<br>"+c);  var e = '"eeeee"'; //外面用单引号，里面双引号。或者外面双引号，里面单引号。假如子字符串包含引号的常见的处理技巧！  var f = "U're right!"; //var f = 'U're right!'; //这么写会出错！  alert(e+"\n"+f);  }    function testNumber(){  var a = 123;  var b = 12.345;  var c = NaN;  var d = 1/0;  document.write(a+"<br>"+b+"<br>"+c+"<br>"+d);  if(c==c){  alert("c==c");  }else{  alert("c是NaN值，他是js中唯一一个跟自己不相等的值！我们可以利用这个特点判断某个值是不是NaN！");  }  if(isNaN(parseInt("sss"))){  alert("刚刚运算的结果是：NaN！");  }  }    function testObject(){  var d = new Date();  alert(d);  //其他对象，以及js基于对象编程，以后会讲！今天先了解下！  }  </script>  </head>  <body>  <input type=button value=测试null和undefined onclick="testNullUndefined();" />  <input type=button value=测试boolean onclick="testBoolean();" />  <input type=button value=测试String onclick="testString();" />  <input type=button value=测试Number onclick="testNumber();" />  <input type=button value=测试Object onclick="testObject();" />  </body>  </html> |

**数据类型的自动转化**

* 布尔型↔字符串↔数字
  + true↔”true” ↔1
  + false↔”false” ↔0
* 数字↔字符串
  + 1234↔”1234”
* null ↔布尔型↔数字↔字符串
  + null↔false↔0↔”null”
* undefined ↔布尔型↔数字↔字符串
  + undefined ↔false↔0↔” undefined ”

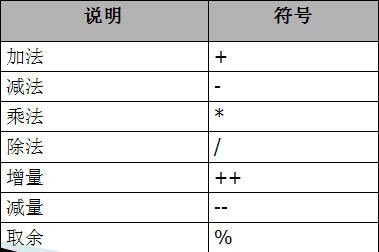
|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂JavaScript初级</title>  <script>  function test(){  //布尔型、字符串、数字的转化  var a = true;  var b = "888"+a;  var c = 99+a;  document.write(b+"<br/>");  document.write(c+"<br/>");    //字符串和数字转化  var d= 33;  var e = "44";  var f = d+e; //加号在此处，变为字符串连接符！这个跟java是一样的！  var g = 3\*"4"; //字符串4，自动变成数字4  document.write(f+"<br>");  document.write(g+"<br>");    //null、布尔型、字符串、数字的转化  var h = null;  var h2 = h+3; //null自动转化数字0  document.write(h2+"<br>");  var h3 = "aa"+h;  document.write(h3+"<br>"); //null自动转化为字符串null  if(!h){  alert("h自动转化为boolean值：false！");  }  }    function test2(){  var a;  var b =null;  if(a==b){  alert("null和undefined可以互相自动转化！");  }  }    </script>  </head>  <body>  <input type=button value=测试 onclick="test();" />  <input type=button value=测试 onclick="test2();" />  </body>  </html> |

# 表达式与运算符

* === 等同符：不会发生类型的自动转化！
* == 等值符：会发生类型自动转化、自动匹配！

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂JavaScript初级</title>  <script>  function test(){  var a = 33;  var b = "33";  if(a==b){  alert("a==b");  }  if(a===b){  alert("a===b");  }  }  </script>  </head>  <body>  <input type=button value=测试 onclick="test();" />  </body>  </html> |

**算术运算符**

**赋值运算符**

* 基本的赋值运算符为“=”，用于将右侧的操作数或表达式的值赋给“=”左侧的变量。
* 在程序中还经常使用到扩展赋值运算符，如：+=，-=，\*=，/=，%=等。

**字符串运算符**

* 字符串运算符主要是字符串运算符“+”和它的变形形式：“+=”
* 数字类型和字符串进行“＋”操作时，实际是进行了字符串的连接操作。
  + var a=1; var b="3"; alert(a+b);//显示的是13,而不是4.
  + var b=true; var c="ddd"; alert(b+c); //显示的是trueddd；
* “我爱”+“北京”+“天安门”产生的结果是一个新的字符串“我爱北京天安门”

**条件运算符**

* **条件表达式的基本的语法是：**
  + **（条件表达式1）？表达式2：表达式3**
  + **执行此表达式时，先判断条件即:逻辑表达式1的值，若为true，则整个三目运算的结果为表达式2的值，否则整个运算结果为表达式3的值。**

**运算符的优先级**

* 在复杂的表达式中往往会综合运用各种不同类型的运算符，而表达式的运算顺序有一定的规则：首先应按照运算符的优先次序从高到低的顺序，其次，优先级相同的运算符按照事先约定的结合方向（从左向右或从右向左）进行。

# 分支语句

* if…else语句完成了程序流程中的分支功能，如果其中的条件成立，则程序执行相应的语句。
  + if…else语句的语法如下：
    - **if(条件){**
    - **执行语句**
    - **}else{**
    - **执行语句**
    - **}**
* 分支语句swith可以根据一个变量的不同取值而采取不同的处理方法，在程序设计中提供不同的分支，这样的语句叫做分支语句。
* Switch语句的语法如下：

switch (expression){

* + case const1:
  + 语句块1
  + case const1:
  + 语句块2
  + ……
  + default:
  + 语句块N
  + }
* 在同时可以使用switch和if时推荐使用switch以获得更好的效率

# 循环语句

* for语句的基本语法如下：
  + **for (初始化部分；条件部分；更新部分){**
  + **语句块…**
  + **}**
* **While循环**
  + **while(条件) {**
  + **语句块；**
  + **}**
* **Do-While**
  + **do{**
  + **语句块；**
  + **}while(条件)**
* break语句是结束当前的循环，并把程序的控制权交给循环的下一条语句。
  + 这里是**结束循环，循环到此为止**
* continue语句是结束当前的某一次循环，但是并没有跳出整个的循环。
  + 这里是**结束一次循环，整个循环还在进行**

# 数组

* “数组”即一组数据的集合。
* js数组更加类似java的容器。长度可变，元素类型也可以不同！
* **Js数组长度随时可变！随时可以修改！我们可以通过修改数组的长度清空数组。**
* 通过array.length获得数组长度。

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂JavaScript初级</title>  <script>  function test(){  //两种定义数组的方式  var a = new Array();  var a2 = []; //更常用！    //初始化方式  a[0] = 123;  a[1] = new Date();    var a3 =[123,"45",new Date(),true];    alert(a);  alert(a3.length);    a3.length = 2; //改变数组长度，清掉后面的元素！  alert(a3);  a3.length = 0; //直接清空数组  alert(a3);  }    function test2(){  var a = [1,2,3,4,5];  a.push("ppp"); //数组末尾添加一个项  alert(a);  a.pop(); //删除数组最后一个项  alert(a);  a.unshift("aa");//在数组第一个元素位置添加元素  alert(a);  a.shift(); //删除数组第一个元素    }  </script>  </head>  <body>  <input type=button value=测试 onclick="test();" />  <input type=button value=测试 onclick="test2();" />  </body>  </html> |

# 函数及深化

* 一个典型的JavaScript函数定义如下：
  + function 函数名称（参数表）
  + {
  + 函数执行部分：
  + }
  + 注意：参数列表直接写形参名即可，不用写var!
* return语句：
  + return返回函数的返回值并结束函数运行
  + return语句的语法如下：
    - return 表达式
* 当使用return语句的时候，要得到函数的返回值，只要利用函数名对一个变量赋值就可以了。
* 函数也是对象！
* 函数的另一个定义方式！
  + 函数可以作为参数传递！

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂JavaScript初级</title>  <script>  function test(){  var a = test2; //把test2指向的函数对象引用赋值给了a  a(); //执行的仍然是test2函数  }  var test2 = function(){ //更能体现函数也是对象的含义！根据此种定义方式，我们可以发现test2是一个全局变量，他只想了匿名的函数对象！  alert("test2");  }      function test3(){  test4(234,test2);  }    function test4(a,b){  alert(a);  alert(b);  b(); //执行b指向的函数对象！  }    </script>  </head>  <body>  <input type=button value=测试 onclick="test();" />  <input type=button value=测试2 onclick="test3();" />  </body>  </html> |

# 内部函数

* 使用eval(String str)来执行javascript代码
  + eval参数字符串是和eval函数上下文相关的
* parseInt(String str), parseFloat(String str);
* isNaN()
* isFinite()
* Escape(), unescape()

