第 14 章　CSS 基本布局

CSS+DIV 是 Web 标准中一种新的布局方式，它正逐渐代替传统的表格（table）布局。CSS+DIV 模式具有比表格更大的优势，结构与表现相分离，代码简洁，利于搜索，方便后期维护和修改。本章主要讲解 CSS+DIV 的基本布局种类，同时还在每种类型的布局后面，列举一个实例形象说明布局样式，以帮助读者快速掌握 CSS+DIV 布局的应用方法。

14.1　一列固定宽度

一列式布局是所有布局的基础，也是最简单的布局形式。本节将要讲到的一列式布局是一种固定宽度的布局样式，XHTML 代码相当简单，只需要编写一段 div 即可。

1．实现原理

div 默认状态下占据整行显示，当为其设置宽度的时候，div 的宽度就会变成所设置的宽度，以实现固定宽度的布局。

2．制作实例

制作实例的步骤很简单，也同样是分为制作 XHTML 代码和添加 CSS 样式表。

（1）制作页面的 XHTML 代码，如下。

```
<div id="layout">
   一列固定宽度布局
</div>
```

这里给 div 使用了 layout 作为 id 名称，下一步就是为一列式布局定义样式。

（2）制作布局的 CSS 样式，代码如下。

```
div{
    background-color: #FFFF00;                    /*设置背景颜色*/
    border: 1px solid #000000;                    /*设置边框样式*/
    text-align: center;                           /*设置文本居中对齐*/
    padding-top: 40px;                            /*设置上内边距*/
    font-size: 14px;                              /*设置字体大小*/
    font-family: "宋体";                          /*设置字体类型*/
    font-weight: bold;                            /*设置为粗体*/
}
#layout{
    height: 300px;                                /*设置高度*/
    width: 400px;                                 /*设置宽度*/
}
```

在以上代码中，为了便于讲解，首先使用标签选择符 div 定义了一套公共样式属性，用于接下来的每一节当中。使用了 background-color 将 div 设定为黄色背景，并使用 border 属性

将 div 设置成了黑色的 1px 的边框。同样，为了布局美观，将 div 内部的字体设定了字体大小、居中显示以及上内边距，字体与边框上边界有一定空间。

然后，由于是宽度固定的布局，所以直接为#layout 元素设置了宽度属性 400px 与高度属性 300px。在前面讲到过，div 默认状态下，宽度将占据整行的空间，因此当设置了 width: 400px 之后，当前的 div 宽度将变为设置的宽度，这样便形成了一列式的固定宽度布局，也是最简单的布局形式。

在接下来讲解 CSS 布局样式的时候都将使用这样的代码以方便讲解。这样一列固定宽度的布局就实现了，预览效果如图 14-1 所示。

很多网站都是在一列固定宽度基础上衍生出来的布局类型，无论是结构简单的还是结构复杂的，如图 14-2 所示的网站是首先在一列固定宽度的布局基础上，再进行细致划分布局。

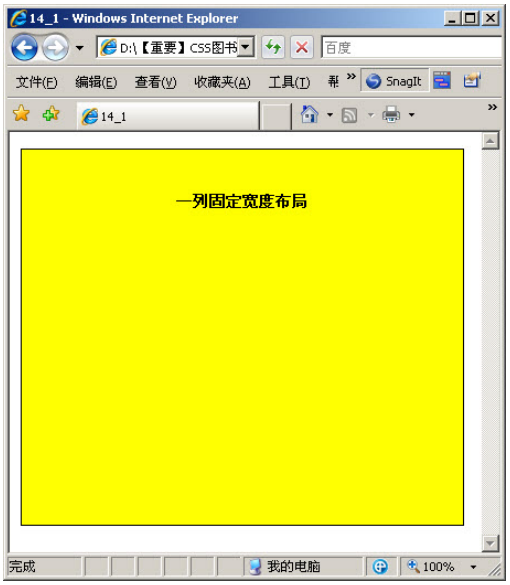

图 14-1　一列固定宽度布局

图 14-2　一列固定宽度的 CSS 布局网站

14.2　一列宽度自适应

自适应布局是在网页设计中最常见的一种布局形式，自适应的布局能够根据浏览器窗口的大小，自动改变其宽度或高度值，是一种非常灵活的布局形式，良好的自适应布局网站对

不同分辨率的显示器都能提供最好的显示效果。

1．实现原理

实际上 div 的默认状态占据整行的空间，便是宽度为 100%的自适应布局的表现形式，一列自适应布局需要做的工作也非常简单，只需要将宽度由固定值改为百分比宽度值。

2．制作实例

制作实例的步骤很简单，同样是分为制作 XHTML 代码和添加 CSS 样式表。

（1）制作页面的 XHTML 代码，如下。

```
<div id="layout">
   一列固定宽度布局
</div>
```

结构代码还是使用这样的代码。

（2）制作布局的 CSS 样式，代码如下。

```
div{
    [沿用第一节的样式]
}
#layout{
    height: 300px;                                    /*设置高度*/
    width: 80%;                                       /*设置宽度，单位使用百分比*/
}
```

自适应布局，大部分使用数值作为参数的样式属性都提供百分比值，width 宽度属性也不例外。在以上代码中，将宽度值重新设置为了 80%，这样 div 的宽度就变成了浏览器宽度的 80%，而宽度自适应的优势就是当扩大或者缩小浏览器窗口大小时，宽度还将维持在于浏览器当前宽度比例的 80%。这样宽度自适应的 CSS 布局就实现了，预览效果如图 14-3 所示。

