第5章 CSS 定位布局

课程提要

- 定位（position）
- 网站整体布局
- BFC
- IFC

5.1 定位（position）

> position 属性规定元素的定位类型。这个属性定义建立元素布局所用的定位机制。任何元素都可以定位，不过绝对或固定元素会生成一个块级框，而不论该元素本身是什么类型。相对定位元素会相对于它在正常流中的默认位置偏移。

属性值：

- static：默认值，没有定位，元素出现在正常的文档流中，这时给这个元素设置的left,right,bottom,top这些偏移属性都是没有效果的 。
- relative：相对定位。
- absolute：绝对定位。
- fixed：固定定位。

5.1.1 相对定位

> 生成相对定位的元素他会跟其它的元素一样，出现在文档流中它该出现的位置，可以设置它的水平或垂直偏移量，让这个元素相对于它在文档流中的位置的起始点进行移动。有一点要注意， 在使用相对定位时，就算元素被偏移了，但是他仍然占据着它没偏移前的空间。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。

- 属性值：relative

```
#box_relative {
  position: relative;
  left: 30px;
  top: 20px;
}
```

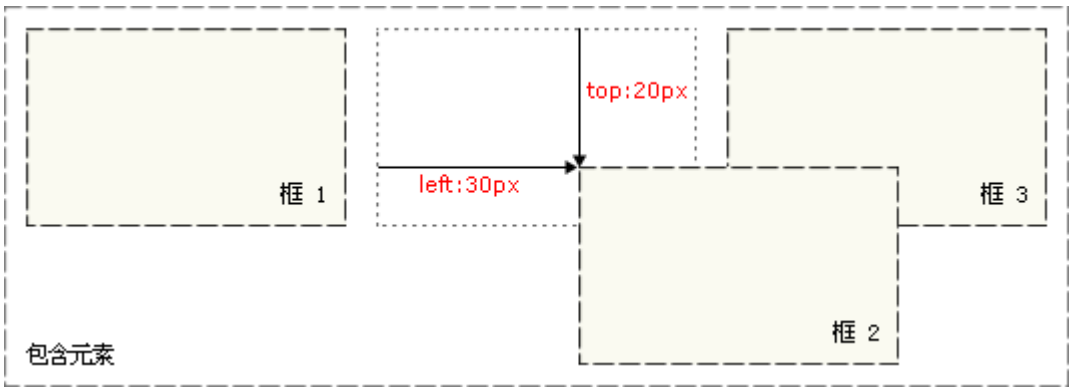

图5-1 相对定位

5.1.2 绝对定位

> 绝对定位使元素的位置与文档流无关，因此不占据空间。 生成绝对定位的元素，相对于 static 定位以外的第一个父元素（body）进行定位。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。

- 属性值：absolute

```
#box_relative {
  position: absolute;
  left: 30px;
  top: 20px;
}
```

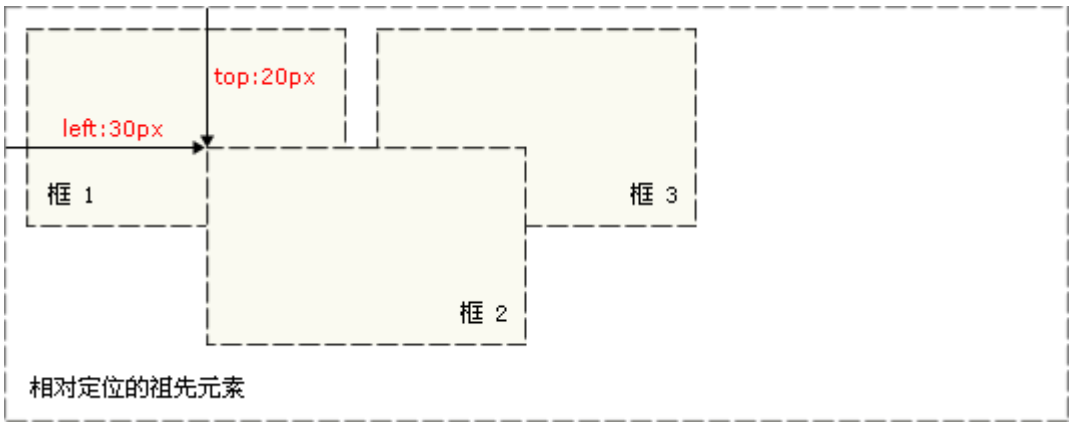

图5-2 绝对定位

5.1.3 固定定位

> 生成固定定位的元素，相对于浏览器窗口进行定位。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。

- 属性值：fixed

```
<!--div会出现在屏幕的右下角-->
<div style='position:fixed;right:0px;bottom:0px;'></div>
```

5.1.4 相对定位&绝对定位

> 父元素使用相对定位，子元素使用绝对定位后，这样子元素的位置不在浏览器左上角，而是相对于父容器左上角。

- 属性值：relative&absolute

```
<!--元素p会一直在父元素div的右上角-->
<div style='position:relative;'>
    <p style='position:absolute;right:0;top:0'></p>
</div>
```

