# 2017年9月23日00:05:24

1. **html的规范**
2. 一个html文件开始标签和结束标签<html> </html>
3. Html包含两部分内容

<head>设置相关信息</head>

<body>显示在页面上的内容都写在body里面</body>

1. Html标签有开始标签，也要有结束标签
2. Html的代码不区分大小写
3. 有些标签，没有结束标签

比如：换行 <br> 在标签内结束

<hr/> 画一条水平线

<html>

<head>

<title>大家好</title>

</head>

<body>

<font size="5" color="red">这是我的第一个html程序1！</font> <br>

这是我的第一个html程序1！

</body>

</html>

1. Html的头标签,在头标签里面head

**a.<meta>标签**，设置页面的一些相关内容

<meta http-equiv="refresh" content="3;url=hello.html"/>定时跳转的标签，3秒后跳转

<meta name="keywords" content="毕姥爷，熊出没，刘翔"/> 搜索引擎用的关键字

**b. title标签**，在标题栏显示内容

**c. base标签，**设置超链接的

<a href="hello.html" target="\_blank">超链接路径1</a>

这是一个超链接标签，用了target属性，则会打开新的页面，否则在当前页面打开

<h1>头标签</h1>

<a href="hello.html" target="\_blank">超链接路径1</a>

<a href="hello.html">超链接路径2</a>

<a href="hello.html">超链接路径3</a>

<a href="hello.html">超链接路径4</a>

连接比较少的话，可以直接使用target，若有很多则不方便，此时需要在头标签里面用base标签

<html>

<head>

<title>大家好</title>

<meta name="keywords" content="毕姥爷，熊出没，刘翔"/>

<!--<meta http-equiv="refresh" content="3;url=hello.html"/>-->

<base target="\_blank"/>

</head>

<body>

<!--<font size="5" color="red">哈哈哈哈哈哈哈哈！</font> <br>-->

<h1>头标签</h1>

<a href="hello.html">超链接路径1</a>

<a href="hello.html">超链接路径2</a>

<a href="hello.html">超链接路径3</a>

<a href="hello.html">超链接路径4</a>

<hr>

</body>

</html>

此时点击任何一个连接，都会跳转到新的页面

**d． link标签**

**引入外部文件，可以引入css文件**

# 2017年9月23日09:33:23

1. 文字标签

<font></font>

属性：size：文字大小，范围是1-7

Color：颜色，两种表示方式: 英文单词red blue green

使用16进制数表示 #fffffff RGB,通过工具来实现不同的颜色

1. 注释标签

Java注释方法有3种，单行 多行 文档

HTML的注释： <!—html注释 -->

注意在浏览器里面用F12查看的时候也能看到注释

1. 标题标签、水平线标签和特殊字符

标签标签 <h1></h1> <h2></h2> <h3></h3> … <h6></h6>

标题标签可以自动换行

<html>

<head>

<title>大家好</title>

</head>

<body>

<h1>标题1</h1>

<h2>标题2</h2>

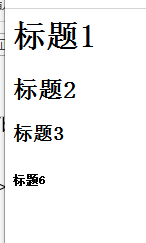
<h3>标题3</h3>

<h6>标题6</h6>

</body>

</html>

效果：



从h1到h6依次变小

1. 水平线标签，<hr/> 注意是在标签内结束，也就是斜杠写在内部

<hr size=5 color="red"/>

1. 特殊字符
2. 尖括号的转义

想要再页面上显示这样的内容： <html>:是网页的开始

需要特殊字符来转义

< 用&lt; 表示

> 用 &gt; 表示

&lt; html &gt; :是网页的开始

1. 空格的转义，输入多个空格，但页面只显示一个空格

&nbsp

&lt; html &gt; :是网&nbsp;&nbsp;&nbsp;&nbsp;页的开始

1. &符号的转义 &amp

版权符号的转义©： &copy

注册符®： &reg

1. **列表标签**

<dl></dl> 表示列表的范围

在dl里面写：<dt></dt> 上层内容

在dl里面写：<dd></dd> 下层内容

<dl>

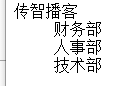
<dt>传智播客</dt>

<dd>财务部</dd>

<dd>人事部</dd>

<dd>技术部</dd>

</dl>



1. <ol></ol>标签：有序列表的

在ol标签里面 <li>具体内容</li>

属性：type，设置排序方式，数字是默认的

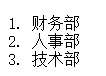
<ol>

<li>财务部</li>

<li>人事部</li>

<li>技术部</li>

</ol>



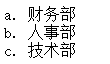
<ol ="a">

<li>财务部</li>

<li>人事部</li>

<li>技术部</li>

</ol>



**无序列表 ul标签**

**Type :默认是实心圆，circle空心圆，disc实心圆，square实心方块**

<ul type=“circle”></ul>

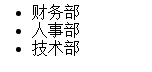
<ul>

<li>财务部</li>

<li>人事部</li>

<li>技术部</li>

</ul>



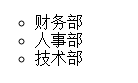
<ul type="circle">

<li>财务部</li>

<li>人事部</li>

<li>技术部</li>

</ul>



1. 图像标签

<img src=“图片的路径”>

属性：src表示路径

Width表示图片的宽度

Height表示图片的高度

Alt 图片上现实的文字

1. 路径介绍

绝对路径

相对路径

图片在html文件的同层或下层目录

<img src="img\22.jpg" alt="这是一个美女"/>

图片在html的上层文件，用两个点

<img src="..\timg.jpg" alt="这是一个美女"/>

<img src="../timg.jpg" alt="这是一个美女"/>

实测斜杠的方向不影响结果

想要表示上层的上层 ../../

1. 案例1，列表标签的使用

<html>

<head>

<title>商品列表</title>

</head>

<body>

<img src="img/1.jpg" height=200 />

<br/><br/>

首页>中国馆>女装/女士精品>所有商品

<br/><br/>

<img src="img/2.jpg" height=150 />

<h1>热点推荐</h1>

<dl>

<dt><img src="img/3.jpg" width=100/></dt>

<dd>

一口价：49.00 <br/>

全国包邮！韩版修身长袖T恤 打底衫 纯棉圆领T恤

</dd>

</dl>

<hr size=5 color="red">

<img src="img/4.jpg" width=100/><br/>

CopyRight&copy 2010易趣版权所有

</body>

</html>

1. 超链接标签

<a href=“连接到资源的路径”>显示在页面上的内容</a>

在前面讲解base标签的时候已经提到过超链接标签，

Target属性可以选择是否跳转到新的页面

<a href="test.html" target="\_blank">超链接</a>

Target属性，默认是当前页打开

\_blank 在一个新的页面打开

\_self 在当前页面打开，默认

当超链接里面不写任何内容的时候

<a href="">超链接2</a>

教程上说的是打开一个选择对话框，实测是没有任何反应（多个浏览器测试）

<a href="#">超链接2</a>

教程上说用一个#作占位符，就不会打开对话框

1. **定位资源**

如果要定位资源：定义一个位置

<a name=”top”>顶部</a>

回到这个位置

<a href=”#top”>回到顶部</a>

**引入一个标签pre:原样输出**

比如我在html里面输入：

public static void main(String[] args)

{

System.out.println("hello world");

}

这默认不会换行，全部在一行，如果把它包裹在pre标签里面，则表示，写的是什么格式，输出就是什么格式

<pre>

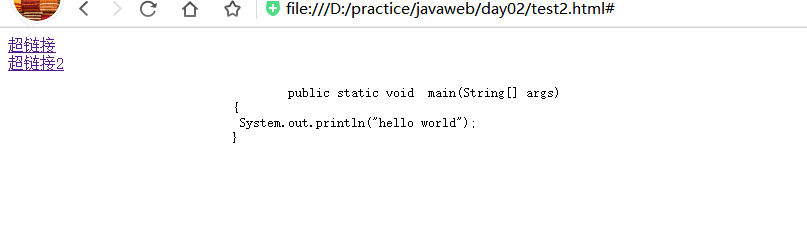
public static void main(String[] args)

{

System.out.println("hello world");

}

</pre>



<a name=”top”>顶部</a>

回到这个位置

<a href=”#top”>回到顶部</a>

将网页拉到最下面，点击回到顶部，可以回到top位置

1. **表格标签**

<table></table> 表示表格的范围

在table里面 <tr></tr> 表一行

在tr里面 <td></td> 表示一行的一格

<table border="1">

<tr>

<td>哈哈</td>

<td>哈哈</td>

<td>哈哈</td>

</tr>

<tr>

<td>哈哈</td>

<td>哈哈</td>

<td>哈哈</td>

</tr>

<tr>

<td>哈哈</td>

<td>哈哈</td>

<td>哈哈</td>

</tr>

</table>

</body>

Table的属性，border表示表格线的粗细，如果没有这个属性，则没有表格线

<table border="1" bordercolor="blue">



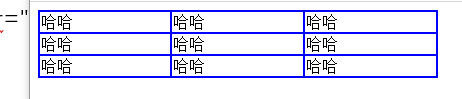
每个单元格都有单独的线，可以设置cellspacing属性，就没有间隙了

<table border="1" bordercolor="blue" cellspacing="0">



但是变粗了，并且表格太小了，可以设置宽和高

<table border="1" **bordercolor**="blue" cellspacing="0" width="400",height="150">



总结表格标签table属性：

Border 表格线

Bordercolor 表格线颜色

Cellspaceing 单元格之间的距离

Width 表格的宽度

Height 表格的高度

Tr标签属性，

设置显示方式 align，left center right左中右

<table border="1" bordercolor="blue" cellspacing="0" width="400",height="150">

<tr align="left">

<td>哈哈</td>

<td>哈哈</td>

<td>哈哈</td>

</tr>

<tr align="center">

<td>哈哈</td>

<td>哈哈</td>

<td>哈哈</td>

</tr>

<tr align="right">

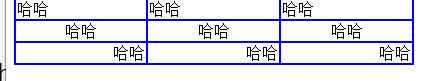
<td>哈哈</td>

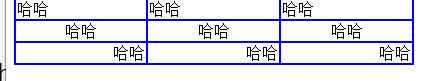
<td>哈哈</td>

<td>哈哈</td>

</tr>

</table>





也可以设置每一个单元格的对齐方式

设置td的属性

<td align="right">哈哈</td>

使用th也可以表示单元格

<tr align="right">

<td>哈哈</td>

<td>哈哈</td>

<th>哈哈</th>

</tr>



Th可以实现加粗

表格的标题 <captin><caption>写到table里面

<table border="1" bordercolor="blue" cellspacing="0" width="400",height="150">

<caption>人员信息</caption>

<tr align="left">

<td align="right">哈哈</td>

<td>哈哈</td>

<td>哈哈</td>

</tr>

<tr align="center">

<td>哈哈</td>

<td>哈哈</td>

<td>哈哈</td>

</tr>

<tr align="right">

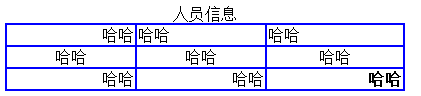
<td>哈哈</td>

<td>哈哈</td>

<th>哈哈</th>

</tr>

</table>



合并单元格

**Rowspan 跨行**

**Colspan 跨列**

<table border="1" bordercolor="red" cellspacing="0" width="400",height="150">

<!--<caption>人员信息</caption>-->

<tr align="center">

**<td colspan="3">人员信息</td>**

</tr>

<tr >

<td>哈哈</td>

<td>哈哈</td>

<td>哈哈</td>

</tr>

<tr >

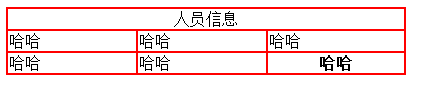
<td>哈哈</td>

<td>哈哈</td>

<th>哈哈</th>

</tr>

</table>



<table border="1" bordercolor="red" cellspacing="0" width="400",height="150">

<!--<caption>人员信息</caption>-->

<tr >

<td>哈哈</td>

<td>哈哈</td>

**<td rowspan="3">哈哈</td>**

</tr>

<tr >

<td>哈哈</td>

<td>哈哈</td>

</tr>

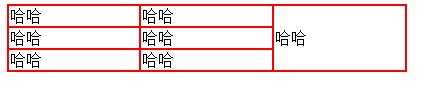
<tr>

<td>哈哈</td>

<td>哈哈</td>

</tr>

</table>



注意，本身是三行三列，但是从第一行的第三列开始往下跨了三列，那么，第二行和第三行就只有两个单元格了

Table的width属性还可以用百分数表示，比如100%，表示完全沾满屏幕

Td标签，如果某个单元格没有内容，需要用空格填充，需要用到转义字符&nbsp

<td>&nbsp</td>

1. **表单标签（非常重要）**

可以提交数据到开心网的服务器，这个过程可以使用表单标签

<form></form> 定义一个表单的范围

输入项：可以输入内容或者选择内容的部分

大部分的输入项 使用<input type=”输入项的类型”/>

1. 普通输入项 <input type=”text”/>

<html>

<head>

<title>表单标签</title>

</head>

<body>

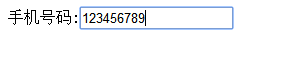
<form>

手机号码:<input type="text"/>

</form>

</body>

</html>



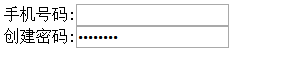
1. 密码输入项 type=“password”

<form>

手机号码:<input type="text"/> <br/>

创建密码:<input type="password"/>

</form>



1. 单选输入项 type=“radio”

<form>

手机号码:<input type="text"/> <br/>

创建密码:<input type="password"/> <br/>

性别：<input type="radio"/>女 <input type="radio"/>男

</form>



在这里需要属性 name

性别：<input type="radio"name="sex1"/>女 <input type="radio" name="sex2"/>男

教程上讲的如果没有加name属性，则默认不能选，加上了就能选，实测加不加都能选，但是两个都可以同时选，并且选后就不能取消

性别：<input type="radio" name="sex"/>女 <input type="radio" name="sex"/>男

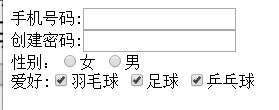
将两个选项的name属性设置为一样，表示这两个选项在一个组里面，只能同时选一个



1. 复选输入项 type=“checkbox”

在里面必须要有name属性，并且name属性必须要相同

爱好:<input type="checkbox" name="love"/>羽毛球 <input type="checkbox" name="love"/>足球 <input type="checkbox" name="love"/>乒乓球



1. 文件输入项，在文件上传的时候用到

Type=“file”

文件:<input type="file"/>



1. 下拉输入项，不是在input标签里面

生日：<select name=“birth”>

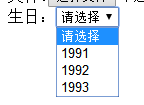
<option value ="0">请选择</option>

<option value ="1991">1991</option>

<option value="1992">1992</option>

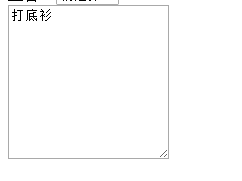
<option value="1993">1993</option>

</select>



1. 文本域

<textarea cols=”10” rows=”10”></textarea>



1. 隐藏项，不会显示在页面上，但是会存在于html代码里面

<input type=”hidden”/>

1. 提交按钮

<input type=”submit”/>

<input type="sbumit" value="注册"/>



点击注册按钮后，地址栏会有变化

<file:///D:/practice/javaweb/day02/test6.html?sex=on&love=on>

必须要在输入项里面写上name属性，才能提交

<file:///D:/practice/javaweb/day02/test6.html?phone=1615&pwd=ffdfsd&sex=on&love=on&birth=1991>

