转 div+css基础教程

2013年11月21日 00:29:05　阅读数：3472

本文存下来作为备忘。

第一节　了解div+css

一、什么是div+css

div元素是html(超文本语言)中的一个元素，是标签，用来为html文档内大块（block-level）的内容提供结构和背景的元素。div的起始标签和结束标签之间的所有内容都是用来构成这个块的，其中所包含元素的特性由div标签的属性来控制，或者是通过使用样式表格式化这个块来进行控制。

css是英语cascadingstyle sheets（层叠样式表单）的缩写，它是一种用来表现 html 或 xml 等文件式样的计算机语言。

div+css是网站标准（或称“web标准”）中常用术语之一，通常为了说明与html网页设计语言中的表格（table）定位方式的区别，因为xhtml网站设计标准中，不再使用表格定位技术，而是采用div+css的方式实现各种定位。

html语言自html4.01以来，不再发布新版本，原因就在于html语言正变得越来越复杂化、专用化。即标记越来越多，甚至各个浏览器生产商也开发出只适合于其特定浏览器的html标记，这显然有碍于html网页的兼容性。于是w3c组织进而重新从sgml中获取营养，随后，发布了xml，xml是一种比html更加严格的标记语言，全称是可扩展标记语言。但是xml过于复杂，且当前的大部分浏览器都不完全支持xml。于是xhtml这种语言就派上了用场，xhtml语言就是一种可以将html语言标准化，用xhtml语言重写后的html页面可以应用许多xml应用技术。使得网页更加容易扩展，适合自动数据交换，并且更加规整。

二、div+css的优势

1、符合w3c标准。这保证您的网站不会因为将来网络应用的升级而被淘汰。

2、对浏览者和浏览器更具亲和力。由于css富含丰富的样式，使页面更加灵活性，它可以根据不同的浏览器，而达到显示效果的统一和不变形。这样就支持浏览器的向后兼容，也就是无论未来的浏览器大战，胜利的是什么，您的网站都能很好的兼容。

3、使页面载入得更快。页面体积变小，浏览速度变快，由于将大部分页面代码写在了css当中，使得页面体积容量变得更小。相对于表格嵌套的方式，div+css将页面独立成更多的区域，在打开页面的时候，逐层加载。而不像表格嵌套那样将整个页面圈在一个大表格里，使得加载速度很慢。

4、保持视觉的一致性。以往表格嵌套的制作方法，会使得页面与页面，或者区域与区域之间的显示效果会有偏差。而使用div+css的制作方法，将所有页面，或所有区域统一用css文件控制，就避免了不同区域或不同页面体现出的效果偏差。

5、修改设计时更有效率。由于使用了div+css制作方法，使内容和结构分离，在修改页面的时候更加容易省时。根据区域内容标记，到css里找到相应的id，使得修改页面的时候更加方便，也不会破坏页面其他部分的布局样式，在团队开发中更容易分工合作而减少相互关联性。

6、搜索引擎更加友好。相对与传统的table,

采用div+css技术的网页，由于将大部分的html代码和内容样式写入了css文件中，这就使得网页中代码更加简洁，正文部分更为突出明显，便于被搜索引擎采集收录。

三、css＋div网站设计的缺陷

尽管div+css具有一定的优势，不过现阶段css+div网站建设存在的问题也比较明显，主要表现在：

1、对于css的高度依赖使得网页设计变得比较复杂。相对于html4.0中的表格布局（table），css+div尽管不是高不可及，但至少要比表格定位复杂的多，即使对于网站设计高手也很容易出现问题，更不要说初学者了，这在一定程度上影响了xhtml网站设计语言的普及应用。

2、css文件异常将影响整个网站的正常浏览。css网站制作的设计元素通常放在一个或几个外部文件中，这些文件有可能相当复杂，甚至比较庞大，如果css文件调用出现异常，那么整个网站将变得惨不忍睹。