5.1.5 z-index

> 当多个元素添加绝对定位，元素将会叠加在一起，使用z-index可以设置元素显示的层次。

- **注意：**z-index 仅能在定位元素上奏效（例如 position:absolute;）！
- **注意：**元素可拥有负的 z-index 属性值。

  说明：一般元素为普通流，普通流的z-index默认为0，脱离了普通流，在普通流之上（定位，浮动）z-index为0-1之间。如果将z-index值设置为大于或者等于1，元素将会在定位或者浮动流之上。

```
<style>
    div{
        width: 400px;
        height: 200px;
    }
    .div1{
        width: 1000px;
        height: 600px;
        background-color: #f00;
        position: relative;
    }
    .div2{
        background-color: #0f0;
        position: absolute;
        z-index: 2;
    }
    .div3{
        background-color: #00f;
        position: absolute;
        z-index: 3;
    }
</style>
<div class="div1">
    <div class="div2"></div>
    <div class="div3"></div>
</div>
```

5.2 网站整体布局

5.2.1 双飞翼布局

> 双飞翼布局是将一个网页分为左列、中列和右列三部分，并且我们要得到的效果是：左列和右列宽度恒定，中间列的宽度根据浏览器窗口的大小自适应。

- 在外层用一个类名为：container的大的div包裹。而内层分为了三列。

```
<div class="container">
    <div class="column" id="center_panel"></div>
    <div class="column" id="left_panel"></div>
    <div class="column" id="right_panel"></div>
</div>
```

- 让列开始浮动

```
 .container{width:100%;}
 .column{
    float:left;
    height: 200px;
 }
 /*在这种情况下，center_panel已经占据了整个container父元素的全部宽度，并且这个宽度是自适应的*/
 #center_panel{
    width:100%;
    background-color: #f00;
 }
```

- 如何实现左右列固定宽度,在实现左右列固定宽度时，我们这里采用了margin赋以负值。

```
/*假设，固定宽度的左列宽度为300px，当margin-left:-100%之时，这个左边列会脱离自己所在行，向上一行浮动*/
#left_panel{
     width:300px;
     margin-left:-100%;
     background-color: #0f0;
}
```

- 此时我们实现了左列固定宽度，中间列自适应。我们根据同样原理，定义右列的浮动偏移（margin负值）。这样，我们就实现了我们所需要的双飞翼布局，也就是中间列宽度自适应，左列和右列的宽度固定。

```
#right_panel{
   width:300px;
   margin-left:-300px;
   background-color: #00f;
}
```

双飞翼布局的优点：

（1）兼容性好，兼容所有主流浏览器，包括万恶的IE6。

（2）因为在DOM中center_panel在三列结构的最前面，因此可以实现主要内容的优先加载。

5.2.2 圣杯布局

> 圣杯布局和双飞翼布局实现的问题都是三列布局，两边定宽，中间自适应布局。

实现方式：

定义左侧宽度为 200px，右侧宽度为 150px，中间是流动的布局。

```
<!-- html代码 -->
<div id="header">#header</div>

<div id="container">
    <!-- 把三栏塞进去了 -->
    <div id="center" class="column">#center</div>
    <div id="left" class="column">#left</div>
    <div id="right" class="column">#right</div>
</div>

<div id="footer">#footer</div>
```

```
/* css代码 */
body {min-width: 550px;}
/* 设置内容的左右内边距为200和150 */
#container {padding-left: 200px;padding-right: 150px;}
/* 设置元素左浮动和相对定位 */
#container .column {height: 200px;position: relative;float: left;}
#center {background-color: #e9e9e9;width: 100%;}
#left {background-color: red;width: 200px;right: 200px;margin-left: -100%;}
#right {background-color: blue;width: 150px;margin-right: -150px;  }
#footer {clear: both;}
#header, #footer {background-color: #c9c9c9;}
```

原理：

- 1.将 container 的内边距设置为左右两边各自的宽度。如图5-4所示：

```
#container {padding-left: 200px;padding-right: 150px;}
```