必须要有一个value值，否则结果就是on，

性别：<input type="radio" name="sex" value="nv"/>女 <input type="radio" name="sex" value="nan"/>男 <br/>

<file:///D:/practice/javaweb/day02/test6.html?phone=5555&pwd=555&sex=nan&love=y&love=z&birth=1992>

form标签的属性： **action**提交到的地址，默认提交到当前页面

<form action="test5.html">

点击提交按钮，会跳转到test5.html页面

file:///D:/practice/javaweb/day02/test5.html?phone=333&pwd=333&sex=nv&love=y&love=z&birth=1992

**method**属性：表单的提交方式，常用的有两种，get和post，默认是 get请求

**get和post的区别：**

1. get请求会存在安全性的问题，比如提交的密码会直接显示出来

如果使用了psot请求就，不会再地址栏出现那些参数了。Post不会携带参数（请求体里面）

1. post比get安全性更高
2. get请求有数据大小限制。Post没有限制，地址栏大小有有限，因此get有限制

enctype 属性：一般情况下不需要这个属性，在文件上传的时候需要

**使用图片提交** <input type="image" src="img\6.jpg"/>

默认选择值，单选框和复选框，下拉选择框

单选和复选：Checked=“checked”

下拉选择框：selected = "selected"

重置按钮

<input type=”reset”value=”重置”/>

普通按钮

<input type=”button” value=”普通按钮”>

点击没有任何效果，要和js一起使用

实例：

<html>

<head>

<title>欢迎来到人人网</title>

</head>

<body>

<h2>注册传智播客账号</h2>

<form action="test1.html" method="get">

<table width="100%">

<tr>

<td align="right">注册邮箱</td>

<td> <input type="text" name="mail" </td>

</tr>

<tr>

<td>&nbsp</td>

<td>你可以使用<a href="#">账号</a>注册或者使用<a href="#">手机号</a>注册</td>

</tr>

<tr>

<td align="right">创建密码：</td>

<td><input type="password"/></td>

</tr>

<tr>

<td align="right">真实姓名：</td>

<td><input type="text" name="realname"/></td>

</tr>

<tr>

<td align="right">性别：</td>

<td><input type="radio" name="sex" value="nv"/>女<input type="radio" name="sex" value="nan"/>男</td>

</tr>

<tr>

<td align="right">生日：</td>

<td>

<select name="year">

<option value="1991">1991</option>

<option value="1992">1992</option>

<option value="1993">1993</option>

</select>年

<select name="month">

<option value="1”>1</option>

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

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

</select>月

<select name="day">

<option value="1>1</option>

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

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

</select>日

</td>

</tr>

<tr>

<td align="right">我现在：</td>

<td>

<select name="now">

<option value="study">我正在上学</option>

<option value="work">我已经工作</option>

</select>

</td>

</tr>

<tr>

<td>&nbsp;</td>

<td><img src="img/7.jpg"width=100/><a href="#">看不清，换一张</a></td>

</tr>

<tr>

<td align="right">验证码：</td>

<td><input type="text" name="yanzhengma"/></td>

</tr>

<tr>

<td>&nbsp;</td>

<td><input type="image" src="img\6.jpg" width=100/></td>

</tr>

</table>

</form>

</body>

</html>



2017年9月23日21:03:12

1.其他常用标签

b s u ui pre sub sup div span

<html>

<head>

<title>你好</title>

</head>

<body>

<b>天之道</b>

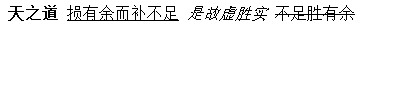
<u>损有余而补不足</u>

<i>是故虚胜实</i>

<s>不足胜有余</s>

</body>

</html>



b：加粗

s：删除线

u：下划线

i: 斜体

pre：原样输出

3 <sub>100</sub> <br/>

4 <sup>200</sup>



sub：下标

sup：上标

<div>这是div1</div>

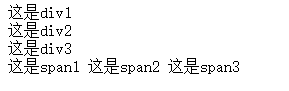
<div>这是div2</div>

<div>这是div3</div>

<span>这是span1</span>

<span>这是span2</span>

<span>这是span3</span>



Div会自动换行，span会在一行显示

P标签：段落标签，比br标签多一行

<p>哈啊哈哈哈哈哈哈哈哈哈</p>

<p>哈啊哈哈哈哈哈哈哈哈哈</p>

<p>哈啊哈哈哈哈哈哈哈哈哈</p>



**2． 框架标签（比较过时）**

<frameset>

属性：rows 按行进行划分

<frameset rows=”80,\*”>

Cols： <frameset cols=”80,\*”>

Frame具体显示的页面

<frame>

**使用框架标签的时候，不能写到body标签里面，需要去掉body标签**

<html>

<head>

<title>你好</title>

</head>

<frameset rows="100,\*">

<frame name="top" src="a.html"/>

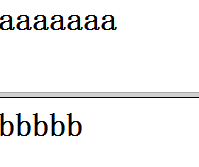
<frame name="bottom" src="b.html"/>

</frameset>

</html>

**a.html和b.html中只有简单的文字**

结果：



<frameset rows="100,\*">

<frame name="top" src="a.html"/>

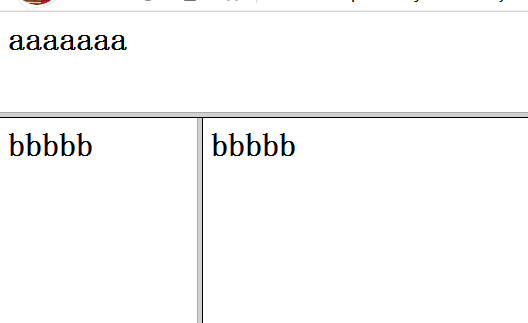
<frameset cols="80,\*">

<frame name="bottom" src="b.html"/>

<frame name="bottom" src="b.html"/>

</frameset>

</frameset>



<html>

<head>

<title>你好</title>

</head>

<frameset rows="100,\*">

<frame name="top" src="a.html"/>

<frameset cols="120,\*">

<frame name="left" src="b.html"/>

<frame name="right" src="b.html"/>

</frameset>

</frameset>

</html>

**b.html的内容：**

<html>

<head>

<title>你好</title>

</head>

<body>

<h1>bbbbb</h1>

<a href="hello.html" target="right">超链接1</a> <br/>

<a href="test.html" target="right">超链接2</a> <br/>

<a href="test8.html" target="right">超链接3</a> <br/>

</body>

</html>



点击左边的超链接，会在右边部分打开相应的html文件

# 2017年9月24日16:01:19

1. ccs 层叠样式表

层叠：一层一层的

样式表：很多属性和属性值

Css将网页内容和显示样式进行分离，提高了显示功能（html和css单独存在自己的文件中）

用来定义网页的显示效果，可以解决html代码样式定义的重复，提高了后期样式代码的可维护性，并增强了网页的显示效果功能。

1. css和html的结合方式（4种）
2. 在每个html标签上面都有一个属性style，把css和html结合在一起

<div style="background-color:red;">

《九阴真经》书中所记载的武学博大精深，威力无穷。

</div>



修改背景颜色为红色

<div style="background-color:white; color:green;">

修改文字前景色为绿色

属性和属性值之间用冒号隔开，属性之间用分号隔开。

1. 使用html的一个标签实现<style>,写在head里面

<style type=”text/css”>

Css 代码；

</style>

示例：

<html>

<head>

<title>HTML示例</title>

<style tyep="text/css">

div{

background-color:red;

}

</style>

</head>

<body>

<div >

《九阴真经》书中所记载的武学博大精深，威力无穷。

</div>

</body>

</html>

1. 在style标签里面使用语句

@import url（css文件路径）

第一步 创建一个css文件

实测好像没有效果，原来是我的 css文件里面的标点符号用成了中文；

<html>

<head>

<title>HTML示例</title>

**<style type="text/css">**

**@import url(div.css);**

**</style>**

</head>

<body>

<div >

《九阴真经》书中所记载的武学博大精深，威力无穷。

</div>

</body>

div{

background-color:yellow;

color:red;

}

**这种方式有缺点，在某些浏览器下没有效果（教程上说的）**

**用的最多的是下面的方式**

1. 使用头标签link，来引入外部css文件

第一步创建css文件

<html>

<head>

<title>HTML示例</title>

**<link rel="stylesheet" type="text/css" href="div.css"/>**

</head>

<body>

<div >

1111《九阴真经》书中所记载的武学博大精深，威力无穷。

</div>

</body>

</html>

1. Css的优先级

由上到下，由外到内，优先级由低到高。

后加载的优先级高。

1. 格式

Css的选择器，要对哪个标签里面的数据进行操作

选择器名称 {属性名：属性值：属性名：属性值 … }

基本选择器（三种）

1. 标签选择器

使用标签名作为选择器的名称

<html>

<head>

<title>HTML示例</title>

<style type="text/css">

**div{**

**background-color:orange;**

**color:white;**

**}**

**p{**

**background-color:gray;**

**color:red;**

**}**

</style>

</head>

<body>

<div >

《九阴真经》书中所记载的武学博大精深，威力无穷。

</div>

<p>

王重阳在将死之前，发现西毒欧阳锋在终南山出没，

</p>

</body>



这里p和div都是单独的标签，可分别设置样式

1. Class选择器 类选择器

每个html标签都有一个属性class

实例1：

<html>

<head>

<title>HTML示例</title>

<style type="text/css">

**div.haha{**

background-color:orange;

color:white;

}

**div.nnn{**

background-color:gray;

color:red;

}

</style>

</head>

<body>

**<div class="haha" >**

《九阴真经》书中所记载的武学博大精深，威力无穷。

</div>

<div class="nnn">

王重阳在将死之前，发现西毒欧阳锋在终南山出没，

</div>

</body>

这个例子，根据不同的类名设置不同的样式

实例2：

<html>

<head>

<title>HTML示例</title>

<style type="text/css">

**.haha**{

background-color:orange;

color:white;

}

</style>

</head>

<body>

<div class="**haha**" >

《九阴真经》书中所记载的武学博大精深，威力无穷。

</div>

<p class="**haha**">

王重阳在将死之前，发现西毒欧阳锋在终南山出没，

</p>

</body>

</html>

这个例子将类名为哈哈的都写成相同的样式

1. ID选择器

每个html标签上面都有一个属性id

<div id=”hehe”></div>

<html>

<head>

<title>HTML示例</title>

<style type="text/css">

**div#hehe{**

background-color:gray;

}

**p#hehe{**

background-color:orange;

}

</style>

</head>

<body>

**<div id="hehe" >**

《九阴真经》书中所记载的武学博大精深，威力无穷。

</div>

**<p id="hehe">**

王重阳在将死之前，发现西毒欧阳锋在终南山出没，

</p>

</body>

</html>

注意，使用id的时候，要用#号，而不是点

和上面一样，如果不同类型的标签的id一样，可以直接使用#

#hehe

{

}

1. Css的基本选择器的优先级

#hehe

{

}

div{

background-color:green;

}

.haha{

background-color:orange;

color:white;

}

<div class="haha" id=“hehe”>

《九阴真经》书中所记载的武学博大精深，威力无穷。

</div>

**Class选择器的优先级大于标签选择器，与谁先加载没关系**

**Id选择器的优先级大于class的优先级**

如果直接在div上面设置style

<div class="haha" style="background-color:blue" >

《九阴真经》书中所记载的武学博大精深，威力无穷。

</div>

**Style的优先级最高**

1. **css扩展选择器**

(1). 关联选择器

<div><p>wwwwww</p></div>

设置div标签里面的p标签里面的样式，嵌套标签里面的样式

Div p{

}

<html>

<head>

<title>HTML示例</title>

<style type="text/css">

**div p{**

**background-color:orange;**

**}**

</style>

</head>

<body>

<div >

<p >

王重阳在将死之前，发现西毒欧阳锋在终南山出没，

</p>

</div>

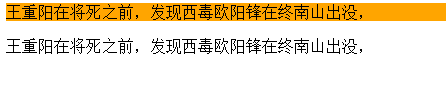
<p>

王重阳在将死之前，发现西毒欧阳锋在终南山出没，

</p>

</body>

</html>



**（2）组合选择器**

<div>11111</div>

<p>44444</p>

将div和p标签设置成相同的样式

把不同的标签设置成相同的样式

用逗号隔开

**div,p{**

background-color:orange;

}

（3）伪元素选择器

Css里面提供了一些定义好的样式，可以拿过来使用

比如 超链接

超链接的状态： 原始状态 鼠标放上去的状态 点击 点击之后

:link :hover :active : :visited

<html>

<head>

<title>HTML示例</title>

<style type="text/css">

a:link{

color:red;

}

a:hover{

color:green;

}

a:active{

color:blue;

}

a:visited{

color:gray;

}

</style>

</head>

<body>

<a href="https://daohang.qq.com/?fr=hmpage">超链接12222</a>

</body>

</html>

注意点击之后，再点后退看到的是点击之后的样式

1. **css的盒子模型**

在进行布局之前需要把数据封装到一块一块的区域内

1. 边框

Border： 2px solid blue； 粗细 样式 颜色 （用空格隔开）

Border 统一设置

上 border-top

下 border-bottom

左 border-left

右 border-right

1. 内边距

Padding 统一设置

Padding-left

Padding-right

Padding-top

Padding-bottom

<style type="text/css">

div{

width:200px;

height:100px;

border:2px solid blue;

}

#div12 {

padding: 20px;

}

</style>

1. 外边距

Margin

Margin-top margin-bottom margin-left margin-right

2017-9-24 23:00:00

1. Css布局的漂浮(了解)

Float

属性值 left right

#div11{

float:left;

}

实测和教程效果不一样

float 属性定义元素在哪个方向浮动

float常常用于制作横向配列的菜单，可以设置大小并且横向排列

1. Css布局的定位

Position

属性值 absolute 可将对象从文档流中拖出，剩余的补齐前面的空位，然后再定位该对象

可以使用top bottom等属性值进行定位

Relative 对象不可层叠，不会把对象从文档流中拖出，也可以使用top bottom等属性定位

div{

width:200px;

height:150px;

border:2px solid blue;

}

#div11

{

background-color:red;

position:absolute;

top:50px;

left:80px;

}

#div12

{

width:250px;

height:150px;

background-color:green;

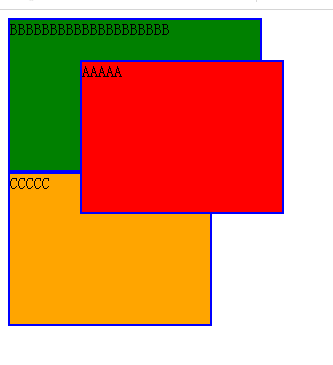
}

#div13

{

background-color:orange;

}



#div11

{

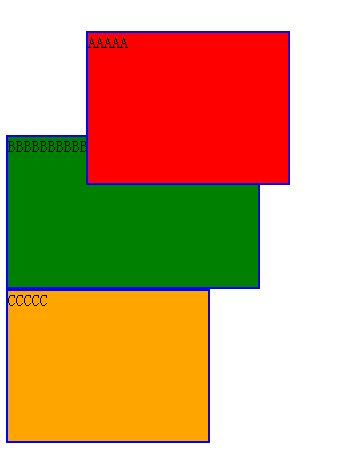
background-color:red;

position:relative;

top:50px;

left:80px;

