千锋教育™
WWW.QFEDU.COM

第7章 CSS浮动与定位

- CSS浮动的原理
- CSS定位

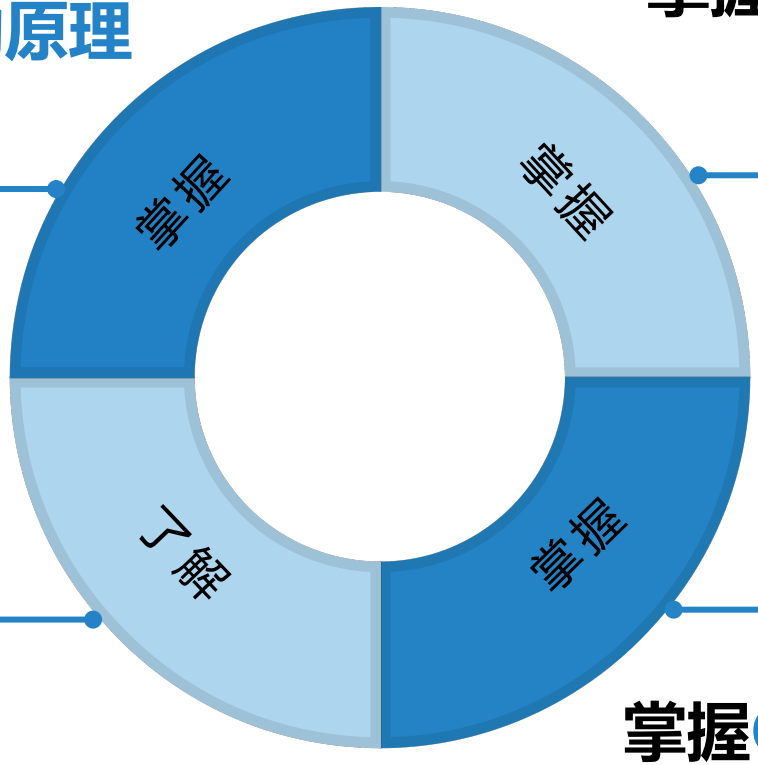
目录

7.1 浮动原理
☞点击查看本小节知识架构

7.2 CSS定位
☞点击查看本小节知识架构

学习目标

1. 掌握浮动的原理
2. 掌握清除浮动的处理
3. 掌握CSS定位的四种模式
4. 了解BFC概念



返回目录

7.1 浮动原理

7.1.1 脱离文档流

7.1.2 float属性

7.1.3 float的注意点

7.1.4 clear属性

7.1.5 清除嵌套中浮动

7.1 浮动原理

7.1.1 脱离文档流

- 文档流是元素在页面中出现的先后顺序。元素在没有任何CSS样式修饰的情况下，元素的排列方式就属于正常文档流，即窗体自上而下分成一行行，并在每行中按从左到右的顺序排放元素。
- 所谓的脱离文档流就是利用CSS样式使元素在HTML结构中的顺序和展示出来的顺序不一致。

7.1.2 float属性

- CSS中利用float属性来设置浮动操作，当被设置成浮动的元素时，会按照一个指定的方向移动，直到到达父容器的边界或者另外一个浮动元素浮动停止，其值有none、left和right三个。下面介绍不同取值和含义。
- 1.none
- none值为默认值，表示不进行浮动操作，元素处于正常文档流。
- 2. left
- left值表示对元素进行左浮动，元素会沿着父容器靠左排列且脱离文档流。
- 3. right
- right值表示对元素进行右浮动，元素会沿着父容器靠右排列且脱离文档流。