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Untitled Document</title>  <script>  function testEval() {  var a="alert('你们好')";  var b="var c=0;";    eval(b); //var c=0;    alert(c);  // var a1 = 'alert("你们好！")';  // var b="var c=11;";  //// alert(a);  // alert("你们好");  // eval("var c=11;");  // var c=11;  // var c=11;  // alert(c);  }    function testParse() {  var d="3.555";  var e1="abd";    var d1= parseFloat(d);  var d2=parseInt(d);  var e2=parseInt(e1);    // alert(1+d);  // alert(1+d1);  alert(d2);  // alert(1+d2);  // alert(e2);    if(isNaN(e2)) {  alert("I'm a NaN");  }    if(e2!=e2) {  alert("我不是我");  }  if(isFinite(e2)) {  alert("我是一个正常的number");  }  }    function testEscape() {  var str="中国";  var str1 = escape(str);  var str2 = unescape(str1);  alert(str1);  alert(str2);  }  </script>  </head>  <body>  <a href="javascript:void(0);" onclick="testEval();">测试eval方法</a>  <a href="javascript:void(0);" onclick="testParse();">测试parse方法</a>  <a href="javascript:void(0);" onclick="testEscape();">测试escape方法</a>  </body>  </html> |

# 事件机制

* 事件是对象发送的消息，以发信号通知操作的发生。操作可能是由用户交互（例如鼠标单击）引起的，也可能是由某些其他的程序逻辑触发的。
* 为什么要使用JavaScript的事件呢？主要有下面的两个用途：
  + 1．验证用户输入窗体的数据。
  + 2．增加页面的动感效果。



|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title></title>  <script>  function testClick(){  alert("click");  }  function testDbclick(){  alert("double click!");  }  function testBlur(){  alert("testOnblur");  }  function testFocus(){  alert("testFocus");  }  function testChange(){  alert("testChange");  }  function testMouseMove(){  alert("onmousemove");  }  function testLoad(){  alert("load");  }  function testUnload(){  alert("Unload");  }  </script>  </head>  <body onload="testLoad();" onunload="testUnload();">  <a href="#" onclick="testClick();">测试onclick</a> <br><br>  <a href="#" ondblclick="testDbclick();">测试ondblclick</a> <br><br>  <a href="#" onmousemove="testMouseMove();">测试onmousemove</a> <br><br>  <input type="text" name="t3" value="测试onFocus" onfocus="testFocus()"/>  <input type="text" name="t1" value="测试onblur" onblur="testBlur()"/>  <input type="text" name="t2" value="测试onChange" onchange="testChange()"/>  </body>  </html> |

|  |  |  |  |
| --- | --- | --- | --- |
| **javascript事件列表** | | | |
| 事件 | | 浏览器支持 | 解说 |
| 一般事件 | onclick | IE3、N2 | 鼠标点击时触发此事件 |
| ondblclick | IE4、N4 | 鼠标双击时触发此事件 |
| onmousedown | IE4、N4 | 按下鼠标时触发此事件 |
| onmouseup | IE4、N4 | 鼠标按下后松开鼠标时触发此事件 |
| onmouseover | IE3、N2 | 当鼠标移动到某对象范围的上方时触发此事件 |
| onmousemove | IE4、N4 | 鼠标移动时触发此事件 |
| onmouseout | IE4、N3 | 当鼠标离开某对象范围时触发此事件 |
| onkeypress | IE4、N4 | 当键盘上的某个键被按下并且释放时触发此事件. |
| onkeydown | IE4、N4 | 当键盘上某个按键被按下时触发此事件 |
| onkeyup | IE4、N4 | 当键盘上某个按键被按放开时触发此事件 |
| 页面相关事件 | onabort | IE4、N3 | 图片在下载时被用户中断 |
| onbeforeunload | IE4、N | 当前页面的内容将要被改变时触发此事件 |
| onerror | IE4、N3 | 出现错误时触发此事件 |
| onload | IE3、N2 | 页面内容完成时触发此事件 |
| onmove | IE、N4 | 浏览器的窗口被移动时触发此事件 |
| onresize | IE4、N4 | 当浏览器的窗口大小被改变时触发此事件 |
| onscroll | IE4、N | 浏览器的滚动条位置发生变化时触发此事件 |
| onstop | IE5、N | 浏览器的停止按钮被按下时触发此事件或者正在下载的文件被中断 |
| onunload | IE3、N2 | 当前页面将被改变时触发此事件 |
| 表单相关事件 | onblur | IE3、N2 | 当前元素失去焦点时触发此事件 |
| onchange | IE3、N2 | 当前元素失去焦点并且元素的内容发生改变而触发此事件 |
| onfocus | IE3 、N2 | 当某个元素获得焦点时触发此事件 |
| onreset | IE4 、N3 | 当表单中RESET的属性被激发时触发此事件 |
| onsubmit | IE3 、N2 | 一个表单被递交时触发此事件 |
| 滚动字幕事件 | onbounce | IE4、N | 在Marquee内的内容移动至Marquee显示范围之外时触发此事件 |
| onfinish | IE4、N | 当Marquee元素完成需要显示的内容后触发此事件 |
| onstart | IE4、 N | 当Marquee元素开始显示内容时触发此事件 |
| 编辑事件 | onbeforecopy | IE5、N | 当页面当前的被选择内容将要复制到浏览者系统的剪贴板前触发此事件 |
| onbeforecut | IE5、 N | 当页面中的一部分或者全部的内容将被移离当前页面[剪贴]并移动到浏览者的系统剪贴板时触发此事件 |
| onbeforeeditfocus | IE5、N | 当前元素将要进入编辑状态 |
| onbeforepaste | IE5、 N | 内容将要从浏览者的系统剪贴板传送[粘贴]到页面中时触发此事件 |
| onbeforeupdate | IE5、 N | 当浏览者粘贴系统剪贴板中的内容时通知目标对象 |
| oncontextmenu | IE5、N | 当浏览者按下鼠标右键出现菜单时或者通过键盘的按键触发页面菜单时触发的事件 |
| oncopy | IE5、N | 当页面当前的被选择内容被复制后触发此事件 |
| oncut | IE5、N | 当页面当前的被选择内容被剪切时触发此事件 |
| ondrag | IE5、N | 当某个对象被拖动时触发此事件 [活动事件] |
| ondragdrop | IE、N4 | 一个外部对象被鼠标拖进当前窗口或者帧 |
| ondragend | IE5、N | 当鼠标拖动结束时触发此事件，即鼠标的按钮被释放了 |
| ondragenter | IE5、N | 当对象被鼠标拖动的对象进入其容器范围内时触发此事件 |
| ondragleave | IE5、N | 当对象被鼠标拖动的对象离开其容器范围内时触发此事件 |
| ondragover | IE5、N | 当某被拖动的对象在另一对象容器范围内拖动时触发此事件 |
| ondragstart | IE4、N | 当某对象将被拖动时触发此事件 |
| ondrop | IE5、N | 在一个拖动过程中，释放鼠标键时触发此事件 |
| onlosecapture | IE5、N | 当元素失去鼠标移动所形成的选择焦点时触发此事件 |
| onpaste | IE5、N | 当内容被粘贴时触发此事件 |
| onselect | IE4、N | 当文本内容被选择时的事件 |
| onselectstart | IE4、N | 当文本内容选择将开始发生时触发的事件 |
| 数据绑定 | onafterupdate | IE4、N | 当数据完成由数据源到对象的传送时触发此事件 |
| oncellchange | IE5、N | 当数据来源发生变化时 |
| ondataavailable | IE4、N | 当数据接收完成时触发事件 |
| ondatasetchanged | IE4、N | 数据在数据源发生变化时触发的事件 |
| ondatasetcomplete | IE4、N | 当来子数据源的全部有效数据读取完毕时触发此事件 |
| onerrorupdate | IE4、N | 当使用onBeforeUpdate事件触发取消了数据传送时，代替onAfterUpdate事件 |
| onrowenter | IE5、N | 当前数据源的数据发生变化并且有新的有效数据时触发的事件 |
| onrowexit | IE5、N | 当前数据源的数据将要发生变化时触发的事件 |
| onrowsdelete | IE5、N | 当前数据记录将被删除时触发此事件 |
| onrowsinserted | IE5、N | 当前数据源将要插入新数据记录时触发此事件 |
| 外部事件 | onafterprint | IE5、N | 当文档被打印后触发此事件 |
| onbeforeprint | IE5、N | 当文档即将打印时触发此事件 |
| onfilterchange | IE4、N | 当某个对象的滤镜效果发生变化时触发的事件 |
| onhelp | IE4、N | 当浏览者按下F1或者浏览器的帮助选择时触发此事件 |
| onpropertychange | IE5、N | 当对象的属性之一发生变化时触发此事件 |
| onreadystatechange | IE4、N | 当对象的初始化属性值发生变化时触发此事件 |

# Event对象

event代表事件的状态，例如触发event对象的元素、鼠标的位置及状态、按下的键等等。

event对象只在事件发生的过程中才有效。

event的某些属性只对特定的事件有意义。比如，fromElement 和 toElement 属性只对 onmouseover 和 onmouseout 事件有意义。

**典型写法：**

|  |
| --- |
| <script>  function test(event){  var evt = window.event||event;  var src = evt.srcElement||evt.target;  // var code = evt.keyCode||evt.charCode;  alert(src.value);    }  </script>  </head>    <body>  <input type=button value=测试 onclick=test(event) />  </body> |