}



1. 图文混排效果

# 2017年9月25日08:06:37

1. JavaScript简介

JavaScript是基于对象和事件驱动的语言，主要应用于客户端

提供好了很多对象，可以直接拿过来使用

Html做网站静态效果，JavaScript做动态效果

客户端指浏览器

1. Js的特点
2. 交互性

信息的动态交互

1. 安全性

不能访问本地磁盘的文件

1. 跨平台性

Java里面的跨平台性，虚拟性

只要能够支持js的浏览器，都可以运行

1. JavaScript和java的区别

没有任何关系（雷锋和雷峰塔）

1. 是不同的公司开发的工具，java是sun，现在是oracle，javascript是网景公司
2. JavaScript是基于对象的，java是面向对象的
3. Java是强类型的语言，js是弱类型的语言

比如在java里面 int i=“10” 是不正确的

在js里面 var i=10； var m=“10”；是正确的

1. JavaScript只需要解析就可以执行，而java需要先编译成字节码文件，再执行
2. JavaScript的组成(3部分)
3. ECMAScript

ECMA:欧洲计算机协会

由ecma组织制定的js的语法，语句

1. BOM

broswer object model 浏览器对象模型

1. DOM

Document object model 文档对象模型

1. Js和html的结合方式（两种）
2. 使用一个标签 <script type=”text/javascript”>js代码</script>

<html>

<head>

<title>大家好</title>

</head>

<body>

<script type="text/javascript">

//向页面弹出一个框，显示内容

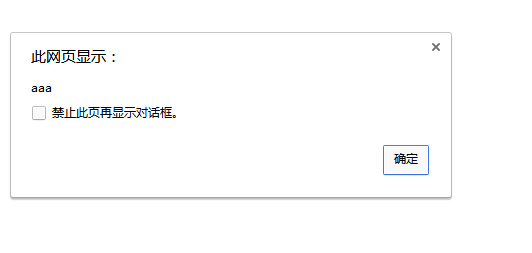
**alert("aaa");**

</script>

</body>

</html>

弹出一个框，显示内容



1. 使用script标签引入一个外部js文件

JavaScript的注释和java类似，单行，多行，但没有文档注释

<html>

<head>

<title>大家好</title>

</head>

<body>

<script type="text/javascript" src="1.js">

</script>

</body>

</html>

1.js文件内容

alert("bbbbbb");

使用第二种方式的时候，如果在内部写了js代码，也不会执行

<script type="text/javascript" src="1.js">

alert("bbbbbb");

</script>

1. Js 的原始类型和变量声明

Java的基本数据类型 byte short int long float double char Boolean

定义变量 都使用关键字 var

Js的原始类型（5个）

String 字符串 var str=“abc”

Number 数字 var m=123；

Boolean true和false var flag=true；

Null var date = new Date();

获取对象的引用，null表示对象引用为空，所有对象的引用也是object

Undefined

定义一个变量，但没有赋值

Var aa；

var str="abc";

// alert(str);

var m=123;

//alert(m);

var flag=true;

alert(flag);

1. Typeof 查看当前变量的数据类型
2. Js语句

Java里面的 语句

If，switch，for，while，do-while

**Js里面的 语句**

1. If判断语句

注意一个等于号和两个等于号的区别

var a=5;

if(a==5)

{

alert("5");

}

else

{

alert("6");

}

1. Switch语句

Java里面的支持的数据类型，string支持吗？在jdk1.7开始支持

在js里面没有限制，所有类型都支持

用法和java一样

var b=6;

switch(b)

{

case 3: alert("3");break;

case 4: alert("4");break;

case 6: alert("6");break;

default:alert("other");break;

}

1. 循环语句 for while

注意i++ 和++i的区别，这个特点和java一样

var i=5;

while(i>1)

{

alert(i);

i--;

}

for(**var** j=0;j<10;j++)

{

alert(j);

}

注意不能写int，要写var

1. Js的运算符

+=

X+=y x=x+y

大部分和java一样

不一样的：

1. var j=123;

alert(j/1000\*1000);

输出123，如果在java里面，应该输出0

**这里js不区分小数和整数**，先除以1000得到0.123，再乘以1000，得到123

（2） 字符串的相加和相减操作

var str="456";

alert(str+1);

输出：4561

var str="456";

alert(str-1);

输出：455 执行减法的运算

var str="abc";

alert(str-1);

输出:NaN 错误提示，表示不是一个数字

总结：如果是相加，做的是字符串的连接，

如果是相减，做的是真正的相减

1. Boolean也可以相加相减

var flag=true;

alert(flag+1);

输出：2

var flag=false;

alert(flag+1);

输出:1

如果把flag设置为true，等价于1，false等价于0

1. ==和===的区别

都是做判断的，

var a="5";

if(a==5)

{

alert("5");

}

else

{

alert("other");

}

输出：5

将a 定义成字符串或者数字，用==结果一样，是比较的数值

如果用===

var a="5";

if(a===5)

{

alert("5");

}

else

{

alert("other");

}

输出：other

var a="5";

if(a==="5")

{

alert("5");

}

else

{

alert("other");

}

输出：5

总结：==比较的只是值，不管类型

===比较的是值和类型

1. 引入知识

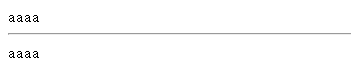
直接向页面输出语句，可以吧内容显示在页面上

document.write("aaaa");

document.write("<hr/>")

document.write("aaaa");

不仅可以输出变量，字符串，还可以输出html代码



案列，实现乘法表

<html>

<head>

<title>大家好</title>

</head>

<body>

<script type="text/javascript" >

document.write("<table border='1' bordercolor='blue'>");

for(var i=1;i<=9;i++)

{

document.write("<tr>");

for(var j=1;j<=i;j++)

{

document.write("<td>");

document.write(j+"\*"+i+"="+j\*i);

document.write("</td>");

}

document.write("</tr>");

}

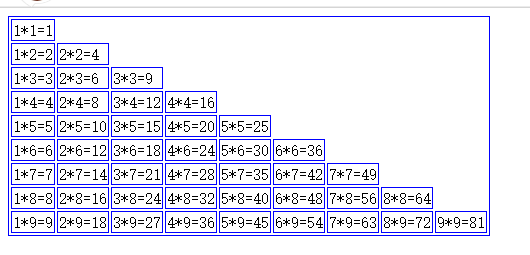
document.write("</table>");

</script>

</body>

</html>

如果不加table，那么，输出的结果没有对齐，因此要加上table，另外 document.write里面本身是双引号，因此table的 border和bordercolor属性要用单引号



1. Js数组

定义方式（3种）

1. 数据类型任意

Var arr=[1,2,3];

var arr=[1,2,3];

alert(arr);

输出：1，2，3

var arr=[1,2,"3"];

alert(arr);

也输出：1,2,3

1. 使用内置array对象

Var arr= new Array(5);

Arr[0]=”1”;

定义一个数组，参数表示元素个数

(3) var arr2= new Array(3,4,5);

定义一个数组，数组里面的元素是3 4 5

var arr2= new Array(3);

arr2[0]="a";

arr2[1]="b";

arr2[2]="c";

alert(arr2);

输出：a,b,c

数组里面有个属性 length 表示数组的长度

var arr3= new Array(5);

alert(arr3.length);

1. Js的函数

在js里面定义函数有三种方式

1. 使用到一个关键字 function

Function 方法名（参数列表）

{

方法体；

返回值可有可无；

}

function test()

{

alert("qqqq");

}

//调用方法

test();

//有参数的

**function add(a,b) //注意不要加var**

{

var sum=a+b;

alert(sum);

}

//调用方法

add(2,4);

**//有返回值**

function add2(a,b,c)

{

var sum1=a+b+c;

**return sum1;**

}

alert(add2(3,4,5));

1. 匿名函数

Var add=Function（参数列表）

{

方法体和返回值；

}

var add3= function(m,n)

{

alert(m+n);

}

//调用

add3(5,6);

（3）（用的少，了解）

使用js里面的一个内置对象 Function 注意第一字母大写

Var add= New function（“参数列表”，“方法体和返回值”）

var add4 = new Function("x,y","var sum; sum=x+y;return sum;")

alert(add4(3,4));

var canshu="x,y";

var fangfati="var sum;sum=x+y;return sum";

var add5=new Function(canshu,fangfati);

alert(add5(4,6));

12． JS的全局变量和局部变量

（1） 全局变量：在script标签里面定义一个变量，这个变量在页面中js部分都可以使用

包括不同的script标签

<html>

<head>

<title>大家好</title>

</head>

<body>

<script type="text/javascript">

var aa=10;

alert("在方法外部调用aa:"+aa);

function test()

{

alert("在方法内部调用aa"+aa);

}

test();

</script>

<script type="text/javascript">

alert("在另外的一个script标签使用："+aa);

</script>

</body>

</html>

结果：三个地方都能输出aa

（2）局部变量：在方法内部定义一个变量，只能在方法内部使用

function test11()

{

var nn=10;

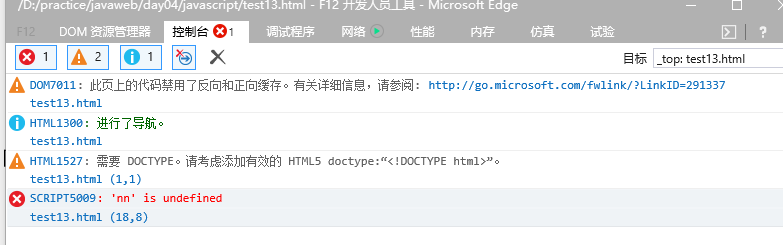
alert("在方法内部"+nn);

}

test11();

alert("在方法外部"+nn);

只能输出第一个



在ie浏览器里面，按F12，打开控制台，提示错误，nn没有定义

1. Script标签应该放的位置

<html>

<head>

<title>大家好</title>

<script type="text/javascript">

function test11()

{

var nn=10;

alert("在方法内部"+nn);

}

test11();

</script>

</head>

<body>

</body>

</html>

之前都是放到body标签里面的，现在放到head标签也可以

<html>

<head>

<title>大家好</title>

</head>

<body>

</body>

</html>

<script type="text/javascript">

function test11()

{

var nn=10;

alert("在方法内部"+nn);

}

test11();

</script>

放到html的下面也可以

<html>

<head>

<title>大家好</title>

</head>

<body>

<input type="text" name="name" id="nameid" value="aaa"/>

</body>

<script type="text/javascript">

**var input1= document.getElementById("nameid");**

alert(input1.value);

</script>

</html>

这里放到</body>的后面，如果放到其他位置，会提示错误,因为代码是从上到下解析的，下面的还没解析到，肯定就读取不到。一般放到body后面。

1. JS的重载

function add(a,b)

{

return a+b;

}

function add(a,b,c)

{

return a+b+c;

}

function add(a,b,c,d)

{

return a+b+c+d;

}

alert(add(1,2));

alert(add(1,2,3));

alert(add(1,2,3,4));

输出两次NaN,最后输出正确值

# 2017年9月25日15:28:20

1. **JS的string对象**

属性： <script type="text/javascript">

var str="abcdef";

document.write(str.length);

</script>

方法：

1. 与html相关的方法

bold（） 加粗

var str1="1321423";

document.write(str1.bold());

fontcolor（）设置字符串的颜色

var str2="1321423";

document.write(str2.fontcolor("red"));

fontsize（） 设置字符串的大小

var str3="1321423";

document.write(str3.fontsize(6));

link（） 将字符串显示成超链接

var str4="1321423";

document.write(str4.link("hello.html"));

实测，如果将link里面设置为百度的网址，无法打开，他是在本html后面加上引号中的内容

Sub和sup 表示设置下标和上标

var s1="100";

var s2="200";

var s3="300";

document.write(s1.sub());

document.write(s3);

document.write(s2.sup());



1. 与java相似的方法

Concat（）连接字符串

var str="abcdef";

var str2="123456";

document.write(str.concat(str2));

charAt（） 返回指定位置的字符，从0开始

var str3="hello";

document.write(str3.charAt(1));

document.write(str3.charAt(10)); 如果字符位置不存在，返回空字符串

indexOf() 返回字符串的位置，不存在，返回-1

var str4="hello";

document.write(str4.indexOf("o"));

注意Of的O要大写

Split（） 切分字符串成数组

var str5="a-b-c-d";

var arr1= str5.split("-")

document.write(arr1.length);

replace（） 替换字符串

传递两个参数

第一个参数是原始的字符串，第二个是要替换成的值

var str6="abcdef";

document.write(str6);

document.write("<br/>");

document.write( str6.replace("a","b"));

这样会输出bbcdef

如果写成

var str6="abcdef";

document.write(str6);

document.write("<br/>");

str6.replace("a","b");

document.write( str6);

这样两次输出结果是没有变化的

substr()

substring()

var str7="abcdefghigk";

document.write(str7.substr(5,3));

document.write("<br/>");

document.write(str7.substring(5,3));

输出：fgh

de

substr（）的两个参数，第一个表示起始位置，第二个表示截取个数

substring（）的两个参数，第一个表示其起始位置，第二个表结束位置，但不包括结束位置，并且两个参数的大小可以交换

1. JS的Array对象

创建数组（三种）

Var arr1=[1,2,3]

Var arr2= new Array(3)

Var arr3=new Array(1,2,3)

属性：length

方法： concat（） 连接数组

var arr2=[4,5,6];

document.write(arr1.concat(arr2));

**注意 JS里面的代码是区分大小写的，而html的代码是不区分大小写的**

Join() 根据指定的字符来分割数组

var arr3= new Array(3);

arr3[0]="a";

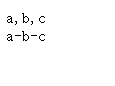
arr3[1]="b";

arr3[2]="c";

document.write(arr3);

document.write("<br/>");

document.write(arr3.join("-"));



默认是用逗号连接，使用join可以设置连接诶字符

push（） 向数组末尾添加一个或跟多元素，并返回新的长度

var arr4= new Array(3);

arr4[0]="tom";

arr4[1]="lucy";

arr4[2]="jack";

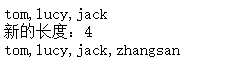
document.write(arr4);

document.write("<br/>");

document.write("新的长度："+arr4.push("zhangsan"));

document.write("<br/>");

document.write(arr4);



var arr5=["aa","bb","cc"];

var arr6=["www","xxx"];

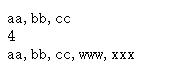
document.write(arr5);

document.write("<br/>");

document.write(arr5.push(arr6));

document.write("<br/>");

document.write(arr5);



**注意push后的长度是4，把arr6的两个字符串当成了一个整体**

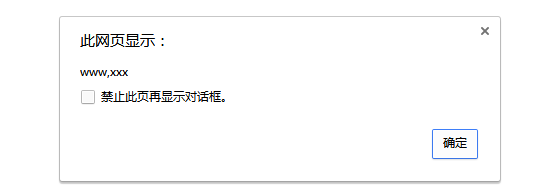
在上面的代码后面加上for循环

for(var i=0;i<arr5.length;i++)

{

alert(arr5[i]);

}



**最后一个结果是把arr6整体输出**

pop（）方法，删除最后一个元素，并返回删除的元素，没有参数

var arr7=["zhangsan","lishi","wangwu"];