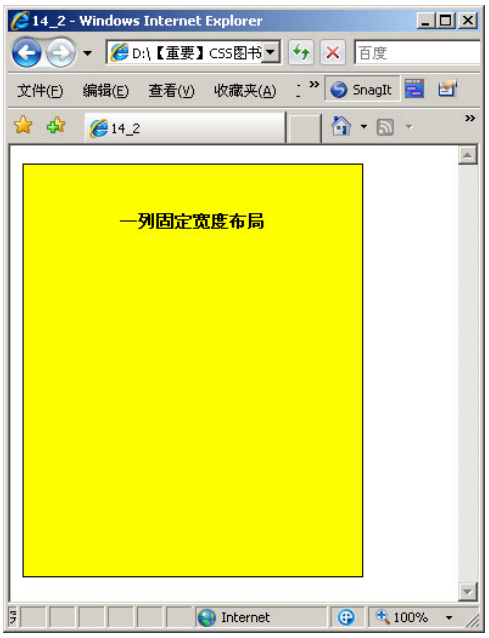

图 14-3　一列宽度自适应布局

有一些网站都是在一列宽度自适应的基础上衍生出来的布局类型，无论是结构简单的还是结构复杂的，如图 14-4 和图 14-5 所示是首先在一列宽度自适应的布局基础上，再进行细致划分布局。

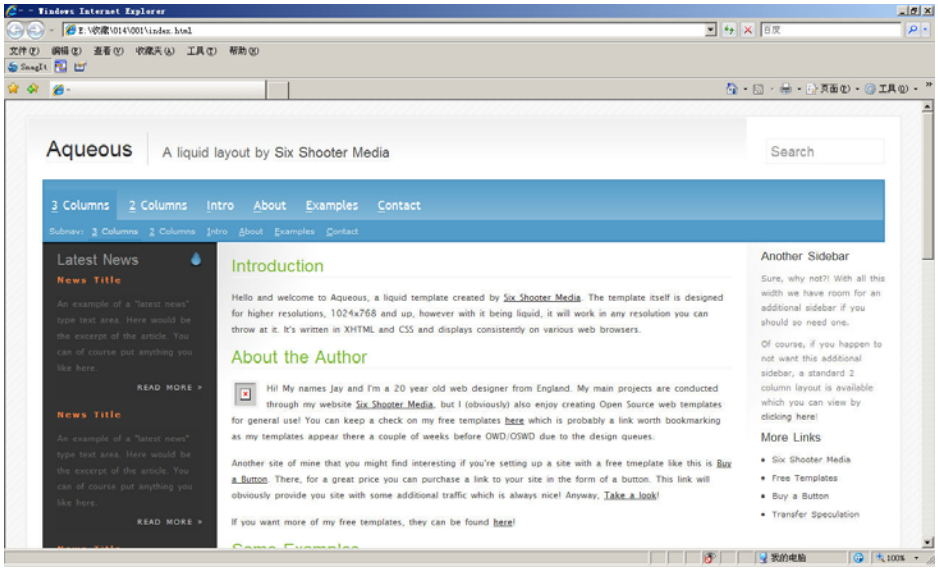

图 14-4　改变浏览器的宽度前的页面自适应

图 14-5　改变浏览器的宽度后的页面自适应

14.3　一列居中

页面整体居中是网页设计中常见的形式。在使用 XHTML 表格布局中，使用表格的

align="center"属性来实现整体居中。div 实现居中布局可以使用 align="center"属性，呈现居中状态。然而，CSS 布局是了实现表现与内容的分离，而 align 对齐属性是一种样式代码，写在 XHTML 的 div 属性之中，违背了分离原则，因此应当使用 CSS 的方法实现内容的居中。

1．实现原理

margin 的一个属性值 auto 是让浏览器自动判断边距，为 div 的左右边距设置了 auto，浏览器就会将 div 的左右边距设为相同，并且呈现为居中显示效果。

2．制作实例

现在以 14.2 节的宽度自适应布局的代码为例，使其居中的 CSS 代码如下。

```
div{
    [沿用第一节的样式]
}
#layout{
    height: 300px;                                    /*设置高度*/
    width: 80%;                                       /*设置宽度*/
    margin: 0px auto;                                 /*设置外边距，实现对象的居中显示*/
}
```

在以上代码中，最后一行添加了 margin 属性，margin 属性用于控制对象的上、右、下、左 4 个方向的外边框，当 margin 使用了两个参数时，第一个参数表示上下边距，第二个参数表示左右边距。margin 的一个属性值 auto 是让浏览器自动判断边距，在这里就给当前的 div 的左右边距设置了 auto，浏览器就会将 div 的左右边距设为相同，并且呈现为居中状态，从而实现居中布局效果，预览效果如图 14-6 所示。

有很多网站都是在一列居中的基础上衍生出来的布局类型，无论是结构简单的还是结构复杂的，网站要求就是居中显示，如图 14-7 所示。

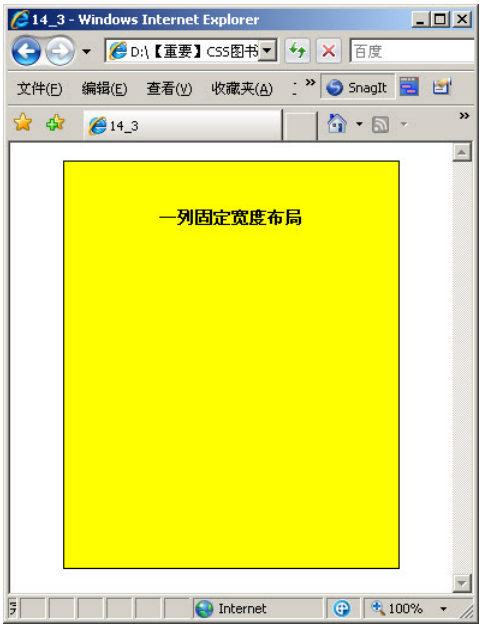

图 14-6　一列居中布局

图 14-7　一列居中网站实例

14.4 二列固定宽度

有了一列固定宽度的基础，二列固定宽度就很简单了。制作一列布局使用到一个 div，那么制作二列布局，自然就需要用到两个 div。本节将讲解如何制作二列固定宽度的 CSS 布局。