3、对于css网站设计的浏览器兼容性问题比较突出。虽然说div+css解决了大部分浏览器兼容问题，但是也有在部分浏览器中使用出现异常，css+div还有待于各个浏览器厂商的进一步支持。

4、css+div对搜索引擎优化与否取决于网页设计的专业水平而不是css+div本身。css+div网页设计并不能保证网页对搜索引擎的优化，甚至不能保证一定比html网站有更简洁的代码设计。因为对于搜索引擎而言，网站结构、内容、相关网站链接等因素始终是网站优化最重要的指标。

如何更有效、更合理的运用web2.0设计标准，这需要很长时间的学习和锻炼。而如何将div+css运用的更好，需要通过不断的实践和体检，积累丰富的设计经验，才能很好的掌握这门技术。

第二节　css布局标签

一、什么时候应该用div?

div元素是一个标签，但多层嵌套的div会严重影响代码的可阅读性。什么时候应该用div虽然没有什么硬性的规定，但div更适用于大体框架的定位。

例如我们要定义一块头部的区域，一般会这样定义一个div：

```html
<ul id="navbar">
<li id="articles">articles</li>
<li id="topics">topics</li>
<li id="about">about</li>
<li id="contact">contact</li>
<li id="contribute">contribute</li>
<li id="feed">feed</li>
</ul>
<h1 id="masthead">
<a href="/">
<img src="/pix/alalogo.gif" alt="a list apart: for peoplewho make websites"
/>
</h1>
no. 214
```

这个例子定义了最上面的导航(ul部分)，左边的logo和no.214的标记.

二、最常用的布局标签

h1：这个标签表达的意思就如同它原先的作用一样明显(大标题)。

h2:使用最多的地方应该不是布局上,而是副标题,但有些地方需要定义栏目样式的话,用这个标签比较合适,栏目内容就使用p

ul：这个标签很多情况是用来定义导航部分的，当然也可以用ol来代替，但导航连接没有什么顺序之分,所以还是用ul来的比较确切。

b:这个已经不再推荐使用的标签，但因为短小，在布局上却能带来不少的方便，有些时候(比如细小地方的布局定义)还是不错的选择。

css放入网页的方式，可以在html文件内直接宣告样式，也可以在外部连接套用。外部连接套用时，所有的css样式都存在另外一个文件中，文件名称为.css。

第三节 页面布局设计

在网页制作设计中，第一步就是构思，可以用photoshop或fireworks等图片处理软件将需要制作的页面布局简单的构画出来，然后根据布局图设计。

一般来说，页面包括：顶部部分，其中又包括了logo、menu和一幅banner图片；内容部分，又可分为侧边栏、主体内容；底部，包括一些版权信息。可以命名为：顶部层header、内容层pagebody、侧边栏层sidebar、主体内容层mainbody、底部层footer。层的嵌套关系div结构设计为：

```
|body {}  /*这是一个html元素
└#container {}  /*页面层容器*/
├#header{}  /*页面头部*/
├#pagebody {}  /*页面主体*/
|  ├#sidebar {}  /*侧边栏*/
|  └#mainbody{}  /*主体内容*/
└#footer {}  /*页面底部*/
```

接下来就可以书写html代码和css了。

第四节 写入整体层结构与css

新建一个文件夹，命名为“div+css布局练习”，在文件夹下新建两个空的记事本文档。

将一个记事本文档命名为index.htm，输入以下内容：

```html
<!doctype html public"-//w3c//dtd xhtml 1.0 transitional//en"
http://www.w3.org/tr/xhtml1/dtd/xhtml1-transitional.dtd">
<htmlxmlnshtmlxmlns="http://www.w3.org/1999/xhtml">
<head>
<metahttp-equivmetahttp-equiv="content-type" content="text/html;charset=gb2312" />
<title>无标题文档</title>
<link href="css.css"rel="stylesheet" type="text/css" />
</head>
<body>
</body>
</html>
```

