【DIV+CSS 入门教程】DIV+CSS 的叫法是不准确的

第一章：应知道

1.1 DIV+CSS 的叫法是不准确的

我想凡是来到“这个专题”的同学，很大部分是冲着 DIV+CSS 来的，目的就是学习 DIV+CSS 的，说的再直接一些就是学习如何用 DIV+CSS 布局页面，如何从一张图片制作成标准的 DIV+CSS 页面。

如果你看完第一段还没有发现错误的话，那你就很有必要，接着往下看。

DIV+CSS 这种叫法其实是一种很错误的叫法，这是国人一厢情愿的叫法，而标准的叫法是什么呢？

呵呵，没错，是 xHTML+CSS，不理解吧，我来细细给你说，如果下面的你能理解，保证面试的时候会有很大的帮助，同时也可以让你后面的学习更轻松。

为什么国人将这种页面布局的方法叫做 DIV+CSS？

因为过去布局页面基本上都是用 Table 布局，也可以说是 Table+CSS，而现在布局页面呢，用 DIV，所以叫 DIV+CSS，听起来也挺合理，认为这样布局出来的页面也就是标准页面，甚至有些人走了个极端，看到其他网站用到 Table，就会嘲笑页面做的不够标准，好似用不用 Table 成为了页面是否标准的一个标尺。现在我可以告诉大家，凡是有着这种行为的，都学得不咋样，很皮毛！

用了 Table 页面就不标准了?!纯粹无稽之谈，那什么才是标准页面呢？先看一个专业概念，WEB 标准，然后我会问三个问题，你来回答：

WEB 标准不是某一个标准，而是一系列标准的集合。网页主要由三部分组成：结构（Structure）、表现（Presentation）和行为（Behavior）。对应的标准也分三方面：结构化标准语言主要包括 XHTML 和 XML，表现标准语言主要包括 CSS，行为标准主要包括对象模型（如 W3C DOM）、ECMAScript 等。这些标准大部分由 W3C 起草和发布，也有一些是其他标准组织制订的标准，比如 ECMA（European Computer Manufacturers Association）的 ECMAScript 标准。

看明白了没有？问题来了¯先不要看答案，从上面的概念中找出

问题一：WEB 标准有几部分组成？

问题二：结构化标准语言是什么？

问题三：表现标准语言是什么？

答案一：三部分，结构、表现、行为

答案二：XHTML 和 XML

答案三：CSS

看完上面三个问题，哪什么是标准页面呢？呵呵，说白了就是按照 WEB 标准制作的页面，从第二个问题和第三个问题中，我们又可以说，用 XHTML 和 CSS 制作的页面就是标准页面，也就是说 xHTML+CSS 制作的页面就是标准页面。怎么样，理解了吧

为什么不说 XML+CSS 呢？

很简单，因为 XML 过于复杂，且当前的大部分浏览器都不完全支持 XML。所以就不用它来布局页面喽¯

既然 xHTML+CSS 制作页面就是标准页面了，又因为 xHTML 中不只有 DIV 标签，还有 span、p、a、ul、li、dl、dt、dd….，即使我不用 DIV，用其他标签(比如:ul、li)制作出来的页面也是标准页面！所以说用 DIV+CSS 来制作标准页面这句话就很狭隘喽¯如果满屏全部都是 DIV 那也算不上标准页面，曾经由一个朋友告诉我，说他的页面全部用的 DIV，每个

模块，每个功能区域，就连一条线都是纯 DIV 实现，并且相当自豪的告诉我，没有人比他做的页面更标准的了，他不但对 WEB 标准页面的理解有差错还犯了一个很大的错误，xHTML 中的每一个标签都有其作用，各司其职，各守其责，要用的恰到好处，这才算是标准页面，DIV 不是万能的哟¯！

说到这里大家应该明白，这种 Web2.0 时代的布局页面的方法，叫 DIV+CSS 是不准确的，应该叫 xHTML+CSS。

凡是看到这节的同学们，以后尽可能说 xHTML+CSS，不要再说 DIV+CSS 喽，如果非要说，也要加上一句说明哟，比如

面试官：你对 DIV+CSS 了解么？

应聘者：DIV+CSS 准确的说应该叫 xHTML+CSS，我对这种页面布局方法非常了解！……

如果你是面试官，你对这个应聘者，感觉如何呢？

【DIV+CSS 入门教程】使用 Table 布局是不明智的

使用 Table 布局页面为什么是不明智的？

大家看到标题，不要误解认为在页面中不能使用 Table，而是可以使用 Table，但是尽量不要用 Table 去布局页面，为什么这么说呢，因为使用 Table 布局页面会使页面失去灵活性，怎么个灵活法呢，比如今天你好不容易做出来的页面，第二天老板说我不喜欢登录模块放到右边，还是放到左边，通知板块放到右侧去，页面风格最好一个月换一种，如果遇到这种老板，提出这种要求，并且你的页面是用 Table 布局的，那么你会崩溃的，工作量那是大大滴¯，如果不相信的话，你们自己可以找个页面，用 Table 布局出来，然后变换板块和风格，你就会体会到 Table 布局的不灵活性，这是为什么呢，因为 Table 的诞生是为存储数据用的，功能和 Execel 差不多，不是用来布局用的，只不过后来大家发现用 Table 可以把想放的页面元素，比如图片，放到任何自己想放的地方，且做出来的页面可以兼容多种浏览器，于是 Table 就承担起了布局页面的重担，这一做就是好几年... ...直到 Web2.0 时代的到来，Table 才从布局页面的工作中逐渐解脱，专心的去存储数据^_^

既然 Table 是为存储数据诞生的，那谁的诞生是为了页面布局呢？

答案就是：**DIV**，DIV 就是为布局页面而诞生的，只不过一直不被人认同，原因就是 DIV 去布局页面需要 CSS 的配合，使用比较繁琐，还不如 Table 拖拖拽拽页面就布局 OK 了，感觉还不如 Table 方便，从而 DIV 被人们放置在一个无人问津的昏暗角落里，暗暗的等待着伯乐的出现，直到 2003 年美国加州 Scott Design 公司参加了在旧金山举办的有关网页排版和设计的一个研讨会上的演讲，使 DIV 看到了阳光，走出了阴霾... ...

说了那么多，我们对比一下 Table 布局页面和 DIV 布局页面的优缺点

使用表格进行页面布局会带来很多问题：

* 把格式数据混入你的内容中。这使得文件的大小无谓地变大，而用户访问每个 页面时都必须下载一次这样的格式信息，带宽并非免费。
* 这使得重新设计现有的站点和内容极为消耗劳力 (且昂贵)。
* 这还使我们保持整个站点的视觉的一致性极难，花费也极高。
* 基于表格的页面还大大降低了它对残疾人和用手机或 PDA 浏览者的亲和力。

而使用 CSS 进行网页布局，它会:
* 使你的页面载入得更快
* 降低你的流量费用
* 让你在修改设计时更有效率而代价更低
* 帮助你的整个站点保持视觉的一致性
* 让你的站点可以更好地被搜索引擎找到
* 使你的站点对浏览者和浏览器更具亲和力
* 在世界上越来越多人采用 Web 标准时,它还能 提高你的职场竞争实力 (事实上也就是降低失业的风险)。

网上有一篇文章，转过来，文章着重介绍 DIV 三点优势,也许看完文章后，就像社区元老 heflyaway 说的感觉作者比较迷恋 Table，每篇文章都不可避免的带有个人色彩，而转出来的目的，其实就是想给大家降降 DIV+CSS 的温度，免得“走火入魔”，视 DIV+CSS 是为万能的，如果想学好 CSS 布局页面，就要从多个方面看它，好了，不多说了，下面是作者对 CSS 布局页面的三点优势及理解:
1、内容和形式分离，网页前台只需要显示内容就行，形式上的美工交给 CSS 来处理。生成的 HTML 文件代码精简，更小打开更快。
2、改版网站更简单容易了，不用重新设计排版网页，甚至于不用动原网站的任何 HTML 和程序页面，只需要改动 CSS 文件就完成了所有改版。对于门户网站来说改版就像换件衣服一样简单容易。
3、搜索引擎更友好，排名更容易靠前。

第一点、内容和形式分离

网页前台只需要显示内容就行，形式上的美工交给 CSS 来处理。生成的 HTML 文件代码精简，更小打开更快。
这个是 DIV+CSS 技术最显著的特点，也是 CSS 存在的根源。完全的颠覆现在传统（table）网页设计的技术。所有现在用 table 制作的内容，都可以用 CSS 来解决掉，而且解决的更完美，更强大。不需要大家再表格套表格，让生成的网页文件大小更精简，更小。table 时代，一个页面表格达到 10 个以上是非常普遍的事情，但是现在用 DIV+CSS，一个 table 都可以不用，就完全达到之前的效果，这就直接导致网页文件大小比使用 table 时减少 50%-80%，更节约各位站长的硬盘空间，访问者打开网页时更快，而且用 div+CSS 时，不像以往使用 table 时，必须把全部 table 读取完了才显示页面内容，现在是可以读一个 div 就显示一个效果，大家打开网页不用等。好处真是明显而强大。
这个优点的确是显著的，凡是使用传统 table 建的网页，内容多的话，有时候达到 30K 左右都有可能，文件打了打开时，肯定就有 0.0 几秒的延迟。使用 DIV+CSS，你前台打开看到的全是直接内容，CSS 文件都是导入链接的，是另一个文件，根本和 HTML 文件大小没关系，这种生成的 HTML 文件，一个也就 10K 左右大小。

第二点，改版网站更简单容易了

不用重新设计排版网页，甚至于不用动原网站的任何 HTML 和程序页面，只需要改动 CSS 文件就完成了所有改版。
DIV+CSS 对于门户网站来说改版就像换件衣服一样简单容易，改版时，不用改动全站 HTML 页面，只需要重新写 CSS，再用新 CSS 覆盖以前的 CSS 就可以实现改版了。方便吧。

第三点，搜索引擎更友好，确实能够对 SEO 起到一定的帮助。

通过 DIV+CSS 对网页的布局，可以让一些重要的链接、文字信息，优先让搜索引擎蜘蛛爬取，这对于 SEO 也有帮助。

综上所述，个人感觉 DIV+CSS 不能太迷信它的很好很强大，它作为制作网页，美化网页的一个重要辅助是很强大方便的。可以弥补 table 制作框架和表格时的很多不足和美工上的缺点，但是完全只用它来做，太费时费力，对于全国中小型网站长来说，真的不太适合。我个人觉得用 table+DIV+CSS 是最好的组合，也是最省时省力的办法。

还需要再说明一下，本节讲得是 Table 布局页面和 CSS 布局页面的问题，讨论的是“布局页面”上用谁更好，并不是说在 CSS 布局的页面内不能用 Table，真正厉害的人物是 DIV、Table、CSS 用得恰到好处，他们三个各做各的事情，DIV 布局页面，Table 存储数据，CSS 给页面穿衣服！

【DIV+CSS 入门教程】xHTML+CSS 与 SEO

xHTML+CSS 与 SEO 的内容，后面章节会详细给大家介绍，这里就先说一些，让大家对 xHTML+CSS 与 SEO 有一定的认识，为后面制作页面打基础，毕竟我们做出来的页面还是要给搜索引擎看的，所以不能不提提 xHTML+CSS 与 SEO 的关系。

1)将页面中最重要的内容用 h1 标签括起来，h1 的内容就和页面 title 很自然的包含了站点或者页面的核心关键词，搜索引擎很重视 h1 标签的内容哟

2)合理的运用 h2、h3 等标题标签，他们对于页面来说就是文章不同的等级或者不同的功能区域的标志性元素

3)页面 meta 信息不可忽视，一定要包含页面核心的内容

4)为了便于搜索引擎更方便的抓取，要尽可能的保证 HTML 页面代码纯净，强调一下，既然是 xHTML+CSS 布局页面，所以 CSS 代码要单独写在一个文件内，保证 CSS 部分和 HTML 部分彻底分离；html 页面中使用 id 和 class，尽可能的避免 style="";尽量使用标准的 CSS 命名规范，从这里就可以看出你这个页面重构师是否专业哟；尽量使用 CSS 的缩写以节省代码，例如 padding:10px 20px 10px 20px;缩写为 padding:10px 20px;最好不要在 HTML 页面用 font、center 这种标签。

5)在 HTML 页面中 strong 标签是可以使用的，可以进一步强化关键词和相应的文字信息。

6)页面中的 javascript 代码会对搜索引擎分析页面内容产生干扰，可以将 javascript 代码封装在一个.js 文件中外部调用。

7)<img />尽可能的加入 alt 注释，因为百度和 google 都有搜索图片的功能，如果加了 alt，就更方便搜索蜘蛛的爬行，搜索相应关键词，就可能出现你网站上的图片，点击图片不就进入你的网站了嘛，就又多了点流量吧。

【DIV+CSS 入门教程】CSS 如何控制页面

本节主要讲解，两个内容，

第一：CSS 如何控制页面样式，有几种方式；

第二：这些方式出现在同一个页面时的优先级。

使用 xHTML+CSS 布局页面，其中有个很重要的特点就是内容与表象相分离，内容指 HTML 页面代码，表象就是 CSS 代码了，如果把页面看成穿着衣服的人的话，人就是 HTML，是内容，而衣服呢就是 CSS，是表
象，现在出现的问题是，如何让 CSS 去控制页面？或者说，如何让衣服穿在人身上，好体现出人得风格特点；不同的 CSS 就可以使页面出现不同的风格适用不同的网站，而不同的衣服，人穿上后就会体现
出不同的职业。

第一：如何让 CSS 去控制 HTML 页面效果呢？

有这么 4 种方式，行内方式、内嵌方式、链接方式、导入方式

1）行内方式

行内方式是 4 种样式中最直接最简单的一种，直接对 HTML 标签适用 style=""，例如：

```
1. <p style="color:#F00; background:#CCC; font-size:12px;"></p>
```

虽然这种方法比较直接，在制作页面的时候需要为很多的标签设置 style 属性，所以会导致 HTML 页面不够纯净，文件体积过大，不利于搜索蜘蛛爬行，从而导致后期维护成本高。

2）内嵌方式

内嵌方式就是将 CSS 代码写在<head></head>之间，并且用<style></style>进行声明，例如：

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2. <html xmlns="http://www.w3.org/1999/xhtml">
3. <head>
4. <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
5. <title>无标题文档</title>
6. <style type="text/css">
7. <!--
8. #div1{width:64px; height:64px; float:left;}
9. #div1 img{width:64px; height:64px;}
10. -->
11. </style>
12. </head>
13. <body>
14. <div id="div1"><img
   src="http://www.cssxuexi.cn/index/images/ico/2days.gif" /></div>
```

```
15. 全国的 CSS 爱好者汇聚于此，如果不来，你就 OUT 喽¯我们的口号是:
16. “分享自己的欢乐与痛苦，分享自己的经验与心得，分享自己的资料与资源”
17. 如果您也愿意，就加入我们吧¯
18. </body>
19. </html>
```

内嵌方式，大家应该也能意识到，即使有公共 CSS 代码，也是每个页面都要定义的，如果一个网站有很多页面，每个文件都会变大，后期维护也大，如果文件很少，CSS 代码也不多，这种方式还是很不错
的。

3）链接方式

链接方式是使用频率最高，最实用的方式，只需要在<head></head>之间加上