Event对象常用属性(仅供参考，不必记忆)：

属性：

altKey, button, cancelBubble, clientX, clientY, ctrlKey, fromElement, keyCode, offsetX, offsetY, propertyName, returnValue, screenX, screenY, shiftKey, srcElement, srcFilter, toElement, type, x, y

--------------------------------------------------------------------------------

1.altKey  
描述：  
检查alt键的状态。

语法：  
event.altKey

可能的值：  
当alt键按下时，值为 TRUE ，否则为 FALSE 。只读。

2.button   
描述：  
检查按下的鼠标键。

语法：  
event.button

可能的值：   
0 没按键   
1 按左键   
2 按右键   
3 按左右键   
4 按中间键   
5 按左键和中间键   
6 按右键和中间键   
7 按所有的键

这个属性仅用于onmousedown, onmouseup, 和 onmousemove 事件。对其他事件，不管鼠标状态如何，都返回 0（比如onclick）。

3.cancelBubble  
描述：  
检测是否接受上层元素的事件的控制。

语法：  
event.cancelBubble[ = cancelBubble]

可能的值：  
这是一个可读写的布尔值:

TRUE 不被上层原素的事件控制。   
FALSE 允许被上层元素的事件控制。这是默认值。

例子：  
下面的代码片断演示了当在图片上点击（onclick）时，如果同时shift键也被按下，就取消上层元素（body）上的事件onclick所引发的showSrc()函数。

<SCRIPT LANGUAGE="JScript">  
function checkCancel() {  
if (window.event.shiftKey)  
window.event.cancelBubble = true;  
}  
function showSrc() {  
if (window.event.srcElement.tagName == "IMG")  
alert(window.event.srcElement.src);  
}  
</SCRIPT>  
<BODY onclick="showSrc()">  
<IMG onclick="checkCancel()" SRC="sample.gif">

4.clientX  
描述：  
返回鼠标在窗口客户区域中的X坐标。

语法：  
event.clientX

注释：  
这是个只读属性。这意味着，你只能通过它来得到鼠标的当前位置，却不能用它来更改鼠标的位置。

5.clientY  
描述：  
返回鼠标在窗口客户区域中的Y坐标。

语法：  
event.clientY

注释：  
这是个只读属性。这意味着，你只能通过它来得到鼠标的当前位置，却不能用它来更改鼠标的位置。

6.ctrlKey  
描述：  
检查ctrl键的状态。

语法：  
event.ctrlKey

可能的值：  
当ctrl键按下时，值为 TRUE ，否则为 FALSE 。只读。

7.fromElement  
描述：  
检测 onmouseover 和 onmouseout 事件发生时，鼠标所离开的元素。 参考：18.toElement

语法：  
event.fromElement

注释：  
这是个只读属性。

8.keyCode  
描述：  
（请参考 <http://www.joyist.com/forum/showtopic.asp?TOPIC_ID=31&Forum_ID=2>）   
检测键盘事件相对应的内码。  
这个属性用于 onkeydown, onkeyup, 和 onkeypress 事件。

语法：  
event.keyCode[ = keyCode]

可能的值：  
这是个可读写的值，可以是任何一个Unicode键盘内码。如果没有引发键盘事件，则该值为 0 。

9.offsetX   
描述：  
检查相对于触发事件的对象，鼠标位置的水平坐标

语法：  
event.offsetX

10.offsetY   
描述：  
检查相对于触发事件的对象，鼠标位置的垂直坐标

语法：  
event.offsetY

11.propertyName  
描述：  
设置或返回元素的变化了的属性的名称。

语法：  
event.propertyName [ = sProperty ]

可能的值：  
sProperty 是一个字符串，指定或返回触发事件的元素在事件中变化了的属性的名称。   
这个属性是可读写的。无默认值。

注释：  
你可以通过使用 onpropertychange 事件，得到 propertyName 的值。

例子：  
下面的例子通过使用 onpropertychange 事件，弹出一个对话框，显示 propertyName 的值。

<HEAD>  
<SCRIPT>  
function changeProp()  
{  
btnProp.value = "This is the new VALUE";  
}

function changeCSSProp()  
{  
btnStyleProp.style.backgroundColor = "aqua";  
}  
</SCRIPT>  
</HEAD>  
<BODY>  
<P>The event object property propertyName is   
used here to return which property has been   
altered.</P>

<INPUT TYPE=button ID=btnProp onclick="changeProp()"  
VALUE="Click to change the VALUE property of this button"  
onpropertychange='alert(event.propertyName+" property has changed value")'>  
<INPUT TYPE=button ID=btnStyleProp  
onclick="changeCSSProp()"  
VALUE="Click to change the CSS backgroundColor property of this button"  
onpropertychange='alert(event.propertyName+" property has changed value")'>  
</BODY>

12.returnValue  
描述：  
设置或检查从事件中返回的值

语法：  
event.returnValue[ = Boolean]

可能的值：   
true 事件中的值被返回   
false 源对象上事件的默认操作被取消

例子见本文的开头。

13.screenX   
描述：  
检测鼠标相对于用户屏幕的水平位置

语法：  
event.screenX

注释：  
这是个只读属性。这意味着，你只能通过它来得到鼠标的当前位置，却不能用它来更改鼠标的位置。

14.screenY   
描述：  
检测鼠标相对于用户屏幕的垂直位置

语法：  
event.screenY

注释：  
这是个只读属性。这意味着，你只能通过它来得到鼠标的当前位置，却不能用它来更改鼠标的位置。

15.shiftKey  
描述：  
检查shift键的状态。

语法：  
event.shiftKey

可能的值：  
当shift键按下时，值为 TRUE ，否则为 FALSE 。只读。

16.srcElement  
描述：  
返回触发事件的元素。只读。例子见本文开头。

语法：  
event.srcElement

17.srcFilter   
描述：  
返回触发 onfilterchange 事件的滤镜。只读。

语法：  
event.srcFilter

18.toElement  
描述：  
检测 onmouseover 和 onmouseout 事件发生时，鼠标所进入的元素。 参考：7.fromElement

语法：  
event.toElement

注释：  
这是个只读属性。

例子：下面的代码演示了当鼠标移到按钮上时，弹出一个对话框，显示“mouse arrived”

<SCRIPT>  
function testMouse(oObject) {  
if(oObject.contains(event.toElement)) {  
alert("mouse arrived");  
}  
}  
</SCRIPT>  
:  
<BUTTON ID=oButton onmouseover="testMouse(this)">Mouse Over This.</BUTTON>

19.type  
描述：  
返回事件名。

语法：  
event.type

注释：  
返回没有“on”作为前缀的事件名，比如，onclick事件返回的type是click  
只读。

20. x  
描述：  
返回鼠标相对于css属性中有position属性的上级元素的x轴坐标。如果没有css属性中有position属性的上级元素，默认以BODY元素作为参考对象。

语法：  
event.x

注释：  
如果事件触发后，鼠标移出窗口外，则返回的值为 -1  
这是个只读属性。这意味着，你只能通过它来得到鼠标的当前位置，却不能用它来更改鼠标的位置。

21. y  
描述：  
返回鼠标相对于css属性中有position属性的上级元素的y轴坐标。如果没有css属性中有position属性的上级元素，默认以BODY元素作为参考对象。

语法：  
event.y

注释：  
如果事件触发后，鼠标移出窗口外，则返回的值为 -1  
这是个只读属性。这意味着，你只能通过它来得到鼠标的当前位置，却不能用它来更改鼠标的位置。

# 对话框

* 警告框
  + alert, 常用。
* 询问框
  + 返回提示框中的值。
* 确认框
  + 返回true/false.

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>  <title>Untitled Document</title>  <script>  function testFrame(){  var name = prompt("您没有登录，请输入用户名：");  alert("your name: " + name);  }    function testFrame1(){  var isLogin = confirm("您确认登录吗？");  if (isLogin) {  alert("您同意登录");  }  else {  alert("您不同意登录");  // alert("您不同意登录2");  }  }  </script>  </head>  <body>  <a href="#" onclick="testFrame();">测试询问框</a>  <a href="#" onclick="testFrame1();">测试确认框</a>  </body>  </html> |

# 常见内置对象

## 时间对象

* 使用以前必须先申明：
  + var curr=new Date();
* 主要的方法
  + getyear, getmonth, …
  + setyear, setmonth, …
  + toLoacaleString();

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Untitled Document</title>  <script>  function testDate() {    var date2 = new Date();  // date2.setYear(1999);  // date2.setMonth(10-1) ;  // date2.setDate(10); //不设的话，就是以当前时间的日期算，其他的类似。  document.write(date2.getYear()+"年"+"<br>");  document.write(date2.getMonth()+1+"月"+"<br>");  document.write(date2.getDate()+"日"+"<br>");  document.write("星期"+date2.getDay()+"<br>");  document.write(date2.getHours()+"时"+"<br>");  document.write(date2.getMinutes()+"分"+"<br>");  document.write(date2.getSeconds()+"秒"+"<br>");  document.write("从1970.01.01至今"+date2.getTime()+"毫秒"+"<br>");  document.write(date2.toLocaleString()); //本地时间，即北京时间  document.write("<br>");  document.write(date2.toGMTString()); //格林尼治时间。    }    </script>  </head>  <body>  <a href="javascript:void(0);" onclick="testDate();">测试date对象的方法</a>  </body>  </html> |