这是xhtml的基本结构。下面，我们在<body></body>标签对中写入div的基本结构，为了使以后阅读代码更简易，我们添加相关注释：

```html
 <div id="container">[color=#aaaaaa]<!--页面层容器-->[/color]
<div id="header">[color=#aaaaaa]<!--页面头部-->[/color]
</div>
<div id="pagebody">[color=#aaaaaa]<!--页面主体-->[/color]
<div id="sidebar">[color=#aaaaaa]<!--侧边栏-->[/color]
</div>
<div id="mainbody">[color=#aaaaaa]<!--主体内容-->[/color]
</div>
</div>
<div id="footer">[color=#aaaaaa]<!--页面底部-->[/color]
</div>
 </div>
```

将另一个记事本文档命名为css.css，写入css信息，代码如下：

```html
/*基本信息*/
 body {font:12pxtahoma;margin:0px;text-align:center;background:#fff;}
 /*页面层容器*/
 #container {width:100%}
 /*页面头部*/
 #header {width:800px;margin:0auto;height:100px;background:#ffcc99}
 /*页面主体*/
 #pagebody {width:800px;margin:0auto;height:400px;background:#ccff00}
 /*页面底部*/
 #footer {width:800px;margin:0auto;height:50px;background:#00ffff}
```

把以上文件保存，用浏览器打开，这时我们已经可以看到基础结构了，这个就是页面的框架了。

温馨提示：

1、请养成良好的注释习惯，这是非常重要的；

2、body是一个html元素，页面中所有的内容都应该写在这标签对之内；

3、一些常用的css代码的含义：

font:12px tahoma；这里使用了缩写，说明字体为12像素大小，字体为tahoma格式。完整的代码是：font-size:12px;font-family:tahoma；

margin:0px;使用了缩写,完整的是:margin-top:0px;margin-right:0px;margin-bottom:0px;margin-left:0px,也可以写成margin:0px 0px 0px 0px,顺序是 上 / 右 / 下 / 左,还可以书写为margin:0(缩写);这个样式说明body部分对上右下左边距为0像素,如果使用auto则是自动调整边距,另外还有以下几种写法:margin:0pxd auto;说明上下边距为0px,左右为自动调整;以后使用到的padding属性和margin有许多相似之处,他们的参数是一样的，只不过各自表示的含义不相同，margin是外部距离，而padding则是内部距离。

text-align:center 文字对齐方式，可以设置为左、右、中，这里设置为居中对齐。

background:#fff 设置背景色为白色,这里颜色使用了缩写,完整的应该是background:#ffffff。background可以用来给指定的层填充背景色、背景图片，以后会用到这个格式：background:#ccc:

url('bg.gif') topleft no-repeat;表示使用#ccc(灰度色)填充整个层,使用bg.gif做为背景图片,top left表示图片位于当前层的左上端,no-repeat表示仅显示图片大小而不填充满整个层。top/right/bottom/left/center用于定位背景图片，分别表示上 /右 / 下 / 左 / 中；还可以使用background:url('bg.gif') 20px100px;表示x座标为20像素,y座标为100像素的精确定位;repeat/no-repeat/repeat-x/repeat-y分别表示填充满整个层 / 不填充 / 沿x轴填充 / 沿y轴填充。height / width / color 分别表示高度(px)、宽度(px)、字体颜色(html色系表)。

4、可以看到，整个页面是居中显示的，因为我们在#container中使用了以下属性：

margin:0 auto;