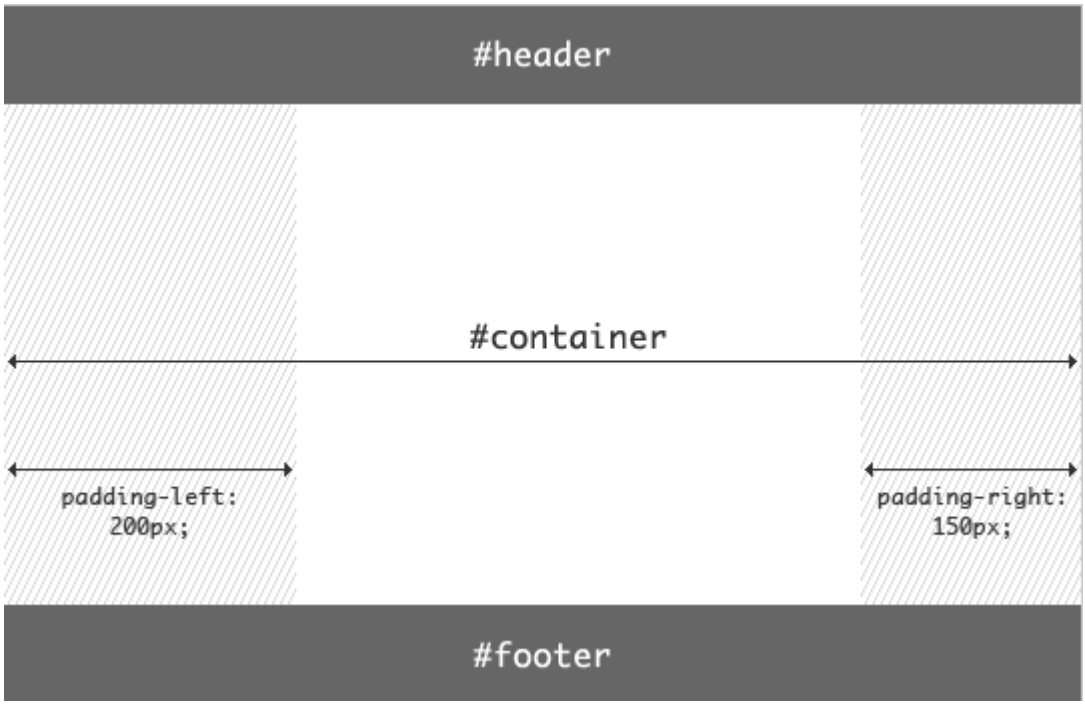

图5-3 圣杯布局1

- 2.每一栏配上合适的宽度，并将它们设为浮动。同时我们需要清除 footer 的上下环境，以免遭跟上面三栏一起浮动。

```
/*中间一栏的 100% 宽是基于它的父容器 container 的宽度而言的，由于 container 设置了内边距，因此中间栏看起来就处在了网页的中间，但左右两栏由于排在中间栏的后面，且因为空间不够被挤到了中间栏的下面*/

#container .column {float: left;}
#center {width: 100%;}
#left {width: 200px;}
#right {width: 150px;}
#footer {clear: both;}
```

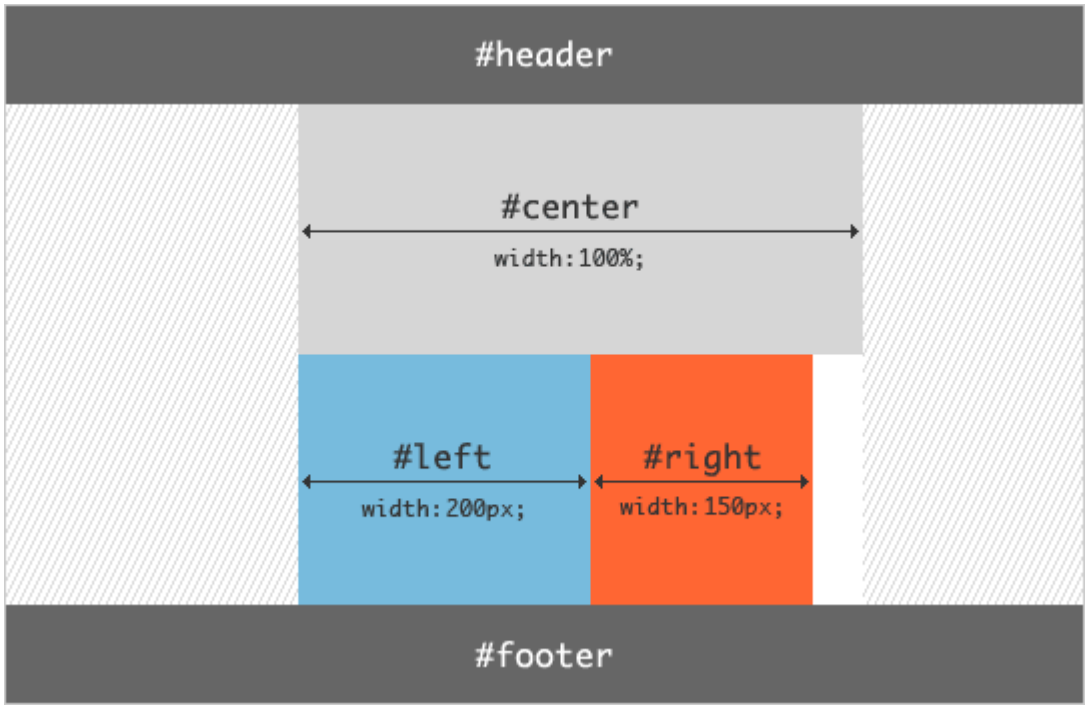

图5-4 圣杯布局2

- 3.把左侧栏放上去

```
/* 将左侧栏的外边距设置为 -100%，这样一来，由于浮动的关系，左侧栏就能上位，与中间栏交叠在一起，并占据了左边。而右侧栏由于左侧栏的上位，自动向前浮动到了原来左侧栏的位置。*/
/* 接着我们要用到相对定位属性（relative），并设置一个与左侧栏等宽的偏移量。*/
#container .columns {float: left;position: relative;}
#left {width: 200px; margin-left: -100%;  right: 200px;}
```