## Math对象

* **内置的Math对象可以用来处理各种数学运算**
* **可以直接调用的方法：**
  + **Math.数学函数（参数）**
* **求随机数方法**
  + **Math.random(),产生【0，1）范围的一个任意数**

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Untitled Document</title>  <script>  function testMath() {    var random1=Math.random();    var int1 = Math.round(random1\*100); //产生0-100之间的任意一个整数。较常用    document.write(random1);  document.write("<br>");  document.write(int1);  }  </script>  </head>  <body>  <a href="javascript:void(0);" onclick="testMath()">测试随机产生0-10之间的数值</a>  </body>  </html> |

## String对象

* **我们一般利用String对象提供的函数来处理字符串。String对字符串的处理主要提供了下列方法：**
  + **charAt(idx)：返回指定位置处的字符**
  + **indexOf(Chr)：返回指定子字符串的位置，从左到右。找不到返回-1**
  + **substr(m,n)：返回给定字符串中从m位置开始，取n个字符，如果参数n省略，则意味着取到字符串末尾。**
  + **substring(m,n)：返回给定字符串中从m位置开始，到n位置结束，如果参数n省略，则意味着取到字符串末尾。**
  + **toLowerCase()：将字符串中的字符全部转化成小写。**
  + **toUpperCase()：将字符串中的字符全部转化成大写。**
  + **length: 属性，不是方法，返回字符串的长度。**

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Untitled Document</title>  <script>  function testString() {  var str = "abcdefghigklmnopqrstuvwxyz";  // var str = new String("abcdefghigklmnopqrstuvwxyz");    document.write(str.charAt(2)+"<br>");  document.write(str.indexOf("f")+"<br>");  document.write(str.substring(3,5)+"<br>"); //返回给定字符串中从m位置开始，到n位置结束，如果参数n省略，则意味着取到字符串末尾。  document.write(str.substr(3,5)+"<br>"); //返回给定字符串中从m位置开始，取n个字符，如果参数n省略，则意味着取到字符串末尾。    document.write(str.toUpperCase()+"<br>");;  //document.write(str.toLowerCase());  document.write(str.length);  }  </script>  </head>  <body onload="testString();">  </body>  </html> |

# 对象机制

1. Js只是基于对象并非面向对象。它的面向对象特性没有JAVA那么强大，下表做了对比：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 抽象 | 继承 | 多态 | 封装 |
| JAVA | 具备：类，接口，抽象类四个概念 | 支持,  Extends | 支持 | 支持。四个级别封装 |
| javascript | 只有：  类，一个概念 | 支持，  prototype | 不支持 | 支持较差。两个级别封装：private,public |

# JS中类的典型定义方式

|  |
| --- |
| function Person(name,age){  this.name = name;  this.age = age;    this.study = function(){  alert("good good study! day day up!--我是："+this.name);  }  } |
| var c = new Person(“aa”,18); |

# 继承的实现

Js中继承的实现:prototype原型继承

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂JavaScript初级</title>  <script>  function test(){  var p1 = new Person("张三",29);  alert(p1.name);  p1.study();  }    function Person(name,age){  this.name = name;  this.age = age;    this.study = function(){  alert("good good study! day day up!--我是："+this.name);  }  }    function Boy(){    }  Boy.prototype = new Person("小张",18);//所有的Boy实例都以new出来的person对象为原型构造!因此具有这个对象的所有属性和方法！    String.prototype.trim = function(){ //为String类型的原型添加trim方法，这样所有的String对象都会有这个方法！  alert("我要去除首尾空格啦！");  return this.replace(/(^\s+)|(\s+$)/g,"");  };  function test2(){  var b = new Boy();  // b.name="小利斯";  b.study();    alert("aaa ".trim().length);  }    </script>  </head>  <body>  <input type=button value=测试 onclick="test();" />  <input type=button value=测试prototype继承方式 onclick="test2();" />  </body>  </html> |

# JSON

* JSON就是指：Javascript object native javascript原生对象
* JSON也是一种轻量级数据交换格式。JSON非常易于人阅读与编写，同时利于机器解析与生成。JSON是在AJAX中代替XML交换数据的更佳方案。

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂JavaScript初级</title>  <script>  function test(){  var a = {  name:"李四",  age:24  };  var b = {  name:"张三",  school:"尚学堂",  age:22,  study:function(){ //可以给这个对象定义方法！  alert("我正在学习");  },  friend:a  };    alert(a.name);  alert(b.friend.name);  b.study();  }  </script>  </head>  <body>  <input type=button value=测试 onclick="test();" />  </body>  </html> |

# Aptana开发环境和firebug

Aptana是一个基于Eclipse的[集成开发环境](http://baike.baidu.com/view/14867.htm)，其最广为人知的是它非常强悍的JavaScript[编辑器](http://baike.baidu.com/view/499644.htm)和调试器。去年Aptana吸收了Radrails项目，添加了非常强大的Ruby on Rails支持。但是Aptana前进的脚步并未停[](http://baike.baidu.com/image/d35a10f4fa06722ddcc4743c)止。随着苹果公司iPhone手机的发布，Aptana也推出了功能完备的iPhone集成开发功能，在Adobe公司的RIA产品AIR推出不久之后，Aptana就支持了AIR的开发环境。

对于我们开发人员来说，JavaScript代码经常需要我们开发和调试。Aptana给我们提供了一个相当豪华的开发调试工具，大大提高了我们开发和调试的效率。

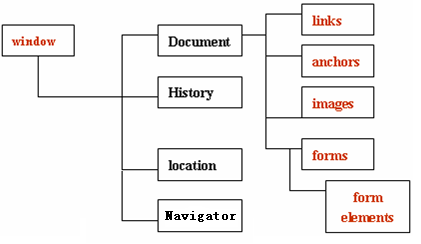
**Aptana开发环境整合及调试：**

1. 下载aptana，装上去。（(会出现一个包含四个复选框的界面，把四个勾化掉。不用aptana打开html,js,css,xml文件。)）
2. 下载firefox，装。
3. 装firebug组件：[**http://getfirebug.com/**](http://getfirebug.com/) **。 点击在线安装按钮，根据提示安装，完成后重启火狐。**

**Aptana常用快捷键：**

1. Ctrl+/ :注释
2. Ctrl+d :删除一行
3. Alt+上下箭头 ：跟上/下行交换顺序
4. Ctrl,alt+上下箭头：向上/下 复制当前行
5. Alt+左右箭头：打开的多个文件中切换
6. Ctrl+w :关闭当前编辑文档

# 浏览器对象(BOM)的层次结构



## window对象

* window对象是浏览器对象中其他大部分对象的共同的祖先，所以一般在JavaScript程序中可以省略window对象
* 常用window对象的方法
  + open(URL,windowName,parameterList)：open方法创建一个新的浏览器窗口，并在新窗口中载入一个指定的URL地址
  + close()：close方法关闭一个浏览器窗口
  + alert()等
  + **setTimeout**
  + Specifies a delay for calling a function or evaluating an expression.
  + setInterval
  + Calls a function or evaluates an expression every time the specified interval elapses.
  + clearTimeout
  + Clears a timeout that was set with the setTimeout method.
  + clearInterval
  + Clears a delay that was set with the setInterval method.

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂-javascript初级</title>  <script>  var a;  var b;  function testSetTimeout(){  // a = setTimeout(function(){ //这样直接定义匿名函数最好！可以解决浏览器差异问题!  // alert("3秒后执行！");  // },3000);      // setTimeout(ttt,3000,"3","5"); //如此传参会有浏览器差异问题。建议仍然使用匿名函数调用。如下：  setTimeout(function(){  ttt(3,4);  },3300);  }    function ttt(a,b){  alert(a+b);  }    function testSetInterval(){  var i = 1;  b = setInterval(function(){  alert("第"+(i++)+"次执行");  },3000);  }    function testClearTimeout(){  clearTimeout(a);  }    function testClearInterval(){  clearInterval(b);  }  </script>  </head>  <body>  <input type=button value=测试setTimeout onclick="testSetTimeout();"/>  <input type=button value=取消setTimeout onclick="testClearTimeout();"/>  <input type=button value=取消setInterval onclick="testClearInterval();"/>  <input type=button value=测试setInterval onclick="testSetInterval();"/>  </body>  </html> |

## History对象

* history含有以前访问过的网页的URL地址。

|  |
| --- |
| <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>history对象</title>  </head>  <body>  <a href="javascript:void(0);" onclick="javascript:history.go(-1);">后退一个</a>  <a href="javascript:void(0);" onclick="javascript:history.go(1);">前进一个</a>  </body>  </html> |