```
1. <link href="style.css" type="text/css" rel="stylesheet" />
```

，就可以了，这种方式将 HTML 文件和 CSS 文件彻底分成两个
或者多个文件，实现了页面框架 HTML 代码与美工 CSS 代码的完全分离，使得前期制作和后期维护都十分方便，并且如果要保持页面风格统一，只需要把这些公共的 CSS 文件单独保存成一个文件，其他的页
面就可以分别调用自身的 CSS 文件，如果需要改变网站风格，只需要修改公共 CSS 文件就 OK 了，相当的方便，这才是我们 xHTML+CSS 制作页面提倡的方式。
HTML 代码

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2. <html xmlns="http://www.w3.org/1999/xhtml">
3. <head>
4. <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
5. <title>无标题文档</title>
6. <link href="style.css" type="text/css" rel="stylesheet" />
7. </head>
8. <body>
9. <div id="div1"><img
    src="http://www.cssxuexi.cn/index/images/ico/2days.gif" /></div>
10. 全国的 CSS 爱好者汇聚于此，如果不来，你就 OUT 喽¯我们的口号是:
11. “分享自己的欢乐与痛苦，分享自己的经验与心得，分享自己的资料与资源”
12. 如果您也愿意，就加入我们吧¯
13. </body>
14. </html>
```

CSS 代码

```
1. #div1{width:64px; height:64px; float:left;}
2. #div1 img{width:64px; height:64px;}
```

4)导入方式

导入样式和链接样式比较相似，采用 import 方式导入 CSS 样式表，在 HTML 初始化时，会被导入到 HTML 文件中，成为文件的一部分，类似第二种内嵌方式。

具体导入样式和链接样式有什么区别，可以参看这篇文章《CSS:@import 与 link 的具体区别》，**不过我还是建议大家用链接方式!**

第二：四种样式的优先级

如果这上面的四种方式中的两种用于同一个页面后，就会出现优先级的问题，这里我就不再举例子来说明了，大家在下面自己证明一下下面的结论

四种样式的优先级别是(从高至低)：**行内样式、内嵌样式、链接样式、导入样式**。

【DIV+CSS 入门教程】CSS 选择器

上节课我们讲了一下 Css 通过什么方式去控制页面，如果不记得，我来帮大家回忆一下，总共有四种方式行内方式、内嵌方式、链接方式、导入方式，大家通过这四种方式就可以实现 CSS 对 HTML 页面样式的控制，如果要让这些样式对 HTML 页面中的元素实现一对一，一对多或者多对一的控制，这就需要用到 CSS 选择器，HTML 页面中的元素就是通过 CSS 选择器进行控制的。

CSS 选择器共有三种：**标签选择器、ID 选择器、类选择器**。

为了后面的对选择器的解释更容易理解，在这里先打个比喻，如果把你所处的环境视为 HTML 页面的话，环境里的每一个人则相当于 HTML 页面内标签元素，每个人都有一个 ID(身份证)，那么 html 中的每一个标签也都有自己的 ID，大家都知道 ID 是唯一的，不可能重复。

【标签选择器】

一个完整的 HTML 页面是有很多不同的标签组成，而标签选择器，则是决定哪些标签采用相应的 CSS 样式，(在大环境中你可能出于不同的位置，但是不管怎么样，你总是穿着同一套衣服，这件衣服就是由标签选择器事先给你限定好的，不管走到哪里都是这身衣服)比如，在 style.css 文件中对 p 标签样式的声明如下:

```
1. p{
2. font-size:12px;
3. background:#900;
4. color:090;
5. }
```

则页面中所有 p 标签的背景都是#900(红色)，文字大小均是 12px，颜色为#090(绿色)，这在后期维护中，如果想改变整个网站中 p 标签背景的颜色，只需要修改 background 属性就可以了，就这么容易!

【ID 选择器】

ID 选择器在某一个 HTML 页面中只能使用一次(当然也可以用好几次，不过就不符合 W3C 标准了，那页面也就不是标准页面喽¯，咱们的目的不就是为了做标准的页面么，所以建议大家不要在同一个 html 页面中多个标签拥有共同的 ID)，就像在你所处的环境中，你只有一

个 ID(身份证)，不可能重复！相信大家也能看出来，ID 选择器更具有针对性，如:
先给某个 HTML 页面中的某个 p 标签起个 ID，代码如下:

```
1. <p id="one">此处为 p 标签内的文字</p>
```

在 CSS 中定义 ID 为 one 的 p 标签的属性，就需要用到#，代码如下:

```
1. #one{
2. font-size:12px;
3. background:#900;
4. color:090;
5. }
```

这样页面中的某个 p 就会是 CSS 中定义的样式。

【类选择器】

这种选择器更容易理解了，就是使页面中的某些标签(可以是不同的标签)具有相同的样式，就像国庆中某个方阵中，肯定都是不同的人，却均穿红色衣服，手中高举花环，样式都是一样的，如果想让这一类人都有共同的样式，该怎么做呢¯呵呵，和 ID 选择器的用法类似，只不过把 id 换做 class，如下:

```
1. <p class="one">此处为 p 标签内的文字</p>
```

如果我还想让 div 标签也有相同的样式，怎么办呢？加上同样的 class 就可以了，如下

```
1. <div class="one">此处为 p 标签内的文字</div>
```

这样页面中凡是加上 class="one"的标签，样式都是一样的喽¯CSS 定义的时候和 ID 选择器差不多，只不过把#换成.，如下

```
1. .one{
2. font-size:12px;
3. background:#900;
4. color:090;
5. }
```

补充:一个标签可以有多个类选择器的值，不同的值用空格分开，如:

```
1. <div class="one yellow leftStyle">此处为 p 标签内的文字</div>
```

这样我们可以将多个样式用到同一个标签中，当然也可以，ID 和 class 一块用

```
1. <div id="div1" class="one yellow leftStyle">此处为 p 标签内的文字
2. </div>
```

【通用选择器】

到这里，前三种基本的选择器说完了，但是还需要给大家介绍一个 CSS 选择器中功能最强大但是用的最少的一种选择器“通用选择器”

```
1.  *{此处为 CSS 代码}
```

强大之处是因为他对父级中的所有 HTML 标签进行样式定义，可对具有共同样式的标签样式进行定义(有点小学数学中的提取公因式)，这样可以大大精简代码；既然有这么强大的功能为什么是用的最少呢，同样还是因为他的强大，他是对父级元素内的所有标签进行定义，所以只要你定义了，那么父级里面的所有的标签，甭管有没有必要，也都相当于加上了通用选择器里面的代码了，能这么说大家不能够完全理解，没关系，我给大家举个例子，请看下面

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2.  <html xmlns="http://www.w3.org/1999/xhtml">
3.  <head>
4.  <meta http-equiv="Content-Type" content="text/html;
    charset=gb2312" />
5.  <title>无标题文档</title>
6.  <style type="text/css">
7.  <!--
8.  #div1 *{
9.  background:#eee;/*设置 DIV1 里面所有的元素背景均为灰色*/
10. color:#333;/*设置置 DIV1 里面所有的元素的字体颜色均为黑色*/
11. }
12. -->
13. </style>
14. </head>
15. <body>
16. <div id="div1">
17. <p>这里是 p 标签区域</p>
18. <div>这里是 a 标签区域</div>
19. </div>
20. <div id="div2">
21. <p>这里是 p 标签区域</p>
22. <div>这里是 a 标签区域</div>
23. </div>
24. </body>
25. </html>
```

大家运行一下上面的例子，div1 里面的两个标签是不是样式一样，这就是通用选择器的强大之处，不管里面有多少个标签都会将样式加到所有标签内，如果 div1 里面得所有的标签都有一部分相同的 CSS 代码，那么可以把这部分代码提取出来，用通用选择器来定义，这样可以大大缩减代码，但是如果 div1 里面只要有一个和其他元素没有相同的代码，就不能用

通用选择器来定义，这也就是 CSS 通用选择器不灵活的一点。现在大家明白为什么通用选择器是选择器里面功能最强大的但又是用的最少的选择器了吧￣呵呵

对于通用选择器还有一个不得不提的用法，就是为了保证作出的页面能够兼容多种浏览器，所以要对 HTML 内的所有的标签进行重置，会将下面的代码加到 CSS 文件的最顶端

```
1.  *{margin:0;  padding:0;}
```

为什么要这么用呢，因为每种浏览器都自带有 CSS 文件，如果一个页面在浏览器加载页面后，发现没有 CSS 文件，那么浏览器就会自动调用它本身自带的 CSS 文件，但是不同的浏览器自带的 CSS 文件又都不一样，对不同标签定义的样式不一样，如果我们想让做出的页面能够在不同的浏览器显示出来的效果都是一样的，那么我们就需要对对 HTML 标签重置，就是上面的代码了，但是这样也有不好的地方，因为 HTML4.01 中有 89 个标签，所以相当于在页面加载 CSS 的时候，先对这 89 个标签都加上了{margin:0; padding:0;}，在这里我不建议大家这么做，因为 89 个标签中需要重置的标签是很少数，没有必要将所有的标签都重置，需要哪些标签重置就让哪些标签重置就可以了，如下

```
1.  body,div,p,a,ul,li{margin:0;  padding:0;}
```

如果还需要 dl、dt、dd 标签重置，那就在上面加上就可以了，如下

```
1.  body,div,p,a,ul,li,dl,dt,dd{margin:0;  padding:0;}
```

用到那些就写那些，这点也可以看做衡量页面重构师制作页面水平的高低，以及是否专业的一个方面到这里大家更应该明白这句话“通用选择器是功能最强大但是用的最少的选择器”了吧￣^_^

OK！选择器的内容我向大家应该都明白了，后面就继续讲解一下“**选择器的集体声明**”和“**选择器的嵌套**”

【选择器的集体声明】

在我们使用选择器的时候，有些标签样式是一样的，或者某些标签都有共同的样式属性，我们可以将这些标签集体声明，不同的标签用“,”分开，比如：

```
1.h1,h2,h3,h4,h5,h6{color:#900;}

1.#one,#three,.yellow{font-size:14px;}
2.#one{background:#ccc;}
3.#three{background:#ccc;}
4..yellow{background:#ccc;}
```

和小学的提取公因式差不多，把共同的部分提取出来，这么做的好处，相同的部分共同定义，不同的部分单独定义，保证风格统一，样式修改灵活，这也是优化 CSS 代码的一块，要记住哟￣

【选择器的嵌套】
选择器也是可以嵌套的，如:

```
1. #div1 p a{color:#900;}/*意思是在 ID 为 div1
2. 内的 p 标签内的链接 a 标签的文字颜色为红色*/
```

这样的好处就是不需要在单独的为 ID 为 div1 的标签内的 p 标签内的 a 标签单独定义 class 选择器或者 ID 选择器，CSS 代码不就少了嘛¯同样也是 CSS 代码优化的一块。

到这里，基本的选择器说完了，但是还需要给大家介绍一个“通用选择器”

```
1. *{此处为 CSS 代码}
```

好，这节课主要讲解了三种 CSS 代码选择器、选择器的声明、选择器的嵌套三块知识，要掌握好

【DIV+CSS 入门教程】CSS 选择器规范化命名

规范的命名也是 Web 标准中的重要一项，标准的命名可以更好的看懂代码，我想大家应该都有这种经历，某日翻出自己过去写的代码居然看不懂了，呵呵，为了避免这种情况我们就要规范化命名，再说了，现在一个项目不是一个人就可以完成的，是需要大家互相合作的，如果没有规范化命名，别人就无法看懂你的代码，大大降低了工作效率，所以必须规范化命名，这样还显着咱专业！

好了不多说了，关于 CSS 命名法，和其他的程序命名差不多，也是有三种：**骆驼命名法，帕斯卡命名法，匈牙利命名法**。

【骆驼命名法】

说到骆驼大家肯定会想到它那明显的特征，背部的隆起，一高一低的，我们的命名也要这样一高一低，怎么才能这样，就用大小写字母呗¯，大写的英文就相当于骆驼背部的凸起，小写的就是凹下去的地方了，但是这个也是有规则的，就是第一个字母要小写，后面的词的第一个字母就要用大写，如下:

```
1.  #headerBlock
2.  .navMenuRedButton
```

【帕斯卡命名法】
这种命名法同样也是大小写字母混编而成，和骆驼命名法很像，只有一点区别，就是首字母要大写，如下

```
1.  #HeaderBlock
2.  .NavMenuRedButton
```

【匈牙利命名法】
匈牙利命名法，是需要在名称前面加上一个或多个小写字母作为前缀，来让名称更加好认，更容易理解，比如：