1. 实现原理

在一列固定宽度的实现基础上，二列固定宽度就是把每一列的 div 都设定为固定宽度值，div 之间的排列使用浮动定位属性。

2. 制作实例

制作实例的步骤很简单，也同样是分为制作 XHTML 代码和添加 CSS 样式表。

（1）制作页面的 XHTML 代码，如下。

```
<div id="left">左列</div>
<div id="right">右列</div>
```

在以上代码中，使用了两个 id，分别是 left 和 right，表示两个 div 的名称。需要为它们设置宽度，然后让两个 div 在水平行中并排显示，从而形成二列式布局。

（2）制作布局的 CSS 样式，代码如下。

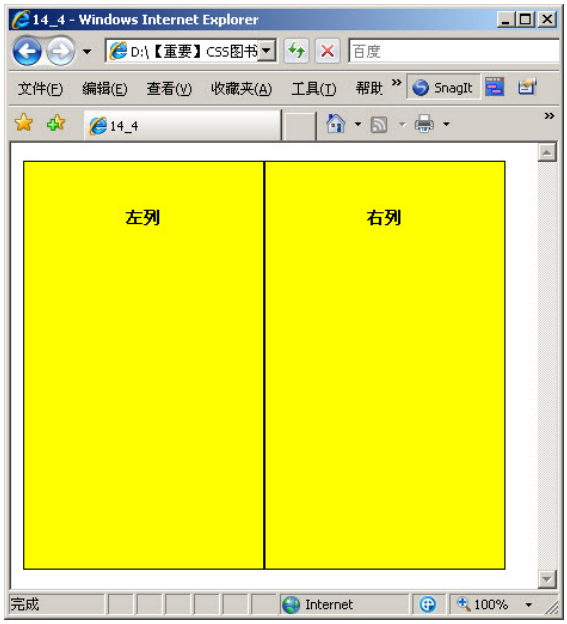

图 14-8 二列固定宽度布局

```
div{
    [沿用第一节的样式]
}
#left
    height: 300px;          /*设置高度*/
    width: 200px;           /*设置宽度*/
    float: left;            /*设置向左浮动定位*/
}
#right{
    height: 300px;          /*设置高度*/
    width: 200px;           /*设置宽度*/
    float: left;            /*设置向左浮动定位*/
}
```

在以上代码中，left 与 right 两个 div 的代码与前面类似，都使用相同宽高，而为了实现二列式布局，使用浮动属性 float。这里使用向左浮动定位，左栏向左浮动定位，它右侧的右栏对象浮动到它的右侧定位。这样使用 float 属性之后，二列固定宽度的布局就实现了，预览效果如图 14-8 所示。

使用二列固定宽度布局的网站也不占少数，如图 14-9 所示。

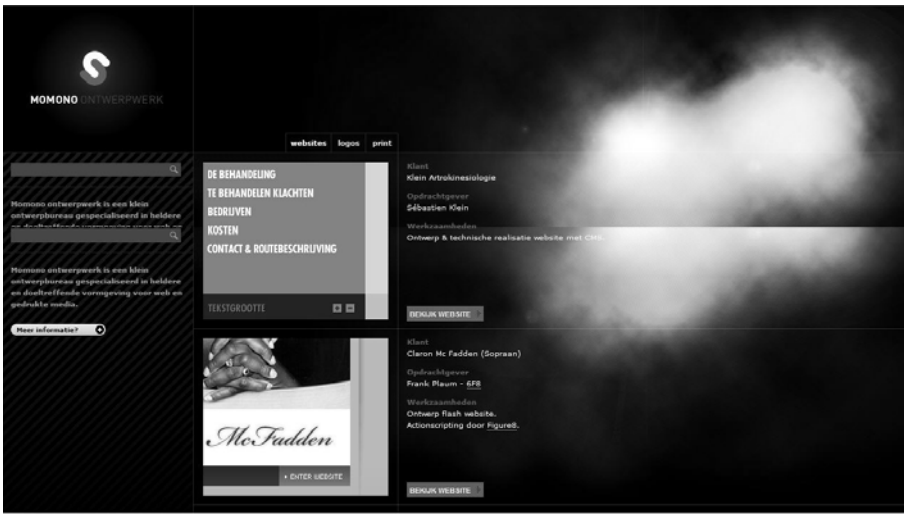

图 14-9　二列固定宽度布局实例

14.5　二列宽度自适应

本节在尝试二列布局的情况下，左右分栏宽度能够做到自适应，从一列宽度自适应布局可知，设定自适应主要通过宽度的百分比值来设置，因此在二列宽度自适应布局中也同样是对百分比宽度值的设计。

1．实现原理

在一列宽度自适应的实现基础上，二列宽度自适应就是把每一列的 div 都设定为百分比值，并且总的宽度百分比值不超过 100%，div 之间的排列使用浮动定位属性。

2．制作实例

继续使用 14.4 节的 XHTML 代码，所做的是重新定义布局的 CSS 样式，如下。

```
div{
    [沿用第一节的样式]
}
#left{
    height: 300px;                                    /*设置高度*/
    width: 20%;                                       /*设置宽度，使用百分比为单位*/
    float: left;                                      /*设置向左浮动定位*/
}
#right{
    height: 300px;                                    /*设置高度*/
    width: 70%;                                       /*设置宽度，使用百分比为单位*/
    float: left;                                      /*设置向左浮动定位*/
}
```