这个表示上下边距为0，左右为自动，因此该层就会自动居中了。如果要让页面居左，则取消掉auto值就可以了，因为默认就是居左显示的。

第五节 页面顶部制作

当我们写好了页面大致的div结构后，我们就可以开始细致地对每一个部分进行制作了。

我们把css.css中的样式全部清除掉，重新写入以下样式代码：

/*基本信息*/

body {font:12px tahoma;margin:0px;text-align:center;background:#fff;}

a:link,a:visited{font-size:12px;text-decoration:none;}

a:hover{}

/*页面层容器*/

#container {width:800px;margin:10px auto}

以上样式说明如下：

a:link,a:visited{font-size:12px;text-decoration:none;}

a:hover {}

这两项分别是控制页面中超链接的样式。

#container {width:800px;margin:10px auto}

指定整个页面的显示区域。

width:800px指定宽度为800像素，这里根据实际所需设定。

margin:10px auto，则是页面上、下边距为10个像素，并且居中显示。

接下来，开始制作top部分，top部分包括了logo、菜单和banner，首先在fw下对设计好的图片进行切片，第一部分

为logo部分，由于logo图片并没有太多的颜色，这里我于是将这一部分保存为gif格式(这样能使页面载入的速度更快，当然使用此格式之前必须确定图片并没有使用太多的颜色)，调色板选择为精确，选择alpha透明度，色版为白色(此处颜色应与背景色相同)，导出为logo.gif，图像宽度为800px。第二部分为banner部分，因为banner部分是一个细致的图片，如果使用gif格式颜色会有太大的损失，所以必须使用jpeg格式，将文件导出为banner.jpg。

然后我们在css.css中再写入以下样式：

/*页面头部*/

#header {background:url(logo.gif)no-repeat}

这个样式给页面头部分加入一个背景图片logo，并且不作填充。这里，我们没有指定header层的高度，因为header层中还有菜单和banner项，所以层的高度暂时是未知的，而层的属性又可以让层根据内容自动设定调整，因此我们并不需要指定高度。

下面我们制作菜单，代码如下：

```html
<div id="menu">
<ul>
<li><a href="#">首页</a></li>
<li class="menudiv"></li>
<li><a href="#">博客</a></li>
<li class="menudiv"></li>
<li><a href="#">设计</a></li>
<li class="menudiv"></li>
<li><a href="#">相册</a></li>
<li class="menudiv"></li>
<li><a href="#">论坛</a></li>
<li class="menudiv"></li>
<li><a href="#">关于</a></li>
  </ul>
</div>
<div id="banner">
</div>
```

以上
、
这两个html元素主要的作用就是在html中以列表的形式来显示一些信息。对菜单使用列表
形式，可以在以后方便对菜单定制样式。<li class="menudiv">
这一段代码是可以方便地对菜单选项之间插入一些分隔样式，例如预览图中的竖线分隔。需要分清楚的是，当在html中定义为id="divid"时，在css对应的设置语法则是#divid{}存，如果在html中定义为class="divid"时，则在css中对应的设置语法是.divid。如果id="divid"这个层中包括了一个<img></img>，则这个img在css中对应的设置语法应该是#dividximg {}，同样，如果是包含在class="divid"这个层中时，则设置语法应该是.divid img {}。另外，html中的一切元素都是可以定义的，例如table、tr、td、th、form、img、input...等等，如果你要在css中设置它们，则直接写入元素的名称加上一对大括号{}就可以了。所有的css代码都应该写在大括号{}中。

然后我们在css.css中再写入以下样式：

#menu ul{list-style:none;margin:0px;}

#menu ul li {float:left;}

第一个样式list-style:none，这一句是取消列表前点，因为我们不需要这些点。margin:0px，这一句是删除ul的缩进，这样做可以使所有的列表内容都不缩进。第二个样式使用了浮动属性(float)，float:left是让内容都在同一行显示。