```
1. #head_navigation
2. .red_navMenuButton
```

以上三种，前两种（骆驼命名法、帕斯卡命名法）在命名 CSS 选择器的时候比较常用，当然这三种命名法可以混合使用，只需要遵守有一个原则就可以，就是“容易理解，容易认，方便协同工作”就 OK 了，没有必要强调是那种命名法。

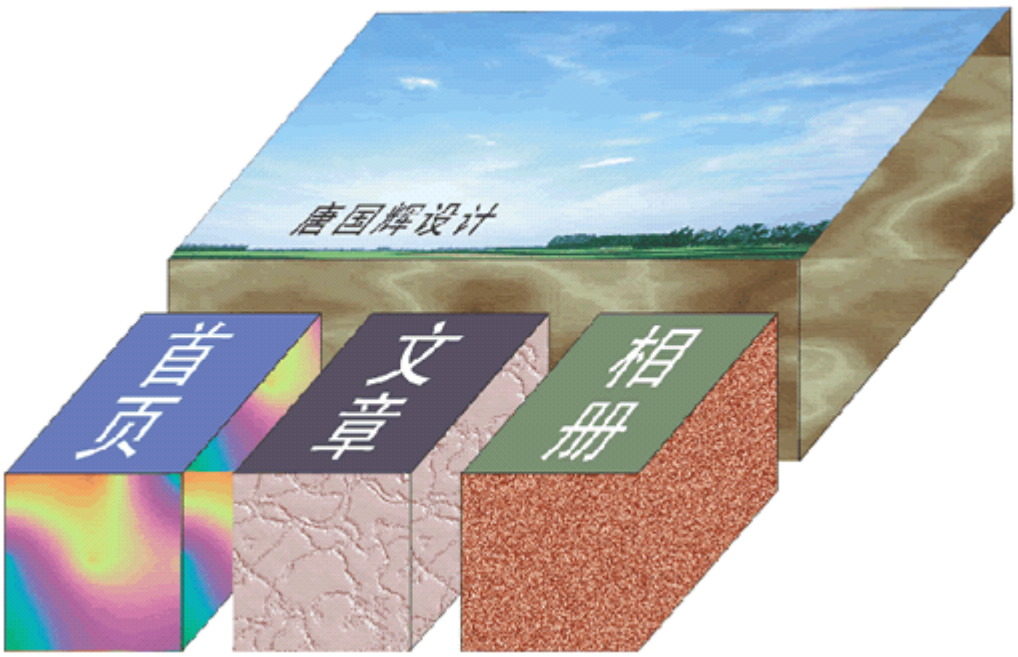
以下为于页面模块的常用命名
头：header
内容：content/container
尾：footer
导航：nav
侧栏：sidebar
栏目：column
页面外围控制整体布局宽度：wrapper
左右中：left right center
登录条：loginbar
标志：logo
广告：banner
页面主体：main
热点：hot
新闻：news
下载：download
子导航：subnav
菜单：menu
子菜单：submenu
搜索：search
友情链接：friendlink
页脚：footer
版权：copyright
滚动：scroll
内容：content
到这节课，都是 CSS 非常基础的知识，是为了照顾没有一点基础的同学，从下节课开始，将介绍 CSS 布局页面中的很重要的两个概念，也是必须要掌握的概念，如果不能很好理解的话后面再布局页面的时候就会出现很多问题。
1）盒子模型
2）内链元素 VS 块状元素

【CSS 入门】理解盒子模型

盒模型，是 XHTML+CSS 布局页面中的核心！是关系到设计中排版定位的关键，都遵循盒模型规范，例如 、<p>、<a>……盒模型包含
(外边距)margin，(内边距)padding ，(内容)content，(边框)border。

下图是 CSS 盒模型的示意图

这些属性我们可以把它转移到我们日常生活中的盒子（箱子）上来理解，日常生活中所见的盒子也具有这些属性，所以叫它盒子模式。那么内容就是盒子里装的东西；而填充就是怕盒子里装的东西（贵重的）损坏而添加的泡沫或者其它抗震的辅料；边框就是盒子本身了；至于边界则说明盒子摆放的时候的不能全部堆在一起，要留一定空隙保持通风，同时也为了方便取出嘛。在网页设计上，内容常指文字、图片等元素，但是也可以是小盒子（DIV 嵌套），与现实生活中盒子不同的是，现实生活中的东西一般不能大于盒子，否则盒子会被撑坏的，而 CSS 盒子具有弹性，里面的东西大过盒子本身最多把它撑大，但它不会损坏的。填充只有宽度属性，可以理解为生活中盒子里的抗震辅料厚度，而边框有大小和颜色之分，我们又可以理解为生活中所见盒子的厚度以及这个盒子是用什么颜色材料做成的，边界就是该盒子与其它东西要保留多大距离。在现实生活中，假设我们在一个广场上，把不同大小和颜色的盒子，以一定的间隙和顺序摆放好，最后从广场上空往下看，看到的图形和结构就类似我们要做的网页版面设计了，如下图。



由“盒子”堆出来的网页版面

传统的前台网页设计是这样进行的：根据要求，先考虑好主色调，要用什么类型的图片，用什么字体、颜色等等，然后再用 Photoshop 这类软件自由的画出来，最后再切成小图，再不自由的通过设计 HTML 生成页面，改用 CSS 排版后，我们要转变这个思想，此时我们主要考虑的是页面内容的语义和结构，因为一个强 CSS 控制的网页，等做好网页后，你还可以轻松的调你想要的网页风格，况且 CSS 排版的另外一个目的是让代码易读，区块分明，强化代码重用，所以结构很重要。如果你想说我的网页设计的很复杂，到后来能不能实现那样的效果？我要告诉你的是，如果用 CSS 实现不了的效果，一般用表格也是很难实现的，因为 CSS 的控制能力实在是太强大了，顺便说一点的是用 CSS 排版有一个很实用的好处是，如果你是接单做网站的，如果你用了 CSS 排版网页，做到后来客户有什么不满意，特别是色调的话，那么改起来就相当容易，甚至你还可以定制几种风格的 CSS 文件供客户选择，又或者写一个程序实现动态调用，让网站具有动态改变风格的功能。

【CSS 入门】DIV CSS 块状元素和内联元素

我们先来分析一下块级元素、内联级元素的定义和解析：

块元素(block element)一般是其他元素的容器元素，块元素一般都从新行开始，它可以容纳内联元素和其他块元素，常见块元素是段落标签'P"。“form"这个块元素比较特殊，它只能用来容纳其他块元素。

如果没有 css 的作用，块元素会顺序以每次另起一行的方式一直往下排。而有了 css 以后，我们可以改变这种 html 的默认布局模式，把块元素摆放到你想要的位置上去。而不是每次都愚蠢的另起一行。需要指出的是，table 标签也是块元素的一种，table based layout 和 cssbased layout 从一般使用者（不包括视力障碍者、盲人等）的角度来看这两种布局，除了页面载入速度的差别外，没有其他的差别。但是如果普通使用者不经意点了查看页面源代码按钮后，两者所表现出来的差异就非常大了。基于良好重构理念设计的 css 布局页面源码，至少也能让没有 web 开发经验的普通使用者把内容快速的读懂。从这个角度来说，css layout code 应该有更好的美学体验。

你能够把块容器元素 div 想象成一个个 box，或者如果你玩过剪贴文载的话，那就更加容易理解了。我们先把需要的文章从各种报纸、杂志总剪 下来。每块剪下来的内容就是一个 block。然后我们把这些纸块按照自己的排版意图，用胶水重新贴到一张空白的新纸上。这样就形成了你自己独特的文摘快报了。作为一种技术的延伸，网页布局设计也遵循了同样的模式。

内联元素(inline element)一般都是基于语义级(semantic)的基本元素。内联元素只能容纳文本或者其他内联元素，常见内联元素 “a”。

需要说明的是: inline element 的中文叫法，有多种内联元素、内嵌元素、行内元素、直进式元素。基本上没有统一的翻译，爱怎么叫怎么叫吧。另外提到内联元素，我们会想到有个 display 的属性是 display: inline;这个属性能够修复著名的 IE 双倍浮动边界问题。

块元素(block element)和内联元素(inline element)都是 html 规范中的概念。块元素和内联元素的基本差异是块元素一般都从新行开始。而当加入了 css 控制以后，块元素和内联元素的这种属性差异就不成为差异了。比如，我们完全可以把内联元素 cite 加上 display: block 这样的属性，让他也有每次都从新行开始的属性。

块元素（block element）

- address - 地址
- blockquote - 块引用
- center - 举中对齐块
- dir - 目录列表
- div - 常用块级容易，也是 css layout 的主要标签
- dl - 定义列表
- fieldset - form 控制组
- form - 交互表单
- h1 - 大标题
- h2 - 副标题
- h3 - 3 级标题
- h4 - 4 级标题
- h5 - 5 级标题
- h6 - 6 级标题
- hr - 水平分隔线
- isindex - input prompt
- menu - 菜单列表
- noframes - frames 可选内容，（对于不支持 frame 的浏览器显示此区块内容）
- noscript - 可选脚本内容（对于不支持 script 的浏览器显示此内容）
- ol - 排序表单
- p - 段落
- pre - 格式化文本
- table - 表格
- ul - 非排序列表

内联元素（inline element）

- a - 锚点
- abbr - 缩写
- acronym - 首字
- b - 粗体(不推荐)
- bdo - bidi override
- big - 大字体
- br - 换行
- cite - 引用
- code - 计算机代码(在引用源码的时候需要)
- dfn - 定义字段
- em - 强调
- font - 字体设定(不推荐)
- i - 斜体
- img - 图片
- input - 输入框
- kbd - 定义键盘文本
- label - 表格标签
- q - 短引用
- s - 中划线(不推荐)
- samp - 定义范例计算机代码
- select - 项目选择
- small - 小字体文本
- span - 常用内联容器，定义文本内区块
- strike - 中划线
- strong - 粗体强调
- sub - 下标
- sup - 上标
- textarea - 多行文本输入框
- tt - 电传文本
- u - 下划线
- var - 定义变量

当内联元素，在 CSS 中定义下列属性中的一种，便具有块元素的特征

1）display: block;

2）float: left;　（不但具有块元素的特征，同时像左侧浮动）

但是这时候的内联元素，虽然具有块状元素的特征，但是这两种有一点区别，第一种，彻头彻尾和块元素一模一样，都要单独占一行，从左至右，前提没有 width 和 height 属性，严格遵循流动布局模型块状元素的流动方式，自上至下流动，第二种，大小是恰好能将内容包含，并且右侧浮动，可以多个在一行。

当加上 **position: absolute/relative** 的时候，块状元素和内联元素，就不受父级区域的限制了，可以移动到任何位置，此时如果加上 width 和 height 属性，那么就具有层的特征了。

（加上 width 和 height 还有一点好处，就是可以兼容 IE 浏览器了，所有的浏览器现实效果都一样了）

【DIV+CSS 入门教程】盒模型、块状元素与内联元素、CSS 选择器

课程关键词：盒模型、块状元素与内联元素、Css 选 择 器
【例子】
要求：
1）宽度、高度均是 200 像素；
2）颜色为红色#900；
自己做做，看看能不能作出来？先不要看我的代码，如果真的做不出来，就下载下来，跟着我下面说的一步一步修改。

下面是我的代码：  例子 1.rar （498 Bytes）
在 IE6 和 FF 中显示效果如下图：

怎么样，比较容易吧¯，但是你们有没有发现，红色区域离浏览器的顶部和左边的边距 IE6 和 FF 的不一样，有没有发现？这样的话，我们作出来的页面浏览器就不兼容了，效果不一样了？为什么会这样？
这是因为每个浏览器都有一个内置的 CSS 文件，当你没有对某个标签的属性设置的时候，浏览器就会应用内置的 CSS 文件，怎么才能做到浏览器兼容？不着急，你只需要在 CSS 文件中，将我们目前应用到得标签 body 和 div 置零就 OK 了，代码这么写

```
body,div{padding: 0; margin: 0;}
```

当把这句话加上之后，是不是两款浏览器显示效果一样了吧¯如下图

好，我们接着来，现在再加一个条件

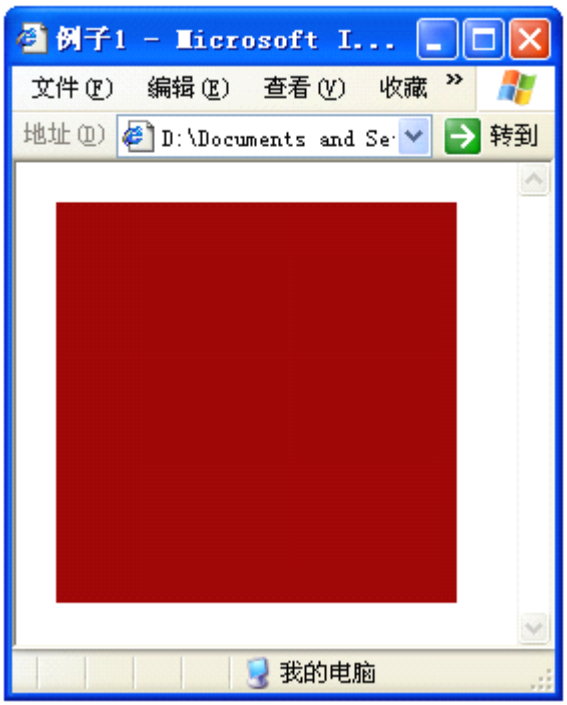
3）让红色区域与浏览器的顶部和左边距离为 20 像素；

怎么样，有没有思路？没有思路没关系，继续向下看

我们，只需要设置红色方块的外边距就可以了，添加如下 CSS 代码：