在以上代码中，左栏设置为宽度 20%，右栏设置为宽度 70%，看上去像一个左侧为导航，右侧为内容的常见网页布局形式。

注意：这里没有将浏览器设置为 80%去实现整体的 100%的效果，是因为为了使布局在预览中更清楚，使用 border 属性，使得左右分栏两个对象都具有 1px 的深色边框线。正如第 6 章提到的，在 CSS 布局之中，一个对象的宽度不仅仅由 width 值来决定，一个对象的真实宽度是由该对象本身的宽、对象的左右外边框以及左右边框，还有内边距这些属性相加而成，因此左侧的对象不仅仅是浏览器窗口的 20%宽度，还应当加上左边的深色边框。这样，左右分栏都超过了自身的百分比宽度，最终的宽度也超过了浏览器窗口的宽度，因此右栏将被浮动定位到第二行显示，那样就达不到左右分栏的效果，因此这里使用了并非 100%的宽度之和。在实际应用中，可以通过避免边框及边距的使用，而达到左右与浏览器填满的效果。

修改 CSS 样式后，修改为二列宽度自适应布局的预览效果如图 14-10 所示。

同样的，CSS 代码也可以如下。

```
#left{
    width: 80%;                                    /*设置宽度，使用百分比为单位*/
    float: left;                                   /*设置向左浮动定位*/
}
#right{
    width: 15%;                                    /*设置宽度，使用百分比为单位*/
    float: left;                                   /*设置向左浮动定位*/
}
```

修改两个列的宽度比，左列为 80%，右列为 15%，左列比右列宽，如图 14-11 所示。

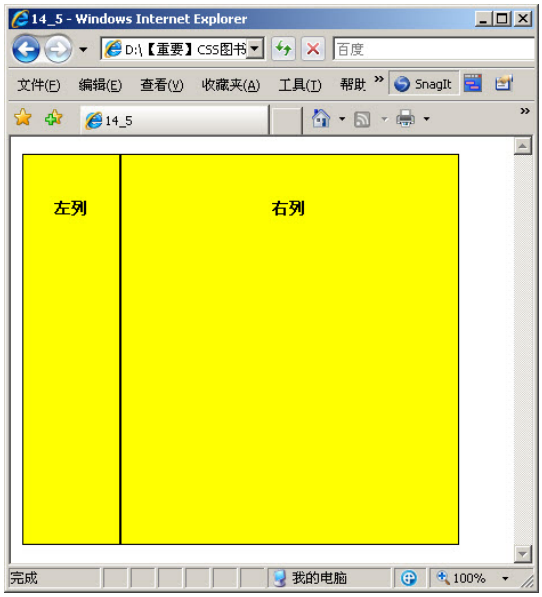

图 14-10　二列宽度自适应布局 1

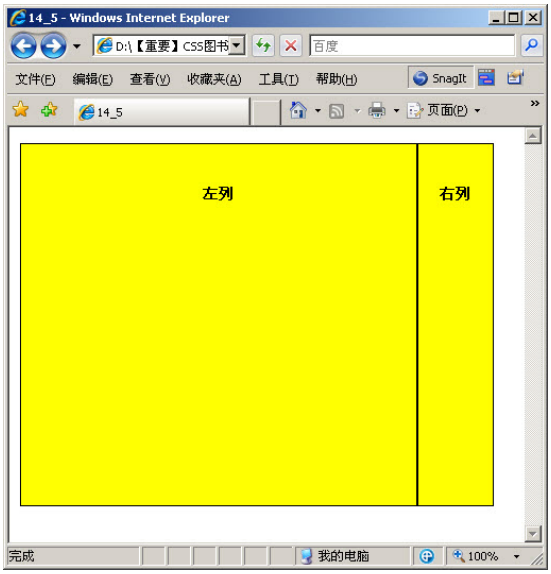

图 14-11　二列宽度自适应布局 2

14.6　两列右列宽度自适应

在实际应用中，有时候需要制作左右两栏的布局，要求右栏固定宽度，左栏根据浏览器窗

口的大小自动适应，在 CSS 中实现这样的布局方式是简单可行的，本节就将介绍其制作方法。

1．实现原理

设置左栏的宽度即可，将左栏宽度设定为固定值，右栏不设置任何宽度值，并且右栏不浮动。

2．制作实例

继续使用以前的 XHTML 代码，所做的是重新定义布局的 CSS 样式，代码如下。

```
div{
    [沿用第一节的样式]
}
#left{
    height: 300px;                                    /*设置高度*/
    width: 100px;                                     /*设置宽度，使用固定宽度*/
    float: left;                                      /*设置向左浮动定位*/
}
#right{
    height: 300px;                                    /*设置高度*/
    float: left;                                      /*设置向左浮动定位*/
}
```