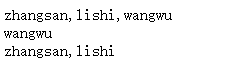
document.write(arr7);

document.write("<br/>");

document.write(arr7.pop());

document.write("<br/>");

document.write(arr7);



reverse（） 表示颠倒数组中的元素顺序

var arr8=["zhangsan","lishi","wangwu","zhaoliu"];

document.write(arr8);

document.write("<br/>");

document.write(arr8.reverse());



1. JS的date对象

在java里面获取当前时间

Date date= new Date();

//格式化

//tolocaleString()

在js里面获取当前时间

var date = new Date（）；

var date= new Date();

document.write(date);



但不是中国人习惯的格式

toLocalString（）

document.write(date.toLocaleString());



getYear() 方法，返回两位数的年份，从ECMAScript v3开始，javascript的实现就不再使用该方法，而使用

getFullYear（）方法

document.write(date.getFullYear());

输出：2017

getMonth() 获取当前的月，0-11月

document.write("<br/>");

document.write(date.getDay()); //得到当前星期

document.write("<br/>");

document.write(date.getDate()); //得到当前天

document.write("<br/>");

document.write(date.getDay()); //得到当前小时

document.write("<br/>");

document.write(date.getMinutes()); //得到当前分钟

document.write("<br/>");

document.write(date.getSeconds()); //得到当前分钟

document.write("<br/>");

document.write(date.getTime()); //**得到1970年1月日至今的毫秒数**

1506388123724

应用场景：使用毫秒数来处理缓存的效果（不有缓存）

<http://www.baidu.com>?毫秒数

2017年9月26日09:12:01

1. JS的Math对象

数学运算

math对象并不像date和string那样是对象的类，因此没有构造函数，像Math.sin()这样的函数只是函数，不是某个对象的方法。

Math.ceil() 向上舍入 （不管小数点是否大于5）

Math.floor() 向下舍入 （不管小数点是否大于5）

Math.round() 四舍五入

var mm =10.4;

document.write(Math.ceil(mm));

document.write("<br/>");

document.write(Math.floor(mm));

document.write("<br/>");

document.write(Math.round(mm));

输出： 11  
10  
10

Math.random（） 得到随机数 （伪随机数） 0.0-1.0之间的随机数

document.write("<br/>");

document.write(Math.random());

输出： 0.7534495589774102

如果要得到0-9之间的随机数,可以乘以10，但得到的仍然是小数，可以使用floor方法去掉小数部分

document.write(Math.random()\*10);

document.write("<br/>");

document.write(Math.floor(Math.random()\*10));

max（x,y）返回最大值

min(x,y) 返回最小值

pow（x,y）返回x的y次方

Math的属性，

PI ： 圆周率

LOG2E : 以2为底e的对数，约1.414

LOG10E : 以10为底e的对数，约0.434

SQRT1\_2 : 返回2的平方根除1（约等于0.707）

SQRT\_2 : 返回2的平方根（约等于1.414）

1. **JS的全局函数**

由于不属于任何一个对象，直接写名称使用

1. eval（）： 执行js代码，如果字符串是一个js代码，使用这个方法直接执行

var str="alert('123')";

eval(str);

1. encodeURI() 对字符进行编码

decodeURI() 对字符进行解码

encodeURIComponent（）

decodeURIComponent()

var str1="测试中文aaa1234";

var encode1 =encodeURI(str1);

document.write(encode1);

document.write("<hr/>");

var decode1= decodeURI(encode1);

document.write(decode1);

输出：%E6%B5%8B%E8%AF%95%E4%B8%AD%E6%96%87aaa1234

测试中文aaa1234

1. isNaN() 判断当前字符串是否是数字

返回true和flase

var str2="123";

alert(isNaN(str2));

输出：false

var str2="aaa";

alert(isNaN(str2));

输出：true

**注意输出结果：是数字则返回false，不是数字返回true**

1. parseInt（） 类型转换

parseFloat（）和上面一个函数一样，因为不区分整数和小数

var str4= "123";

document.write(str4+1);

var a= parseInt(str4)+1;

document.write("<br/>");

document.write(a);

输出：1231  
124

默认字符串加上一个数字是在后面连接该数字，减法是直接减。但可以用parseInt来把字符串转换成数字类型

1. **JS的重载**

什么是重载？方法名相同，参数不同

回顾：什么是重写？子类与父类有相同的方法名

JS的重载：不存在重载

function add1(a,b)

{

return a+b;

}

function add1(a,b,c)

{

return a+b+c;

}

function add1(a,b,c,d)

{

return a+b+c+d;

}

alert(add1(1,2));

alert(add1(1,2,3));

alert(add1(1,2,3,4));

输出：NaN NaN 10

如果把 function add1(a,b)

{

return a+b;

}

写到 alert的前面，则三个输出结果都是3，它们会调用最近的方法

**JS不存在重载，会调用最后一个方法**

**但可以通过其它方式模拟重载**

**把传递的参数保留在一个数组里面. arguments**

function add1(a,b)

{

alert("length:"+ arguments.length);

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

{

alert("value: "+ arguments[i]);

}

return a+b;

}

模拟重载：

function add1()

{

if(arguments.length==2)

{

return arguments[0]+arguments[1] ;

}

else if(arguments.length==3)

{

return arguments[0]+arguments[1] +arguments[2] ;

}

}

alert(add1(1,2));

alert(add1(1,2,3));

1. JS的bom对象

browser object model 浏览器对象模型

对象： navigator

screen

location

history

window（重点掌握）

1. navigator对象的属性

appName 返回浏览器的名称

appCodeName 返回浏览器的代码名称

document.write(navigator.appName); 在360、IE、chrome和qq浏览器里面输出：Netscape

1. screen

获取屏幕的信息，宽，高

document.write(screen.width);

document.write("<br/>");

document.write(screen.height);

输出1920 1080

1. location 对象

href 属性

获取请求的url地址，设置url地址

document.write(location.href);

输出：file:///D:/practice/javaweb/day04/js2/test10.html

设置url： 页面上放置一个按钮，按钮上绑定一个事件，当点击这个按钮，页面可以跳转到另外一个页面

**<input type="button" value="tiaozhuan" onclick="href1()"/>**

<script type="text/javascript">

function href1()

{

location.href="test1.html";

}

</script>

鼠标点击事件，onclick=“js方法”

1. history对象

实现浏览器自带的前进后退效果

到上一个页面history.back(); history.go(-1);

到下一个页面 history.forward(); history.go(1);

1. window 对象

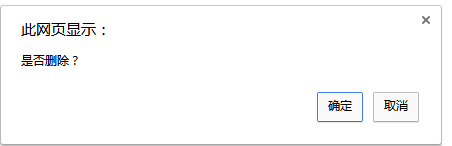
window对象是JavaScript的顶层对象

Window对象代表一个浏览器窗口或一个框架

（1） window.alert() 简写成alert（） 弹出一个对话框

（2） confirm（） 确认提示框

window.confirm("是否删除？");



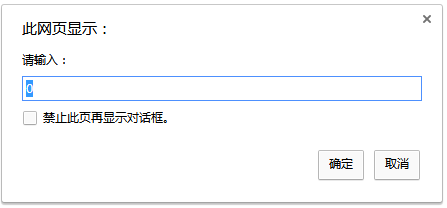
该方法有个返回值，确定返回true，取消返回false

var flag= window.confirm("是否删除？");

alert(flag);

（3） prompt（） 输入对话框

window.prompt("请输入：","0");



有两个参数，第一个在输入框上显示的内容，第二个输入框的默认值

1. **open（）打开一个新的浏览器窗口**

window.open(URL,name,features,replace)

url:新打开的窗口的url地址

name： 一般为空

features：窗口特征，比如窗口宽度和高度

创建一个按钮，打开新的窗口

<input type="button" value="open" onclick="open1()"/>

<script type="text/javascript">

function open1()

{

window.open("test1.html","","width=200,height=100");

}

</script>

这里，打开的是一个新的标签页，不是新的窗口，与浏览器的设置有关

1. close（） 关闭窗口，浏览器兼容性比较差，有些浏览器不好用

winow.close(); 实测可以关闭当前页面

1. 做定时器

setInerval（“js代码”，毫秒数）按照指定的周期（以毫秒计）来调用函数或计算表达式

setTimeout（）在指定的毫秒数后调用函数或计算表达式，只会执行一次

window.setInterval(alert(‘123’,3000);

实测，如果alert对话框没有关闭，下一个对话框不会打开

window.setTimeout("alert('123')",3000);

1. clearInterval（） 清除setInerval设置的定时器

clearInterval（） 清除setTimeout设置的定时器

<input type="button" value="interval" onclick="clear1()"/> <br/>

<input type="button" value="timeout" onclick="clear2()"/>

<script type="text/javascript">

var id1= setInterval("alert('123')",3000);

var id2 = window.setTimeout("alert('vvv')",3000);

function clear1()

{

clearInterval(id1);

}

function clear2()

{

clearTimeout(id2);

}

</script>

1. JS的DOM对象

document object model 文档对象模型

文档： 超文本温文档（超文本标记文档） html xml

对象： 提供了属性和方法

模型： 使用属性和方法操作超文本标记文档

可以使用js里面的dom提供的对象，使用这些对象的属性和方法，对标记型文档进行操作

想要对标记型文档进行操作，首先需要对标记型文档里面的所有内容封装成对象

要想对标记型文档进行操作，解析标记型文档

解析过程： 根据html的层级结构，在内存中分配一个树形结构，需要把html中的每部分封装成对象，

document对象：整个文档

element对象：标签对象

属性对象：

文本对象

node节点对象：这个对象是这些对象的父对象

如果在对象里面找不到想要的方法，这个时候到node对象里面去找

1. DHTML 是很多技术的简称

html 封装数据

css 使用属性和属性值来设置样式

dom 操作html文档

JavaScript 专门指的是js的语法语句 ecmascript

1. document 对象

表示整个文档

常用方法 ：

1. write（）方法： 向页面输出变量，输出html代码

（2） getElementById()

<input type="text" id="nameid" value="aaa"/>

<script type="text/javascript">

var input1= document.getElementById("nameid");

//得到input里面的value值

alert(input1.value);

//向input里面设置一个value值

input1.value="bbbb";

</script>

1. getElementsByName() 通过标签的name属性值得到标签，返回的是一个集合（数组）

<input type="text" name="name1" value="aaa" /> <br/>

<input type="text" name="name1" value="bbb" /> <br/>

<input type="text" name="name1" value="ccc" /> <br/>

<input type="text" name="name1" value="ddd" /> <br/>

<script type="text/javascript">

var inputs = document.getElementsByName("name1");

alert(inputs.length);

for(var i=0;i<inputs.length;i++)

{

var input1= inputs[i];

alert(input1.value);

}

</script>

1. getElementsByTagName(“标签名称”)

<input type="text" name="name1" value="aaa" /> <br/>

<input type="text" name="name1" value="bbb" /> <br/>

<input type="text" name="name1" value="ccc" /> <br/>

<input type="text" name="name1" value="ddd" /> <br/>

<script type="text/javascript">

**var inputs1= document.getElementsByTagName("input");**

alert(inputs1.length);

for(var i=0;i<inputs1.length;i++)

{

var input1= inputs1[i];

alert(input1.value);

}

</script>

只有一个元素，不用遍历

<input type="text" name="name1" value="aaa" /> <br/>

<script type="text/javascript">

var input1= document.getElementsByName("name1");

/\* alert(input1.length);\*/

var input2=input1[0];

alert(input2.value);

</script>

2017年9月28日15:32:29

1. 跨页面操作 opener：得到创建这个窗口的窗口， 得到window页面

var pwin= window.opener;

pwin.document.getElementById(“numid”).value=num1;

window.close(); //关闭窗口

上面的代码，获取创建这个窗口的窗口，然后获取窗口里面的对象，给它赋值，并关闭本窗口。

1. 案例一

点击按钮向列表添加内容

<html>

<head>

<title>大家好</title>

<style type="text/css">

</style>

</head>

<body>

<ul id="ulid">

<li>111</li>

<li>222</li>

<li>333</li>

<li>444</li>

</ul>

<br/>

<input type="button" value="add" onclick="add1();"/>

<script type="text/javascript">

function add1()

{

//获取ul标签

var ul1 = document.getElementById("ulid");

//创建标签

var li1 = document.createElement("li");

//创建文本

var text1 = document.createTextNode("555");

//把文本加入到li下面

li1.appendChild(text1);

//把li加入到ul的下面

ul1.appendChild(li1);

}

</script>

</body>

</html>

1. 获取ulr标签
2. 创建li标签

document.createElement

1. 创建文本

document.createTextNode()

1. 把文本添加到li下面

appendChild

1. 把li添加到ul末尾

appendChild

1. 元素对象 element对象

要操作element对象，首先要获取到element对象，使用document里面对于的方法

方法 ： 获取属性里面的值

getAttribute（“属性名称”）

<input type="text" id="inputid" value="aaa"/>

<script type="text/javascript">

//获取到input标签

var input1= document.getElementById("inputid");

/\* alert(input1.value);\*/

alert(input1.getAttribute("value"));

</script>

设置属性的值： setAttribute（“name”，“value”）

删除属性： removeAttribute（“name”）； 不能删除value

1. 获取ul下面子节点的个数 childNodes

<ul id="ulid"><li>111</li>

<li>222</li>

<li>333</li>

<li>444</li>

</ul>

<br/>

<script type="text/javascript">

var ul1= document.getElementById("ulid");

var lis= ul1.childNodes;

alert(lis.length);

</script>

这个方法兼容性很差，不同浏览器结果不一样，并且它把每个空格换行都当成一元素

可以使用getElementsByTagName（）方法

var ul1= document.getElementById("ulid");

/\* var lis= ul1.childNodes;

alert(lis.length);\*/

var lis =ul1.getElementsByTagName("li");

alert(lis.length);

该方法不是element里面的方法，而是document里面的方法

5． Node对象属性

nodeName

nodeType

nodeValue

使用DOM解析html的时候，需要html里面的标签，属性和文本封装成对象

标签节点对应的值

<span id="spanid">哈哈哈哈</span>

<script type="text/javascript">

var span1 = document.getElementById("spanid");

/\* alert(span1.nodeType);

alert(span1.nodeName);

alert(span1.nodeValue);\*/

//属性

var id1 = span1.getAttributeNode("id");

/\* alert(id1.nodeType);

alert(id1.nodeName);

alert(id1.nodeValue);\*/

//文本

var text1= span1.firstChild;

alert(text1.nodeType);

alert(text1.nodeName);

alert(text1.nodeValue);

</script>

标签节点对应的值：

nodeType ： 1

nodeName ： 大写标签名： 比如SPAN

nodeValue： null

属性节点对应的值：

ndoeType： 2

nodeName ： 属性名称

nodeValue： 属性的值

文本节点对应的值

nodeType ： 3

nodeName： #text

nodeValue ： 文本内容

**父节点 parentNode**

<ul id="ul1">

<li id="li1">aaaa</li>

<li id="li2">bbbb</li>

<li id="li3">cccc</li>

</ul>

<script type="text/javascript" >

var li11= document.getElementById("li1");

**var ul11 = li11.parentNode;**

alert(ul11.id);

</script> </script>

子节点 childNodes firstNode lastNode

同辈节点： nextSibling preSibling

2017年9月29日16:23:59

1. 操作dom树
2. appendChild方法，

<html>

<head>

<title>大家好</title>

<style type="text/css">

#div1{

width:200px;

height: 150px;

border:3px dashed green;