7.1 浮动原理

7.1.2 float属性

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>浮动原理</title>
6  <style>
7       #box1{ width:150px; height:150px; background:red; }
8       #box2{ width:50px; height:50px; background:green; float:right; }
9       #box3{ width:80px; height:80px; background:blue; }
10 </style>
11 </head>
12 <body>
13 <div id="box1">
14      <div id="box2"></div>
15     <div id="box3"></div>
16 </div>
17 </body>
18 </body>
19 </html>
```

- box2沿着它的父容器box1靠右侧进行排列。
- 当给box2和box3都添加浮动时，它们都会脱离文档流。它们处于同一平台，所以会紧挨在一起，从而实现左右排列的布局方案。可以同时靠左、靠右或是左右分开排列。

7.1 浮动原理

7.1.2 float属性

- （1）靠左排列
- box2和box3的float属性同时设置为靠左排列，两个元素会紧挨在一起，排列在左侧。
- （2）靠右排列
- box2和box3的float属性同时设置为靠右排列，两个元素会紧挨在一起，排列在右侧。
- （3）左右分开排列
- box2设置为左浮动，box3设置为右浮动，两个元素会左右分开排列。

7.1.3 float的注意点

- CSS中的float属性比较复杂，有很多需要注意的点，下面依次进行讲解。
- 1.只会影响后面的元素
- float属性只会影响到后面元素的布局，而对之前的元素不会造成任何的影响。如为box3元素添加浮动，不会影响到前面的box2元素。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>浮动原理</title>
<style>
      #box1{ width:150px; height:150px; background:red; }
      #box2{ width:50px; height:50px; background:gray;  }
      #box3{ width:80px; height:80px; background:blue; float:right; }
</style>
</head>
<body>
<div id="box1">
     <div id="box2"></div>
    <div id="box3"></div>
</div>
</body>
</body>
</html>
```

7.1.3 float的注意点

- 2.内容默认提升半层
- 正常文档流位于页面的底层，而脱离文档流位于页面的上层，那么当有内容存在时，内容默认在底层与上层之间的位置。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>浮动原理</title>
<style>
      #box1{ width:150px; height:150px; background:red; }
      #box2{ width:50px; height:50px; background:green; float:left;}
      #box3{ width:80px; height:80px; background:blue; color:white; }
</style>
</head>
<body>
<div id="box1">
      <div id="box2"></div>
    <div id="box3">内容</div>
</div>
</body>
</body>
</html>
```

- 内容并没有在浮动元素的下面，而是在外面，其实这就是内容默认会提升半层。可以利用这个特点，来实现图文混排的效果。

7.1.3 float的注意点

- 3.默认宽根据内容确定
- 当未给浮动的元素设置宽度时，浮动元素的宽与内容的宽相同。浮动的元素改变了块标签的特点。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>浮动原理</title>
<style>
      #box1{ width:300px; height:100px; background:red; }
      #box2{ background:green; float:left; color:white; }
</style>
</head>
<body>
<div id="box1">
      <div id="box2">这是一个浮动的元素</div>
</div>
</body>
</body>
</html>
```

7.1.3 float的注意点

- 4.换行排列
- 当多个元素设置浮动时，它们会水平排列，但是如果父容器放不下这些浮动元素时，会自动换行进行排列。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>浮动原理</title>
<style>
      #box1{ width:120px; height:150px; background:red; }
      #box2{ width:50px; height:50px; background:gray; float:left; }
      #box3{ width:80px; height:80px; background:blue; float:left; }
</style>
</head>
<body>
<div id="box1">
      <div id="box2"></div>
    <div id="box3"></div>
</div>
</body>
</body>
</html>
```

7.1.4 clear属性

- 有时不希望浮动的元素影响到后面元素的布局，就可以给后面的元素添加清除浮动的操作。在CSS样式中通过clear属性，来设置清除浮动的操作。其有left、right和both三个属性值。
- 1.left
- left值用来清除左浮动。这里要注意clear属性为left值时，只能清除之前元素的左浮动，对右浮动不起作用。
- 2.right
- right值是用来清除右浮动的。

7.1.4 clear属性

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>浮动原理</title>
6  <style>
7       #box1{ width:150px; height:150px; background:red; }
8       #box2{ width:50px; height:50px; background:gray; float:right; }
9       #box3{ width:80px; height:80px; background:blue; clear:right; }
10 </style>
11 </head>
12 <body>
13 <div id="box1">
14      <div id="box2"></div>
15     <div id="box3"></div>
16 </div>
17 </body>
18 </body>
19 </html>
```