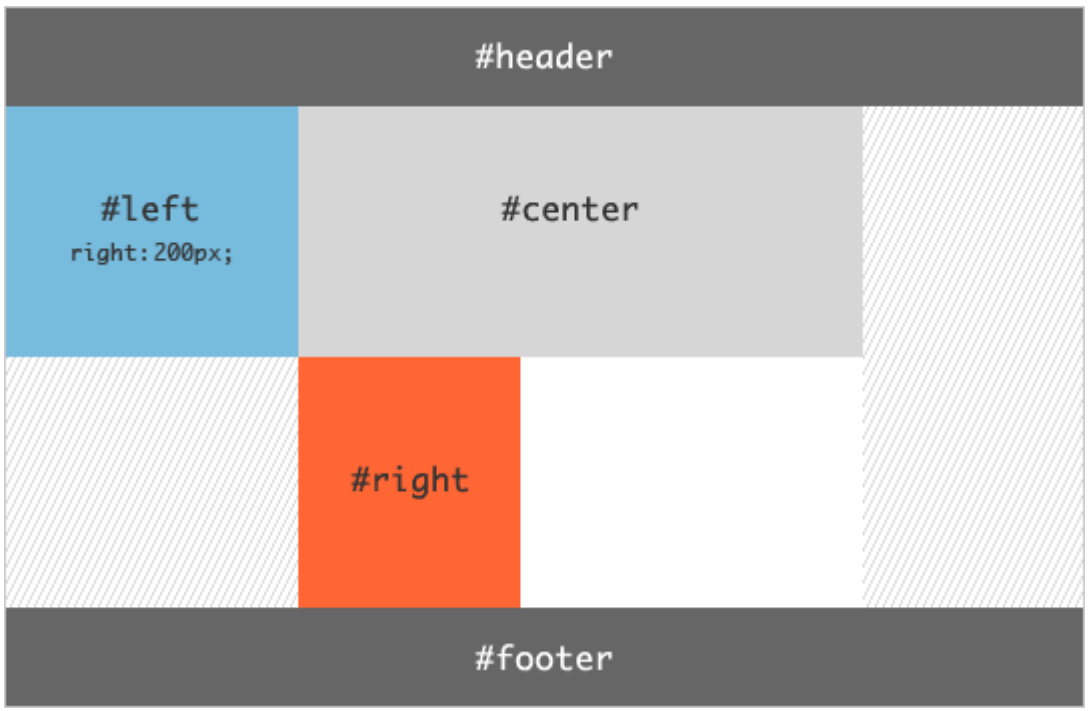

图5-5 圣杯布局3

- 4.把右侧栏放上去

```
/* 利用上面的原理把他放到 container 的右外边距的位置即可，我们需要再一次设置一个负外边距的值，它等于右侧栏的宽度。*/
#right {
  width: 150px;
  margin-right: -150px;
}
```