}

#div2{

width:250px;

height: 150px;

border:2px solid red;

}

</style>

</head>

<body>

<div id="div1">

<ul id="ulid11">

<li>tom</li>

<li>jack</li>

<li>marry</li>

</ul>

</div>

<div id="div2">

</div>

<input type="button" value="add1" onclick="add1();"/>

<script type="text/javascript">

function add1()

{

var div2 = document.getElementById("div2");

var ul11= document.getElementById("ulid11");

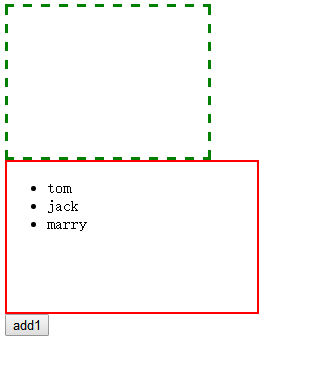
**div2.appendChild(ul11); //类型剪切粘贴效果**

}

</script>

</body>

</html>



点击按钮后，第一个div里面的内容会跑到第二个div里面。

1. insertBefore（newNode，oldNode）；

在某个节点之前插入一个新的节点

两个参数：要插入的节点，在谁之前插入

要插入的节点不存在，要创建

创建标签，创建文本，把文本添加到标签上

<ul id="ulid11">

<li id = "li11">zhangsan</li>

<li id = "li22">lishi</li>

<li id = "li33">wnagwu</li>

</ul>

<input type="button" value="add1" onclick="insert1();"/>

<script type="text/javascript">

function insert1()

{

var li33= document.getElementById("li33");

var li44 = document.createElement("li");

var text44 = document.createTextNode("赵六");

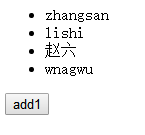
li44.appendChild(text44);

**var ul11= document.getElementById("ulid11");**

**ul11.insertBefore(li44,li33);**

}

</script>



注意：没有insertAfter（）方法

1. removeChild（） 删除节点

通过父节点删除，不能自己删除自己

1. replaceChild（newNode，oldNode） 替换节点

通过父节点替换子节点，不能自己替换，要先创建新的节点

<ul id="ulid11">

<li id = "li11">紫衫龙王</li>

<li id = "li22">白眉鹰王</li>

<li id = "li33">金毛狮王</li>

<li id = "li44">青翼蝠王</li>

</ul>

<input type="button" value="替换" onclick="replace1();"/>

<script type="text/javascript">

function replace1()

{

var li55 = document.createElement("li");

var text55 = document.createTextNode("张无忌");

li55.appendChild(text55);

var ul11= document.getElementById("ulid11");

var li44 = document.getElementById("li44");

ul11.replaceChild(li55,li44);

}

</script>

1. cloneNode（boolean） 复制节点，参数表示是否复制子节点

<ul id="ulid11">

<li id = "li11">紫衫龙王</li>

<li id = "li22">白眉鹰王</li>

<li id = "li33">金毛狮王</li>

<li id = "li44">青翼蝠王</li>

</ul>

<div id="div11">

</div>

<input type="button" value="复制" onclick="copy1();"/>

<script type="text/javascript">

function copy1()

{

// 把ul列表复制到另外一个div里面

//1.获取到ul

//2. 执行复制方法，cloneNode方法复制true

//3. 把复制之后的内容放到div里面

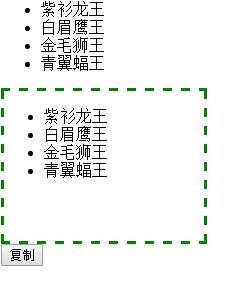
var ul11= document.getElementById("ulid11");

var ulcopy = ul11.cloneNode(true);

var div11= document.getElementById("div11");

div11.appendChild(ulcopy); //如果这里参数是ul11，则原来的就没了

}



总结：

获取节点使用方法：getElementById（） getElementsByTagName（） getElementsByName

插入节点方法： appendChild 在末尾添加，剪切粘贴

insertBefore 在某个节点之前添加

删除节点 ： removeChild 通过父节点删除

替换节点 ：replaceChld 通过父节点替换

1. innerHTML属性

该属性不是DOM的属性，但是大多数浏览器都支持的属性

第一个作用： 获取文本内容

第二个作用： 向标签里面设置内容

（1）

<span id="sid">哈哈哈哈</span>

<script type="text/javascript">

var span1 = document.getElementById("sid") ;

**alert(span1.innerHTML);**

</script>

输出：哈哈哈哈



<div id="div11">

</div>

<script type="text/javascript">

var span1 = document.getElementById("sid") ;

// alert(span1.innerHTML);

var div11= document.getElementById("div11");

**div11.innerHTML="<h1>AAAA</h1>"**

</script>

动态显示时间：

<div id="div11">

</div>

<script type="text/javascript">

function getD1()

{

var date = new Date();

var d1 = date.toLocaleString();

var div1 = document.getElementById("div11");

div1.innerHTML = d1;

}

setInterval("getD1()",1000);

</script>

**全选全不选案例:**

<html>

<head>

<title>大家好</title>

<style type="text/css">

</style>

</head>

<body>

<input type="checkbox" id="boxid" onclick="selAllNo();">全选/全不选</input> <br/>

<input type="checkbox" name="love">篮球</input> <br/>

<input type="checkbox" name="love">排球</input> <br/>

<input type="checkbox" name="love">羽毛球</input> <br/>

<input type="checkbox" name="love">乒乓球</input> <br/>

<input type="button" value="全选" onclick="selAll();"/>

<input type="button" value="全不选" onclick="selNo()"/>

<input type="button" value="反选" onclick="setOther()"/>

<script type="text/javascript">

function selAll()

{

var loves= document.getElementsByName("love");

for(var i=0;i<loves.length;i++)

{

var love1= loves[i];

love1.checked=true;

}

}

function selNo()

{

var loves= document.getElementsByName("love");

for(var i=0;i<loves.length;i++)

{

var love1= loves[i];

love1.checked=false;

}

}

function setOther()

{

var loves= document.getElementsByName("love");

for(var i=0;i<loves.length;i++)

{

var love1= loves[i];

if(love1.checked==true)

{

love1.checked=false;

}

else

{

love1.checked=true;

}

}

}

function selAllNo()

{

var box1= document.getElementById("boxid");

if(box1.checked==true)

{

selAll();

}

else

{

selNo();

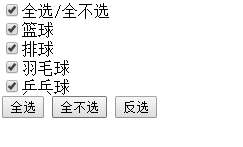
}

}

</script>

</body>

</html>



**案列下拉列表左右选择**

下拉选择框有个multiple属性，如果设置multiple="multiple" 则会将下拉选择框展开

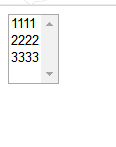
<select multiple="multiple">

<option>1111</option>

<option>2222</option>

<option>3333</option>

</select>



<html>

<head>

<title>大家好</title>

<style type="text/css">

</style>

</head>

<body>

<div id="s1" style="float:left;">

<div>

<select id="select1" multiple="multiple" style="width:100px;height:100px">

<div>

<option>1111</option>

<option>2222</option>

<option>3333</option>

<option>4444</option>

<option>5555</option>

</div>

</select>

<div>

<input type="button" value="选中添加到右边" onclick="selToRight();"/><br/>

<input type="button" value="全部添加到右边" onclick="allToRight();"/>

</div>

</div>

</div>

<div>

<div>

<select id="select2" multiple="multiple" style="width:100px;height:100px">

<option>aaaaa</option>

</select>

</div>

<div>

<input type="button" value="选中添加到左边" onclick="selToLeft();"/> <br/>

<input type="button" value="全部添加到左边" onclick="allToLeft();"/>

</div>

</div>

<script type="text/javascript">

function selToRight()

{

var select1 = document.getElementById("select1");

var select2 = document.getElementById("select2");

var options1 = select1.getElementsByTagName("option");

for(var i=0;i<options1.length;i++)

{

var option1=options1[i];

if(option1.selected==true)

{

select2.appendChild(option1);

i--;

}

}

}

function allToRight()

{

var select1 = document.getElementById("select1");

var select2 = document.getElementById("select2");

var options1 = select1.getElementsByTagName("option");

for(var i=0;i<options1.length;i++)

{

var option1=options1[i];

select2.appendChild(option1);

i--;

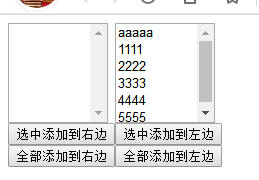
}

}

</script>

</body>

</html>



案例：城市联动

<html>

<head>

<title>大家好</title>

<style type="text/css">

</style>

</head>

<body>

<select id="country" onchange="add1(this.value);">

<option value="0">--请选择--</option>

<option value="中国">中国</option>

<option value="美国">美国</optioN>

<option value="日本">日本</option>

<option value="德国">德国</option>

</select>

<select id="city"></select>

<script type="text/javascript">

**//二维数组**

var arr=new Array(4);

arr[0]=["中国","南京","抚州","临洮","日喀则","哈密"];

arr[1]=["美国","华盛顿","底特律","休斯顿","纽约"];

arr[2]=["德国","慕尼黑","柏林","法兰克福","狼堡"];

arr[3]=["日本","北海道","东京","大阪","广岛","长崎"];

function add1(var1)

{

var city1= document.getElementById("city");

//得到city里面的option

var options1=city1.getElementsByTagName("option");

for(var m=0;m<options1.length;m++)

{

var op=options1[m];

city1.removeChild(op);

m--;

}

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

{

var arr1=arr[i];

var firstvalue= arr1[0];

if(var1==firstvalue)

{

for(var j=1;j<arr1.length;j++)

{

var value1=arr1[j];

var option1= document.createElement("option");

var text1= document.createTextNode(value1);

option1.appendChild(text1);

city1.appendChild(option1);

}

}

}

}

</script>

</body>

</html>

总结：selcet里面有个onchange事件，表示改变选择的事件，然后有个参数this.value,这样就能将选择的option的value传递个该事件方法。

案例：动态生成表格

<html>

<head>

<title>大家好</title>

<style type="text/css">

</style>

</head>

<body>

行：<input type="text" id="h"/>

列：<input type="text" id="l"/>

<br/>

<div id="divv">

</div>

<input type="button" value="生成" onclick="add2();"/>

<script type="text/javascript">

function add2()

{

var h = document.getElementById("h").value;

var l = document.getElementById("l").value;

var tab= "<table border='1' bordercolor='blue'>";

for(var i=1;i<h;i++)

{

tab+="<tr>";

for(var j=1;j<=l;j++)

{

tab+="<td>aaaa</td>";

}

tab+="</tr>";

}

tab+="</table>";

var divv= document.getElementById("divv");

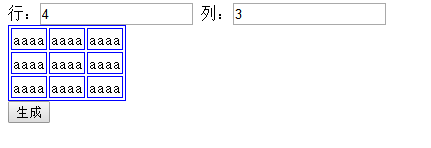
**divv.innerHTML=tab;**

}

</script>

</body>

</html>



2017年9月30日16:40:42

1. 表单提交方式

（1）使用submit提交

<form>

<input type=“submit”/>

</form>

（2）使用button提交

<html>

<head>

<title>dajiahao</title>

</head>

<body>

<form id="form1" action="hello.html">

<input type="text" name="username"/> <br/>

<input type="button" value="tijao" onclick="form1();"/>

</form>

</body>

<script type="text/javascript">

function form1()

{

var form1= document.getElementById("form1");

form1.submit();

}

</script>

</html>

1. 使用超链接提交

<form id="form1" action="hello.html">

<input type="text" name="username"/> <br/>

<input type="button" value="tijao" onclick="form1();"/>

</form>

<a href="hello.html?username=13456">使用超链接提交</a>

1. onfocus（） 事件： 得到焦点

onblur（） 事件: 失去焦点

1. xml extensive markup language 可扩展标记型语言

html也是标记型语言

可扩展： html里面的标签是固定的，每个标签都有特定的韩式

标签可以自定义，可以写中文标签<person></person>

xml的用途： html是用于显示数据，xml也可以显示数据（不是主要功能）

xml的主要功能是存储数据

xml的应用： 不同系统之间传输数据

qq之间的数据传输

用来表示生活中有关系的数据

经常用在配置文件

1. xml的语法
2. xml的文档声明

创建一个文件，后缀名是.xml

如果写xml，第一步，必须要有一个文档声明，（写了文档声明后，表示写xml文件内容）

<？xml version=“1.0” encoding=“gbk”？>

**文档声明必须要写在第一行第一列，不能空行或空格**

属性 version有1.0和1.1，我们使用1.0

encoding:xml编码gbk utf-8 iso8859-1（不包含中文）

standalone： yes或no 是否需要依赖其他文件

1. 定义元素（标签）
2. 定义属性（\*\*\*）
3. 注释
4. 特殊字符
5. CDATA
6. PI指令

<?xml version="1.0" encoding="gbk"?>

<person>

<name>zhangsan</name>

<age>20</age>

</person>

能运行

<?xml version="1.0" encoding="utf-8"?>

<person>

<name>张三</name>

<age>20</age>

</person> 这个不能运行

因为第二个文件，默认是保存为gbk的，与设置的编码方式不一致，在编辑器里面设置保存编码方式和设置的编码方式一致，就不会出现乱码。

1. xml的元素定义

标签定义

标签定义有开始必须要有结束。<person></person>

标签没有内容，必须在标签内结束

标签可以嵌套，但必须合理嵌套

合理嵌套：<aa><bb></bb></aa>

不合理嵌套： <aa><bb></aa></bb>

一个xml中，只能有一个根标签，其他标签都是这个标签下面的标签

在xml中吧空格和换行都当成内容来解析

下面这两段代码含义是不同的

<aa>1111111</aa>

<aa>

1111111

</aa>

xml种标签的名称规则

1. xml种的代码区分大小写

<p></P> 这两个标签是不一样的

(2) xml的标签不能以数字和下划线开头

不能以xml或XML开头

(3) xml的标签不能包含空格和冒号

(4) xml的标签可以是中文

1. xml中的属性的定义

html是标记型文档，可以有属性

xml也是标记型文档，可以有属性

<person id1=“aa”></person>

属性定义要求

1. 一个标签上可以有多个属性
2. 属性名称不能相同
3. 属性名称和属性值之间用=，属性值使用引号包起来（可以是单引号，也可以是双引号）
4. xml属性的名称规范和元素的名称规范一致
5. xml的注释

<! -- xml的注释 -->

注释里面不能嵌套

1. 转义字符

& &amp

< &lt

> &gt

“ &quot

‘ &apos

如果想要再xml中显示a<b,直接写不能正常显示，因为把<当作了标签

1. CDATA区

可以解决多个字符都需要转义的操作 if（a<b&&b<c）

把这些内容放到CDATA区里面，不需要转义了

写法：

<![CDATA[内容 ]]>

1. PI 指令 （处理指令）

可以在xml中设置样式

<?xml version="1.0" encoding="utf-8"?>

**<?xml-stylesheet type="text/css" href="1.css"?>**

<person>

<name>zhangsan</name>

<age>20202022020</age>

<![CDATA[if(a<b&&b<c&&d>f){}]]>

</person>

name

{

background-color:red;

}

age

{

background-color:green;