在以上代码中，左栏将呈现为 100px 的宽度，右栏将根据浏览器窗口的大小自动适应。这样实现了二列右列宽度自适应布局，预览效果如图 14-12 所示。

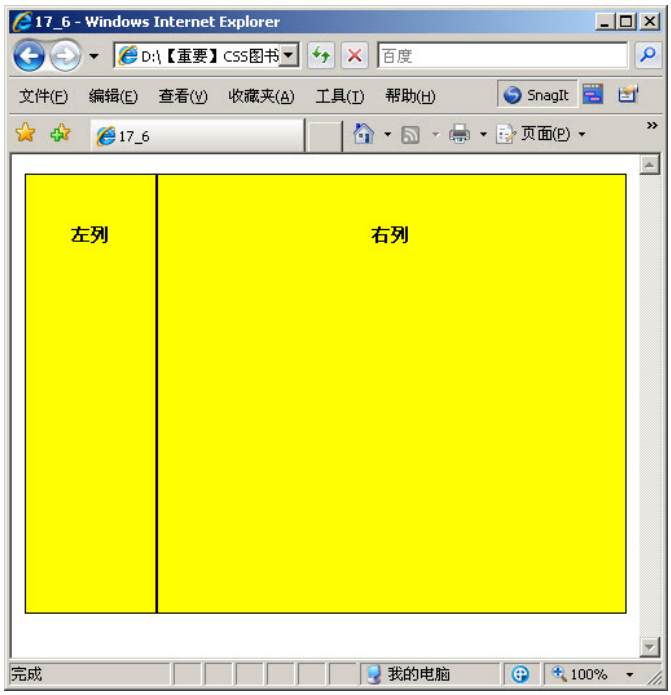

图 14-12　两列右列宽度自适应布局

二列右列宽度自适应的布局主要应用在博客、wiki 这样的日志、信息类网站，如图 14-13 所示。

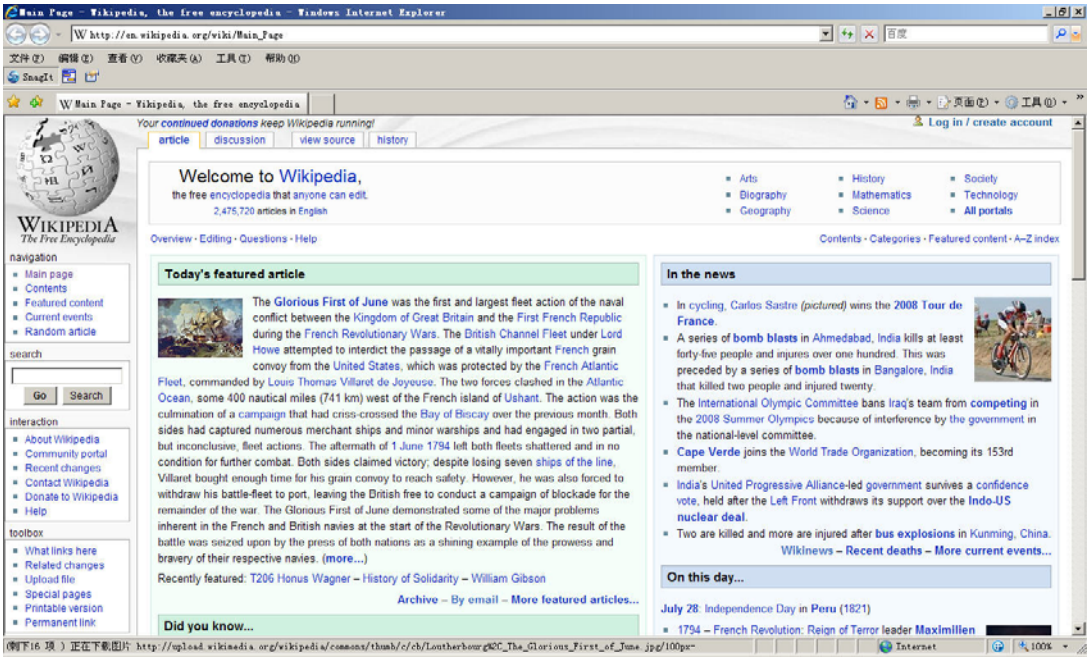

图 14-13　二列宽度自适应的实例

14.7　两列固定宽度居中

在一列固定宽度居中时候，使用了外边距 margin: 0px auto 来控制，使一个 div 得以达到居中显示，而二列分栏中，需要控制的是左分栏的左边和右分栏的右边相等，因此使用 margin: 0px auto 就不能达到想要的要求了，这时就需要进行 div 的嵌套设计来完成。

1．实现原理

使用一个居中的 div 作为主容器，将二列分栏的两个 div 放置在容器中，从而实现二列的居中显示。

2．制作实例

制作实例的步骤很简单，也同样是分为制作 XHTML 代码和添加 CSS 样式表。

（1）制作页面的 XHTML 代码，如下。

```
<div id="layout">
   <div id="left">左列</div>
   <div id="right">右列</div>
</div>
```

在以上代码中，将分栏的两个 div 加上了一个 id 为 layout 的 div 容器。

（2）制作布局的 CSS 样式，代码如下。

```
div{
    [沿用第一节的样式]
}
#layout {
       margin: 0px auto;                                    /*设置外边距，实现对象的居中*/
       width: 404px;                                        /*设置固定宽度*/
}
#left {
       height: 300px;                                       /*设置高度*/
       width: 200px;                                        /*设置固定宽度*/
       float: left;                                         /*设置向左浮动定位*/
}
#right {
       height: 300px;                                       /*设置高度*/
       width: 200px;                                        /*设置固定宽度*/
       float: left;                                         /*设置向左浮动定位*/
}
```

在以上代码中，首先在#layout 元素中将#layout 定义成了居中，这样里面的内容也能够做到居中。

需要注意的是，在#layout 的宽度定义时，将#layout 的宽度设定为 404px。在前面盒模型的时候讲到过，一个对象的真正宽度是由它的各种属性相加而成，而#left 的宽度为 200pz，但左右都有 1px 的边距，因此实际宽度是 202px，#right 同样如此。这样，为了让#layout 作为容器能够装下它们两个，宽度则变为#left 和#right 的实际宽度，便得到了 404px 的结构。这样二列固定宽度居中的布局实现了，预览效果如图 14-14 所示。