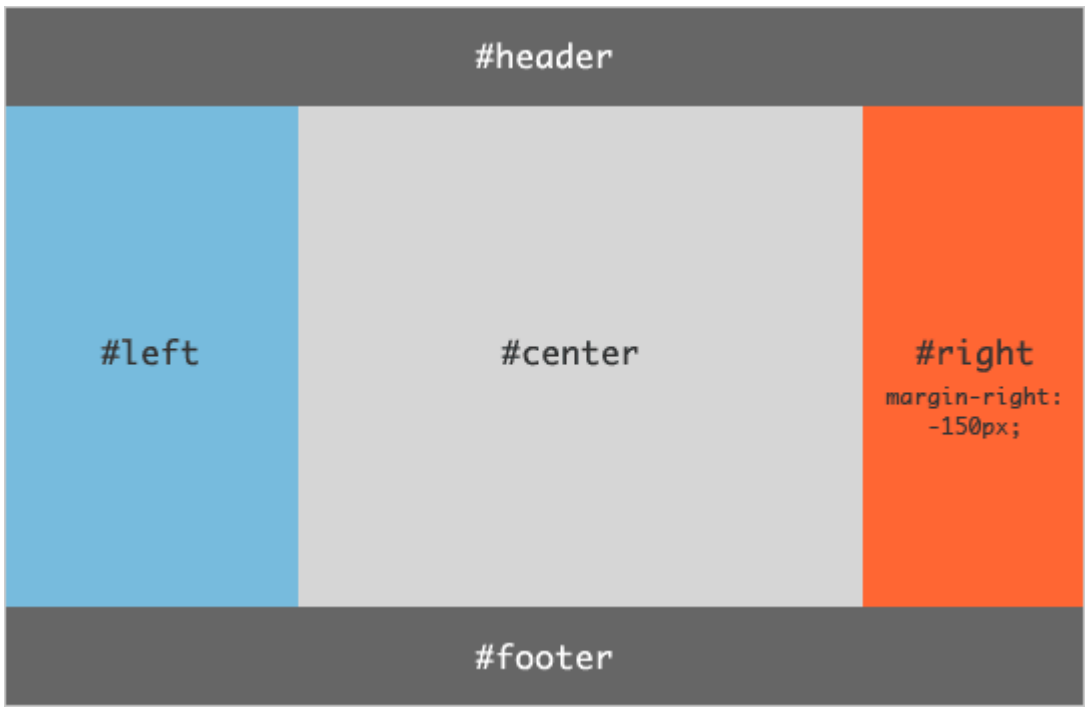

图5-6 圣杯布局4

5.2.3 侧边栏固定布局

> 侧边栏布局是一种比较常见的布局方案，它在空位紧缺的界面内可以提供更多快捷的入口供用户操作，简化了操作层级。

图5-7 侧边栏固定布局1

1.两列右侧自适应布局

图5-8 侧边栏固定布局2

```
<div class="g-bd1">
    <div class="g-sd1" style="background: red;">
        <p>左侧定宽</p>
    </div>
    <div class="g-mn1" style="background: blue;">
        <p>右侧自适应</p>
    </div>
</div>
/* 两列右侧自适应布局 */
body{min-width:550px;}
.g-sd1{position:relative;float:left;width:190px;margin-right:-190px;}
.g-mn1{float:right;width:100%;}
.g-mn1 p{margin-left:200px;}
```

2.两列左侧自适应布局

图5-9 两列左侧自适应布局

```
<div class="g-bd2">
    <div class="g-mn2" style="background: red;">
        <div class="g-mn2c">
            <p>左侧自适应</p>
        </div>
    </div>
    <div class="g-sd2" style="background: blue;">
        <p>右侧定宽</p>
    </div>
</div>

/* 两列左侧自适应布局 */
.g-bd2{margin:0 0 10px;}
.g-sd2{position:relative;float:right;width:230px;margin-left:-230px;}
.g-mn2{float:left;width:100%;}
.g-mn2c{margin-right:240px;}
```

3.两列定宽布局

图5-10 两列定宽布局

```
<div class="g-bd">
    <div class="g-mn" style="background: red;">
        <p>左侧定宽</p>
    </div>
    <div class="g-sd" style="background: pink;">
        <p>右侧定宽</p>
    </div>
</div>

/* 两列定宽布局 */
.g-bd{width:950px;margin:0 auto;}
.g-sd{float:right;width:230px;}
.g-mn{float:left;width:710px;}
```

4.三列中间自适应布局

图5-11 三列中间自适应布局

```
<div class="g-bd5">
    <div class="g-sd51" style="background: red;">
        <p>左侧定宽</p>
    </div>
    <div class="g-mn5">
        <div class="g-mn5c" style="background: yellow;">
            <p>中间自适应</p>
        </div>
    </div>
    <div class="g-sd52" style="background: greenyellow;">
        <p>右侧定宽</p>
    </div>
</div>

/* 三列中间自适应布局 */
.g-bd5{margin:0 0 10px;}
.g-sd51,.g-sd52{position:relative;float:left;width:230px;margin:0 -230px 0 0;}
.g-sd52{float:right;width:190px;margin:0 0 0 -190px;}
.g-mn5{float:left;width:100%;}
.g-mn5c{margin:0 200px 0 240px;}
```

5.三列左侧自适应布局

图5-12 三列左侧自适应布局

```
<div class="g-bd4">
    <div class="g-mn4" style="background: red;">
        <div class="g-mn4c">
            <p>左侧自适应</p>
        </div>
    </div>
    <div class="g-sd41" style="background: blue;">
        <p>右侧定宽</p>
    </div>
    <div class="g-sd42" style="background: green;">
        <p>右侧定宽</p>
    </div>
</div>

/* 三列左侧自适应布局 */
.g-bd4{margin:0 0 10px;}
.g-sd41,.g-sd42{position:relative;float:right;width:190px;}
.g-sd41{width:230px;margin-left:10px;}
.g-mn4{float:left;width:100%;margin-right:-430px;}
.g-mn4c{margin-right:440px;}
```

6.三列右侧自适应布局

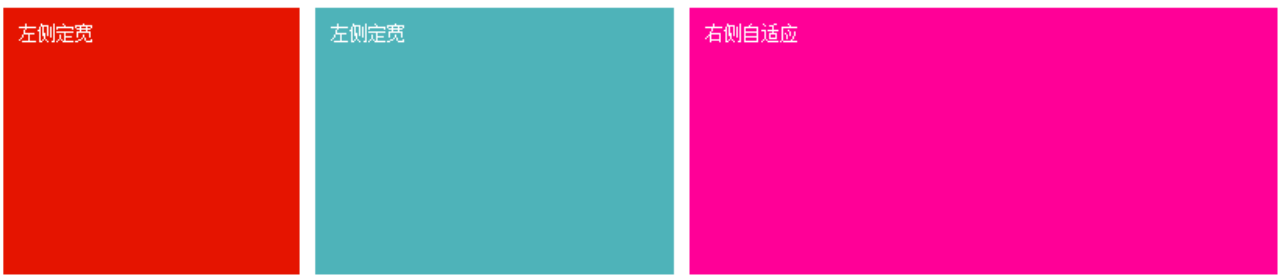

图5-13 三列右侧自适应布局

```
<div class="g-bd3">
    <div class="g-sd31" style="background: red;">
        <p>左侧定宽</p>
    </div>
    <div class="g-sd32" style="background: green;">
        <p>左侧定宽</p>
```

```
    </div>
    <div class="g-mn3">
        <div class="g-mn3c" style="background: blue;">
            <p>右侧自适应</p>
        </div>
    </div>
</div>

/* 三列右侧自适应布局 */
.g-bd3{margin:0 0 10px;}
.g-sd31,.g-sd32{position:relative;float:left;width:230px;}
.g-sd31{width:190px;margin-right:10px;}
.g-mn3{float:right;width:100%;margin-left:-430px;}
.g-mn3c{margin-left:440px;}
```