- 3.both
- both值的作用是左右浮动一起清除，一般情况下，都采用both值，这样就不用担心之前元素究竟设置了左浮动还是右浮动。

7.1.5 清除嵌套中浮动

- 元素嵌套中，如果父元素不设置高度，而子元素box2添加浮动，则子元素浮动使其脱离文档流，导致子元素和父元素不在同一个层面上，这时父元素内没有任何内容，无法撑开。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>嵌套中的清除浮动</title>
<style>
      #box1{ width:300px; border:1px black solid; }
      #box2{ width:100px; height:100px; background:red; float:left;}
</style>
</head>
<body>
<div id="box1">
      <div id="box2"></div>
</div>
</body>
</body>
</html>
```

- 例中可以看出子元素无法撑开父元素的容器。这会影响到后面元素的布局，想要在子元素浮动的情况下，使父元素保持原有位置大小，就涉及到清除嵌套中浮动的操作，下面就将讲解在嵌套中清除浮动的方法。

7.1.5 清除嵌套中浮动

- 1. 父元素固定宽高
- 通过给父元素设定固定的高度方式清除嵌套中浮动，这种方式其实把父元素高度固定，进而对容器大小进行了限制，这样对于内容的扩展不是很方便，因此这种方式是不建议使用。在上述案例的基础上，为父元素固定宽高，具体代码如下。

```
#box1{ width:300px; height:100px; border:1px black solid; }
#box2{ width:100px; height:100px; background:red; float:left; }
```

- 2. 父元素浮动
- 通过给父元素设置浮动操作方式清除嵌套中浮动，这种方式可以保证子元素与父元素处于同一平面上，父元素可以被撑开，但这种方式存在父元素浮动会影响后面元素布局的问题，因此也不推荐使用。在上述案例的基础上，为父元素设置浮动，具体CSS代码如下。

```
#box1{ width:300px; border:1px black solid; float:left; }
#box2{ width:100px; height:100px; background:red; float:left; }
```

7.1.5 清除嵌套中浮动

- 3. 父元素overflow属性
- 通过将父元素overflow属性值设置为hidden或scroll方式清除嵌套中浮动，但overflow会对溢出的元素进行隐藏或添加滚动条，因此也是不推荐使用的方式。在上述案例基础上，将父元素overflow属性值设置为hidden，具体CSS代码如下。

```
#box1{ width:300px; border:1px black solid; overflow:hidden; }
#box2{ width:100px; height:100px; background:red; float:left; }
```

- 4. 父元素设置display属性
- 通过将父元素设置display属性值设置为inline-block方式清除嵌套中浮动。但inline-block值会使元素具备块和内联的特点，对布局有影响，因此也是不推荐使用的方式。在上述案例基础上，将父元素display属性值设置为inline-block，具体CSS代码如下。

```
#box1{ width:300px; border:1px black solid; display:inline-block; }
#box2{ width:100px; height:100px; background:red; float:left; }
```

7.1 浮动原理

7.1.5 清除嵌套中浮动

- 5. 设置空标签
- 通过在父元素中添加一个空标签清除嵌套中浮动，利用这个空标签来撑开父元素。

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

- 给空标签添加清除浮动操作，使得空标签保持在正常位置上，同时空标签和父元素在同一个层面上，因此父元素被空标签所撑开。这种方式非常巧妙，但需要多添加一个标签元素，也不是很方便。因此不建议在实际工作中使用。
- 6. after伪类
- after伪类方式是优化空标签方式的一种做法，也是目前最流行的处理方式。下面先来了解after伪类的用法。
- 它可以通过CSS方式给HTML标签添加内容，内容会被添加到HTML标签内的最后位置。

7.1 浮动原理

7.1.5 清除嵌套中浮动

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>嵌套中的清除浮动</title>
6  <style>
7       #box:after{ content : "_css"; }
8  </style>
9  </head>
10 <body>
11 <div id="box">hello</div>
12 </body>
13 </body>
14 </html>
```

- after伪类通过content属性来添加内容，并且把内容加到了hello内容的后面。
- 通过after伪类的方式在box1中添加一个空内容，相当于添加一个空标签，默认是内联特点，因此通过将display属性值设置为block转化成块的特点，然后就可以通过清除浮动的方式，使得父元素被撑开。
- 与after伪类类似的还有一个before伪类，它是把内容添加到hello内容的前面。
- 清除嵌套结构的浮动方法，after是最优化的一种方式，推荐使用。