采用两列固定宽度居中的布局网站也有很多，如图 14-15 所示。

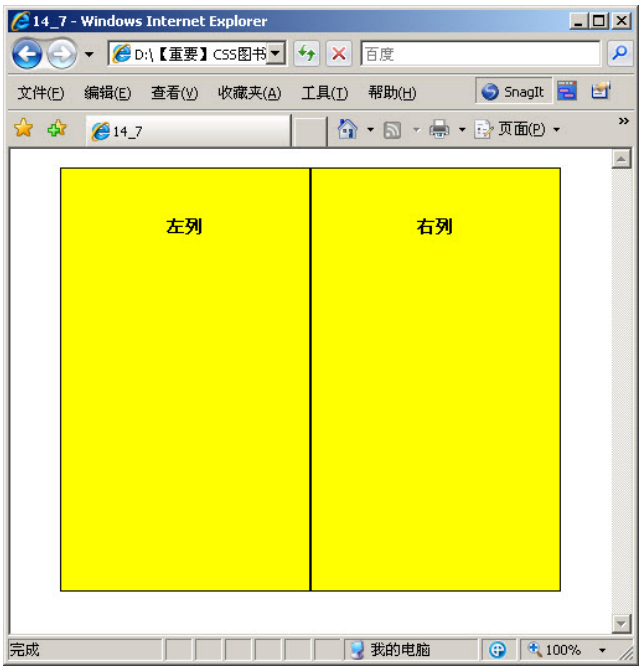

图 14-14　二列固定宽度居中布局

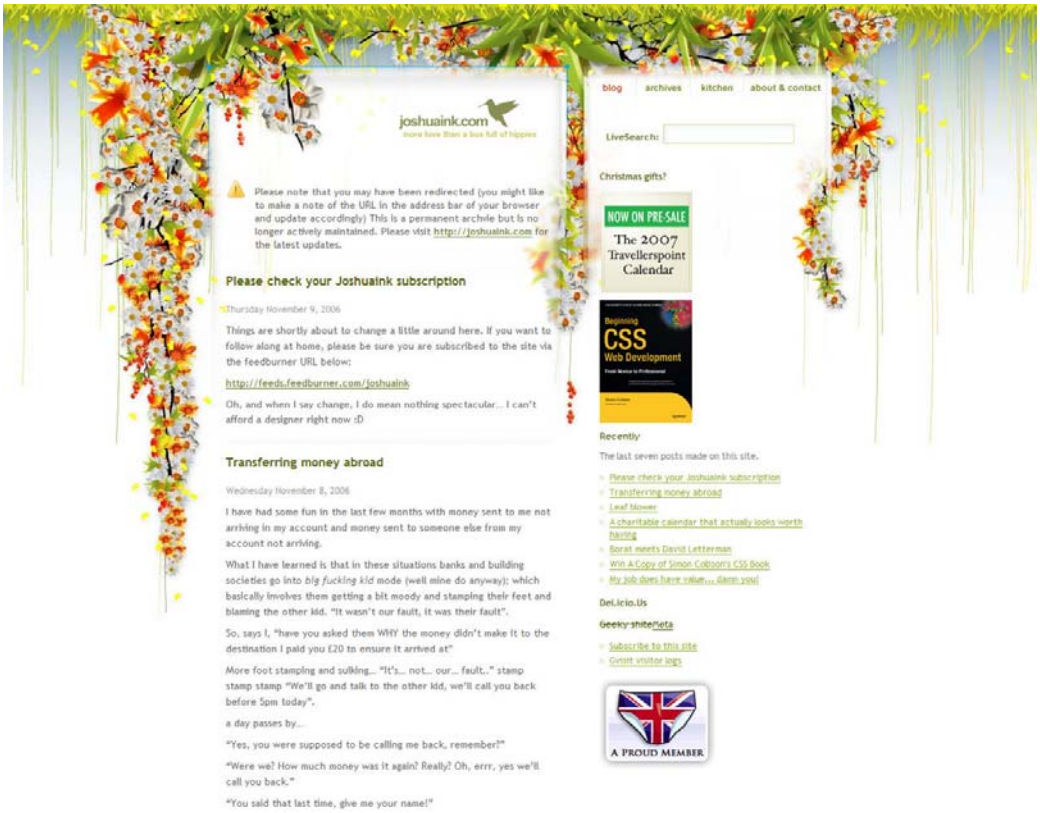

图 14-15　两列固定宽度居中的实例

14.8　三列浮动中间列宽度自适应

使用浮动定位方式，从一列到多列的固定宽度以及自适应，基本上可以简单实现，包括三列的固定宽度。

本节将制作一个三列式布局，其中左栏要求固定宽度，并且居左显示，右栏要求固定宽度并居右显示。而中间栏需要在左栏和右栏的中间，根据左右栏的间距变化自适应宽度。这样的布局要求，单纯使用 float 属性与百分比属性并不能够实现，CSS 目前还不支持百分比的计算精确到考虑左栏与右栏的占位，如果对中间栏使用 100%的宽度的话，它将使用浏览器窗口的宽度，而非左栏和右栏的中间间距。现在就将使用定位属性 position 来解决这个问题。

1．实现原理

使用了绝对定位属性 position，将左栏与右栏进行位置控制。左栏贴进左边缘进行显示。而右栏居右显示。中间一栏，不需要再设定其浮动方式，让它的左外边距保持左栏的宽度和右外边距保持右栏的宽度，而左右两边让出的距离，刚好让中栏显示在这个空间中，从而实现布局要求。

2．制作实例

制作实例的步骤很简单，也同样是分为制作 XHTML 代码和添加 CSS 样式表。

（1）制作页面的 XHTML 代码，如下。

```
<div id="left">左列</div>
<div id="center">中列</div>
<div id="right">右列</div>
```

在以上代码中，使用了 3 个 div 形式所需要的 3 个分栏结构。

（2）制作左栏的 CSS 样式，代码如下。

```
div{
    [沿用第一节的样式]
}
#left {
        height: 300px;                                          /*设置高度*/
        width: 100px;                                           /*设置高度*/
        position: absolute;                                     /*设置相对定位*/
        top: 0px;                                               /*相对定位，设置顶部距离*/
        left: 0px;                                              /*相对定位，设置左边距离*/
}
```

在以上代码中，关键的方法是使用了绝对定位 position: absolute，将左栏进行位置控制，左栏将距浏览器左边界 left: 0px，贴进左边缘进行显示，同样它也距浏览器上边界 top: 0px，贴进上边缘进行显示。

（3）制作右栏的 CSS 样式，代码如下。

```
#right {
        height: 300px;                                  /*设置高度*/
        width: 100px;                                   /*设置宽度*/
        position: absolute;                             /*设置相对定位*/
        right: 0px;                                     /*相对定位，设置右边距离*/
        top: 0px;                                       /*相对定位，设置顶部距离*/
}
```