}



对中文的标签名称不起作用

1. xml的约束

dt 约束 schema 约束

1. dtd的快速入门

创建一个文件，后缀名.dtd

1. 看xml中有多少个元素，有几个元素，在dtd中写几个<!ELEMENT>
2. 判断元素是简单元素还复杂元素

简单元素：没有子元素

<!ELEMENT 元素名称（#PCDATA）>

复杂元素：有子元素

<!ELEMENT 元素名称（子元素）>

1. 需要在xml文件中引入dtd文件

<!DOCTYPE 根元素名称 SYSTEM “dtd文件路径”>

打开xml文件使用浏览器打开的，浏览器只负责校验xml语法，不负责校验约束

如果想要校验xml的约束，需要使用工具

1. dtd的三种引入方式
2. 引入外部的dtd文件

<!DOCTYPE person SYSTEM "1.dtd">

1. 使用内部的dtd文件
2. 使用外部的dtd文件，使用网络上的dtd文件

<!DOCTYPE 根元素 PUBLIC “DTD名称”“DTD文档的url” >

2017年10月6日16:47:09

1. 使用dtd定义属性

语法： <!ATTLIST 元素名称

属性名称 属性类型 属性约束

>

属性类型：

CDATA: 字符串

枚举： 表示只能在一定的范围内出现，但是只能每次出现其中的一个

（aa|bb|cc）

ID: 值只能是字母或者下划线开头，不是是数字

<!ATTLIST name

ID3 ID #REQUIRED

>

属性的约束

#REQUIRED 属性必须存在

#IMPLIED 属性可有可无

#FIXED 表示一个固定值 #FIXED “AAA”

属性值必须是一个固定的值

直接值：

1. 定义引用实体

<!ENTITY 实体名称 “实体的值”>

<!ENTITY TEST “HAHAHEHE”>

使用实体 &实体名称； 比如 &TEST; 注意后面的分号

注意：定义实体需要写在内部dtd里面，如果写在外部dtd里面，有些浏览器下，内容得不到

1. xml的解析

xml是标记型文档

js使用dom解析标记型文档

根据html的层级结构，在内存中分配一个树型结构，属性和文本都封装成对象

document对象、element对象、属性对象、文本对象、node节点对象

**xml的解析方式：dom和sax**

使用dom方式解析xml的时候，如果文件过大，造成内存溢出

使用dom的优点：很方便实现增删改操作

sax解析过程：

采用事件驱动，边读边解析，从上到下，一行一行的解析，解析到某一个对象，把对象名称返回，

使用sax方式不会造成内存溢出，实现查询。使用sax方式不能实现增删改操作。

想要解析xml，首先需要解析器

不同的公司和组织提供了针对dom和sax方式的解析器，通过api方式提供

sun公司提供了针对dom和sax的解析器： jaxp

dom4j组织，提供的解析器：dom4j (实际开发中使用最多)

jdom组织：jdom

1. jaxp的api查看

jaxp是javase的一部分

jaxp解析器在jdk的javax.xml.parsers包里面

四个类：分别是针对dom和sax解析使用的类

dom：DocumentBuilder 解析器类，这是一个抽象类，不能new，

此类的实例可以从DocumentBuilderFactory.newDocumentBuilder()方法获取

一个方法，可以解析xml parse（“xml路径”） 返回Document整个文档

返回的document是一个接口，父节点是node

在document里面方法

getElementsByTagName（string tagname）这个方法可以得到标签

返回集合nodelist

createElement（string tagName）创建标签

createTextNode（string data）创建文本

appendChild（Node child）把文本添加到标签下面

removeChild（Node oldChild）删除节点

getParentNode（） 获取父节点

NodeList

getLength（）得到集合的长度

item（Int index）下标取到具体的值

for（int i=0;i<list.getLength();i++）

{

list.item(i);

}

getTextContent（）方法：得到标签里面的内容，返回String

DocumentBuilderFactory 解析器工厂，则也是一个抽象类

newInstance（） 获取DocumentBuilderFactory 的实例

sax： SAXParser：解析器类

SAXParserFactory：解析器工厂

1. 使用jaxp实现查询操作

在myeclipse里面使用alt加斜线：alt+/ 可以实现代码提示

输出System.out.println(s);的快捷键： 输入syso然后输入atl+/

查询xml中所有的name元素的值

步骤：

1. 创建解析器工厂
2. 根据解析器工厂创建解析器
3. 解析xml返回document
4. 得到所有的name元素
5. 返回集合，遍历集合，得到每一一个name元素

先创建一个xml文件：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<person>

<p1>

<name>zhangsan</name>

<age>20</age>

</p1>

<p2>

<name>lishi</name>

<age>30</age>

</p2>

</person>

再创建一个java文件，包含main方法

package cn.itcast.jaxptest;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

public class TestJaxp {

public static void main(String[] args) throws Exception

{

//查询所有name元素的值

//创建解析器工厂

//根据解析器工厂创建解析器

//解析xml返回document

DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder builer= builderFactory.newDocumentBuilder();

Document document = builer.parse("src/tv.xml");

NodeList list = document.getElementsByTagName("name");

for(int i=0;i<list.getLength();i++)

{

Node name1= list.item(i);

String s = name1.getTextContent();

System.out.println(s);

}

}

}

查询xml中第一个name元素的值

在myeclipse里面提取方法的快速方法，即将一段代码放到一个新的方法里面，然后调用这个方法。首选选中这段代码，（这段代码必须符合要求，例如不能多了一个括号），然后右键选择refactor，然后选择extract method，然后输入方法名即可。最后这段代码会转到一个新的方法里面，并且原代码的位置变成了这个新方法。

//查询xml中第一个name元素的值

**public** **static** **void** selectSin() **throws** ParserConfigurationException, SAXException, IOException

{

DocumentBuilderFactory builderfactory = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder= builderfactory.newDocumentBuilder();

Document document= builder.parse("src/tv.xml");

NodeList list= document.getElementsByTagName("name");

Node name1 = list.item(0);

String s = name1.getTextContent();

System.***out***.println(s);

}

6. 使用jaxp添加节点

**public** **static** **void** addSex() **throws** Exception

{

/\*

\* 1. 创建解析器工厂

\* 2. 根据解析器工厂创建解析器

\* 3. 解析xml，返回document

\* 4. 得到第一个p1，使用item方法下标得到

\* 5. 创建sex标签createElement

\* 6. 创建文本createTextNode

\* 7. 把文本添加到sex下面appendChild

\* 8. 把sex添加到第一个p1下面

\* 9. 回写xml

\*

\*/

DocumentBuilderFactory builderfactoyr = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = builderfactoyr.newDocumentBuilder();

Document document= builder.parse("src/tv.xml");

NodeList list = document.getElementsByTagName("p1");

Node p1= list.item(0);

Element sex1 = document.createElement("sex");

Text text1 = document.createTextNode("nv");

sex1.appendChild(text1);

p1.appendChild(sex1);

TransformerFactory transformerFactory = TransformerFactory.*newInstance*();

Transformer transformer = transformerFactory.newTransformer();

transformer.transform(**new** DOMSource(document), **new** StreamResult("src/tv.xml"));

}

注意，前面的操作都是在内存中的操作，最后必须要回写，即将修改结果写入xml文件中，否则看不到结果。

回写需要用到TransformerFactory和Transformer类，然后调用transform方法

TransformerFactory transformerFactory = TransformerFactory.*newInstance*();

Transformer transformer = transformerFactory.newTransformer();

transformer.transform(**new** DOMSource(document), **new** StreamResult("src/tv.xml"));

7.修改节点内容

//修改第一个p1下面的sex内容是nan

**public** **static** **void** modifySex() **throws** Exception{

/\*

\* 1. 创建解析器工厂

\* 2. 根据解析器工厂创建解析器

\* 3. 解析xml，返回document

\* 4. 得到sex，item方法

\* 5. 修改sex里面的值，setTextContent

\* 6. 回写xml

\*/

DocumentBuilderFactory builderfactoyr = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = builderfactoyr.newDocumentBuilder();

Document document= builder.parse("src/tv.xml");

Node sex1 = document.getElementsByTagName("sex").item(0);

sex1.setTextContent("nan");

TransformerFactory transformerFactory = TransformerFactory.*newInstance*();

Transformer transformer = transformerFactory.newTransformer();

transformer.transform(**new** DOMSource(document), **new** StreamResult("src/tv.xml"));

}

1. 使用jaxp删除节点

//删除<sex>nan</sex>节点

**public** **static** **void** delSex() **throws** Exception

{

/\*

\* 1. 创建解析器工厂

\* 2. 根据解析器工厂创建解析器

\* 3. 解析xml，返回document

\* 4. 得到sex，item方法

\* 5. 获取sex的父节点

\* 6. 使用父节点删除removeChild方法

\* 7. 回写xml

\*/

DocumentBuilderFactory builderfactoyr = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = builderfactoyr.newDocumentBuilder();

Document document= builder.parse("src/tv.xml");

Node sex1 = document.getElementsByTagName("sex").item(0);

Node p1= sex1.getParentNode();

p1.removeChild(sex1);

TransformerFactory transformerFactory = TransformerFactory.*newInstance*();

Transformer transformer = transformerFactory.newTransformer();

transformer.transform(**new** DOMSource(document), **new** StreamResult("src/tv.xml"));

}

1. 使用jaxp遍历节点

//遍历节点，把所有元素名称打印出来

**public** **static** **void** listElement() **throws** Exception

{

/\*

\* 1. 创建解析器工厂

\* 2. 根据解析器工厂创建解析器

\* 3. 解析xml，返回document

\* 4. 得到根节点

\* 5. 得到根节点的 子节点

\* 6. 得到根节点的子节点的子节点

\*

\*/

DocumentBuilderFactory builderfactoyr = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = builderfactoyr.newDocumentBuilder();

Document document= builder.parse("src/tv.xml");

//编写一个方法实现遍历操作

*list1*(document);

}

**private** **static** **void** list1(Node node) {

//判断是元素类型时才打印

**if**(node.getNodeType()==Node.***ELEMENT\_NODE***)

{

System.***out***.println(node.getNodeName());

}

//得到一层子节点

NodeList list = node.getChildNodes();

//遍历list

**for**(**int** i=0;i<list.getLength();i++)

{

//得到每一个节点

Node node1 = list.item(i);

*list1*(node1);

}

}

使用了递归的方法来遍历。并且判断节点是否是元素类型，是的话才打印。因为xml会将空格和换行当成一个节点，不判断的话会打印很多#text。

注意上面的document相关包都是导入w3c的包

2017年10月7日15:22:59

1. xml的schema约束

dtd语法： <!ELEMENT 元素名称 约束>

schema 符合xml语法

一个xml中可以有多个schema，只能有一个dtd

多个schema使用名称空间来区分，类似于java里面的包名

dtd里面有PCDATA类型，但是在schema里面可以支持更多的数据类型

比如：年龄 只能是整数，在schema里面可以直接定义一个整数类型

schema语法更加复杂，schema目前不能替代dtd

1. schema的快速入门

创建一个schema文件,后缀名是.xsd

根节点： <schema>

在schema文件里面：

属性 xmlns=<http://www.w3.org/2001/XMLSchema> 表示当前xml文件是一个约束文件

targetNamespace=<http://www.itcast.con/2017> 使用schema约束文件，直接通过这个地址引入约束文件

elementFormDefault="qualified"

步骤：（1）看xml中有多少个元素

<element>

（2）看简单元素和复杂元素

如果复杂元素

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<schema xmlns=*"http://www.w3.org/2001/XMLSchema"*

targetNamespace=*"http://www.itcast.con/2017"*

elementFormDefault=*"qualified"*>

<element name=*"person"*>

<complexType>

<sequence>

<element name=*"name"* type=*"string"*></element>

<element name=*"age"* type=*"int"*></element>

</sequence>

</complexType>

</element>

</schema>

如果是简单元素，写在复杂元素的sequence里面

1. 在被约束的xml文件里面引入约束文件

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<person xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://www.itcast.con/2017"*

xsi:schemaLocation=*"http://www.itcast.con/2017 1.xsd"*

>

<name>zhangsan</name>

<age>20</age>

</person>

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

表示xml是一个被约束的文件

xmlns=*"http://www.itcast.con/2017"*

是schema里面的targetNameSpace

xsi:schemaLocation=*"http://www.itcast.con/2017 1.xsd"*

两部分：targetNamespace和约束文档的地址路劲，两个之间有空格

**xmlschema复杂元素指示器**

<sequence> 表示元素的出现顺序

注意sequence里面是按照name和age的顺序来的，如果在被约束的文件里面先写age，再写name是要报错的。

<all> 表示元素只能出现一次

<choice> 表示元素只能出现其中的一个

maxOccurs="unbounded" 表示元素的出现的次数

<any></any> 表示任意元素

可以约束属性

写在复杂元素里面，在写</complexType>之前

<attribute name="id1" type="int" use="required"></attribute>

name：属性名称

type ： 属性类型，int string

use ： 属性是否必须出现 required

**复杂的schema约束**

对于多个schema约束，需要引入多个命名空间，不同的命名空间可以使用别名

xmlns =“http：//www.exmaple.org/company”

xmlns :xsi =“http：//www.exmaple.org/company”

xsi就是第二个名称空间的别名

如果每个schema约束下都有相同的元素，比如name，name调用的时候要加别名来区分

<xsi:name>zhangsan</xsi:name>

myeclipse注释的快捷键 ctrl+shift+/

取消注释的快捷键 ctrl+shift+\

1. **sax解析原理**

解析xml有两种技术dom和sax

sax方式： 事件驱动哦，边读边解析

在javax.xml.parsers包里面

SAXParser类，是一个抽象类，此类的实例可以从SAXParserFactory.newSAXParser（）方法获得

parse（File f,DefaulHandler dh）

两个参数: 第一个xml的路劲

第二个事件处理器

SAXParserFactory

实例newInstance（）方法得到

sax的执行过程

当解析到开始标签的时候，自动执行startElement方法

当解析到文本的时候，自动执行characters方法

当解析到结束标签的时候，自动执行endElement方法

1. 使用jaxp的sax方式解析xml

sax方式不能是现在增删改操作，只能实现查询操作

在myeclipse里面生成main方法的快捷键，先输入main，然后输入alt+/

在myeclipse里面快速实现父类父方法，右键空白处，选择source，选择overrid/implement methods

执行parse方法，第一个参数xml路径，第二个参数事件处理器

创建一个类，继承DefaultHandler，重写类里面的三个方法startElement characters

endElement

**package** cn.itcast.jaxpsax;

**import** java.io.IOException;

**import** javax.xml.parsers.ParserConfigurationException;

**import** javax.xml.parsers.SAXParser;

**import** javax.xml.parsers.SAXParserFactory;

**import** org.xml.sax.Attributes;

**import** org.xml.sax.SAXException;

**import** org.xml.sax.helpers.DefaultHandler;

**public** **class** TestSax {

**public** **static** **void** main(String[] args) **throws** SAXException, IOException, ParserConfigurationException {

/\*

\* 1. 创建解析器工厂

\* 2. 创建解析

\* 3. 执行parse方法

\* 4. 自己创建一个类，继承DefaultHnadler

\* 5. 重写类里面的三个方法

\*/

SAXParserFactory saxparserFactory= SAXParserFactory.*newInstance*();

SAXParser saxparser = saxparserFactory.newSAXParser();

saxparser.parse("src/p1.xml",**new** MyDefault1());