## Navigator对象

使用navigator获取浏览器信息//判断浏览器类型

|  |
| --- |
| **function** method() {  **var** c=window.navigator.userAgent.toLowerCase();  **if**(c.indexOf("msie")>0) {  alert("您用的是微软的ie浏览器");  }**else** **if**(c.indexOf("firefox")>0) {  alert("您用的是火狐");  }**else** {  alert("不知道");  }  } |

## location 对象

* location对象是当前网页的URL地址。我们可以使用Location对象来让浏览器打开某页
* 具体的语法为
  + window.location=“xxxx”
  + 这里的xxxx可以是一页也可以是一个网站的IP地址。具体请看下面的案例。

|  |
| --- |
| <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Untitled Document</title>  <script Language="JavaScript">  function testLocation() {  // alert(window.location);  alert(location);  }    function testLocation1() {  location="http://www.baidu.com";  }  </script>  </head>  <body>  <a href="javascript:void(0);" onclick="testLocation()">测试location</a>  <a href="javascript:void(0);" onclick="testLocation1()">测试location</a>  </body>  </html> |

# JAVASCRIPT操作DOM模型

## DOM模型介绍

* DOM模型的全称是：Document Object Model, 即：文档对象模型，它定义了操作文档对象的接口。
* WEB页面的HTML文档，document就是根节点，其它的子对象就是子结点.
* DOM模型在AJAX开发中的作用
  + 在ajax中，DOM模型其实是最核心的结构，是所有ajax开发的基础架构．如果没有DOM模型，就没有办法在客户端改变页面的内容，所有的局部刷新，异步请求也就无从实现。熟练掌握DOM模型的相关技术，才算真正掌握了ajax开发精髓。

**浏览器装载和显示页面的过程：**

1. 下载源代码
2. 通过页面源代码加载相关内容到内存，也就是根据HTML源码在内存中构建相关**DOM对象**。
3. 根据DOM对象的相关属性，来进行显示。

## 结点

* 结点的概念
  + 在DOM模型中，整个文档就是由层次不同的多个节点组成，可以说结点代表了全部内容。
* 结点的类型
  + 元素结点、文本结点和属性结点
    - <font id=“font1” color=“red”>hello DOM!</font>
    - 文本节点和属性结点都看作元素结点的子结点

**我们一般所说的结点指的就是元素结点**。

## 元素结点的引用

* 直接引用结点
  + 使用document.getElementById()引用指定id的结点
  + 使用document.getElementsByTagName(“a”)，将所有<a>元素结点放到一个数组中返回。
  + 使用document.getElementsByName(“abc”)，将所有name属性为”abc”的元素结点放到一个数组中返回。
* 间接引用节点
  + 引用子结点
    - 每个结点都有一个**childNodes**集合属性，类型是数组对象，表示该结点的所有子结点的集合。这些子结点按照它在文档中出现的顺序排列，因此可以通过索引来依次访问各个子结点。
    - **firstChild, lastChild**
  + 引用父节点
    - DOM模型中，除了根结点，每个结点都仅有一个父节点，可以用**parentNode**属性来引用。
  + 引用兄弟结点
    - **element.nextSibling;** //引用下一个兄弟结点
    - element.**previousSibling**; //引用上一个兄弟结点
    - 如果该结点没有相应的兄弟结点，则属性返回null.

## 浏览器差异问题

原则：如果ie和火狐发生差异冲突，尽量解决。如果太复杂不想去做，一切以IE为准。

## 结点的属性

\* nodeName:元素结点返回结点类型(即，标记名称)；属性结点返回undefined；文本节点返回"#text".

\* **nodeType:元素节点返回1，属性节点返回2，文本节点返回3**

\* nodeValue:元素节点返回null, 属性节点返回undefined, 文本结点返回文本值。

通过nodeType解决火狐和ie关于空白文本是否作为子节点的差异，比如：

|  |
| --- |
| 要得到div2的下一个元素节点：  <div id=div1 >  <div id=div2 name="div22" >aaaaa</div>  <div id=div3 >bbbb</div>  <div id=div4 >cccc</div>  </div>  //通过while循环判断nodetype类型，解决浏览器差异问题！  **while**(div2.nextSibling.nodeType!=1){  div2 = div2.nextSibling;  }  div3 = div2.nextSibling; |

## 处理属性节点

* + 元素结点.属性名称（可以读写属性值）
  + 使用setAttritbute(), getAttribute()添加和设置属性

|  |
| --- |
| **function** testAttr() {  **var** f = document.getElementById("font1");  alert(f.color);  f.color="blue";//直接操作    alert(f.getAttribute("color"));  f.setAttribute("color","green");//可动态传入操作  } |

## 处理文本结点

* 要获取一个结点内的文本，一般使用innerHTML属性
* **innerHTML属性不局限于操作一个结点，而是可以使用HTML片段直接填充页面或直接获取HTML片段，大大提高了开发的灵活性。**

|  |
| --- |
| **function** testInnerHTML(){  alert(document.getElementById("div3").innerHTML);  } |
| <div id="div3">将得到的文本</div> |

因为innerHTML属性可写，所以可以HTML片段来直接填充页面

|  |
| --- |
| **function** testInnerHTML(){  **var** dd = document.getElementById("div3");  dd.innerHTML = "<b>奥运会马上要开了！<b><h1>什么时候啊？</h1><h2>是8.8号</h2>";  } |

## 改变文档的层次结构

* 使用document.createElement方法创建元素结点
  + var divElement = document.createElement(“div")
* 使用appendChild方法添加结点
  + parentNode.appendChild(childElement);
* 使用insertBefore方法插入子结点
  + parentNode.insertBefore(newNode, beforeNode)
* 使用replaceChild方法取代子结点
  + parentNode.replaceChild(newNode, oldNode)
  + oldNode必须是已经存在的，不然会发生异常。
* 使用removeChild方法删除子结点
  + parentNode.removeChild(childNode);
* **下拉列表和表格不能用这样的DOM方法，而要用DHTML接口中的方法。**

# Javascript操作CSS

* 通过style对象改变节点的CSS
  + 利用style对象来控制元素的css, 每个style对象对应为该元素指定的css, 而每个css属性—对应于style对象的属性。
    - 对于单个单词的css属性，在style对象中的属性名称不变。
    - 对于双单词或多单词的属性改写为骆驼写法。例如：css中的font-size对应style属性为fontSize.
* 使用className属性指定结点样式
  + 结点的className属性对应于css中的类选择器。

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>javascript操作css</title>  <style type="text/css">  .red { color:red;  font-style: italic;  font-size: 32px;  }  .blue { color:blue;  font-style: italic;  font-size: 100px;  }  .green{  color:green;font-weight:bold;font-size:150px;  }  </style>  <script>    function testCSS1() {  var t = document.getElementById("div1");    // t.style.color="green";  // t.style.fontWeight="bold"; //font-weight:bold;  // t.style.fontSize="150px"; //font-size:150px;    t.style.cssText="color:green;font-weight:bold;font-size:150px;";  // alert(t.style.cssText);    }      function testCSS() {  var t = document.getElementById("div1");  t.className="green";  }  </script>  </head>    <body>  <div class="red" id="div1" >我是红色吗？</div>    <a href="javascript:void(0);" onclick="testCSS1();">测试style属性</a><br/>  <a href="javascript:void(0);" onclick="testCSS();">测试className属性</a>  </body>  </html> |

# 表单操作

## 获取表单对象的方法

* 利用表单在文档中的索引或表单的name属性来引用表单
  + document.forms[i] //得到页面中的第i个表单。
  + document.forms[formName] //得到页面中相应name的表单
* 利用表单的id属性：
  + document.getElementById(formId);
* document.formName;

## Form对象的常用属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| action | 返回或者设置action属性 |
| elements | 获取一个数组，包括该表单中所有的表单域 |
| length | 返回表单的表单域的数量 |
| method | 返回或设置表单的method属性 |
| name | 返回或设置表单的name属性 |

## Form对象的方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| submit() | 相当于单击submit按钮，表示表单提交到指定页面 |
| reset() | 相当于单击reset按钮，表示所有表单域到初始值 |

## Form对象的事件

|  |  |
| --- | --- |
| 事件 | 描述 |
| onsubmit | 在表单提交之前触发 |
| onreset | 在表单被重置之前触发 |

## 表单域通用方法

* 引用表单域的四种方法
  + form.name; form.elements[i];
  + form.elements[name]; document.getElementById(id);
* disabled ：使表单域不能接受用户的操作，变为只读
* name ：获取或设置表单域的名称
* form：获取该表单域所属的表单
* value：获取或设置表单域的值
* type ：判断表单域的类型
* focus():使表单域获得焦点
* blur()：使表单域失去焦点