在以上代码中，和左栏设置方法类似，使用了绝对定位 position: absolute，将右栏进行位置控制，右栏将距浏览器右边界 right: 0px，贴进右边缘进行显示，同样它也距浏览器上边界 top: 0px 贴进上边缘进行显示。

（4）制作中栏的 CSS 样式，代码如下。

```
#center{
        height: 300px;                                  /*设置高度*/
        margin-left: 102px;                             /*设置左外边距*/
        margin-right: 102px;                            /*设置右外边距*/
}
```

在以上代码中，对于#center，不需要再设定其浮动方式，只需要让它的左外边距永远保持#left 和#right 元素的宽度，便可实现两边各让出 102px 距离的自适应宽度，而左右两边让出的距离，刚好让左栏和右栏显示在这个空间中，从而实现布局要求。这样三列浮动中间列宽度自适应布局实现了，预览效果如图 14-16 所示。

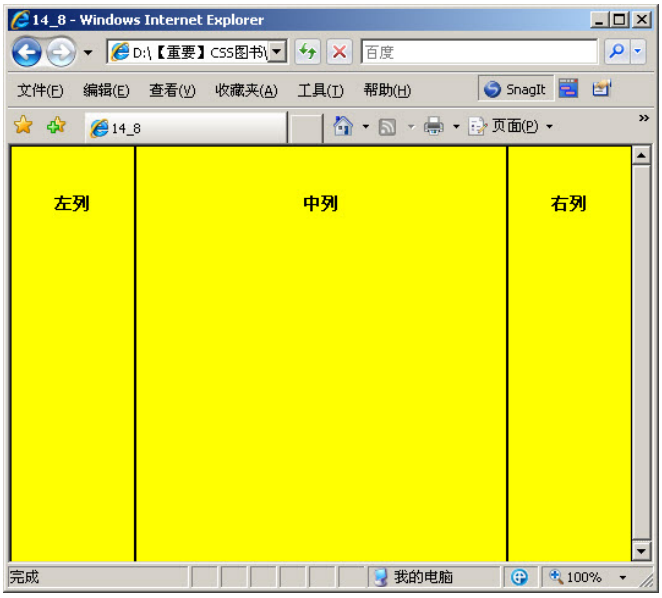

图 14-16　三列浮动中间列宽度自适应布局

三列浮动中间列宽度自适应的布局一般是应用在网站的一部分，例如网站的主体内容，在整个网站的布局基础上，主体部分重新进行布局，所谓布局嵌套布局，如图 14-17 和图 14-18 所示。

图 14-17　三列浮动中间列宽度自适应的网站实例改变宽度前

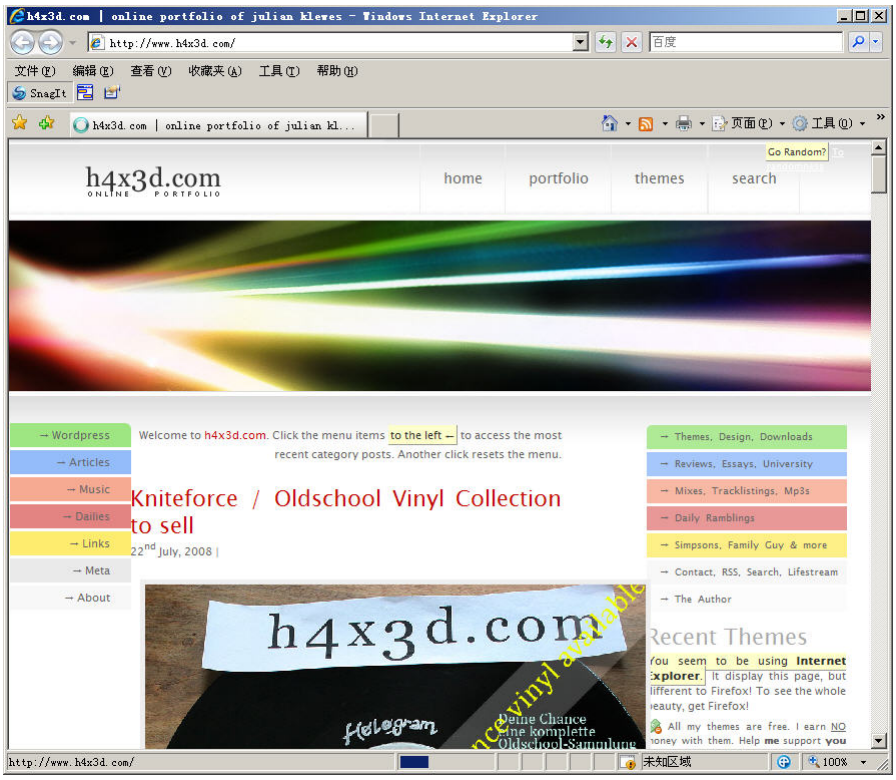

图 14-18　三列浮动中间列宽度自适应的网站实例改变宽度后

14.9 高度自适应

前面探讨的都是横向对象之间的排列组合方式，包括横向的宽度自适应。本节将探讨如何进行高度自适应问题。

实际上，很多人在初次尝试高度自适应会遇到这样的问题，对象的 height: 100%并不能够直接产生高度自适应的效果，产生这样的原因不是浏览器不支持 height: 100%的编写方法。高度值可以使用百分比进行设置，不同的是，之所以直接使用 height: 100%不能达到效果，与浏览器的解析方式有一定关系。