}

}

**class** MyDefault1 **extends** DefaultHandler

{

@Override

**public** **void** startElement(String uri, String localName, String qName, Attributes attributes) **throws** SAXException {

// **TODO** Auto-generated method stub

//super.startElement(uri, localName, qName, attributes);

System.***out***.println("<"+qName+">");

}

@Override

**public** **void** characters(**char**[] ch, **int** start, **int** length) **throws** SAXException {

// **TODO** Auto-generated method stub

//super.characters(ch, start, length);

System.***out***.println(**new** String(ch,start,length));

}

@Override

**public** **void** endElement(String uri, String localName, String qName) **throws** SAXException {

// **TODO** Auto-generated method stub

//super.endElement(uri, localName, qName);

System.***out***.println("</"+qName+">");

}

}

获取所有name元素的值

**class** MyDefault2 **extends** DefaultHandler

{

**boolean** flag = **false**;

@Override

**public** **void** startElement(String uri, String localName, String qName, Attributes attributes) **throws** SAXException {

//判断qName是否是name元素

**if**("name".contentEquals(qName))

{

flag=**true**;

}

}

@Override

**public** **void** characters(**char**[] ch, **int** start, **int** length) **throws** SAXException {

// **TODO** Auto-generated method stub

//super.characters(ch, start, length);

//当flag值是true的时候，表示解析到name元素

**if**(flag==**true**)

{

System.***out***.println(**new** String(ch,start,length));

}

}

@Override

**public** **void** endElement(String uri, String localName, String qName) **throws** SAXException {

// **TODO** Auto-generated method stub

//super.endElement(uri, localName, qName);

//把flag设置成false，表示name元素结束

**if**("name".contentEquals(qName))

{

flag=**false**;

}

}

}

获取第一个name元素的值

**class** MyDefault2 **extends** DefaultHandler

{

**boolean** flag = **false**;

int idx=1;

@Override

**public** **void** startElement(String uri, String localName, String qName, Attributes attributes) **throws** SAXException {

// **TODO** Auto-generated method stub

//super.startElement(uri, localName, qName, attributes);

//判断qName是否是name元素

**if**("name".contentEquals(qName))

{

flag=**true**;

}

}

@Override

**public** **void** characters(**char**[] ch, **int** start, **int** length) **throws** SAXException {

// **TODO** Auto-generated method stub

//super.characters(ch, start, length);

//当flag值是true的时候，表示解析到name元素

**if**(flag==**true**&&idx==1)

{

System.***out***.println(**new** String(ch,start,length));

}

}

@Override

**public** **void** endElement(String uri, String localName, String qName) **throws** SAXException {

// **TODO** Auto-generated method stub

//super.endElement(uri, localName, qName);

//把flag设置成false，表示name元素结束

**if**("name".contentEquals(qName))

{

flag=**false**;

idx++;

}

}

}

2017年10月7日19:00:20

1. 使用dom4j解析xml

dom4j 是一个组织，针对xml解析，提供解析器 dom4j

dom4j不是java se的部分，想要使用第一步需要怎么做?

导入dom4j提供的jar包

创建一个文件夹lib，复制jar包到lib下面，右键点击jar包，build path – add to buildpath

看到jar包变成了奶瓶的样子，表示导入成功

得到document

SAXReader reader= new SAXReader（）；

Document document = reader.read(url) ;

document的父接口是Node，如果在document里面找不到想要的方法，可以在node里面去找

document里面的getRootElement（）方法 获取根节点，返回的是Element，

Element也是一个接口，父接口是node  
 element和node里面的方法：

getParent（） 获取父节点

addElement（） 添加标签

2. 使用dom4j实现查询操作

查询所有name元素里面的值

1. 创建解析器
2. 得到document
3. 得到根节点 getRootElement()
4. 得到所有的p1标签

element (qname) 表示获取标签下面的第一个子标签，qname是标签名称

elements(qname) 获取标签下面的所有子标签（一层标签），qname是标签名称

elements() 获取标签下面的所有一层子标签

1. 得到name 在p1下面执行element（“name”）方法，返回Element
2. 得到name里面的值 getText方法得到值

**package** cn.itcast.dom4jtest;

**import** java.util.List;

**import** org.dom4j.Document;

**import** org.dom4j.DocumentException;

**import** org.dom4j.Element;

**import** org.dom4j.io.SAXReader;

**public** **class** TestDom4j {

**public** **static** **void** main(String[] args) **throws** DocumentException {

// **TODO** Auto-generated method stub

*selectName*();

//System.out.println("fsdfdfs");

}

//查询xml中所有name元素的值

**public** **static** **void** selectName() **throws** DocumentException

{ /\*

\* 1. 创建解析器

\* 2. 得到document

\* 3. 得到根节点

\* 4. 得到p1

\* 5. 得到p1下面的name

\* 6. 得到name里面的值

\*/

SAXReader saxReader = **new** SAXReader();

Document document = saxReader.read("src/p1.xml");

Element root = document.getRootElement();

List<Element> list = root.elements("p1");

**for**(Element element :list)

{

Element name1 = element.element("name");

String s = name1.getText();

System.***out***.println(s);

}

}

}

查询第一个name元素的值

//获取到第一个name元素里面的值

**public** **static** **void** selectSin() **throws** DocumentException

{

/\*

\* 1. 创建解析器

\* 2. 得到document

\* 3. 得到根节点

\* 4. 得到第一个p1元素

\* 5. 得到p1下面的name

\* 6. 得到name里面的值

\*/

SAXReader saxReader = **new** SAXReader();

Document document = saxReader.read("src/p1.xml");

Element root = document.getRootElement();

Element p1 = root.element("p1");

Element name1 = p1.element("name");

String s = name1.getText();

System.***out***.println(s);

}

查询第二个name元素的值

//获取第二个name元素里面的值

**public** **static** **void** selectSecond() **throws** Exception

{/\*

\* 1. 创建解析器

\* 2. 得到document

\* 3. 得到根节点

\* 4. 得到说哟的p1

\* 5. 遍历的得到第二个p1

\* 6. 得到第二个p1下面的name

\* 7. 得到name的值

\*/

SAXReader saxReader = **new** SAXReader();

Document document = saxReader.read("src/p1.xml");

Element root = document.getRootElement();

List<Element> list = root.elements("p1");

//得到第二个p1 list集合下标从0开始

Element p2 = list.get(1);

Element name2 = p2.element("name");

String s = name2.getText();

System.***out***.println(s);

}

1. 使用dom4j实现添加需求

在第一个p1标签末尾添加一个元素<sex>nv</sex>

//在第一个p1标签末尾添加一个元素<sex>nv</sex>

**public** **static** **void** addSex() **throws** Exception

{ /\*

\* 1. 创建解析器

\* 2. 得到document

\* 3. 得到根节点

\* 4. 获取到第一个p1 ，使用element方法

\* 5. 在第一p1下面添加元素

在p1上面直接使用addElement（“标签名称”）方法，返回一个element

\* 6. 在添加完成之后的元素下面添加文本

在sex上直接使用setText(“文本内容”)方法

\* 7. 回写xml

格式化 OutputFormat，使用createPrettyPrint方法，表示一个漂亮的格式

使用XMLWriter 直接new这个类，传递两个参数

第一个参数：是xml文件路径，new FileOutputStream（“路径”）

第二个参数是格式化类的值

\*/

SAXReader saxReader = **new** SAXReader();

Document document = saxReader.read("src/p1.xml");

Element root = document.getRootElement();

Element p1 = root.element("p1");

Element sex1 = p1.addElement("sex");

sex1.setText("nv");

OutputFormat format = OutputFormat.*createPrettyPrint*();

XMLWriter xmlWriter = **new** XMLWriter(**new** FileOutputStream("src/p1.xml"),format);

xmlWriter.write(document);

xmlWriter.close();

}

其中

OutputFormat format = OutputFormat.*createPrettyPrint*();

设置xml的格式化，带缩进，样式非常好。

如果是

OutputFormat format = OutputFormat.createCompactFormat();

则结果是全部压缩成一行了

1. 使用dom4j在特定的位置添加一个元素

在第一个p1下面的age标签之前添加<school>ecit.edu.cn</school>

//在第一个p1下面的age标签之前添加<school>ecit.edu.cn</school>

**public** **static** **void** addAgeBefore() **throws** Exception{

/\*

\* 1. 创建解析器

\* 2. 得到document

\* 3. 得到根节点

\* 4. 获取到第一个p1

\* 5. 获取p1下面的所有元素

\* elements()方法，返回list集合

\* 使用list里面的方法，在特定的位置添加元素

\* add(int index,E element)

\* 第一个参数是位置下标，从0开始

\* 第二个参数是要添加的元素

首先要创建元素，在元素下面创建文本

使用DocumentHelper类方法createElement创建标签

把文本添加到标签下面，使用steText（“文本内容”）方法

\* 6. 回写xml

\*/

SAXReader saxReader = **new** SAXReader();

Document document = saxReader.read("src/p1.xml");

Element root = document.getRootElement();

Element p1 = root.element("p1");

List<Element> list = p1.elements();

//创建元素

Element school = DocumentHelper.*createElement*("school");

school.setText("ecit.edu.cn");

list.add(1, school);

OutputFormat format = OutputFormat.*createPrettyPrint*();

XMLWriter xmlWriter = **new** XMLWriter(**new** FileOutputStream("src/p1.xml"),format);

xmlWriter.write(document);

xmlWriter.close();

}

1. 使用dom4j修改节点的操作

//修改第一个p1下面age的值

**public** **static** **void** modifyAge() **throws** Exception

{

/\*

\* 1. 创建解析器

\* 2. 得到document

\* 3. 得到根节点

\* 4. 获取到第一个p1

\* 5. 得到第一个p1下面的age

\* 6. 修改值

\* 6. 回写xml

\*/

SAXReader saxReader = **new** SAXReader();

Document document = saxReader.read("src/p1.xml");

Element root = document.getRootElement();

Element p1 = root.element("p1");

Element age1 = p1.element("age");

age1.setText("30");

OutputFormat format = OutputFormat.*createPrettyPrint*();

XMLWriter xmlWriter = **new** XMLWriter(**new** FileOutputStream("src/p1.xml"),format);

xmlWriter.write(document);

xmlWriter.close();

}

1. 使用dom4j实现删除节点的操作

myeclipse里面导入包的快捷键： ctrl+shift+o, 需要先用鼠标点中要到包的类

//删除第一个p1下面的school>ecit.edu.cn</school>

**public** **static** **void** delSch() **throws** Exception

{ /\*

\* 1. 创建解析器

\* 2. 得到document

\* 3. 得到根节点

\* 4. 获取到第一个p1

\* 5. 得到第一个p1下面的school

\* 6. 使用p1删除school

\* 7. 回写xml

\*/

SAXReader saxReader = **new** SAXReader();

Document document = saxReader.read("src/p1.xml");

Element root = document.getRootElement();

Element p1 = root.element("p1");

Element sch = p1.element("school");

p1.remove(sch);

OutputFormat format = OutputFormat.*createPrettyPrint*();

XMLWriter xmlWriter = **new** XMLWriter(**new** FileOutputStream("src/p1.xml"),format);

xmlWriter.write(document);

xmlWriter.close();

}

删除要使用父节点来删除，要得到父节点可以直接得到，因为是通过父节点找到子节点的。另外一种是通过子节点的getParent方法来得到父节点。

1. 7. 使用dom4j实现获取属性的操作
2. 获取第一个p1里面的属性id1的值

//获取第一个p1里面的数学id1的值

**public** **static** **void** getValues() **throws** Exception

{ /\*

\* 1. 创建解析器

\* 2. 得到document

\* 3. 得到根节点

\* 4. 获取到第一个p1

\* 5. 得到p1里面的属性的值

\*/

SAXReader saxReader = **new** SAXReader();

Document document = saxReader.read("src/p1.xml");

Element root = document.getRootElement();

Element p1 = root.element("p1");

String s = p1.attributeValue("id1");

System.***out***.println(s);

}

<person>

<p1 id1=*"aaa"*>

<name>zhangsan</name>

<age>30</age>

<sex>nv</sex>

</p1>

<p1>

<name>lishi</name>

<age>60</age>

</p1>

</person>

重点方法：attributeValue

1. 使用dom4j支持xpath的操作

可以直接获取到某个元素，不需要一层一层的解析

第一种形式 /AAA/DDD/BBB 表示一层一层的，AAA下面的DDD下面的BBB

第二种形式 //BBB 表示和这个名称相同，表示只要是BBB，都能得到

第三种形式 /\* 表示所有元素

第四种形式 BBB[1]表示第一个BBB元素 BBB[last()] 表示最后一个BBB元素

第五种形式 //BBB[@id] 表示只要BBB元素上面有id属性，就都能得到

第六种形式 //BBB[@id=’b1’] 表示元素的名称是BBB，在BBB上面有id属性，并且id属性的值的b1

1. 在默认的情况下，dom4j不支持xpath，如果想要在dom4j里面使用xpath

引入支持xpath的jar包 jaxen-1.1.6.jar

在dom4j里面提供了两个方法，用来支持xpath

selectNodes（“xpath表达式”） 获取多个节点

selectSingleNode（“xpath表达式”） 获取单一节点

使用xpath实现：查询xml中所有name元素的值

所有name元素的xpath表示： //name

使用selectNodes（“//name”）

//查询xml中所有name元素的值

**public** **static** **void** test1() **throws** Exception

{

/\*

\* 1.得到document

\* 2.直接使用selectNodes("//name")方法得到所有name元素

\*

\*/

SAXReader saxReader = **new** SAXReader();

Document document = saxReader.read("src/p1.xml");

List<Node> list = document.selectNodes("//name");

**for**(Node node : list)

{

String s= node.getText();

System.***out***.println(s);

}

}

使用xpath实现： 获取第一个p1下面的name的值

//p1[@id=’aaa’]/name

//使用xpath实现： 获取第一个p1下面的name的值

**public** **static** **void** test2() **throws** Exception

{

/\*

\* 1.得到document

\* 2.直接使用selectSingleNode()方法

\*/

SAXReader saxReader = **new** SAXReader();

Document document = saxReader.read("src/p1.xml");

Node name1 =document.selectSingleNode("//p1[@id1='aaa']/name");

String s = name1.getText();

System.***out***.println(s);

}

1. 实现简单的学生管理系统

使用xml当做数据库，存储学生信息

创建一个xml文件，写一些学生信息

在myeclipse里面快速生成get和set方法，右键空白处选择source，然后选择generate getter and setter

在myeclipse里面快速生成toString方法，右键空白处，选择source，然后选择generate toString

增加操作：

/\*

\* 1.创建解析器

\* 2.得到document

\* 3.获取到根节点

\* 4.在根节点上面创建stu标签

\* 5.在stu标签上面依次添加id name age

addElement方法添加

\* 6.在id name age上面依次添加值

setText方法

\*/

删除操作，根据id来删除

/\*

\* 1.创建解析器

\* 2.得到document

\* 3.获取到所有的id，使用xpath //id 返回list 集合

\*

\* 4.遍历list集合

\* 5.判断集合里面的id和传递的id是否相同

\* 6.如果相同，把id所在的stu删除

\*/

查询操作，根据id来查询

/\*

\* 1.创建解析器

\* 2.得到document

\* 3.获取到所有的id，使用xpath //id 返回list 集合

\*

\* 4.遍历list集合

\* 5.判断集合里面的id和传递的id是否相同

\* 6.如果相同，获取id的父节点stu

\* 7.通过stu获取到name age的值

\*/

把这些值封装到一个对象里面，返回这个对象

2017年10月8日17:20:58

1. eclipse： 是一个免费的开发工具

myeclipse：是一个收费的插件，下载的是装好插件的eclipse，可以破解。

myeclipse里面切换行号的显示，右键任意行号的位置，选择 show line numbers

myeclipse设置断点，在行号的左边双击会出现一个很小的蓝色小点，表示设置了断点。运行的时候选择debug as java application，不要选run as