## 文本域 (text, password, textarea )

* 利用value属性获取和设置文本域内容
* 利用defaultValue获得文本域的默认值。reset方法就是调用该属性。

## 单选按钮组和复选框常用操作

* 通过checked属性获得选中和未选中的状态。
* 获得单选按钮组的值：
  + 只能通过遍历单选按钮来获取被选中的值
* **复选框的处理类似单选按钮**

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Untitled Document</title>  <Script Language="JavaScript">  function getRadioValue() {  var radioGroup = document.forms[0].r;  // alert(radioGroup.value); //不能想当然用这个。必须遍历才能取到被选中的单选按钮的值  // alert(radioGroup.length);    var selected = null;    for(var i=0;i<radioGroup.length;i++) {    if(radioGroup[i].checked){  selected = radioGroup[i];  alert(selected.value);  break;  }  }  }    function getCheckboxValue(){  var group = document.form1.hobby;    var a1 = new Array();  var j=0;    for(var i=0;i<group.length;i++) {    if(group[i].checked) {  alert(group[i].value);  a1[j]=group[i].value;  j++;  }    }    }  </Script>  </head>    <body>  <form name="form1" >  <input type="radio" name="r" value="1" >奥迪</input>  <input type="radio" name="r" value="2">宝马</input>  <input type="radio" name="r" value="3">劳斯莱斯</input> <br>    驾驶技术:<br>  <input type="checkbox" name="hobby" value="1">开车</input><br>  <input type="checkbox" name="hobby" value="2">开飞机</input><br>  <input type="checkbox" name="hobby" value="3">开坦克</input><br>  <input type="checkbox" name="hobby" value="4">开轮船</input><br>  <input type="button" onclick="getRadioValue();" value="选择汽车"/><br/>  <input type="button" onclick="getCheckboxValue();" value="选择驾驶技术"/>  </form>  </body>    </html> |

## 下拉列表的使用

* 使用value属性**获取和设置**下拉列表选中选项的值
* 使用selectedIndex属性获取当前被选中选项的索引
* 使用options属性获取所有选项集合
* 使用option对象的value属性和text属性, 可以读写这两个属性。
* 使用option对象的selected属性选中该option

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Untitled Document</title>  <script>  function selectValue() {  // var cars = document.forms[0].elements["cars"];  var cars = document.form1.cars;  // alert(cars.value);  // cars.value="6";  // alert(cars.value);  // alert("selectedIndex:"+cars.selectedIndex);    var a = cars.options; //返回的是一个数组  // a[1].text="奇瑞";  // a[1].value="99";  // alert(a[1].text);  // alert(a[1].value);  // var a = cars.options;  // alert(a[1].selected);  // a[1].selected=true;  }  </script>  </head>  <body>  <form name="form1">  <select name="cars" id="idCars">  <option value="4" selected>劳斯莱斯</option>  <option value="5">宝马</option>  <option value="6">奔驰</option>  </select>  </form>    <a href="#" onclick="selectValue();">测试下拉列表</a>    </body>  </html> |

## 表单验证操作

* + 应用验证函数的两种常用的方法：
    - <input type="submit" onclick="return validate()"/>
    - <form action="a.jsp" onsubmit="return validate()"/>
    - 若返回false,则不提交表单。

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Untitled Document</title>  <script>  function validate() {  alert("validate");  //常见的验证操作  // return false;  return true;  }  </script>  </head>  <body>  <form action="05.html" name="form1">  用户名:<input type="text" name="text1" value=""/>  <input type="submit" name="submit1" value="提交" onclick="return validate();"/>  </form>  <br><br>  <form action="05.html" name="form2" onsubmit="return validate();">  用户名:<input type="text" name="text2" value=""/>  <input type="submit" name="submit2" value="提交" />  </form>  </body>  </html> |

* 注册表单：
  + 用户名长度为：5-10
  + 密码长度为：5-10
  + 确认密码框必须跟密码框的值相等
  + 爱好：学java，用java，教java。 必须至少选中一项

不符合，旁边给出提示。并且不能提交！

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <title>尚学堂-javascript初级</title>  <script>  function check(frm){  var uname = frm.uname.value;  var pwd1 = frm.pwd1.value;  var pwd2 = frm.pwd2.value;  var fs = frm.favorite;  var fs\_no = false;  var uname\_ok = true;  var pwd1\_ok = true;  var pwd2\_ok = true;    document.getElementById("unameError").innerHTML = "";  document.getElementById("pwd1Error").innerHTML = "";  document.getElementById("pwd2Error").innerHTML = "";  document.getElementById("favoriteError").innerHTML = "";      if(uname.length<5||uname.length>10){  document.getElementById("unameError").innerHTML = "用户名长度应在5-10";  uname\_ok = false;  }  if(pwd1.length<5||pwd1.length>10){  document.getElementById("pwd1Error").innerHTML = "密码长度应在5-10";  pwd1\_ok = false;  }  if(pwd1!=pwd2){  document.getElementById("pwd2Error").innerHTML = "两次输入密码不一致！";  pwd2\_ok = false;  }    for(var i=0;i<fs.length;i++){  if(fs[i].checked){  fs\_no = true; //如果有一个被选中！  break;  }  }  if(!fs\_no){  document.getElementById("favoriteError").innerHTML = "请至少选择一个爱好！";  }    return uname\_ok&&pwd1\_ok&&pwd2\_ok&&fs\_no;  }  </script>  </head>  <body>  <form action="a.jsp" onsubmit="return check(this);" >  用户名：<input type=text name=uname /> <span style="color:red;" id="unameError"></span> <br/>  密码：<input type=password name=pwd1 />  <span style="color:red;" id="pwd1Error"></span>  <br/>  确认密码：<input type=password name=pwd2 />  <span style="color:red;" id="pwd2Error"></span>  <br/>  爱好：<input type=checkbox name=favorite value=1 />学java  <input type=checkbox name=favorite value=2 />用java  <input type=checkbox name=favorite value=3 />教java  <span style="color:red;" id="favoriteError"></span>  <br/>  <input type=submit value=提交 />    </form>  </body>  </html> |

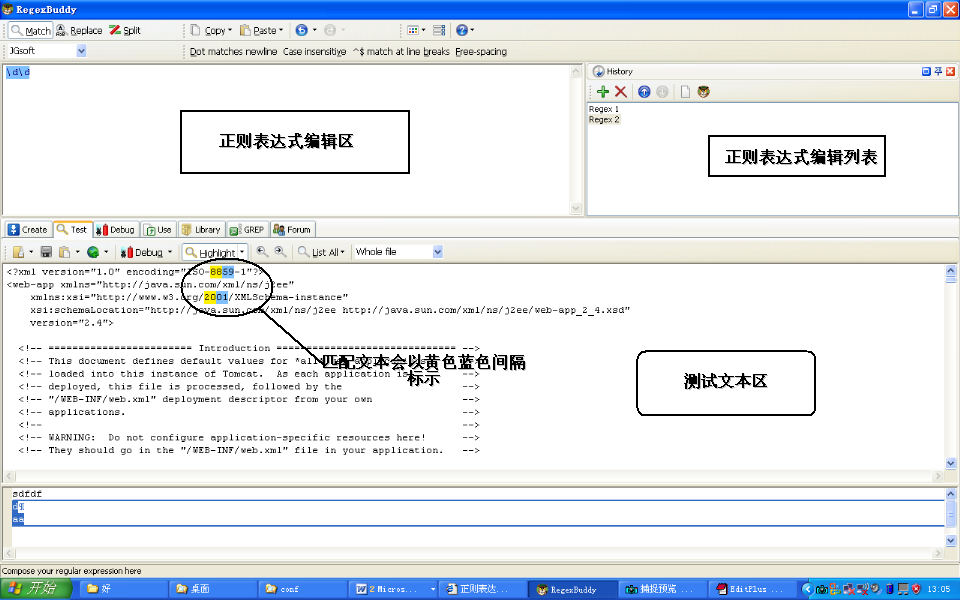
# 正则表达式基础(Regular Expression)

## 正则表达式简介

* 为什么需要正则表达式？
  + **文本的复杂处理。**
* 正则表达式的优势和用途？
  + 一种强大而灵活的文本处理工具；
  + 提供了一种紧凑的、动态的方式，能够以一种**完全通用的方式**来解决各种字符串处理（例如：验证、查找、替换等）问题；
  + 大部分语言、数据库都支持正则表达式。
* 正则表达式定义：
  + 正如他的名字一样是描述了一个规则，通过这个规则可以匹配一类字符串。
* 正则表达式的用处：
  + **验证**给定字符串是否符合指定特征，比如验证是否是合法的邮件地址。
  + 用来**查找**字符串，从一个长的文本中查找符合指定特征的字符串。
  + 用来**替换**，比普通的替换更强大

## 工具软件RegexBuddy的使用

* 为了提高开发效率，一般都先在工具软件中测试正则表达式，通过测试后，才在程序中使用。



## 正则表达式规则

* 普通字符
  + 字母、数字、汉字、下划线、以及没有特殊定义的标点符号，都是**“普通字符”**。表达式中的普通字符，在匹配一个字符串的时候，**匹配与之相同的一个字符**。
* 简单的转义字符

|  |  |
| --- | --- |
| \n | 代表换行符 |
| \t | 制表符 |
| \\ | 代表\本身 |
| \^ ，\$,\.，\(， \) ， \{， \} ， \? ， \+ ， \\* ， \| ，\[， \] | 匹配这些字符本身 |