5.3 BFC&IFC

> 先说说FC，FC的含义就是Fomatting Context。它是CSS2.1规范中的一个概念。它是页面中的一块渲染区域，并且有一套渲染规则，它决定了其子元素将如何定位，以及和其他元素的关系和相互作用。BFC和IFC都是常见的FC。分别叫做Block Fomatting Context 和Inline Formatting Context。

5.3.1 BFC

> BFC（Block Formatting Context）叫做“块级格式化上下文”。

- BFC的布局规则如下：

  1.内部的盒子会在垂直方向，一个个地放置；

  2.盒子垂直方向的距离由margin决定，**属于同一个BFC的两个相邻Box的上下margin会发生重叠**；

  3.每个元素的左边，与包含的盒子的左边相接触，即使存在浮动也是如此；

  4.BFC的区域不会与float重叠；

  5.BFC就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素，反之也如此；

  6.计算BFC的高度时，浮动元素也参与计算。

- 哪些元素会产生BFC:

  1.根元素；

  2.float的属性不为none；

  3.position为absolute或fixed；

  4.display为inline-block，table-cell，table-caption，flex；

  5.overflow不为visible。

5.3.1.1 自适应两栏布局

先定义两个div：

```
<div class="aside"></div>
<div class="main"></div>
```

然后定义css：

```
div {
  width:300px;
}
.aside {
  width: 100px;
  height: 150px;
  float: left;
  background: black;
}
.main {
  height:200px;
  background-color:red;
}
```

效果图如下：

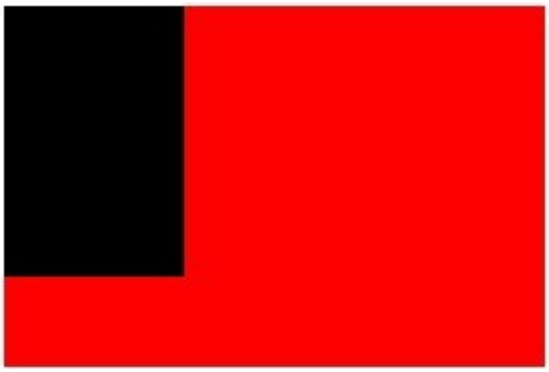

图5-14 自适应两栏布局1

这正满足了规范的第三条：

每个元素的左边，与包含的盒子的左边相接触，即使存在浮动也是如此。

所以如果我们需要将黑色区域撑到红色的左边，就需要利用规范的第四条：

BFC的区域不会与float重叠。

也就是说我们需要创造BFC区域。我们通过将红色区域的overflow设为hidden来触发BFC：

```
.main {
  overflow:hidden;
  height:200px;
  background-color:red;
}
```

效果如图：

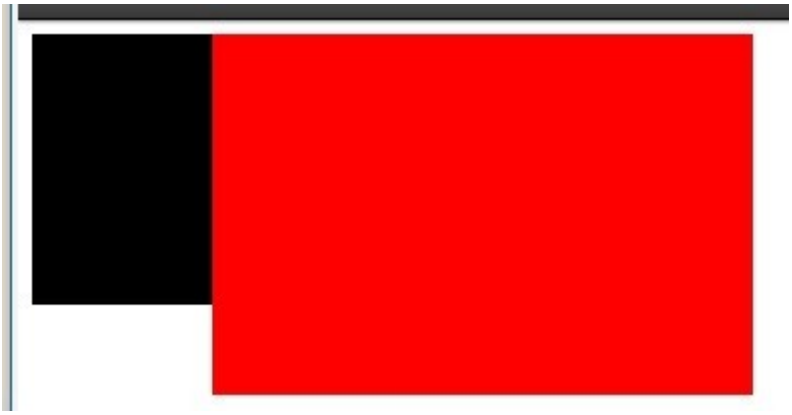

图5-15 自适应两栏布局2

接下来看看**清除内部浮动**

首先是父div套子div：

```
<div class="parent">
  <div class="child"></div>
</div>
```

然后是css：

```
.child {
  border:1px solid red;
  width:100px;
  height:100px;
  float:left;
}
.parent {
  border:1px solid black;
  width:300px;
}
```

效果如图：

图5-16 自适应两栏布局3

可以看到，父div压根就没有被撑开。

我们再回顾一下BFC规范中的第六条：

计算BFC的高度时，浮动元素也参与计算。

所以我们需要将父div触发为BFC，也就是将其overflow设为hidden：

```
.parent {
  border:1px solid black;
  width:300px;
  overflow:hidden;
}
```