我们在#menu ul li {}再加入代码margin:0 10px，改成：

#menu ul li {float:left;margin:010px}

margin:0 10px的作用是让列表内容之间产生一个20像素的距离(左：10px，右：10px)。

明白了这些，我们再来固定菜单的位置，把代码改成如下：

#menu {padding:20px 20px 0 0}

/*利用padding:20px 20px 0 0来固定菜单位置*/

#menu ul{float:right;list-style:none;margin:0px;}

/*添加了float:right使得菜单位于页面右侧*/

#menu ul li {float:left;margin:010px}

这样，位置就确定了。下面在菜单间加入竖线，对留好的空的<liclass="menudiv">，再添加以下代码：

.menudiv{width:1px;height:28px;background:#999}

保存预览一下，竖线已经出来。不过，菜单选项的文字却在顶部，我们再修改成以下代码：

#menu ul li {float:left;margin:010px;display:block;line-height:28px}

效果基本上已经实现了，剩下的就是修改菜单的超链接样式，在css.css中添加以下代码：

#menu ul li a:link,#menu ul lia:visited {font-weight:bold;color:#666}

#menu ul li a:hover{}

第六节 页面顶部的完整代码

下面给出本例页面顶部的完整代码，供学习参考：

index.htm的内容：

```html
<!doctype html public"-//w3c//dtd xhtml 1.0 transitional//en"
"http://www.w3.org/tr/xhtml1/dtd/xhtml1-transitional.dtd">
 <htmlxmlnshtmlxmlns="http://www.w3.org/1999/xhtml">
 <head>
 <meta http-equiv="content-type"content="text/html; charset=gb2312" />
 <title>无标题文档</title>
 <link href="css.css"rel="stylesheet" type="text/css" media="all"/>
 </head>

 <body>
 <div id="container">
 <div id="header">
 <div id="menu">
 <ul>
 <li><a href="#">首页</a></li>
 <li class="menudiv"></li>
 <li><a href="#">博客</a></li>
 <li class="menudiv"></li>
 <li><a href="#">设计</a></li>
 <li class="menudiv"></li>
 <li><a href="#">相册</a></li>
 <li class="menudiv"></li>
 <li><a href="#">论坛</a></li>
 <li class="menudiv"></li>
 <li><a href="#">关于</a></li>
 </ul>
 </div>
 <div id="banner">
 </div>
 </div>
 </div>
 </body>
 </html>
```

css.css的代码：

```
/*基本信息*/
 body {font:12pxtahoma;margin:0px;text-align:center;background:#fff;}
 a:link,a:visited{font-size:12px;text-decoration: none;}
 a:hover{}

 /*页面层容器*/
 #container{width:800px;height:600px;margin:10px auto}

 /*页面头部*/
 #header {background:url(logo.gif)no-repeat}
 #menu {padding:20px 20px 0 0}
 #menu ul{float:right;list-style:none;margin:0px;}
 #menu ul li {float:left;display:block;line-height:30px;margin:010px}
 #menu ul li a:link,#menu ul lia:visited {font-weight:bold;color:#666}
 #menu ul li a:hover{}
 .menudiv{width:1px;height:28px;background:#999}
 #banner {background:url(banner.jpg)0 30px
no-repeat;width:730px;margin:auto;height:240px;border-bottom:5px solid
#efefef;clear:both}

 /*页面主体*/
 #pagebody {width:800px;margin:0auto;height:400px;background:#ccff00}

 /*页面底部*/
 #footer {width:800px;margin:0auto;height:50px;background:#00ffff}
```

第七节 div+css的border和clear属性

如果你曾用过table制作网页，你就应该知道，如果要在表格中绘制一条虚线该如何做，那需要制作一个很小的图片来填充，其实我们还有更简单的办法，只要在<td></td>中加入这么一段就可以了，你可以试试：

<divstyle="border-bottom:1px dashed #ccc">

其实利用dashed、solid、dotted...等可以制作出许多效果来，实线、虚线、双线、阴影线等等。

<divid="banner">

以上代码便可以实现banner，在css.css中加入以下样式：

#banner {

background:url(banner.jpg) 0 30pxno-repeat; /*加入背景图片*/

width:730px; /*设定层的宽度*/

margin:auto; /*层居中*/

height:240px; /*设定高度*/

border-bottom:5px solid #efefef; /*画一条浅灰色实线*/

clear:both /*清除浮动*/

}

通过border很容易就绘制出一条实线了，并且减少了图片下载所占用的网络资源，使得页面载入速度变得更快。

再看看clear:both，表示清除左、右所有的浮动，在接下来的布局中我们还会用这个属性：clear:left/right。在这里添加clear:both是由于之前的ul、li元素设置了浮动，如果不清除则会影响banner层位置的设定。