* **标准字符集合**：能够与 ‘多种字符’ 匹配的表达式
  + （**注意区分大小写，大写是相反的意思**）

|  |  |
| --- | --- |
| \d | 任意一个数字，0~9 中的**任意一个** |
| \w | 任意一个字母**或**数字**或下划线**，也就是 **A~Z,a~z,0~9,\_ 中任意一个** |
| \s | 包括空格、制表符、换行符等**空白字符**的其中任意一个 |
| **.** | 小数点可以匹配**除了换行符（\n）以外**的任意一个字符 |

* **自定义字符集合**：[ ]方括号匹配方式，能够匹配方括号中**任意一个**字符

|  |  |
| --- | --- |
| [ab5@] | 匹配 "a" 或 "b" 或 "5" 或 "@" |
| [^abc] | 匹配 "a","b","c" 之外的任意一个字符 |
| [f-k] | 匹配 "f"~"k" 之间的任意一个字母 |
| [^A-F0-3] | 匹配 "A"~"F","0"~"3" 之外的任意一个字符 |

**注意事项：**

1. 正则表达式中的特殊符号，如果被包含于中括号中，则失去特殊意义，**但 \ [ ] : ^ - 除外**。

2. **标准字符集合，除小数点(.)外，**如果被包含于中括号中，自定义字符集合将包含该集合。  
比如：**[\d.\-+]**，将可以匹配数字，小数点和 + - 符号。**（小数点和 + 号失去特殊意义）**

* 修饰匹配次数的特殊符号

|  |  |
| --- | --- |
| {n} | 表达式重复n次 |
| {m,n} | 表达式**至少**重复m次，**最多**重复n次 |
| {m,} | 表达式至少重复m次 |
| ? | 匹配表达式0次或者1次，相当于 {0,1} |
| + | 表达式至少出现1次，相当于 {1,} |
| \* | 表达式不出现或出现任意次，相当于 {0,} |

* 匹配次数中的**贪婪模式(匹配字符越多越好)**
  + “{m,n}”, “{m,}”, “?”, “\*”, “+”，具体匹配的次数随被匹配的字符串而定。这种重复匹配不定次数的表达式在匹配过程中，总是尽可能多的匹配。
* 匹配次数中的**非贪婪模式（匹配字符越少越好）**
  + **在修饰匹配次数的特殊符号后再加上一个 "?" 号**，则可以使匹配次数不定的表达式尽可能少的匹配，使可匹配可不匹配的表达式，尽可能的 "不匹配"。
* **字符边界**(本组标记匹配的不是字符而是位置，符合某种条件的位置)

|  |  |
| --- | --- |
| ^ | 与字符串开始的地方匹配 |
| $ | 与字符串结束的地方匹配 |
| \b | 匹配一个单词边界 |

* 选择符和分组

|  |  |
| --- | --- |
| 表达式 | 作用 |
| | | 左右两边表达式之间 "或" 关系，匹配左边或者右边 |
| ( ) | (1). 在被修饰匹配次数的时候，**括号中的表达式可以作为整体**被修饰  (2). 取匹配结果的时候，括号中的表达式匹配到的内容可以被单独得到  (3). 每一对括号会分配一个编号，使用 () 的捕获根据**左括号的顺序从 1 开始自动编号**。捕获元素编号为零的第一个捕获是由整个正则表达式模式匹配的文本 |

* **反向引用（\nnn）**
  + 每一对()会分配一个编号，使用 () 的捕获**根据左括号的顺序从 1 开始自动编号**。
  + 通过反向引用，可以对**分组已捕获的字符串**进行引用。
* 非捕获组(**?:**xxx)
  + 与捕获组的区别在于不捕获匹配的文本，仅仅作为分组。**其他地方一致。**
* 模式修改符 (?ismg)\*\*\*\*\*(?-ismg) (用的不多，听听就行！)
  + 在正则表达式中间，对匹配模式进行修改。

**大小写模式修改，比如匹配a，A：**

1. **[aA]**
2. **把整个模式修改为大小写不敏感。**
3. **(?i)a(?-i)**

* **预搜索(零宽断言)** 
  + 判断当前位置的前后字符，是否符合指定的条件，但不匹配前后的字符。**是对位置的匹配。**

|  |  |
| --- | --- |
| (?=exp) | 断言自身出现的**位置的后面能**匹配表达式exp |
| (?<=exp) | 断言自身出现的**位置的前面能**匹配表达式exp |
| (?!exp) | 断言此**位置的后面不能**匹配表达式exp |
| (?<!exp) | 断言此**位置的前面不能**匹配表达式exp |

## 正则表达式的匹配模式

* IGNORECASE 忽略大小写模式
  + 匹配时忽略大小写。
  + 默认情况下，正则表达式是要区分大小写的。
* SINGLELINE 单行模式
  + 整个文本看作一个字符串，只有一个开头，一个结尾。
  + 使小数点 "." 可以匹配包含换行符（\n）在内的任意字符。
* MULTILINE 多行模式
  + 每行都是一个字符串，都有开头和结尾。
  + 在指定了 MULTILINE 之后，如果需要仅匹配字符串开始和结束位置，可以使用 \A 和 \Z

## 开发中使用正则表达式的流程

1. **分析所要匹配的数据，写出测试用的典型数据**
2. **在工具软件中进行匹配测试**
3. **在程序中调用通过测试的正则表达式**

## 课堂练习

* **电话号码**验证
  + (1)电话号码由数字和"-"构成
  + (2)电话号码为7到8位
  + (3)如果电话号码中包含有区号，那么区号为三位或四位, 首位是0.
  + (4)区号用"-"和其他部分隔开
  + (5)移动电话号码为11位
  + (6)11位移动电话号码的第一位和第二位为"13“，”15”，”18”
* **电子邮件地址验证**

1.用户名：字母、数字、中划线、下划线组成。

2.@

3.网址：字母、数字组成。

4. 小数点：**.**

5. 组织域名：2-4位字母组成。

**6.不区分大小写。**

[a-zA-Z0-9\_\-][+@[A-Za-z0-9]+(\.[a-zA-Z]{2,3}){1,2}](mailto:+@[A-Za-z0-9]+(\.%5ba-zA-Z%5d%7b2,3%7d)%7b1,2%7d)

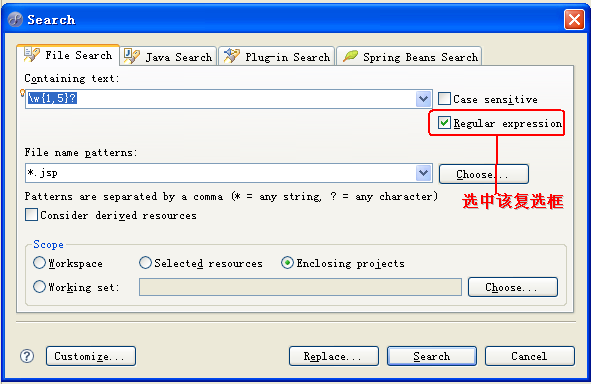
使用模式修改符：(?i)[a-z0-9\_\-][+@[a-z0-9]+(\.[a-z]{2,3}){1,2}(?-i)](mailto:+@[a-z0-9]+(\.%5ba-z%5d%7b2,3%7d)%7b1,2%7d(?-i))

* 将页面：<http://www.baidu.com/s?wd=%C6%FB%B3%B5> ，所有的以推广结尾的url抓出来！



## 开发软件中正则表达式的使用

* 文本编辑器editplus, ultraedit中使用正则表达式
  + **很多文本编辑器中并不一定支持所有正则表达式规则，使用前最好查看该软件相关文档**
* **Eclipse中使用正则表达式**