效果如图：

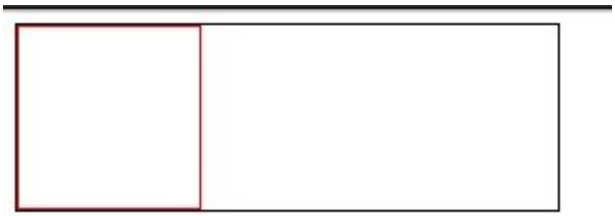

图5-17 自适应两栏布局4

可以看到父div已经撑开了。

5.3.1.2 margin重叠

先定义两个垂直的div：

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

然后定义margin:

```
.p {
  width:200px;
  height:50px;
  margin:50px 0;
  background-color:red;
}
```

可以看到margin重叠后的效果：

图5-18 margin重叠1

我们再看看BFC规范的第二条：

盒子垂直方向的距离由margin决定，属于用一个BFC的两个相邻Box的上下margin会发生重叠。

说明两者属于同一个BFC，所以我们需要两个div不属于同一个BFC。

为第二个div套一个父亲div，然后讲其overflow设为hidden来激活一个BFC就可以使margin不再重叠。

```
<div class="p"></div>
<div class="wrap">
  <div class="p"></div>
</div>
```

```
.wrap {
  overflow:hidden;
}
```

效果如下：

图5-19 margin重叠2

5.3.2 IFC

> IFC(Inline Formatting Contexts)意为“内联格式化上下文”,IFC中，盒子依次水平放置，从包含块的顶部开始。

- IFC布局规则：

  1.框会从包含块的顶部开始，一个接一个地水平摆放。

  2.摆放这些框的时候，它们在水平方向上的外边距、边框、内边距所占用的空间都会被考虑在内。在垂直方向上，这些框可能会以不同形式来对齐：它们可能会把底部或顶部对齐，也可能把其内部的文本基线对齐。能把在一行上的框都完全包含进去的一个矩形区域，被称为该行的行框。水平的margin、padding、border有效，垂直无效。不能指定宽高。

  3.行框的宽度是由包含块和存在的浮动来决定。行框的高度由行高计算这一章所描述的规则来决定。

- 结论：

1. 一个line box总是足够高对于包含在它内的所有盒子。然后，它也许比包含在它内最高的盒子高。(比如，盒子对齐导致基线提高了)。

2. 当盒子B的高度比包含它的line box的高度低，在line box内的B的垂值对齐线通过'vertical align'属性决定。当几个行内级盒子在一个单独的line box内不能很好的水平放置，则他们被分配成了2个或者更多的垂直重叠的line boxs.因此,一个段落是很多个line boxs的垂直叠加。Line boxs被叠加没有垂直方向上的分离(特殊情况除外)，并且他们也不重叠。
3. 通常，line box的左边缘挨着它的包含块的左边缘，右边缘挨着它的包含块的右边缘。然而，浮动盒子也许会在包含块边缘和line box边缘之间。因此，尽管line boxs在同样的行内格式上下文中通常都有相同的宽度(就是他的包含块的宽度)，但是水平方向上的空间因为浮动被减少了，它的宽度也会变得复杂。Line boxs在同样的行内格式上下文中通常在高度上是多样的(比如，一行也许包含了一个最高的图片然后其他的也可以仅仅只包含文字)
4. 当在一行中行内级盒子的总宽度比包含他们的line box的宽度小，他们的在line box中的水平放置位置由'text align'属性决定。如果属性是'justify'，用户代理可能会拉伸空间和文字在inline boxs内。
5. 当一个行内盒子超过了line box的宽度，则它被分割成几个盒子并且这些盒子被分配成几个横穿过的line boxs。如果一个行内盒子不能被分割。则行内盒子溢出line box。
6. 当一个行内盒子被分割，分割发生则margins,borders,和padding便没有了视觉效果。
7. 在同样的line box内的行内盒子也许会被分割成几个盒子因为双向的文字。
   Line boxs在行内格式上下文中档需要包含行内级内容时被创造。Line boxs包含没有文字，没有空格，没有带着margins,padding和borders，以及没有其他在流中的内容(比如图片，行内盒子和行内表格)，也不会以新起一行结尾。对于在他们内的任何盒子的位置都以他们决定并且必须将他们视作没有高度的line boxs。

主要印象IFC内布局的CSS：

font-size line-height height vertical-aligin

5.3.2.1 line box

> 行盒模型，这是一个显示区域，根据块状容器内，每一行的多个内联元素（inline-level element）都会共同生成一个行盒模型。

5.3.2.2 font-size

> 常见的属性，用来指定文本类型节点的大小。IFC内的很多属性的值是基于这个的。

5.3.2.3 line-height & height

> 在一个由多个内联元素组成的块状容器内，'line-height'为内联元素的行盒模型指定了一个最低高度。 这个最低高度是分别由基线上的最小高度和基线下的最小深度组成。
>
> 从上到下四条线分别是顶线、中线、基线、底线。 那么行高是指上下文本行的基线间的垂直距离，即图中两条红线间垂直距离（实际在数值上，行高也等于其它相同颜色间的距离）。