```
margin-top:20px;
margin-left:20px;
```

效果如下图



这样就使红色区域定位于页面坐标(20,20)处了，与浏览器上边距和左边距都为 20 像素。
不过上面的这种写法我们可以精简为
margin:20px 0 0 20px;
其中的数值顺序是：上右下左。
而 margin:20px 0;则和 margin:20px 0 20px 0;是等价的哟¯只不过是更加精简而已，这样写 CSS 加载速度会更快。

我们接着将问题延伸，怎样才能让红色区域水平定位于浏览器的正中间，无论浏览器窗口的大小，显示器分辨率的大小。
也很简单，刚刚加的两句话"margin-top:20px;margin-left:20px;"修改为
margin:0 auto;
怎么样，有意思吧，红色区域是不是位于浏览器的正中间了¯

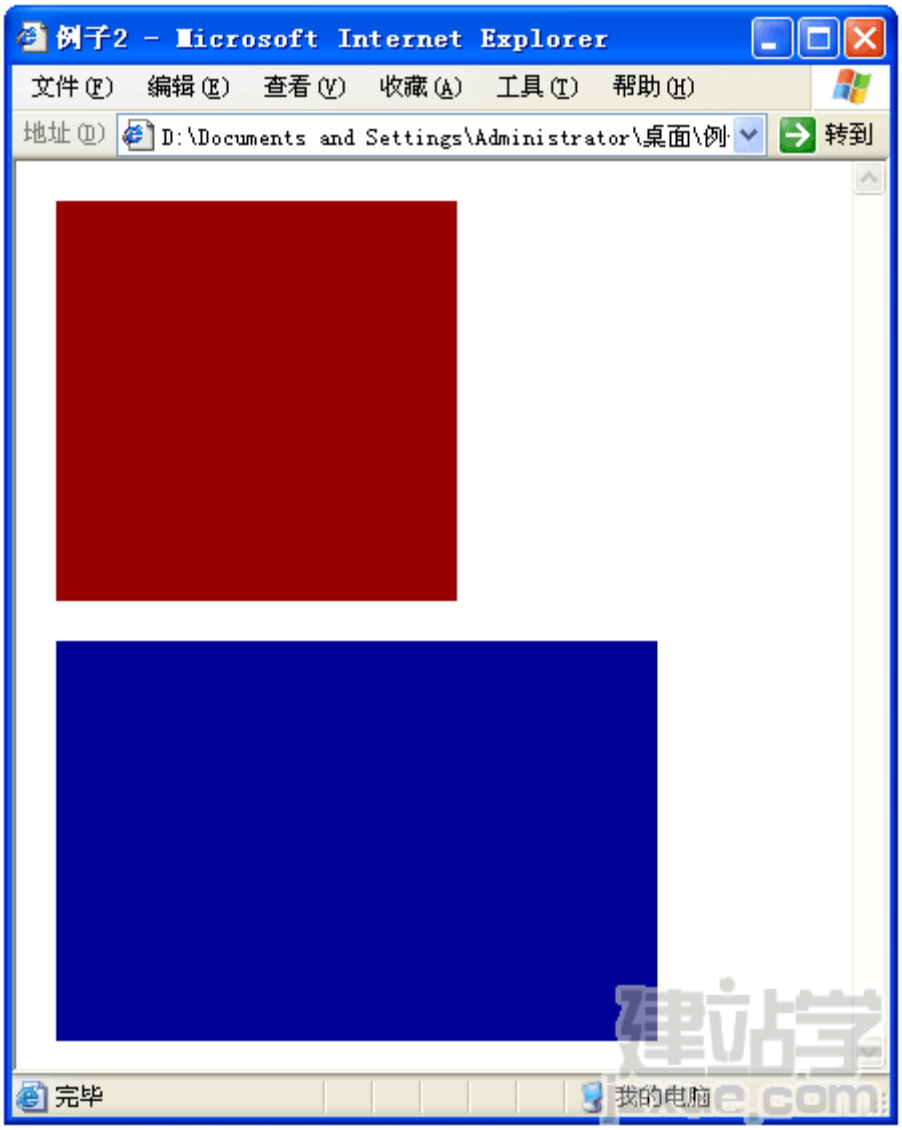
好¯！到这里第一节课结束，是不是很简单，或者太简单了!!!KwooJan 可以保证,后面的课程照样很简单，当你看完教程，肯定会说 XHTML+CSS 就这么简单！easy!
如果你有看不懂的，赶紧点击文章最上面的课程关键词，只要你看了这些关键词，相信你绝对能明白！

第一次写教程，不知道大家能不能看懂，能不能接受，如果你觉着不错，就顶我吧，如果你有建议或者想法，就直接留言，我会在下节课改进！

【DIV+CSS 入门教程】理解 Float 的含义

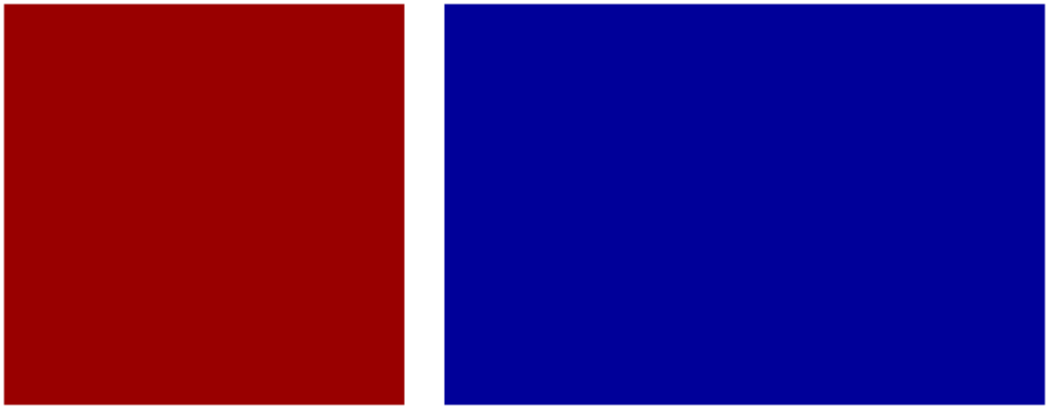
页面布局有两种方式
1）浮动 Float
2）定位 Position

今天就来一个小小的练习，让大家理解 Float 的含义
【例子】
要求:
1）两个方块，一个红色#900，一个蓝色#009;
2）红色方块宽度和高度均为 200 像素，蓝色方块
宽度为 300 像素，高度为 200 像素;
3）红色方块蓝的上外边距(margin-top)和左外边距(margin-left)均为 20 像素;
页面效果如下:

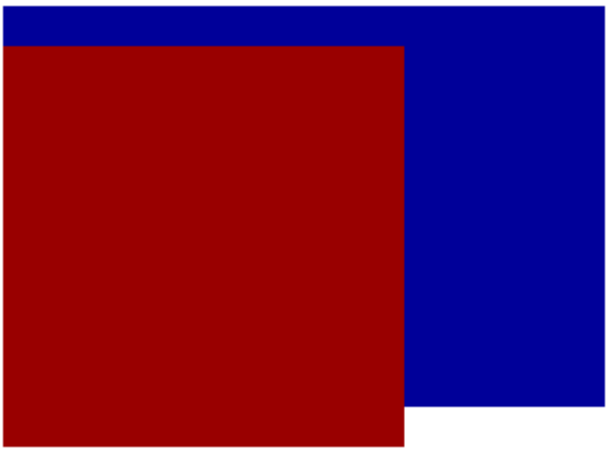


源代码： 例子 2.rar

大家应该注意到了，虽然红色方块的宽度并不是 100%，但是蓝色并未和红色处于同一行，这就是块状元素比较“霸道”的一点，（**即使块状元素的宽度不是 100%，它也不允许其他元素和他同在一行**）为了消除这种“霸权”，让红色和蓝色方块都处在一行，如图



此时就需要拿出我们的利器 Float！只需要在红色方块的 CSS 里面加上“float:left;”，这时候在 IE6 中可以看到蓝色方块的确跑到红色方块的后面了，并且处于一行了，但是在 FireFox 中却变成了如下效果：



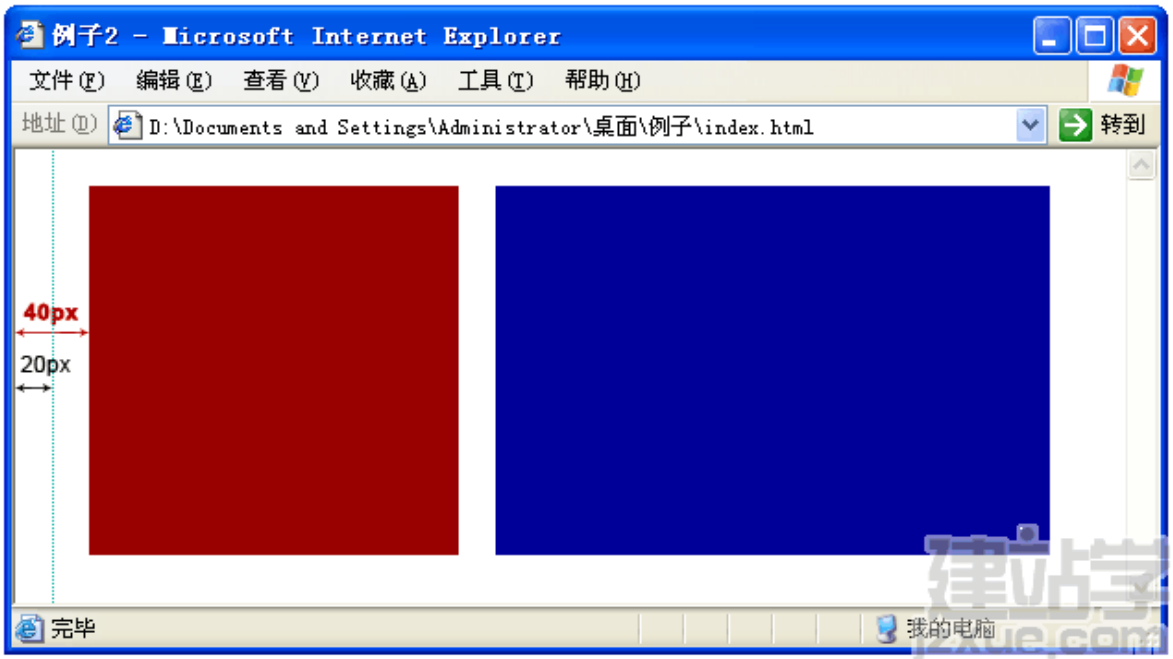
这时候就需要注意了，FF 中如果前面的区域浮动了，后面的那个区域很有可能就会和前面的区域发生重叠并错位。
怎么才能解决这个问题，解决这个浏览器兼容的问题，很容易，只需要在蓝色方块的 CSS 代码中也加入“Float:left;”，问题就解决了，加上试试，看看在 FF 中蓝色方块是不是和红色方块处于一行了¯

到这里，大家应该明白 Float 的作用了吧，就是为了消除块状元素“霸权主义”的一把利器！在布局页面的时候有时候是需要消除块状元素霸权主义才能布局好哟，比如 KwooJan 的博客中间内容部分，分为左边(LEFT)和右边(RIGHT)，就是要用上面这个方法布局的哟，如下图

下载 (105.27 KB)
2009-5-24 23:19

下载 (105.27 KB)
2009-5-24 23:19

细心的同学会注意到，在 IE6 中红色方块距离浏览器的左边距并不是 CSS 代码中定义的 20 像素，而是 40 像素，如下图：



其实这是 IE6 的一个 BUG，（IE6 双倍边距 BUG），只要满足下面 3 个条件才会出现这个 BUG:

1）要为块状元素;
2）要左侧浮动;
3）要有左外边距(margin-left);

解决这个 BUG 很容易，只需要在相应的块状元素的 CSS 树形中加入“display: inline;”，代码如下:

```
#redBlock{
width: 200px;
height: 200px;
background: #900;
margin-top: 20px;
margin-left: 20px;
float: left;
display: inline;
}
```

现在再看看，是不是 IE6 和 FF 显示一样了呢¯

呵呵，这节课也比较容易吧，如果大家有不明白的可以留言，我会做进一步解释。
下节课，我们讲讲“浮动清除(Clear)”问题!

最终代码: **最终代码.rar** (587 Bytes)
精简后的代码 CSS 加载更快，大家一看就明白了^_^

CSS 标签重置

对于 XHTML+CSS 布局起着一个重要的决定性作用，它是提高页面浏览器兼容性的第一步。

那么什么是标签重置呢?

顾名思义，就是对 HTML 中的所有标签属性重置，因为每个浏览器都有一个自己默认的 CSS 文件，对 HTML 中的所有的标签进行定义，以便没有定义 CSS 的页面能够正常显示在页面，页面在加载的时候如果没有找到自带的 CSS 文件，浏览器就用事先为您准备好的 CSS 样式，但是这个对于页面布局，没有什么用，所以我们需要将它们重新设置，不过标签有那么多，属性那么多，怎么设置？！这都是个问题!

哈哈，不要着急，问题很容易解决，其实在我们布局页面的时候，将最容易影响页面布局的是 HTML 标签中的内外边距，只要我们将最常用的标签的内外边距设为零就 OK 了¯比如一个页面中用到下面 div,p,ul,li 四个标签那么我们的重置代码就要这么写

```
body,div,p,ul,li{margin:0;  padding:0;}
```

因为 body 标签不同的浏览器定义的内边距是不一样的，所以在上面代码中加入 body。
如果后面我用到 form,h1,h2 标签的话，我就再添加上去就 OK 了

```
body,div,p,ul,li,form,h1,h2{margin:0;  padding:0;}
```

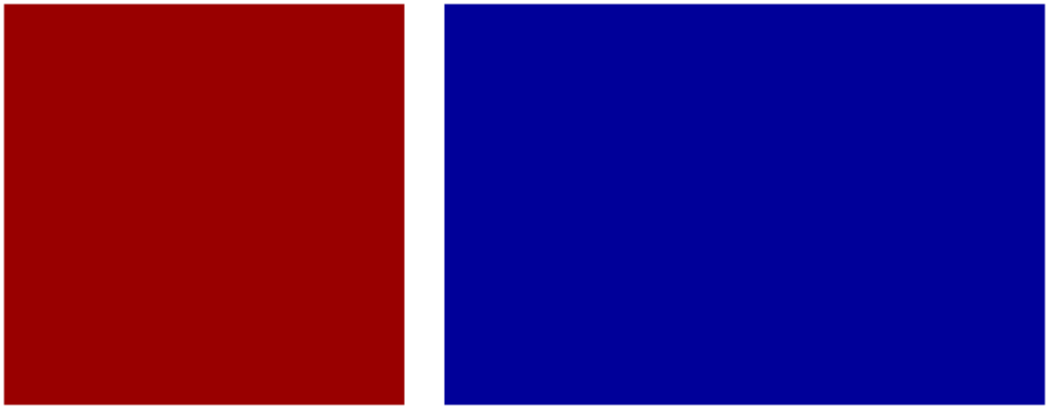
用到几个标签就写几个!

有些人很喜欢用*{margin:0; padding:0;},其实 KwooJan 不建议这么用，因为 HTML 标签太多了，HTML4.01 参考手册中就多达 89 个，我们平时常用的也就那么几个，所以没有必要将所有标签重置，这样反而使页面加载速度变慢！特别是对于大站点，更不提倡!

不过呢，如果页面出了问题，用这个可以检验一下，是不是有标签没有重置而导致的布局错位哟¯!

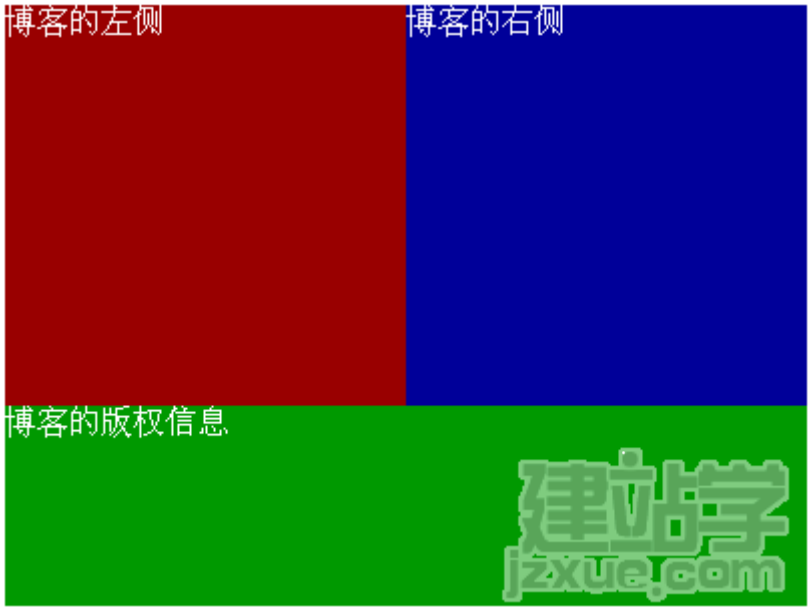
【DIV+CSS 入门教程】清除浮动 Clear

还记得第二课我们做的例子的效果么？最后效果是，红色方块和蓝色方块都处于一行，我们使用“Float:left”，打击了块状元素的“霸道”即块状元素不允许其他元素和它处于同一行。我们将红色方块的 CSS 代码中加入了“Float:left;”后，红色方块终于允许蓝色方块和它处于同一行。如图:

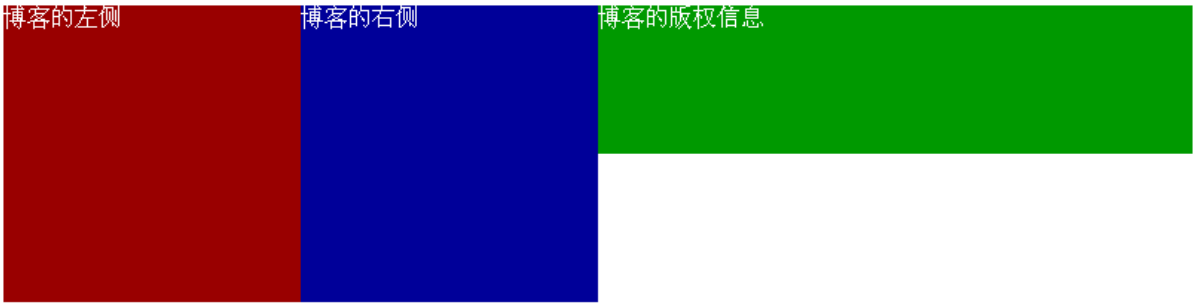


我们换一种方法表达上面的意思，因为红色方块的“左侧浮动”，才导致蓝色方块上移至红色方块的尾后；

在上个例子中，为了达到浏览器兼容性，我们分别在红色蓝色方块 CSS 代码中分别加了“Float:left;”，这样 IE 和 FF 中显示效果一样，如果此时我们还想放一个宽度 400 像素，高度 100 的绿色方块，并让其处于第二行，效果如下图：



可是这时候不管怎么放，在 IE 中的效果始终是



导致绿色拍到蓝色的后面这种情况就是因为蓝色方块 CSS 代码中含有"Float:left;"，但是为了浏览器兼容性，又不能去掉(什么？这句话看不明白，只能说明第二节课你没有好好学，好好品味！)，怎么办？

好办¯！只要在 CSS 代码中加入下面这段代码：