# JAVA程序中使用正则表达式

* 相关类位于：**java.util.regex**包下面
* **类 Pattern**：
  + 正则表达式的编译表示形式。
  + Pattern p = Pattern.*compile(r,int)*; //建立正则表达式，并启用相应模式
* **类 Matcher**：
  + 通过解释 [Pattern](mk:@MSITStore:F:\306\JDK_API6.0.CHM::/java/util/regex/Pattern.html) 对 [character sequence](mk:@MSITStore:F:\306\JDK_API6.0.CHM::/java/lang/CharSequence.html) 执行匹配操作的引擎
  + Matcher m = p.matcher(str); //匹配str字符串
* 编程中使用正则表达式常见情况：
  + 验证表达式是否匹配整个字符串
  + 验证表达式是否可以匹配字符串的子字符串
  + 返回给定字符串中匹配给定正则表达式所有子字符串
  + 替换给定字符串中匹配正则表达式的子字符串
  + 根据正则表达式定义规则截取字符串

|  |
| --- |
| package test;  import java.util.ArrayList;  import java.util.List;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  public class TestReg {    /\*\*  \* 验证传入的字符串是否整个匹配正表达式  \* @param regex: 正则表达式  \* @param decStr：要匹配的字符串  \* @return ：若匹配，则返回true;否则，返回false;  \*/  public static boolean validate(String regex, String decStr) {  // 表达式对象  Pattern p = Pattern.compile(regex,Pattern.CASE\_INSENSITIVE);  // 创建 Matcher 对象  Matcher m = p.matcher(decStr);  // 是否完全匹配  boolean yesorno = m.matches(); //该方法尝试将整个输入序列与该模式匹配  return yesorno;  }    /\*\*  \* 验证传入的字符串是否有子字符串匹配正表达式  \* @param regex: 正则表达式  \* @param decStr：要匹配的字符串  \* @return ：若匹配，则返回true;否则，返回false;  \*/  public static boolean validate2(String regex, String decStr) {  // 表达式对象  Pattern p = Pattern.compile(regex, Pattern.CASE\_INSENSITIVE);  // 创建 Matcher 对象  Matcher m = p.matcher(decStr);  // 是否完全匹配  boolean yesorno = m.find(); //该方法扫描输入序列以查找与该模式匹配的下一个子序列。  return yesorno;  }    /\*\*  \* 给定字符串中是否有符合给定正则表达式的子字符串，返回匹配的第一个子字符串  \* @param regex：正则表达式  \* @param decStr：要匹配的字符串  \* @return :返回匹配的第一个字符串，若不匹配则null  \*/  public static String search(String regex, String decStr) {  Pattern p = Pattern.compile(regex, Pattern.CASE\_INSENSITIVE);  Matcher m = p.matcher(decStr);  // 是否找到匹配  boolean found = m.find(); //该方法扫描输入序列以查找与该模式匹配的下一个子序列。  String foundstring="";  if(found){  foundstring = m.group();  String foundstring0 = m.group(0); //group(),group(0)返回符合整个表达式的子字符串  String foundstring1 = m.group(1); //group(1)返回符合整个表达式的子字符串中匹配第一个表达式的子字符串  String foundstring2 = m.group(2); //group(2)返回符合整个表达式的子字符串中匹配第一个表达式的子字符串  // String foundstring3 = m.group(3);  System.out.println("foundstring:"+foundstring);  System.out.println("foundstring0:"+foundstring0);  System.out.println("foundstring1:"+foundstring1);  System.out.println("foundstring2:"+foundstring2);  // System.out.println("foundstring3:"+foundstring3);  }  return foundstring;  }    /\*\*  \* 返回给定字符串中匹配给定正则表达式所有子字符串  \* @param regex  \* @param decStr  \* @return List：返回所有匹配正则表达式的子字符串  \*/  public static List searchSubStr(String regex,String decStr) {  Pattern p = Pattern.compile(regex, Pattern.CASE\_INSENSITIVE);  Matcher m = p.matcher(decStr);  List<String> list = new ArrayList<String>();  while(m.find()){ //12a345b666cwer  list.add(m.group());  }  for (String string : list) {  System.out.println(string);  }  return list;  }    /\*\*  \* 替换给定字符串中匹配正则表达式的子字符串  \* @param regex：正则表达式  \* @param decStr：所要匹配的字符串  \* @param replaceStr：将符合正则表达式的子串替换为该字符串  \* @return：返回替换以后新的字符串  \*/  public static String replace(String regex,String decStr,String replaceStr) {  Pattern p = Pattern.compile(regex, Pattern.CASE\_INSENSITIVE);  Matcher m = p.matcher(decStr);  // 替换  String newstring = m.replaceAll(replaceStr);  System.out.println(newstring);  return newstring;  }    public static void testSplit() {  String str = "abc5Adefghi7Ajklmn";  // 分割  String [] strs = str.split("(\\d)A");  for(int i=0;i<strs.length;i++) {  System.out.println(strs[i]);  }  }  public static void main(String[] args) {  replace("\\d", "dsfd;sa;ksd12a34b567c890d888e999f","\*");  // System.out.println(validate("\\d+", "334455aaa"));  // System.out.println(validate2("\\d+", "334455aaa"));  // System.out.println(searchSubStr("(\\d+)([a-z]+)", "334455aaa--3232423aaa-32324bbb"));  testSplit();  }  } |

# JAVASCRIPT中使用正则表达式

* Javascript中**正则表达式对象**的创建
  + **re = /pattern/[“flags”] 🡪 var** re1 = /\d{3}/ig;
  + **re = new RegExp("pattern",["flags"]) 🡪 var** re2 = **new** RegExp("\\d{3}","ig");
  + Flags可选项常见值：
    - **g** **（全文查找出现的所有pattern）**
    - **i （忽略大小写）**
* **RegExp对象常用方法：**
  + exec()**:** 返回的是一个数组。该数组包含了匹配该模式的第一个子字符串以及该子字符串中匹配相关分组的字符串。比如：

**var** re = **new** RegExp("(\\d+)([a-z]+)","ig");

**var** result = re.exec("33ff55tt77uu88yy");

返回的数组为：**[33ff,33,ff]**

* + **test():** **返回一个 Boolean 值，它指出在被查找的字符串中是否存在模式匹配的子字符串**
* **字符串中常用的使用正则表达式的方式：**
  + **match():使用正则表达式模式对字符串执行查找，并将符合该模式的所有子字符串包含到数组中返回。**

**var** re = **new** RegExp("(\\d+)([a-z]+)","i**g**");

**var** t = "33ff55tt77uu88yy".match(re);

**结果为数组：[33ff,55tt,77uu,88yy]**

* + search(): **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**
  + **split(regex)：**按照指定的模式切割字符串，返回一个数组。

**var** t = "asdfd33ff55tt77uu88yy".split(/\d+/);

* + **replace()**：

**var** t = "asdfd33ff55tt77uu88yy".replace(/\d+/g,"\*\*");

# 针对表单域的验证，封装一个通用的函数

1. 分析用户名和邮箱验证的共同点：
   1. 错误提示文字都会放到某一个元素中。
   2. 都不能为空
2. 分析用户名和邮箱验证的不同点：
   1. 不为空时，还需要有其他的格式验证。（用户名：长度不能超过30. 邮箱：符合邮箱规格）

**封装成的函数：**

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=GB18030" />  <title>Untitled Document</title>  <script>  **function** check(frm){  /\*  var uname = frm.userName.value.replace(/(^\s+)|(\s+$)/g,"");  var flag = true;  document.getElementById("unameMsg").innerHTML="";  document.getElementById("pwdMsg").innerHTML="";  document.getElementById("ageMsg").innerHTML="";  if(uname.length==0){  document.getElementById("unameMsg").innerHTML="用户名不能为空！";  flag = false;  }else{  if(!(/^\w+$/.test(uname))){  document.getElementById("unameMsg").innerHTML="用户名必须是：字母、数字、下划线！";  flag = false;  }  }    var pwd = frm.pwd.value.replace(/(^\s+)|(\s+$)/g,"");  if(pwd.length==0){  document.getElementById("pwdMsg").innerHTML="pwd不能为空！";  flag = false;  }else{  if(!(/^\w+$/.test(pwd))){  document.getElementById("pwdMsg").innerHTML="pwd必须是：字母、数字、下划线！";  flag = false;  }  }    var age = frm.age.value.replace(/(^\s+)|(\s+$)/g,"");  if(age.length==0){  document.getElementById("ageMsg").innerHTML="age不能为空！";  flag = false;  }else{  if(!(/^\d{3}$/.test(age))){  document.getElementById("ageMsg").innerHTML="年龄不能超过3位数或者不是数字";  flag = false;  }  }    return flag;  \*/  **var** unameFlag = checkField(frm.userName,$("unameMsg"),/^\w+$/,"用户名不能为空","用户名必须为字母、数字、下划线组成！");  **var** pwdFlag = checkField(frm.pwd,$("pwdMsg"),/^\w{6,16}$/,"pwd不能为空","用户名必须为字母、数字、下划线组成，并且长度在6-16位之间！");  **var** ageFlag = checkField(frm.age,$("ageMsg"),/^((\d{1,2})|(1[0123]\d))$/,"age不能为空","年龄不能高于139！");  **return** unameFlag&&pwdFlag&&ageFlag;  }    **function** $(a){  **return** document.getElementById(a);  }      **function** checkField(fieldObj,msgObj,re,nullMsg,errorMsg){  msgObj.innerHTML = "";  **var** v = fieldObj.value.replace(/(^\s+)|(\s+$)/g,"");  **var** flag = true;  **if**(v.length==0){  msgObj.innerHTML=nullMsg;  flag = false;  }**else**{  **if**(!(re.test(v))){  msgObj.innerHTML=errorMsg;  flag = false;  }  }    **return** flag;  }    </script>  </head>  <body>  <form action="formValidate.htm" name=testFrm onsubmit="**return** check(**this**);" >  用户名：<input type=text name=userName /> <span id=unameMsg ></span> <br>  密码：<input type=password name=pwd /> <span id=pwdMsg></span> <br>  年龄：<input type=text name=age /> <span id=ageMsg></span> <br>  <input type=submit value=注册 />  </form>  </body>  </html> |