（1）制作页面的 XHTML 代码，如下。

```
<div id="layout">高度自适应</div>
```

（2）制作布局的 CSS 样式，代码如下。

```
html,body{
      margin: 0px;                                    /*设置外边距*/
       height: 100%;                                  /*设置高度，使用百分之百高度*/
}
div{
[沿用第一节的样式]
}
#layout{
      height: 100%;                                   /*设置高度，同样使用百分之百高度*/
      width: 300px;                                   /*设置宽度*/
      float: left;                                    /*设置向左浮动定位*/
}
```

在以上代码中，对#layout 元素设置了 height: 100%，同时为标签选择符 html 和 body 设置了 height:100%，这个就是高度自适应的关键方法所在。

一个对象高度是否可以使用百分比显示，取决于对象的父级对象，#layout 在页面中直接放置在 body 元素之中，因此它的父级就是 body，而浏览器默认状态下，是没有给 body 元素一个高度属性的，因此当直接为#layout 元素设置 height: 100%时，是不会产生任何效果的，而当为 body 设置了 height: 100%之后，它的子级对象#layout 的 height: 100%变发生作用了，这便是浏览器解析规则引发的高度自适应问题。

在 CSS 代码中，除了给 body 设置了高度以外，给 html 元素也设置了相同的样式，这样做是使 IE 与 Firefox 浏览器都能够实现高度自适应，IE 与 Firefox 对页面中对象的解析方法存在一定的差异，这也是第 6 章讲到的。在 IE 中 html 元素默认为 100%高度，而 body 却不是。在 Firefox 中 html 元素就不是 100%高度，因此给两个标签都定义为 height: 100%以保证两个浏览器下均能够正常显示。这样高度自适应布局就实现了，预览效果如图 14-19 所示。

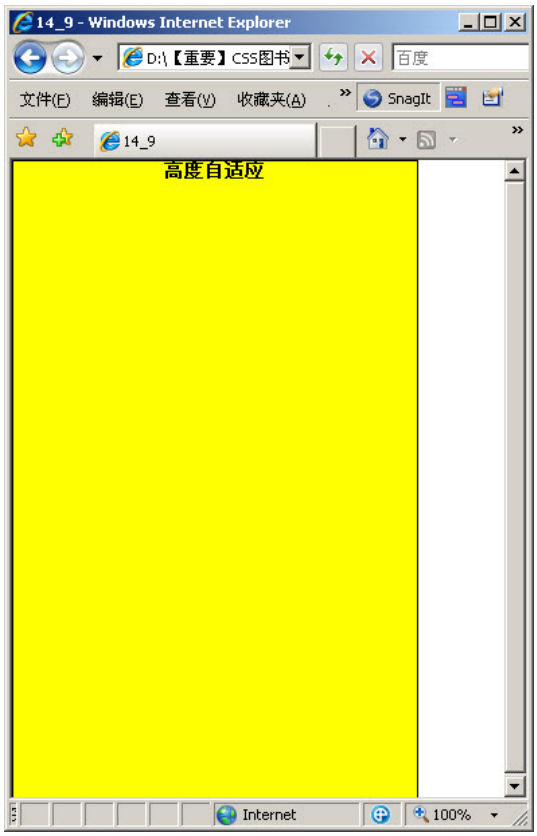

图 14-19　高度自适应布局

14.10　小结

本章讲解的是使用 CSS+DIV 进行网页的基本布局形式，主要涉及到以下 9 种常用的布局样式。

- 一列固定宽度。
- 一列宽度自适应。
- 一列居中。
- 二列固定宽度。
- 二列宽度自适应。
- 两列右列宽度自适应。
- 两列固定宽度居中。
- 三列浮动中间列宽度自适应。
- 高度自适应。

对本章的知识点前后联系进行归纳总结，知识点结构导图如图 14-20 所示。

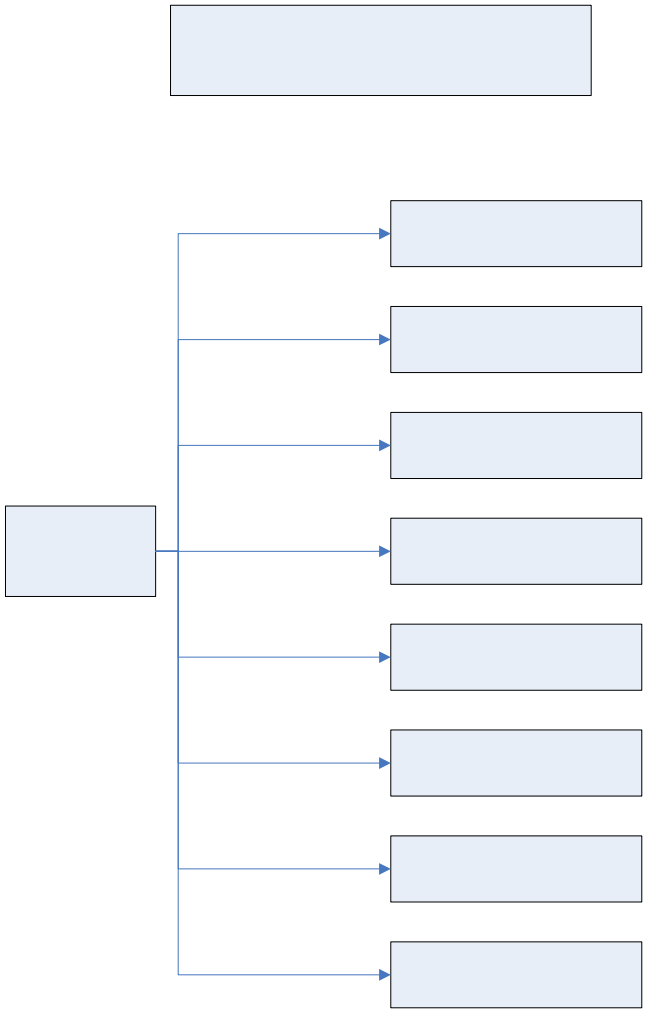

图 14-20　本章知识点结构导图

9种基本布局方式