```
.clear{clear:both;}
```

并在 HTML 代码中加入下面代码：

```
<div class="clear"></div>
```

上面这句话究竟加在那个位置呢，要加在蓝色方块和绿色方块中间，然后看看效果是不是我们想要的了¯^_^

```
<div id="redBlock">博客的左侧</div>
<div id="blueBlock">博客的右侧</div>
<div class="clear"></div>
<div id="greenBlock">博客的版权信息</div>
```

目的就是为了清除蓝色方块的浮动对下面绿色方块的影响！是影响哟¯是清除影响，而不是清楚蓝色方块的浮动，或者说清除蓝色方块的浮动对下面区域块产生的作用！（仔细品品我说的这句话！）

如果还是不明白，你就在红色方块和蓝色方块中间加上“ ”，看看效果变成什么样子，然后再品品我刚才说的话！

这节课就到这里，下节课我们做一个导航条，很实用的哟！一定要把前三节吃透，不然第四节会跟不上理解不透！

CSS 标签重置

对于 XHTML+CSS 布局起着一个重要的决定性作用，它是提高页面浏览器兼容性的第一步。

那么什么是标签重置呢？

顾名思义，就是对 HTML 中的所有标签属性重置，因为每个浏览器都有一个自己默认的 CSS 文件，对 HTML 中的所有的标签进行定义，以便没有定义 CSS 的页面能够正常显示在页面，页面在加载的时候如果没有找到自带的 CSS 文件，浏览器就用事先为您准备好的 CSS 样式，但是这个对于页面布局，没有什么用，所以我们需要将它们重新设置，不过标签有那么多，属性那么多，怎么设置？！这都是个问题！

哈哈，不要着急，问题很容易解决，其实在我们布局页面的时候，将最容易影响页面布局的

是 HTML 标签中的内外边距，只要我们将最常用的标签的内外边距设为零就 OK 了¯比如一个页面中用到下面 div,p,ul,li 四个标签那么我们的重置代码就要这么写

```
body,div,p,ul,li{margin:0;  padding:0;}
```

因为 body 标签不同的浏览器定义的内边距是不一样的，所以在上面代码中加入 body。
如果后面我用到 form,h1,h2 标签的话，我就再添加上去就 OK 了

```
body,div,p,ul,li,form,h1,h2{margin:0;  padding:0;}
```

用到几个标签就写几个！

有些人很喜欢用*{margin:0; padding:0;},其实 KwooJan 不建议这么用，因为 HTML 标签太多了，HTML4.01 参考手册中就多达 89 个，我们平时常用的也就那么几个，所以没有必要将所有标签重置，这样反而使页面加载速度变慢！特别是对于大站点，更不提倡！

不过呢，如果页面出了问题，用这个可以检验一下，是不是有标签没有重置而导致的布局错位哟¯！

几个 css 元素的简单解释 div ul dl dt oldiv，这个很常见，块级元素，div 尽量少用，和 table 一样，嵌套越少越好。

ol 有序列表

```
<ol>
<li>……</li>
<li>……</li>
<li>……</li>
</ol>
```

表现为:
1……
2……
3……

ul 无序列表，表现为 li 前面是大圆点而不是 123

```
<ul>
<li>……</li>
<li>……</li>
</ul>
```

很多人容易忽略 dl dt dd 的用法

dl 内容块
dt 内容块的标题

dd 内容

可以这么写:

```
<dl>
<dt>标题</dt>
<dd>内容 1</dd>
<dd>内容 2</dd>
</dl>
```

dt 和 dd 中可以再加入 ol ul li 和 p

理解这些以后，在使用 div 布局的时候，会方便很多，w3c 提供了很多元素辅助布局。

课程开始:

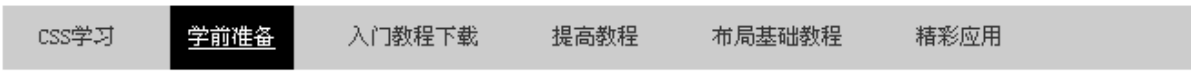
前三节课，我们知道了什么是“内容块状元素和内联元素”，以及 XHTML+CSS 布局的核心概念“盒子模型”，同时又学习了一下页面布局中两种方法中的一种方法“浮动”,这次我们就利用这三个概念，来制作一款，经典的导航条，别看它其貌不扬，可是网上所有的导航条都可以再它的基础上修改而来哟¯，厉害吧¯！其实理论都是一样的，只要你能理解并学会这节课的内容，以后再困难的导航条你都可以很应对，EASY ！！！

OK！我们要做的导航条的效果如下:

鼠标移动上去背景变黑，并且字体颜色变成白色

OK！我们要做的导航条的效果如下:

鼠标移动上去背景变黑，并且字体颜色变成白色



其实做这款导航条很容易的，你只需要动动鼠标敲敲键盘，跟着 KwooJan 做就是了，呵呵

【第一步】我们要先做一个容器（要求: ID 为“nav”，宽度为 960px，高度为: 35px，位于页面水平正中，与浏览器顶部的距离是 30px;),这个容器就是放我们的导航的哟¯代码如下:

HTML 代码:

```
<div id="nav"></div>
```

CSS 代码:

```
#nav{
width:960px;
height:35px;
background:#CCC;/*为了便于查看区域范围大小，故而加个背景色*/
margin:0 auto;/*水平居中*/
margin-top:30px;/*顶部 30px*/
}
```

还有一点需要提醒的是，为了页面在浏览器的兼容性，不要忘记在 CSS 文件顶部加入标签重置代码哟¯

```
代码:
body,div{padding:0; margin:0;}
```

这里就不多说了，不明白的就看，课程顶部的课程关键词
怎么样，作出来了没有，效果是不是一个灰色条，位于页面的正中间，并且所有浏览器效果一样呢¯呵呵
(如果没有做出来证明你没有认真看教程哟¯用这种态度看教程会学不好的，本身我把整个 XHTML+CSS 的理论都压缩到教程里了，或者说教程的“知识点浓度”很高，有时需要你一字不漏的去品我说的话，不要一目十行的去看哟¯只要你把我写的教程逐字逐句的研究透了，KwooJan 保证你以后只要做出个页面就很 Easy 的兼容各种浏览器，并且代码绝对的精简！CSS 文件加载速度大大提升哟¯)

【第二步】
盒子做好了，我们就要往里面放导航条中的内容了“CSS 学习 学前准备 入门教程 提高教程 布局教程 精彩应用”，如果我们把这内容(目前有 6 个)，当成酒杯的话，如果直接放到盒子里面的话，肯定会乱，并且还会东倒西歪，一点顺序都没有，但是我们平时会用一个隔板将每个酒杯隔开，这样就是酒杯很有序的放入盒子，并且牢稳而且防震，方便使用！现在我们把这个隔板叫做“有序列表”起个英文名字叫：ul,里面的每个单元格我们也给起个英文名字叫“li”，大家想想里面的这个 ul 是不是和盒子里面的空间一样大，小了，酒杯放不进去,大了杯子就会不稳,所以我们定义 UL 的时候大小一定也要和外面的盒子一样大哟¯,所以呢，我们的代码就知道怎么写了吧

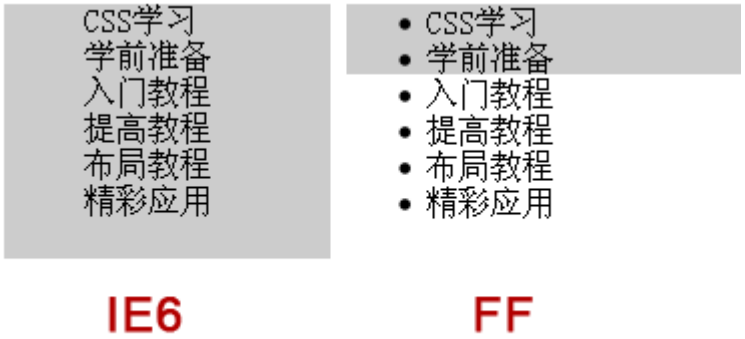
HTML 代码

```
<div id="nav">
    <ul>
        <li>CSS 学习</li>
        <li>学前准备</li>
        <li>入门教程</li>
        <li>提高教程</li>
        <li>布局教程</li>
        <li>精彩应用</li>
    </ul>
</div>
```

CSS 代码:

```
#nav ul{
        width:960px;
        height:35px;
}
```

效果作出来了没有，下面是在 IE6 和 FF 中显示效果(其他浏览器效果大家自己测试，总结规律)：



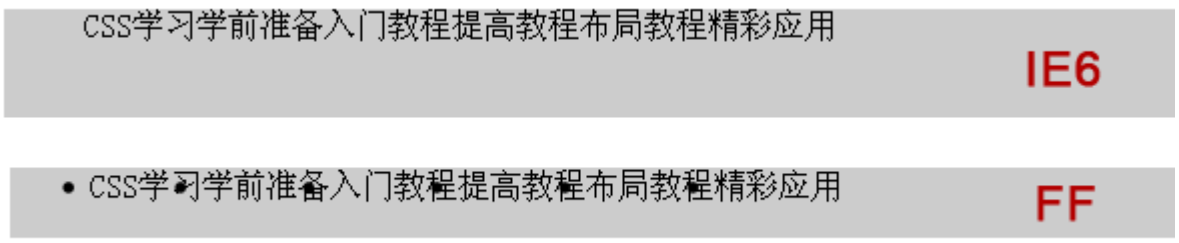
效果不一样吧，没关系，IE6 中盒子被撑大，FF 中却没有，但是我们的“酒杯”却出来了，还有我们不希望我们的酒杯纵向排列，而是横向排列，怎么办呢？给大家一分钟时间想——想出来了没有？什么没有？

没关系，我带着大家想想，因为<li>标签也是块状元素，所以他也有块状元素的“霸道”，不允许其他元素和自己处于同一行，总共六个<li>,所以他们六个就像台阶似的纵向排列起来了，我提示到这里，大家应该知道怎么做才能让这些“酒杯”横向排列了吧！^_^

对喽¯用浮动 Float！可是让谁浮动呢，当然是<li>标签喽¯代码如下：

```
#nav ul li{ float:left;}
```

效果是不是和下面的一样呢



大家会发现虽然“酒杯”横向排列了，但 IE6 和 FF 中的效果还是不一样的

1）盒子(#nav)高度不一样

2）在 FF 中“酒杯”前面有个大黑圆点，而 IE6 中却没有！

解决上面这两个问题，也很容易，如下

1）做到这里标签 ul 和 li 有没有进行重置？只要我们在页面中新写一个标签，就要进行重置，做法是，将 ul、li 标签加入重置代码中“body,div,**ul,li**{padding:0; margin:0;}”

2）“酒杯”前面的大黑圆点，是 FF 给 li 标签定义的默认样式，我们只需要将 li 的默认样式去掉就是了，在 li 标签的 CSS 属性中加入“list-style:none;”就 OK 了

现在在瞅瞅，两种浏览器的显示效果是不是和下图一样了呢¯

CSS学习学前准备入门教程提高教程布局教程精彩应用

如果你做到这里的效果和我说的不一样，没关系，我把做到目前第二步的代码发出来，你对着上面说的再看看，绝对可以学会

【第三步】

第二步的效果还不是我们想要的，所有的“酒杯”都没有保持“车距”，后面的文字全部贴着前面的文字。

好¯！我们现在就将他们分开！设置<li>标签的宽度为 100 像素：

CSS 代码:

```
#nav ul li{
width:100px;
float:left;
list-style:none;
}
```

为了便于观察我们暂且将<li>标签的背景设置成红色(设置背景色，是页面布局中一个很重要的方法，便于查看块状元素区域范围)

CSS 代码:

```
#nav ul li{
width:100px;
float:left;
list-style:none;
background:#900;
}
```

效果如下：

CSS学习 学前准备 入门教程 提高教程 布局教程 精彩应用

瞧瞧，发现问题了吧，我们的<li>标签的高度并没有和我们的盒子的高度一样，这就是为什么在布局页面的时候，经常会设置一下背景色，就是这个道理，不然的话，你是发下不了

隐藏的问题，但是往往这些隐藏的问题就会导致页面浏览器的兼容性大大降低哟¯
现在暂不把<li>标签的背景色去掉，当我们把它调成我们需要的效果的时候再去掉！
继续，我们把 li 的高度设置成盒子的高度 35 像素，代码自己写，怎么样，高度一样了吧，但是文字却位于顶端，如何将它设置成居中呢，对喽¯设置行距**(如果你不会，建议你看看这篇文章《两种方法实现垂直居中》)**，在<li>的 CSS 代码中再加入下面这句代码：

```
line-height: 35px;
```

效果是不是和下图一样呢

好垂直居中解决了，轮到水平居中了，这个就容易了吧，直接在<li>的 CSS 代码中再加入下面这句代码：

```
text-align: center;
```

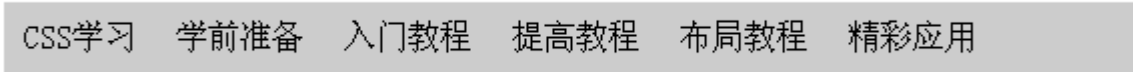
怎么样，效果有点意思了吧¯到这里我再发一次代码，保证大家每个步骤都学会！

好¯！做到这里，大家有没有想过一个问题，因为我们的<li>标签是设置了宽度为 100 像素，已经限定了它的宽度，如果文字多了它不会自动伸缩的自适应的，那这时候我们就需要去掉其宽度，这时候<li>的宽度就会缩小至文字的宽度，也就是说，如果我们再添加一些文字(把我们的酒杯换成一个大个的)，这个<li>也会跟着变大，大家去掉宽度后试试，是不是这个样子，这样我们的导航条就比较灵活了，不会对“酒杯”的大小有所顾忌了！

虽然这个宽度自适应解决了，但是给文字的空间太少，视觉上感觉不舒服，那么我们就帮它扩大一下空间，但是又要保证宽度自适应，解决方法很容易，加上左右内边距就 ok 了，这里设置边距为 10px，在<li>标签加上下面代码，顺便把背景颜色去掉

```
padding: 0 10px;
```

效果是不是这样



无论你的“杯子”是增大还是缩小，<li>不但宽度会随之增大缩小，但是杯子和杯子之间的距离永远不变！怎么样有点意思吧¯！

CSS学习　学前准备　入门教程下载　提高教程　布局基础教程　精彩应用

我们将导航条做成了下面的效果

CSS学习　学前准备　入门教程下载　提高教程　布局基础教程　精彩应用

但是此时的导航条还没有链接，还不能点击，这节课我们就要做一个完整的导航条

【第四步】

我们需要将上面的导航条做以下几个修改

1）给上面的导航加上链接;

2）链接文字大小修改为 12px;

3）并且规定链接样式，鼠标移上去和拿开的效果

修改方法如下

1）导航加链接，HTML 代码如下:

```
<div id="nav">
    <ul>
     <li><a href="#">CSS 学习</a></li>
        <li><a href="#">学前准备</a></li>
        <li><a href="#">入门教程下载</a></li>
        <li><a href="#">提高教程</a></li>
        <li><a href="#">布局基础教程</a></li>
        <li><a href="#">精彩应用</a></li>
    </ul>
</div>
```

2) 文字大小 12 像素，CSS 代码如下

```
a{font-size:12px;}
```

3）鼠标移动上面和拿开效果

```
#nav ul li a{color:#333; text-decoration:none;}
#nav ul li a:hover{color:#fff; text-decoration:underline;}
```

效果是不是和下面一样，鼠标移上去变成白色的有下划线的链接

到这里，基本上一个导航条就出来了¯不过为了能让大家再提高一个层次，KwooJan 就帮大家对上面的导航条进行一下修改，算是抛砖引玉！
我希望鼠标移上去后，链接的背景变成黑色的，下面是我的步骤
首先把链接 a 加上一个背景，以方便看出来链接 a 的区域

```
#nav ul li a{color:#333; text-decoration:none; background:#0FF;}
```

怎么样，知道 a 的区域了吧

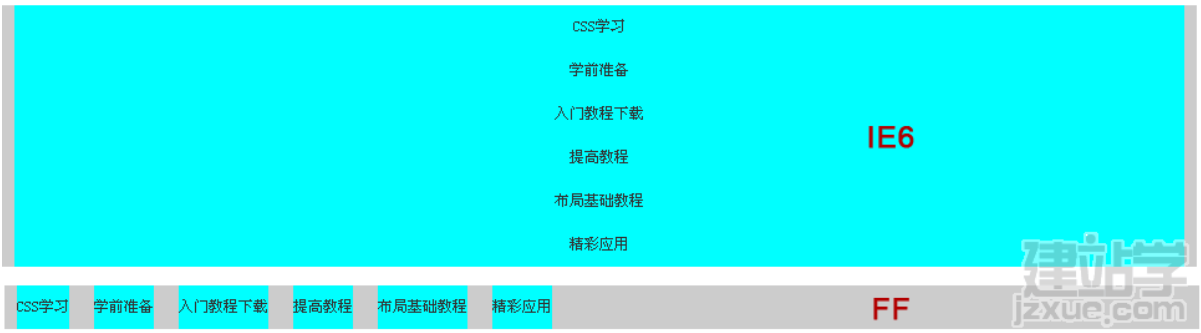
现在我要将 a 的高度设定为 35px 和盒子一样高度,这样我在把刚才的亮蓝色背景就可以完全覆盖下面盒子的灰色了
于是我插入下面红色的代码:

```
#nav ul li a{height:35px; color:#333; text-decoration:none; background:#0FF;}
```

可是不管我怎么刷新浏览器，高度都没有任何变化，这是为什么呢？！
原因就在于 a 属于内联元素，内联元素是无法设置宽度和高度的，width 和 height 只是针对块状元素，说道这里，解决办法就出来了，只要我们把内联元素 a 转化成块状元素就可以了，我们用“display:block;”将内联元素转化成块状元素。大家先不要加这段代码，闭上眼想想界面会变成什么样子？

```
#nav ul li a{display:block; height:35px; color:#333; text-decoration:none;
background:#0FF;}
```

实际效果:



IE6和FF显示效果居然大相径庭,IE6中为什么所有链接纵向排列了呢？其实这个也很简单,IE认为a既然转化成块状元素，就拥有块状元素的特性---霸道，它是不允许其他元素和它同一行，再加上也没有对a的宽度进行设定，所以才导致IE6中这么显示，不过FF中为什么不这样呢，和我们想象的一样，那是因为FireFox认为a即使为块状元素，也应该受到外面<li>元素的影响，所以如此现实，究竟以谁标准，因为大家都认为FF是标准浏览器，所以大家可以以FF为标准，不过KwooJan认为，不用管谁标准不标准，那都是相对的，我认为IE标准，FF就不标准了呢，我不愿意在这个问题上浪费精力，我更喜欢将精力用在思考如何提高页面的浏览器兼容性!

看到这里我想大家应该知道如何让页面在IE6中显示的和FF中一样，很简单，只需要在a的CSS代码中加入“float:left;”

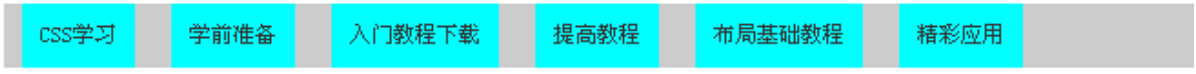
```
#nav ul li a{display:block; height:35px; color:#333; text-decoration:none; background:#0FF; float:left;}
```

问题迎刃而解，这还是用到前三节的课程内容，如果你想不起来如何解决，说明前面的课,特别是第二节的课，你没有真正理解！怎么做，你应该知道....回去再品品去

但是这样你不觉着，每个连接的左边和右边是不是太挤了，紧贴着a区域的左侧和右侧,应该怎么做？还是很简单，只需要再加上一句话“ padding:0 10px;”

```
#nav ul li a{display:block; height:35px; color:#333; text-decoration:none; background:#0FF; float:left; padding:0 10px;}
```

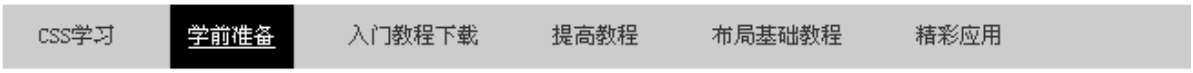
现在再瞅瞅，是不是下面的效果



这样看看是不是不挤了吧，哈哈，看着舒服了吧，但是这离我们的想要的效果只有一步了,因为现在看到的连接效果是，鼠标移上去和拿开背景都是蓝色的，我们现在只需要将，a链接中的背景去掉，移到a:hover的CSS代码中，并且颜色变成“#000”就ok了¯

```
#nav ul li a{display:block; height:35px; color:#333; text-decoration:none;
float:left; padding:0 10px;}
#nav ul li a:hover{color:#fff; text-decoration:underline; background:#000;}
```

怎么样，和下面的效果一样么？



效果好多了吧，这下是我们想要的效果了吧¯
当然！大家还可以把背景不设置成黑色，用个图片也可以！现在大家明白，为什么一开始我说这款导航栏可以演变出成千上万的不同特色的导航栏了吧¯万变不离其宗！

第四课的思路就是这样的，如果吃透了这节课，那么以后什么样子的导航都很轻易作出来，如果你在和 js 很好的结合起来用¯你就可以很自信的向老板提出加薪了！！！^_^

下节节课我将给大家用浮动方法布局一个页面，敬请期待！

【DIV+CSS 入门教程】浮动(float)页面布局

前四节的大练习大家做的怎么样？有没有难度，如果你觉着有难度没有关系，这节课，我带着大家做一下这个练习！

【第一步 整体布局与公共 CSS 定义】

我们先来分析一下这个页面

MyBlog
bbs.cssxuexi.cn

HOME PHOTOS ABOUT LINKS CONTACT

CSS学习互动社区欢迎您！

我们是一群热爱页面前端技术并热衷于推广W3C标准的热心好友，如果您想学或者正在学DIV+CSS布局页面，加入我们！您会很快驯服并驾驭这匹烈马！

虽然我们的论坛正在起步，但是这里的每个人都很热爱页面前端技术并热衷于推广W3C标准在中国的运用，只要你有问题就可以问，一定会有人帮你解答！我们可能不是高手，但是我们都是很乐于帮助，乐于钻研。我们都很热心！

跟KwooJan学习DIV+CSS只需2天

群上很多朋友在刚接触DIV+CSS的时候，很迷茫，不知道从何学起，看网上的教程吧，理论性的东西太多，越看越糊涂，再说时间上也不允许，也没有那个耐心，其实KwooJan也不喜欢看这种视频教程，很枯燥，很乏味，即使耐着性子看完收获也不大，实用性不大，群上的一些朋友告诉我，他们学习DIV+CSS没有思路，不知道怎么去学，如何去学，希望KwooJan能带着他们一步一步走，从今天开始我将写个教程，打算以例子为主，帮主大家更轻松的驾驭DIV+CSS。好了，在这里我必须要给大家纠正一个错误，我们平时说的DIV+CSS其实是一种错误的说法，是中国人自己发明的，并不准确，不能够将所谓的页面布局思想说的很确切，其实应该说XHTML+CSS才对。

加入我们！

CSS学习互动社区QQ群：
1群：5505810　2群：87951377
3群：73513641　4群：72263578
希望有强烈进取精神和互助精神的朋友请加入！一块探讨一块交流一块学习！

页面主要分 5 大块，顶部的 Logo、导航条 Nav、Banner、Content、Footer，如下图

这样 HTML 就很容易写出来了

```
<div id="Logo"></div>
<div id="Nav"></div>
<div id="Banner"></div>
<div id="Content"></div>
<div id="Footer"></div>
```

因为这 5 块的宽度都是 900 像素，并且都是水平居中的，所以相应 CSS 代码如下

```
body, div, a, img, p, form, h1, h2, h3, h4, h5, h6, input, textarea, ul, li, dt, dd, dl {margin: 0; padding: 0;}
/*为什么写这段代码没有忘记吧，作用就是重置可能用到的标签，不明白的去看第四节的课程关键词*/
#logo, #Nav, #Banner, #Content, #Footer {width: 900px;  margin: 0 auto;}
```

【第二步 布局 Logo 栏】

首先我们需要把页面上的 logo 给切割出来，其大小为 173*46,名字为：logo.gif

一般网站都会做到点击 logo，就会回到主页，应该怎么做呢，大家首先会想到，给图片加上链接就可以了，代码一般会这么写

```
<a href="#" id="logoLink"><img src="#" /></a>
```

不过 KwooJan 要介绍另外一种方法,将图片做成链接 a 的背景,同样可以达到上面说的效果,并且 HTML 代码就会更精简，少了<img...>，看看下面 Logo 栏的页面代码，红色的为将 logo.gif 作为背景的链接

HTML 代码:

```
<div id="Logo">
<a href="#" id="logoLink"></a>
</div>
```

CSS 代码

```
#Logo{
height:80px;/*公共代码中没有定义高度，在这里定义*/
}
#logoLink{
display:block;/*将链接 a 转化成块状元素，这样才能显示出背景*/
width:173px;
height:46px;
background:url(../Images/logo.gif) no-repeat;
float:left;/*为了让 ie6 和 ff 显示效果一样,如果不加上这句话,后面的 margin-top:20px;两个浏览器解析不一样，大家可以去掉这句话，看看两者显示效果差别*/
margin-top:20px;/*设置 a 的顶部外边距为 20 像素，这样才能和浏览器顶部有段距离，才能和图片中做的一样*/
}
```

好到这里，头部含有 logo 的区域已经写完。

【第三步 布局导航栏 Nav】

页面上的导航栏和第四节讲的几乎是一样的，并且更简单些，这里我就不再给大家一步一步做，不会做的就去看第四节，这里我就直接把代码发出来供大家学习

HTML 代码:

```
<div id="Nav">
<ul>
```

```
      <li><a href="#">HOME</a></li>
          <li><a href="#">PHOTOS</a></li>
          <li><a href="#">ABOUT</a></li>
          <li><a href="#">LINKS</a></li>
          <li><a href="#">CONTACT</a></li>
     </ul>
</div>
```

CSS 代码

```
#Nav{height:42px;}
#Nav ul{
height:42px;
list-style:none;
background:#56990c;
}
#Nav ul li{height:42px; float:left;}
#Nav ul li a{
display:block;/*转化成块状元素，因链接是内链元素，若想给它定义下面的属性，必须将它转化成块状元素，*/
height:42px;
color:#FFF;
padding:0 10px;
line-height:42px;
font-size:14px;
font-weight:bold;
font-family:Arial;
text-decoration:none;/*去除链接样式，默认是有下划线的，加上这句就没有任何样式，下划线也没有了*/
float:left;/*这句一定要加，不然在 IE6 中会出现，这种效果*/
}
#Nav ul li a:hover{background:#68acd3;}
```

【第四步 Banner 布局】

这个就更简单了，有两种方法

第一种：将图片作为 背景

第二种：直接将图片插入 之间，代码：

大家可以根据需求和实际情况选择用哪一种，在这里我们用第一种

HTML 代码没有什么变化，只需要在 CSS 里面定义一下就 OK 了

CSS 代码:

```
#Banner{
height:290px;
```

```
background: url(../Images/banner.jpg) no-repeat;
}
```

怎么样做到这里比较简单吧，好，接着来

【第五步 内容 Content 板块布局】

从图片上我们看到，内容板块分为左右两个区域，左边 ContentL 宽度是 600px，右边 ContentR 宽度是 300px，但是由于我们要将内边距设置成 15px（这样才会好看），所以 ContentL 的宽度在 CSS 中就要设置成 600-15*2=570px，而右侧的 ContentR 则需要设置成 300-15*2=270px；

HTML 代码：

```
<div id="Content">
    <div id="ContentL">此处为左边 ContentL</div>
    <div id="ContentR">此处为左边 ContentR</div>
</div>
```

CSS 代码：

```
#Content #ContentL, #Content #ContentR {float: left; padding: 15px;} /*为什么都要左侧浮动，如果不明白就去看第二节*/
#Content #ContentL {width: 570px; background: #f0f0f0;}
#Content #ContentR {width: 270px; background: #d3e7f2;}
```

页面效果：

内容部分我们就先做到这里，最后我们再布局里面的具体元素，下面接着来布局版权模块（Footer）

【第六步 Footer 布局】

这部分结构比较简单，大家只需要知道怎么布局就 OK 了

HTML 代码:

```
<div id="Footer">
    <p>版权归 CSS 学习</p>
    <p>CSS 交流 QQ 群: 5505810/87951377/73513641/72263578</p>
</div>
```

CSS 代码:

```
#Footer{
text-align: center;
background: #68acd3;
padding: 10px 0;
font-size: 12px;
font-family: Arial, Helvetica, sans-serif;
color: #fff;
line-height: 20px;
}
```

目前效果如下:

就这样我们页面的整体结构基本出来了，剩下的工作就是内容板块内部元素的具体布局了，我将在下节课接着讲~

我们接着上节课，继续学习，我把页面整体效果发出来，方便大家学习

【第七步 内容左侧板块(ContentL)布局】

我们分析一下他的结构，主要包括标题和文章内容两块，并且标题和内容之间有一条虚线，而第二篇文章的内容部分是图片和文字相结合且文字环绕图片。

好¯！既然搞清楚结构了，后面我们布局就容易了

我打算标题用<h1>标签，为什么这么用呢，原因如下

第一:<h1>标签本身字体就是加粗的这样 CSS 里面就不用再定义字体粗细

第二:如果标题用<h1>的话，搜索引擎会首先抓取<h1>里面的内容，然后提取关键词，这样别人在搜索引擎中输入关键词，会更容易找到你的网站哟¯然后流量就唰唰滴¯^_^

对于文章内容，我们就放到 中就 OK 了，相应的代码如下:

HTML 代码:

```
<div id="ContentL">
        <h1>CSS 学习互动社区欢迎您！</h1>
        <p>我们是一群热爱页面前端技术并热衷于推广 W3C 标准的热心好友，如果您想学或者正在学 DIV+CSS 布局页面，加入我们！您会很快驯服并驾驭这匹烈马！虽然我们的论
```

```
坛正在起步，但是这里的每个人都很热爱页面前端技术并热衷于推广 W3C 标准在中国的运用，只要你有问题就可以问，一定会有人帮你解答！我们可能不是高手，但是我们都是很乐于帮助，乐于钻研。我们都很热心！</p>
</div>
```

CSS 代码:

```
#Content #ContentL h1{
height:40px;
line-height:40px;/*设置行距，目的是保证 h1 中的文字垂直居中*/
font-size:16px;
color:#054d73;
border-bottom:1px #969696 dashed;/*设置 h1 的下边框为宽度 1 像素的虚线*/
margin-bottom:10px;/*设置外边距，让 h1 和下面的内容区域 p 保持 10 像素的距离*/
}
#Content #ContentL p{
font-size:12px;
line-height:20px;
text-indent:2em;/*这句话的目的就是为了让文章第一行缩进两个汉字，记住这句话就 OK 了*/
}
```

效果如下:

CSS学习互动社区欢迎您!

我们是一群热爱页面前端技术并热衷于推广W3C标准的热心好友，如果您想学或者正在学DIV+CSS布局页面，加入我们！您会很快驯服并驾驭这匹烈马！虽然我们的论坛正在起步，但是这里的每个人都很热爱页面前端技术并热衷于推广W3C标准在中国的运用，只要你有问题就可以问，一定会有人帮你解答！我们可能不是高手，但是我们都是很乐于帮助，乐于钻研。我们都很热心！

这里我们第一篇文章已经布局完毕，下面布局一下第二篇文章，估计大家早就注意到了，两篇文章唯一区别就是第二篇文章的内容的左侧有一张图片，哈哈，这就好办了，把第一篇文章的代码复制过来，替换标题和文章内容，然后再文章内容里面插入一张图片就 OK 了，代码如下:

HTML 代码:

```
<h1>跟 KwooJan 学习 DIV+CSS 只需 2 天</h1>
<p>
        <img src="Images/1.gif" />群上很多朋友在刚接触 DIV+CSS 的时候，很迷茫，不知道从何学起，看网上的教程吧，理论性的东西太多，越看越糊涂，再说时间上也不允许，
```

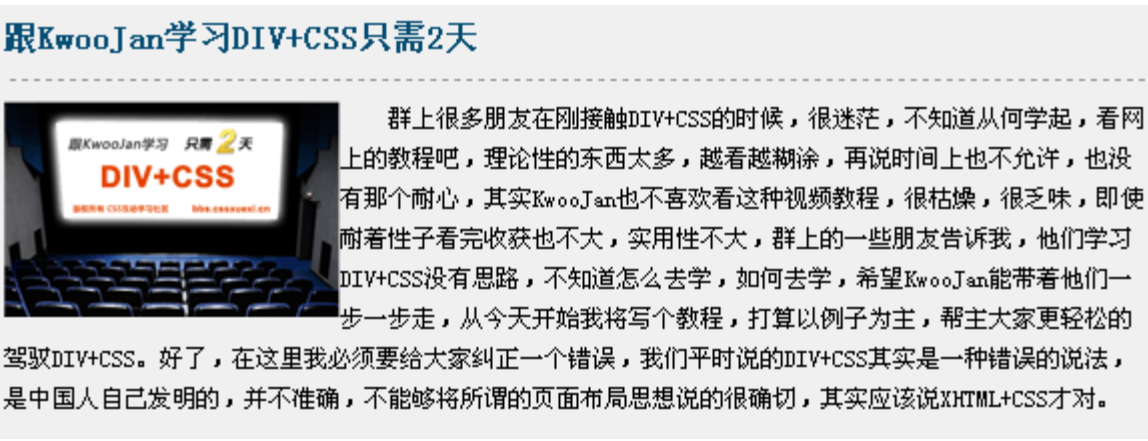
也没有那个耐心，其实 KwooJan 也不喜欢看这种视频教程，很枯燥，很乏味，即使耐着性子看完收获也不大，实用性不大，群上的一些朋友告诉我，他们学习 DIV+CSS 没有思路，不知道怎么去学，如何去学，希望 KwooJan 能带着他们一步一步走，从今天开始我将写个教程，打算以例子为主，帮主大家更轻松的驾驭 DIV+CSS。好了，在这里我必须要给大家纠正一个错误，我们平时说的 DIV+CSS 其实是一种错误的说法，是中国人自己发明的，并不准确，不能够将所谓的页面布局思想说的很确切，其实应该说 XHTML+CSS 才对。
</p>

但是如果我们预览效果的话，确是这样子的

不但图片没有靠左边，而且文字的上方还有一大片空白，应该怎么做呢？很容易，只要我们给图片左侧浮动(float:left;)就可以了,CSS 代码如下:

```
#Content #ContentL p img{
float:left;
}
```

效果如下，很接近了吧，只不过图片的左侧和文字靠的太接近了



这个很好解决，设置图片的右外边距(margin-right)嘛¯,CSS 代码如下:

```
#Content #ContentL p img{
float:left;
margin-right:10px;
}
```

这下效果一样了吧¯！ ¯！ ¯！

OK！到这里 ContentL 板块布局搞定！

【第八步 内容右侧板块(ContentR)布局】

有了 ContentL 板块布局的经验，右侧就会很容易，标题“加入我们！”当然还是用<h1>标签喽,好¯！开工！

标题区域代码如下

HTML 代码:

```
<h1>加入我们！</h1>
```

CSS 代码:

```
#Content #ContentR h1{
height:40px;
line-height:40px;/*设置行距，目的是保证 h1 中的文字垂直居中*/
font-size:16px;
color:#900;
border-bottom:1px #969696 dashed;/*设置 h1 的下边框为宽度 1 像素的虚线*/
margin-bottom:10px;/*设置外边距，让 h1 和下面的内容区域 p 保持 10 像素的距离*/
}
```

而内容的第一句“CSS 学习互动社区 QQ 群：”的代码如下

HTML 代码:

```
<strong>CSS 学习互动社区 QQ 群: </strong>
```

CSS 代码:

```
#Content #ContentR strong{
display:block;/*只有把 strong 标签，转化成块状元素，margin-bottom:5px;才会起作用，才能使<strong>与下面的元素维持一定距离*/
font-size:12px;
color:#333;
margin-bottom:5px;
}
```

好¯！第一行搞定！
下面的两行红色的 QQ 群信息怎么做？其实这个有很多办法

```
方法一：ul、li 或者 dl、dt、dd 来布局
方法二：表格(Table)来布局
方法三：用单纯的标签来布局比如<p>、<span>、<div>等标签
```

其实在这里，我最推荐第二种方法，可能大家看到这里挺想不透的，或者觉着用 Table 很丢人，好似没有什么技术含量似的，其实这时候如果你不用 Table，反而认为你的技术含量低，自己给自己找麻烦，为什么这么说呢
首先你必须知道 DIV 和 Table 的诞生目的不一样，**DIV 的诞生就是用来布局页面，而 Table 的诞生就是为了放数据**，大家看 KwooJan 所有写的代码，只有布局页面大板块的时候才用，还记得上节课一开始布局页面板块的时候代码么

```
<div id="Logo"></div>
<div id="Nav"></div>
<div id="Banner"></div>
<div id="Content"></div>
<div id="Footer"></div>
```

整个页面就这 5 个 DIV，其他地方一般不用，因为 DIV 的使命就是布局页面！
大家经常会进入一个误区，会认为在 Web2.0 时代，只要页面中用了 Table 就是没有技术含量，就是丢人，要是页面中没有一个 table，所有元素全部用 DIV 做，那就是牛人！大家注意了，要是某人对你说，我的整个网站没有应用一个 Table，这时候你就可以认为这个人做页面没有什么技术含量，并且 CSS 代码相当庞杂，根本不能算是一个高手，顶多是一个 DIV 的狂热分子，做的页面也能说是标准，很多用 Table 就可以简简单单实现的效果，非要用 DIV 去实现，不仅使 CSS 文件相当的臃肿，而且使页面加载速度变慢。
所以在这里 KwooJan 提醒大家，一定一定要走出这个误区！
好了说了这么多，这块的代码如下:

HTML 代码:

```
<table width="100%" border="0" cellpadding="0" cellspacing="0">
        <tr>
```

```
            <td width="36%" height="20">1群: 5505810</td>
            <td width="64%">2群: 87951377</td>
        </tr>
        <tr>
            <td height="20">3群: 73513641</td>
            <td>4群: 72263578</td>
        </tr>
</table>
```

CSS 代码:

```
#Content #ContentR table{
font-size:12px;
color:#900;
}
```

最后一句话就更简单了，代码如下

HTML 代码:

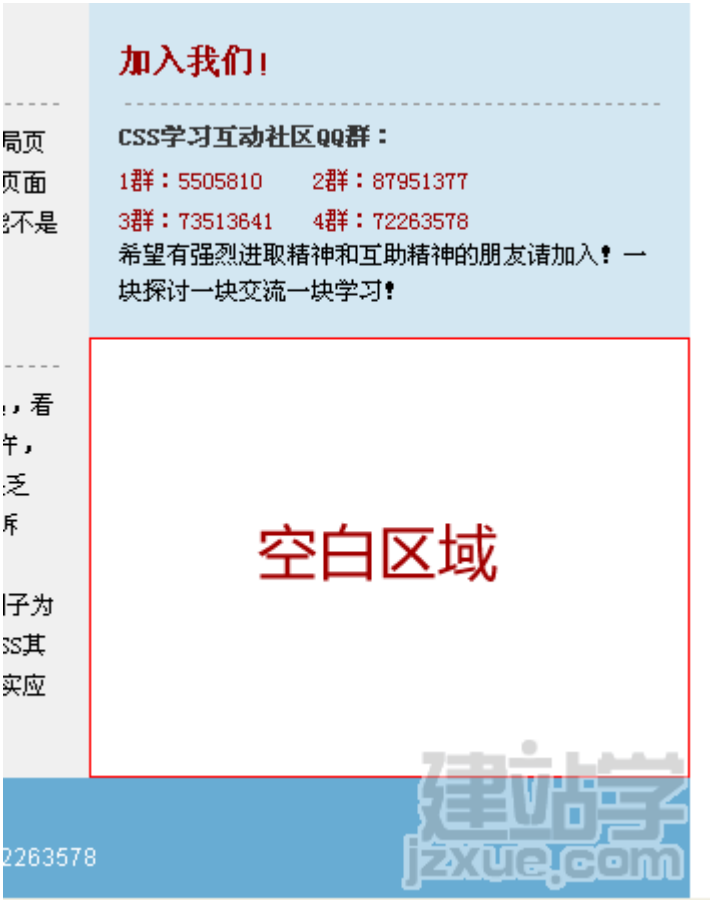
```
<span>希望有强烈进取精神和互助精神的朋友请加入!一块探讨一块交流一块学习!</span>
```

CSS 代码:

```
#Content #ContentR span{
font-size:12px;
}
```

至此我们每个版块均以布局完毕，但是却有两点瑕疵:
1）IE6 和 FF 中有一点却显示的却不一样，底部版权在 FF 中却跑到了的右侧 ContentR 的下面，如图:

产生原因：是因为 id 为 Content 的 div，没有自动适应里面 ContentL 的高度
解决方法：最简便的方法是设置 Content 的 CSS 属性 overflow:hidden;
怎么样问题解决了吧¯
2）因为 ContentR 的高度没有 ContentL 的高度高，所以在 ContentR 的下面留有一块空白，如图：



解决方法：只需要把 Content 的背景颜色设置成和 ContentR 背景颜色一样就 OK 了
这个问题也解决了吧¯
最终效果

至此，整个页面算是布局完成了，感觉怎么样？有不懂的就回复帖子告诉我

顺便说一下：最终代码其实还可以进行精简，这个算是给大家一个思考题了
下节课，我们将要讲讲布局网页的第二种方法---定位

CSS 实例: 定位(position)页面定位详解

注：在做这节教程的时候，我又上网查了相关资料，看了大量的文章，做了大量的测试，最后总结出下面这些文字，洋洋洒洒一整篇，不过需要大家一句话一句话的看，一定要仔细喽！还有对于课程中的说的，最好一边看，一边练，不练绝对看不懂！

定位(position)布局页面说容易非常容易，只需要记住这节课最后一句话就可以了，说困难，那是相当的难理解，需要一定的耐心，不过还好，KwooJan 给大家总结的已经很通俗易懂了。

如果下面的文字实在是无法理解透，那就记住这节课最后总结的一句话“**如果用 position 来布局页面，父级元素的 position 属性必须为 relative，而定位于父级内部某个位置的元素，最好用 absolute，因为它不受父级元素的 padding 的属性影响，当然你也可以用 position，不过到时候计算的时候不要忘记 padding 的值。**”

好，上课！
任何元素的默认 position 的属性值均是 static,静态。这节课主要讲讲 relative（相对）以及 absolute（绝对）。

【position:absolute】
意思是：他的意思是**绝对定位**，他默认参照浏览器的左上角，配合 TOP、RIGHT、BOTTOM、LEFT(下面简称 TRBL)进行定位，有以下属性：
1)如果没有 TRBL，以父级的左上角，在没有父级的时候，他是参照浏览器左上角,如果在没有父级元素的情况下，存在文本，则以它前面
的最后一个文字的右上角为原点进行定位但是不断开文字，覆盖于上方。
2)如果设定 TRBL，并且父级没有设定 position 属性，那么当前的 absolute 则以浏览器左上角为原始点进行定位，位置将由 TRBL 决定。
3)如果设定 TRBL，并且父级设定 position 属性(无论是 absolute 还是 relative)，则以父级的左上角为原点进行定位，位置由 TRBL 决定。即使父级有 Padding 属性，对其也不起作用，说简单点就是：它只坚持一点，就以父级左上角为原点进行定位，父级的 padding 对其根本没有影响。
以上三点可以总结出，若想把一个定位属性为 absolute 的元素定位于其父级元素内，只有满足两个条件，
第一：设定 TRBL
第二：父级设定 Position 属性
上面的这个总结非常重要，可以保证你在用 absolue 布局页面的时候，不会错位，并且随着浏览器的大小或者显示器分辨率的大小，而不发生改变。

只要有一点不满足，元素就会以浏览器左上角为原点，这就是初学者容易犯错的一点，已经定位好的板块，当浏览器的大小改变，父级元素会随之改变，但是设定 Position 属性为 absolute 的板块和父级元素的位置发生改变，错位了，这就是因为此时元素以浏览器的右上角为原点的原因。

初学者很容易犯错的是,不清楚 Position 属性为 absolute 的板块,若想定位到父级板块中,并且当浏览器的大小改变或显示器的分辨率改变,布局不发生改变,是需要满足两个条件的,只要有一点不满足，元素就会以浏览器左上角为原点，从而导致页面布局错位。

【position:relative】
意思是**相对定位**，他是默认参照父级的原始点为原始点，无父级则以文本流的顺序在上一个元素的底部为原始点，配合 TRBL 进行定位，当父级内有 padding 等 CSS 属性时，当前级的原始点则参照父级内容区的原始点进行定位，有以下属性：
1)如果没有 TRBL，以父级的左上角，在没有父级的时候，他是参照浏览器左上角(到这里和 absolute 第一条一样)，如果在没有父级元素的情况下，存在文本，则以文本的底部为原始点进行定位并将文字断开(和 absolut 不同)。

2)如果设定 TRBL，并且父级没有设定 position 属性，仍旧以父级的左上角为原点进行定位(和 absolut 不同)
3)如果设定 TRBL，并且父级设定 position 属性(无论是 absolute 还是 relative)，则以父级的左上角为原点进行定位，位置由 TRBL 决定(前半段和 absolut 一样)。如果父级有 Padding 属性，那么就以内容区域的左上角为原点，进行定位(后半段和 absolut 不同)。
以上三点可以总结出，无论父级存在不存在，无论有没有 TRBL，均是以父级的左上角进行定位，但是父级的 Padding 属性会对其影响。
综合上面对 relative 的叙述，我们就可以将 position 属性为 relative 的 DIV 视成可以用 TRBL 进行定位的的普通 DIV，或者说只要将我们平时
布局页面的 div 的 CSS 属性中加上 position:relative 后，就不只是用 float 布局页面了，还可以用 TRBL 进行布局页面了，或者说加上
position:relative 的 DIV 也可以像普通的 DIV 进行布局页面了，只不过还可以用 TRBL 进行布局页面。但是 position 属性为 absolute 不可以
用来布局页面，因为如果用来布局的话，所有的 DIV 都相对于浏览器的左上角定位了，所以只能用于将某个元素定位于属性为 absolute 的
元素的内部某个位置，这样我们就可以总结比较重要的结论

属性为 relative 的元素可以用来布局页面，属性为 absolute 的元素用来定位某元素在父级中的位置

既然属性为 absolute 的元素用来定位某元素在父级中位置，就少不了 TRBL，这时候根据一开始讲的 absolute 的第三条，如果父级元素没有
position 属性那么 absolute 元素就会脱离父级元素，但是如果是布局页面，父级元素
position 的属性又不能为 absolute，不然就会以浏览
器左上角为原点了，所以父级元素的 position 属性只能为 relative!
==========

总结：如果用 position 来布局页面，父级元素的 position 属性必须为 relative，而定位于父级内部某个位置的元素，最好用 absolute，因为它不受父级元素的 padding 的属性影响，当然你也可以用 position，不过到时候计算的时候不要忘记 padding 的值。

CSS 布局教程:用 position 来页面布局

“如果用 position 来布局页面，父级元素的 position 属性必须为 relative，而定位于父级内部某个位置的元素，最好用 absolute，因为它不受父级元素的 padding 的属性影响，当然你也可以用 position，不过到时候计算的时候不要忘记 padding 的值。”
如果你还是不能太明白这句话，我们就做个实例，把我们第 5 节页面的头部 blog 区域用定位(position)来布局一下

HTML 代码和原来的没有区别:

```
<div id="Logo">
    <a href="#" id="logoLink"></a>
</div>
```

CSS 代码就有变化了，咱先看看原来的 CSS 代码

```
#Logo{
    height:80px;
}
#logoLink{
    display:block;
    width:173px;
    height:46px;
    background:url(../Images/logo.gif) no-repeat;
    float:left;
    margin-top:20px;
}
```

原来是用“浮动+外边距”布局的，将#logoLink 定位在距离顶部 20 像素，左侧 0 像素的地方;

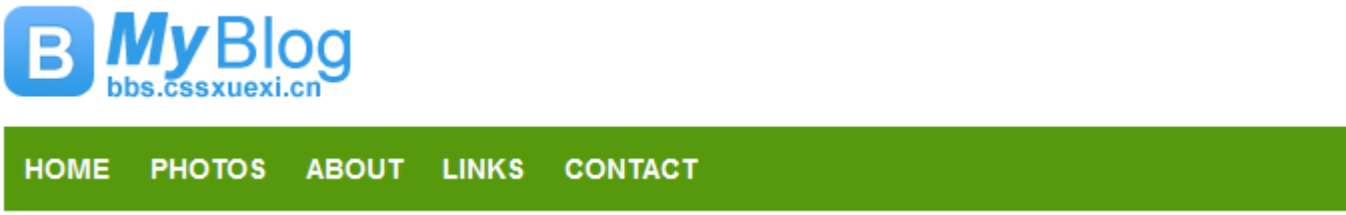
如果用 position 的方法，CSS 就应该这么写，首先给#logo 加上“position:relative;”，给#logoLink 加上“position:absolute;”,然后让#logoLink 距离上面 20 像素，左侧 0 像素，具体代码如下:

```
#Logo{
height:80px;
position:relative;/*相对定位*/
}
#logoLink{
display:block;
width:173px;
height:46px;
background:url(../Images/logo.gif) no-repeat;
position:absolute;
top:20px;
left:0;
}
```

怎么样效果和原来的一样吧，就是这么简单.

什么？CSS 代码多了？的确，由原来的两句话，变成现在的四句，但是有没有发现，就靠 Position 我们就可以将 Logo 随意的定位于任何一个地方！这就是它的灵活的地方！如果在

页面顶部在有一行“加入收藏 | 设为首页 | RSS 订阅”如下图，你是不是也可以很方便的将它们定位于右上角呢¯



但是定位(position)有一个缺点，不会自适应内部元素的高度，所以平时我们在布局页面的时候，如果某个或者某些模块高度永远不变，就可以用定位，但是 KwooJan 建议大家布局页面的时候，还是要以 Float 为主，Position 为辅！这样你才能做出高质量的页面。

==========

最后，请大家记住我这句话：“布局页面，Float 为主，Position 为辅！”

==========

好，我们有关页面布局的教程，算是已经完结。

如果能悟透每节课内容，你现在就可以拍着胸脯说“**我是 DIV+CSS 高手！**”哈哈，不过你要知道，Div+CSS 的叫法是不标准的，应该叫... ...

什么你想不起来了？！

咣当！—

【DIV+CSS 入门教程】CSS Hack

写完第七课，因为时间问题没有继续写第八课有关浏览器兼容方面的文章，以为大家可以自己掌握这方面的知识，不过发现有很多同学找了很多资料，很多文章去研究，费时费力的，效果也不好，今天是星期六，我呢就再给大家补写这篇教程，带领大家用最短的时间掌握 CSS Hack！

说到浏览器兼容性问题，就必须说 CSS Hack！提到 Hack 大家肯定会想到电脑黑客(hacker)、和病毒程序联系到一块，不过在 CSS 中，Hack 是指一种兼容 CSS 在不同浏览器中正确显示的技巧方法。说的更直白一些就是，你平时做个页面，布局正确，CSS 正确，可就是在不同的浏览器中显示的效果不一样，要么错位，要么多几个像素，怎么都找不到原因，这时候我们就会用一些技巧方法来让不同的浏览器显示一样的效果，这种方法我们就称之为 CSS Hack，**记住喽，CSS Hack 是解决页面浏览器不兼容的技巧方法，是一种方法哟，不要理解偏差**。

不过这里需要说明一点，**CSS Hack 都属于个人对 CSS 代码的非官方修改，所以编写的 CSS 代码不会通过官方 W3C 的认证**，这个要知道！以后经常会遇到这种情况，CSS 写的正确，通过 W3C 验证，但是不同浏览器显示效果不一样，用了 CSS Hack，显示的效果一样了，却又通不过 W3C 验证了，很是郁闷，不过不要钻牛角尖就是了，W3C 验证只不过是帮你检查一下 CSS 代码写的有没有语法错误而已，通过验证只不过是说明你到目前写的 CSS 代码没有语法

错误而已，不要太计较是否通过验证，也不要多想，如果通不过 W3C 验证，其他人会不会笑话我，这些想法都是没有必要的，这说明我们的技能更强，因为我们用到了 CSS Hack，再说了你的页面是给网民看的，网民看的是界面好看不好看，内容好不好，有没有找到他要找的东西，他不理会你的页面有没有通过 W3C 验证，所以 KwooJan 在这里提醒大家，不要落入这个误区哟~

好，我们开讲！
这节课我主要讲两个最常用的 CSS Hack，如果这两个能明白，再学其他的 Hack 就容易了
（1）!important　（2）*

!important

```
【例子】
#content{
      height:960px !important;
      height:900px;
}
```

它所附加的生命拥有**最高**优先级，但是由于 IE6 不能识别它,而对于 IE7，FF 均能识别，所以我们就可以用来来解决一些页面,在 IE6 上显示的效果与 IE7、FF 上的效果不一样的情况。

在上面的例子中
IE7 和 FF 遇到附加有!important 的 CSS 属性，就会只解析第一句
“height:960px !important;”将高度定为 960px，而后面的“height:900px;”将不解析，忽略它；
因为 IE6 不认识附加有!important 的语句，所以会跳过第一句，忽略此句，直接解析第二句“height:900px;”将高度定为 900px；
注意：附加有“!important”的语句一定要在没有附加“!important”的语句的上面，顺序一定不能错！

*

```
#content{
      height:960px;
      *height:900px;
}
```

由于 IE6、IE7 可以识别附加有*的 CSS 属性语句，FF 则不能识别，所以我们就可以用来来解决一些页面，在 IE 上显示的效果与 FF 上的效果不一样的情况。
在上面的例子中
因为 FF 不识别*，所以它只读第一句“height:960px;”而忽略第二句，又因 IE6、IE7 识别*，所以它们先读第一句，将高度定为 960px，而后又读第二句“*height:900px;”，将高

度从 960px 修改为 900px，所以我们在 IE 中看到的最终效果就是高度为 900px；
注意：附加有“*”的语句一定要在没有附加“*”的语句的下面，顺序一定不能错！你想想就知道！

好了，这节课很容易，相信你肯定很容易就理解了，后面的就是自己实践喽~加油各位！
==========
为了帮助大家更好的提高，到这个帖子里***提建议，拿金币！*** ，你希望在后面的教程里加上哪方面的内容，你对现在教程有什么建议，说出来，我会给大家写一本电子书，人手一本，更好的帮助大家提高！快来吧，别等了~~~就等你的建议了~

【DIV+CSS 入门教程】单图片按钮实例

一般我们做按钮基本上都需要两张图片，一张正常状态的图片，一张按下去效果图片
做这种按钮思路就是，设置链接 a 的背景为第一张图片，a:hover 的背景为第二章图片

代码如下：

HTML 代码：

```
<a id="theLink"></a>
```

CSS 代码：

```
#theLink{
	display:block;/*因为标签 a 是内链元素，所以利用这句话将内链元素转化成块状元素，后面的 width 和 height 才起作用*/
	width:120px;
	height:41px;
	margin:0 auto;
	background:url(../images/normal.gif) no-repeat;
}
#theLink:hover{background:url(../images/press.gif) no-repeat;}
```

源代码：两张图片按钮的源代码.rar (5.2 KB)

==========

这节课，主要给大家介绍第二种思路，其实也很简单，首先我们将上面的两个图片合并成一张图片，如下

其次，将上面的图片设置成按钮的背景
最后，将 a:hover 的背景向上移动 41 个像素就 OK 了

HTML 代码：

```
<a id="buttonBlock"></a>
```

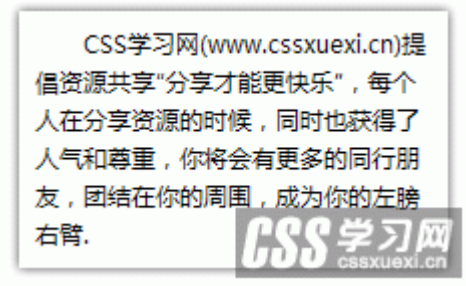
CSS 代码：

```
#theLink{
      display:block;
      width:120px;
      height:41px;
      margin:0 auto;
      background:url(../images/buttonBG.gif) no-repeat;
}
#theLink:hover{ background:url(../images/buttonBG.gif) no-repeat 0 -41px;}
```

可能我讲到这里，你不能完全理解，没关系
下载下来源代码，保你一看就明白源代码：单张图片按钮的源代码.rar (4.48 KB)

【DIV+CSS 入门教程】CSS 首行文字缩进

我记得过去刚开始学习制作页面的时候，要想让一段文字首行缩进两个文字，如下图



总是在前面加上 8 个“ ”，因为过去大家对 CSS 的概念也不太熟悉，也不太重视，这种方法实现虽然比较直接，但是文字多的时候会有很多“ ”充斥在代码中，如果再有些换行
，那代码显着比较乱，现在大家开始主动了解 CSS，学习 CSS，你会发现这个问题很容易解决，只需要在相应的位置加入代码

```
1. text-indent:2em;
```

就很容易实现喽~

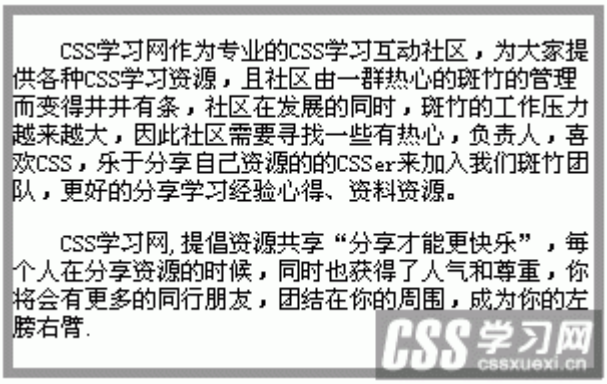
text-indent:2em;解释：text 的意思是文本、indent 在计算机英语中意思是缩进、至于后面的 2em 意思就是 2 个相对单位；

em 解释：相对于当前对象内文本的字体尺寸。(引自 CSS2.0 手册)各位同学，要仔细品一下这个概念！理解喽没有，em 这个单位的意思就是相对文字大小，1em 就是 1 个文字大小，2em 就是两个文字大小，到这里看明白了吧，理解“text-indent:2em;”的意思了吧，就是“文本缩进两个文字大小”，这不就是我们要的效果么？

将上面的代码给大家写出来，CSS 代码是用的行内方式

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2. <html xmlns="http://www.w3.org/1999/xhtml">
3. <head>
4. <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
5. <title>CSS 学习网 - 首行文字缩进</title>
6. </head>
7. <body>
8. <div style="width:300px; text-indent:2em;">CSS 学习网,提倡资源共享"分享才
   能更快乐"，每个人在分享资源的时候，同时也获得了人气和尊重，你将会有更多的同行朋友,
   团结在你的周围，成为你的左膀右臂.
9. </div>
10. </body>
11. </html>
```

【本节作业】



要求：

1. 制作一个宽度为 300px，高度自适应，边框宽度为 5 像素且颜色为灰色(#999)的区域，id 为 div1；
2. 在 div1 内部有两段文字，文字大小为 12 像素，这两段文字首行分别缩进两个文字；
3. CSS 用内嵌方式；
4. 暂不考虑页面对各种浏览器的兼容性，只在一种浏览器中实现上面效果就可以

由 知博网 zhiblog.com 提供