单行注释快捷键： ctrl + /

多行注释快捷键： 选中后 ctrl + shift + /

取消单行注释：ctrl + /

取消多行注释：ctrl + shift + \

删除一行的快捷键：ctrl + d

代码格式化： ctrl + shift + F 这个功能有时候不好使，可以使用菜单栏source下的format

快速导包 ctrl + shift + o

1. junit 的使用

单元测试

测试对象是类中的一个方法

junit不是javase的一部分，想要使用导入jar包，但是，在myeclipse中自带了junit的jar包

单元测试的时候，方法命名规则 public void 方法名（）

**没有返回值，没有参数**

使用注解的方式运行测试方法，在方法的上面@Test

（1）在src下创建一个包，添加一个类，包名为cn.itcast.test02，类名为TetsJunit

**package** cn.itcast.test02;

**public** **class** TetsJunit {

**public** **void** testAdd(**int** a,**int** b)

{

System.***out***.println(a+b);

}

}

1. 新建一个源代码文件夹，在工程名上面右键，new ，source folder。

然后在这个文件夹下创建一个和上面一样的包名，并添加一个测试类

**package** cn.itcast.test02;

**import** org.junit.Test;

**public** **class** TestDemo {

@Test

**public** **void** testAdd1()

{

TetsJunit test01 = **new** TetsJunit();

test01.testAdd(2, 3);

}

}

1. 添加@Test注解后，需要导入junit的包，myeclipse自带了，
2. 运行方式，选中方法名，然后右键选择run as ，junit test，然后会出现junit的窗口，绿色表示测试通过

如果有多个测试方法，右击任意空白处，选择run as ，junit test，就可以测试多个方法了

1. 其他注解

@Ignore 表示不进行单元测试，而@Test表示要进行单元测试

@Before 在每个方法之前运行befor测试方法

@After 在每个方法之后运行after测试方法

1. 断言

**package** cn.itcast.test02;

**import** org.junit.Test;

**import** junit.framework.~~Assert~~;

**public** **class** TestDemo2 {

@Test

**public** **void** test02()

{

**int** a =3;

**int** b =5;

**int** sum=a+b;

~~Assert~~.~~assertEquals~~(8, sum);

}

}

泛型里面不能写基本的数据类型，比如int，写基本数据类型对应的包装类

byte -- Byte

int -- Integer

long -- Long

float -- Float

double -- Double

char -- Char

boolean -- Boolean

2017年10月24日09:38:55

1. Servlet概述

生命周期方法：

void init(ServletConfig) 出生之后1次

void service（ServletRequest request，ServletResponse response）每次处理请求之后都会被调用

void destroy（） 临时之前1次

特性：

单例，一个类只有一个对象

线程不安全的，所以它的效率高

servlet类由我们来写，但对象由服务器来创建，并由服务器来调用相应的方法

1. GenericServlet

2017年10月28日10:05:20

1. jsp的组成

jsp= html+ java脚本+jsp标签

jsp中无需创建即可使用的对象一共有9个，被称之为9大内置对象。例如：**request**对象、out对象

3种java脚本

<%...%>: java代码片段（常用），用于定义0~N条java语句

<%=… %>: java表达式，用于输出（常用），用于输出一条表达式或变量的结果

<%!...%> 声明，用来创建类的成员变量和成员方法（基本不用）

2017年11月1日10:07:22

1. css中 相同属性和值的选择符组合起来称为选择符组。如果需要给选择符组定义属性和值，只要用逗号将选择符分开即可，这样可以减少重复定义样式。

p,table{font-size:10pt}

等效于： p{font-size:10pt}

table{font-size:10pt}

1. css 类选择符

格式： 标记名.类名{样式属性:取值; 样式属性:取值;……}

1. id选择器

标记名#类名{样式属性:取值; 样式属性:取值;……}

1. 包含选择符

包含选择符是对某种元素包含关系定义的样式表。这种方式只对在元素1里面的元素2定义。

table b{font-size:11px} 这里只对表格里面的b设置字体为11像素

**li** **a** **span**{

width: *0*;

height: *0*;

border-bottom: *6px solid #00ccad*;

border-left:*6px solid #fff*;

position: *absolute*;

top: *0*;

left: *0*;

overflow: *hidden*;

}

2017年11月9日20:59:14

1. 让servlet在tomcat启动的时候创建

在servlet的配置里面添加

<load-on-startup>0</load-on-startup>

如果不加这句话，则servlet在调用的时候被创建，servlet只会陪创建一次

web.xml中的servlet配置

<servlet>

<servlet-name>yyy</servlet-name>

<servlet-class>cn.itcast.web.servlet.EServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>yyy</servlet-name>

<url-pattern>/EServlet</url-pattern>

</servlet-mapping>

还可以在<url-pattern>中使用通配符，通配符就是“\*”，可以匹配任何url的前缀或后缀

2017年11月12日16:13:42

1. 域对象的功能

ServletContext是JavaWeb四大域对象之一

PageContext、 ServletRequest、HttpSession、ServletContext

域对象就是用来在多个servlet中传递数据的

所有域对象都有存取数据功能，因为域对象内部有一个Map，用来存储数据

下面是ServletContext对象用来操作数据的方法：

void setAttribute(String name,Object value)：用来存储一个对象，也可以称之为存储一个域属性

object getAttribute(String name):用来获取ServletContext中的数据，当前在获取之前需要先去存储才行

void removeAtrribute(String name):用来移除ServletContext中的域属性，如果参数name指定的域属性不存在，那么本方法什么都不做

Enumeration getAttributeNames(): 获取所有域属性的名称

获取ServletContext的方法：

在doGet()方法中：

ServletContext application= this.getServletContext();

存储数据：

application.setAtrribute(“name”，“张三”)

获取数据：

String name = application.getAttribute(“name”)；

2017年11月20日09:48:48

1. css的伪类

**CSS 伪类用于向某些选择器添加特殊的效果。**

<style type="text/css">

a:link {color: #FF0000}

a:visited {color: #00FF00}

a:hover {color: #FF00FF}

a:active {color: #0000FF}

</style>

1. css的伪元素

CSS 伪元素用于向某些选择器设置特殊效果。

## CSS2 - :before 伪元素

":before" 伪元素可以在元素的内容前面插入新内容。

下面的例子在每个 <h1> 元素前面插入一幅图片：

h1:before

{

content:url(logo.gif);

}

## CSS2 - :after 伪元素

":after" 伪元素可以在元素的内容之后插入新内容。

下面的例子在每个 <h1> 元素后面插入一幅图片：

h1:after

{

content:url(logo.gif);

}

**2017年11月20日10:15:38**

1. margin属性可以取1到4个值

margin:10px 5px 15px 20px;

* 上外边距是 10px
* 右外边距是 5px
* 下外边距是 15px
* 左外边距是 20px

#### 例子 2

margin:10px 5px 15px;

* 上外边距是 10px
* 右外边距和左外边距是 5px
* 下外边距是 15px

#### 例子 3

margin:10px 5px;

* 上外边距和下外边距是 10px
* 右外边距和左外边距是 5px

#### 例子 4

margin:10px;

* 所有 4 个外边距都是 10px

## 可能的值

|  |  |
| --- | --- |
| **值** | **描述** |
| auto | 浏览器计算外边距。 |
| *length* | 规定以具体单位计的外边距值，比如像素、厘米等。默认值是 0px。 |
| *%* | 规定基于父元素的宽度的百分比的外边距。 |
| inherit | 规定应该从父元素继承外边距。 |

注意：margin是外边距，padding是内边距

**margin的属性之间是空格，不是逗号，也不是分号。**

**使元素在父容器中居中显示**

.content{

background-color: #ff1;

width:300px;

**margin: 0 auto;**

}

2017年11月20日11:57:05

1. 在一个元素后面插入元素，使用after伪元素

要设置前一个元素的position属性为relative，否则位置不对。

.content .select{

background-color: #af1;

width:300px;

height: 40px;

margin: 0 auto;

position: relative;

}

.content .select:after{

content:'';

display: block;

width: 10px;

height: 10px;

border-left: 1px #ccc solid;

border-bottom: 1px #ccc solid;

position:absolute;

top:11px;

right: 12px;

transform:rotate(-45deg);

}

2017年11月20日16:02:11

1. css的颜色

color:关键字|RGB值

可以用两种方式来设置颜色，关键字和RGB值。

关键字如：red，blue等

RGB值： #00FF00，是十六进制的RGB值，表示绿色。

#0F0是十六进制RGB的缩写，表示绿色，只要有同样的数字重复出现，就可以省略其中一个不写。

RGB(255,0,0) 是RGB函数值，表示红色。取值为0-255

RGB(0%,100%,0%)也是RGB函数值，表示绿色，取值0%-100%

1. 用css画三角形

先看一个例子：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>练习border</title>

<style type="text/css">

body{

background-color: #00f1f1;

margin: 0;

padding: 0;

}

div{

width: 100px;

height: 100px;

border-left: 50px solid magenta;

border-right: 50px solid red;

border-top: 50px solid blue;

border-bottom: 50px solid green;

}

</style>

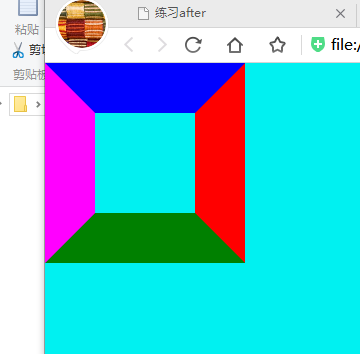
</head>

<body>

<div></div>

</body>

</html>



修改div的高度和宽度为0，左边框设置为透明：

div{

width: 0px;

height: 0px;

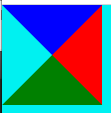
border-left: 50px solid transparent;

border-right: 50px solid red;

border-top: 50px solid blue;

border-bottom: 50px solid green;

}



总结：画三角形的时候，要把width和height设置为0，实际上，上下左右边框分别对应上下左右的三角形，将不要的三角形设置为透明即可。

1. css画三角形例子2

div{

width: 0;

height: 0;

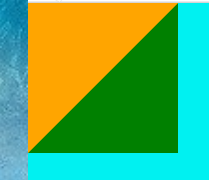
border-left: 50px solid orange;

/\* border-right: 50px solid red;

border-top: 50px solid blue;\*/

border-bottom: 50px solid green;

}



这次边框的左边和下边，结果只有两个三角形。

如果只画左边和右边，则没有结果。

因此如果只想画一个简单的三角形，不用四个边框都画，只需要画两个即可。但这两个边框必须是相邻的。

1. css的position深入理解

之前学习css的position的时候，只知道absolute是脱离文档流，relative是保留文档流，使用absolute定位，后面的元素会移到该元素的位置，只有relative却不会。但实际上是以什么为参照定位的呢。

position的四个属性值：

relative

absolute

fixed

static

设置position属性后，要用left，top，right，bottom属性来设置偏移位置。一般设置top和left

1. relative

使用relative定位，元素不脱离文档流，即该元素按照之前的位置偏移。在偏移之前该元素有个自己的位置，设置position为relative后，结合left和top等属性，该元素按照原来的位置偏移，而其他元素位置不变。

1. absolute

absolute要比relative复杂些，要分两种情况,与父元素的position属性有关。

1. 如果以元素的父元素（包括父元素的所有父级）设置了position属性，并且该属性不是static。当这个元素的position属性设置为absolute，则该元素结合left和top等按照它的父级元素定位。
2. 如果元素的父级没有设置position属性。则该元素会按照body为定位对象。
3. 自己写的一个例子



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>练习三角形</title>

<style type="text/css">

body{

background-color: #0c9;

}

div{

padding-top: 5%;

width: 100px;

text-align: center;

margin: 0 auto;

font-size: 20px;

}

p{

background-color: #ffff00;

**position: relative;**

}

span {

width: 0;

height: 0px;

border-bottom: 10px solid #ffff00;

border-left:10px solid #0c9;

/\* border-top:16px solid red;

border-right:16px solid #0444c9; \*/

position: absolute;

top: 0;

left: 0;

overflow: hidden;

}

</style>

</head>

<body>

<div>

<p>

<span></span>联系我们

</p>

</div>

</body>

</html>

注意p的样式，红色部分。如果不设置position属性，则文字左边的小三角形就会跑到左上角了。

span属性的position属性，如果设置为relative，则高度会出现异常，具体原因未知。



2017年11月21日19:58:35

1. 让ul列表横着显示

使用float属性，设置为left

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>列表样式</title>

<style type="text/css">

li{

list-style: none;

font-size: 20px;

float:left;

margin: 12px;

background-color: #ffff00;

}

</style>

</head>

<body>

<ul>

<li>dreamware</li>

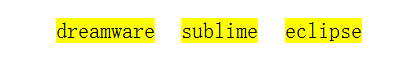
<li>sublime</li>

<li>eclipse</li>

</ul>

</body>

</html>



1. css float属性深入

float，顾名思义就是浮动，设置了float属性的元素会根据属性值向左或向右浮动，我们称设置了float属性的元素为浮动元素。  
浮动元素会从普通文档流中脱离，但浮动元素影响的不仅是自己，它会影响周围的元素对齐进行环绕。

不管一个元素是行内元素还是块级元素，如果被设置了浮动，那浮动元素会生成一个块级框，可以设置它的width和height，**因此float常常用于制作横向配列的菜单，可以设置大小并且横向排列**。

1. 模仿的一个简单的导航条（实际上应该用a标签做超链接的）



特点：a.用border-left和border-bottom属性画三角形

b.使用列表的 float:left;属性使列表横向显示

c.使用hover属性设置鼠标移入和移除的不同颜色

d.使用position属性设置span的位置

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>导航条</title>

<style type="text/css">

body{

background-color: white;

}

ul{

list-style: none;

}

li {

float:left;

background-color: #00BFFF;

color: red;

position: relative;

text-align: center;

padding: 6px;

}

li:hover{

background-color: #bdbbbb;

color:#FFD700;

}

li span{

border-left: 10px solid white;

border-bottom: 10px solid #00BFFF;

position: absolute;

top: 0;

left:0;

}

li:hover span{

border-bottom: 10px solid #bdbbbb;

}

</style>

</head>

<body>

<div class="content">

<ul>

<li><span></span>今日头条</li>

<li><span></span>今日头条</li>

<li><span></span>今日头条</li>

<li><span></span>今日头条</li>

</ul>

</div>

</body>

</html>

**2018年1月21日23:09:13**

1. 让ul无序列表水平显示

通常ul列表是竖着一次换行显示的，可以通过float：left来让其水平显示

<html>

<head>

<title>firstDemo</title>

<style type="text/css">

ul{

list-style:none;

color:red;

}

li{

padding:5px;

float:left;

}

</style>

</head>

<body>

<ul>

<li>one</li>

<li>two</li>

<li>three</li>

<li>four</li>

</ul>

</body>

</html>