<divid="pagebody"><!--页面主体-->

<divid="sidebar"><!--侧边栏-->

<divid="mainbody"><!--主体内容-->

以上是页面主体部分，我们在css.css中添加以下样式：

```
#pagebody {
width:730px; /*设定宽度*/
margin:8px auto; /*居中*/
}
#sidebar {
width:160px; /*设定宽度*/
text-align:left; /*文字左对齐*/
float:left; /*浮动居左*/
clear:left; /*不允许左侧存在浮动*/
overflow:hidden; /*超出宽度部分隐藏*/
}
#mainbody {
width:570px;
text-align:left;
float:right; /*浮动居右*/
clear:right; /*不允许右侧存在浮动*/
overflow:hidden
}
```

为了可以查看到效果，建议在#sidebar和#mainbody中加入以下代码，预览完成后可以删除这段代码：

border:1px solid #e00;

height:200px

保存预览效果，可以发现这两个层完美的浮动，在达到了我们布局的要求，而两个层的实际宽度应该160+2(border)+570+2=734px，已经超出了父层的宽度，由于clear的原因，这两个层才不会出现错位的情况，这样可以使我们布局的页面不会因为内容太长（例如图片）而导致错位。而之后添加的overflow:hidden则可以使内容太长（例如图片）的部份自动被隐藏。通常我们会看到一些网页在载入时，由于图片太大，导致布局被撑开，直到页面下载完成才恢复正常，通过添加overflow:hidden就可以解决这个问题。

css中每一个属性运用得当，就可以解决许多问题，或许它们与你在布局的页并没有太大的关系，但是你必须知道这些属性的作用，在遇到难题的时候，可以尝试使用这些属性去解决问题。

第八节 div+css常见错误

1. 检查html元素是否有拼写错误、是否忘记结束标记

即使是老手也经常会弄错div的嵌套关系。可以用dreamweaver的验证功能检查一下有无错误。

2. 检查css是否正确

检查一下有无拼写错误、是否忘记结尾的 } 等。可以利用cleancss来检查 css的拼写错误。cleancss本是为css减肥的工具，但也能检查出拼写错误。

3. 确定错误发生的位置

如果错误影响了整体布局，则可以逐个删除div块，直到删除某个div块后显示恢复正常，即可确定错误发生的位置。

4. 利用border属性确定出错元素的布局特性

使用float属性布局一不小心就会出错。这时为元素添加border属性确定元素边界，错误原因即水落石出。

5. float元素的父元素不能指定clear属性

macie下如果对float的元素的父元素使用clear属性，周围的float元素布局就会混乱。这是macie的著名的bug，倘若不知道就会走弯路。

6. float元素务必指定width属性

很多浏览器在显示未指定width的float元素时会有bug。所以不管float元素的内容如何，一定要为其指定width属性。另外指定元素时尽量使用em而不是px做单位。

7. float元素不能指定margin和padding等属性

ie在显示指定了margin和padding的float元素时有bug。因此不要对float元素指定margin和padding属性(可以在float元素内部嵌套一个div来设置margin和padding)。也可以使用hack方法为ie指定特别的值。

8. float元素的宽度之和要小于100%

如果float元素的宽度之和正好是100%，某些古老的浏览器将不能正常显示。因此请保证宽度之和小于99%。

9. 是否重设了默认的样式?

某些属性如margin、padding等，不同浏览器会有不同的解释。因此最好在开发前首先将全体的margin、padding设置为0、列表样式设置为none等。

10. 是否忘记了写dtd?

如果无论怎样调整不同浏览器显示结果还是不一样，那么可以检查一下页面开头是不是忘了写下面这行dtd：

<!doctype html public"-//w3c//dtd html 4.01 transitional//en"

"http://www.w3.org/tr/html4/loose.dtd">

文章标签： div css dreamweaver 网页设计 html

个人分类： J2EE