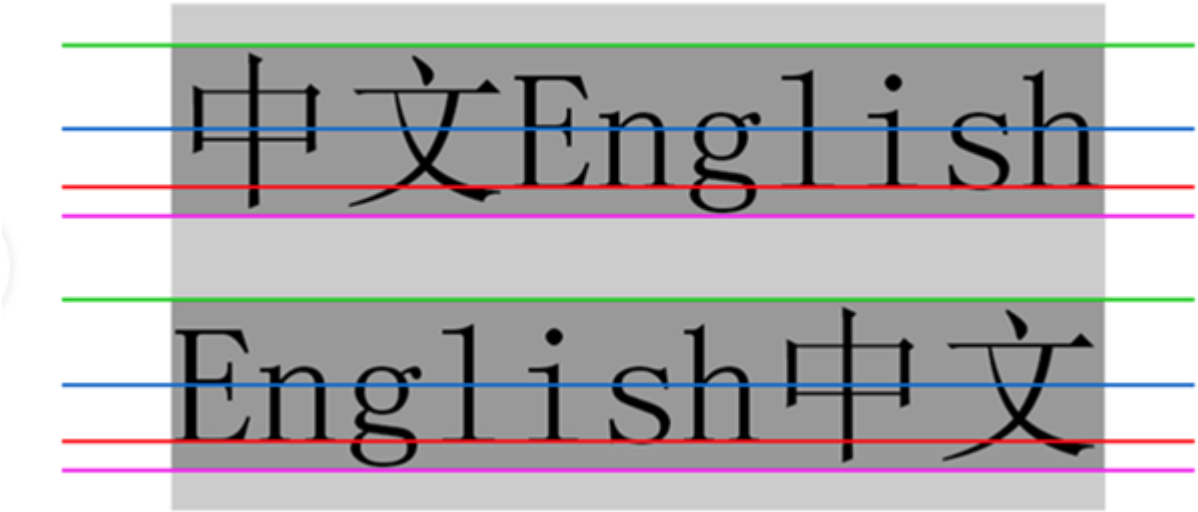

图5-20 line-height & height

5.3.2.4 vertical-align

> 该属性影响由多个内联元素生成的盒模型组成的行内盒模型的垂直定位。 vertical有几个特定的值，或者指定一个值。

```
<p class="a1">
  <span style="vertical-align:60px;">
    English中文
  </span>
  <span>
    中文English
  </span>
</p>
```

给第一个span，设置60px的垂直偏移量，显示如下：

其中，黄色线就是基线（baseline），绿线和黄线的间隔即为60px。 这里会发现，容器（蓝色）的高度被撑高了。

容器的高度 height = line-height + vertical-align

当然同理,如果容器的高度被指定了，那么高度则不变，而超出的部分则不影响布局。如果设置overflow:hidden，超过的部分则不可见。

而vertical-align的其它特殊值，均可以看做一个根据容器高度而变化的相对值。

5.4 课程总结

1. 定位
2. 布局
3. BFC&IFC

5.5 实训

设计如图5-21所示的效果。


招商银行
CHINA MERCHANTS BANK
一网通
ALL IN ONE NET
主页
个人业务
公司业务
小企业
信用卡
i理财
商旅预定
今日招行
金葵花理财 | 私人银行 | 出国金融 | 个人贷款 | 空中银行 | 一卡通 | 财富账户 | 伙伴一生 | 电子银行 | 居家生活 | 储蓄业务 | 投资理财 | 网上个人银行
生意贷
生意一卡通
特色创新功能
一手住房贷款
二手住房贷款
购房专享装修贷款
购房专享车位贷款
个人消费贷款
信用贷款
金葵花客户尊享贷款
金卡客户专享贷款
工资贷款
个人汽车贷款
商业用房贷款
个人留学贷款
全国个贷中心
按揭贷款月供计算器
个人消费贷款
适用客户
所有需要申请个人消费贷款的客户
购车、装修、旅游、留学.....各种用途任您选择！贷款金额最高可达2000万元！30年超长期限，全方位满足您各种消费需求！把您的房产变成提款机，尽情享用！
期限：授信期限最长可达30年
成数：最高7成
办理流程
距您成功贷款，只有三步！
第一步：提交申请
第二步：银行审批
第三步：提款消费
您需要准备的贷款申请资料
1.身份证、婚姻证明
2.房产证
3.住址证明【至少任选其一】：水、电、气、电话或物管等费用账单
4.收入证明【至少任选其一】：工资证明/银行流水/所得税税单/社保记录/其他收入证明
5.用途证明：提供相应的交易证明材料
如何找到招商银行个人贷款？
1.欢迎致电招商银行客户经理。
2.向就近招商银行网点提出申请。
3.拨打全国统一服务热线95555。
安全说明 | 网站声明 | 隐私保密条款 | 网站地图 | 友情链接 | 加入收藏夹 | 人才招聘 | 手机一网通

图5-21 实训效果图