7.1.5 清除嵌套中浮动

- 7. BFC模式
- BFC是“Block fomatting context”的缩写，即“块级格式化上下文”的意思。当创建了BFC的元素就会形成一个独立的盒子。盒子里的布局不受外部影响，当然也不会影响到外面的元素。下面列举了几个能够触发BFC模式的属性。
- float的值为left或right
- overflow的值为hidden或scroll
- display的值为inline-block
- position的值为absolute或fixed

7.1.5 清除嵌套中浮动

- position为定位属性，会在下一节中详细讲解。可以发现，overflow、display等属性可以清除浮动是因为它们不会影响周围的布局，而成为独立的盒子。前面章节中讲过的盒子模型中margin传递的问题，也可以通过设置BFC模式来解决。

7.2 CSS定位

返回目录

7.2.1 定位属性

7.2.2 相对定位

7.2.3 绝对定位

7.2.4 固定定位

7.2.5 定位的层级

7.2 CSS定位

- 在CSS中，通过CSS定位（CSS position）可以实现网页元素的精确定位。CSS定位和CSS浮动类似，也是控制网页布局的操作，CSS定位更加灵活，可以针对更多个性化的布局方案来使用。在网页布局实战中，灵活使用这两种布局方式，能够创建多种高级而精确的布局。本节将对元素的定位属性及常用的几种定位方式进行详细地讲解。

7.2 CSS定位

7.2.1 定位属性

- 制作网页时，CSS可以使用定位属性将一个元素精确地放在页面上指定位置。元素的定位属性由定位模式和位置属性两部分构成。
- 1. 定位模式
- 在CSS中，position属性用来定义元素的定位模式，其常用的属性值有四个，分别表示不同的定位模式，如表所示。

□□□□	□□□
static	□□□□ □□□□□□ □
relative	□□□□ □□□□□□□□□□□□□□□□
absolute	□□□□ □□□□□□□□□□□□□□□□□□□
fixed	□□□□ □□□□□□□□□□□□□

7.2 CSS定位

7.2.1 定位属性

- 表中静态定位就是默认的方式，当position属性值为static时，可以将元素定位于静态位置。静态位置就是各个元素在HTML文档流中默认的位置。
- 在默认状态下任何元素都会以静态定位来确定位置。因此，当元素未设置position属性时，会遵循默认值显示为静态位置。
- 2.位置属性
- 定位模式仅仅定义了元素的定位方式，并不能确定元素的具体位置。在CSS中，位置属性用来精确定义定位元素的位置，其取值为不同单位的数值或百分比，定位属性包括top、bottom、left和right，其具体含义如表所示。

□□□□□□	□□□
top	□□□□□□
bottom	□□□□□□
left	□□□□□□
right	□□□□□□

7.2.2 相对定位

- 相对定位是元素相对于它在原文档流中的位置进行定位，position属性的取值为relative。
- 当给元素只添加relative值时，对元素本身并没有任何影响，只是设置其相对定位，因此还需要通过定位属性改变元素的位置，但它在文档流中的位置仍然保留。
- 相对定位是相对于元素本身的左上角进行偏移操作的。相对定位的偏移并没有影响到其他元素的位置。
- 需要注意一点，定位模式和位置属性是配合在一起使用的，如果只定义一种，则对元素不起任何作用。

7.2.3 绝对定位

- 绝对定位是元素相对于已经定位（相对、绝对或固定定位）的父元素进行定位。若所有父元素都没有定位，则依据浏览器窗口左上角进行定位。当position属性值为absolute时，可以将元素的定位模式设置为绝对定位。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CSS 定位</title>
<style>
      #box1{ width:50px; height:50px; background:red; }
      #box2{ width:50px; height:50px; background:green;
         position:absolute; }
      #box3{ width:50px; height:50px; background:blue; }
</style>
</head>
<body>
<div id="box1"></div>
<div id="box2"></div>
<div id="box3"></div>
</body>
</html>
```

- box2叠加到box3的上面，这说明绝对定位会脱离文档流。例中box2没有父元素，因此其定位根据浏览器窗口左上角进行偏移。

7.2.3 绝对定位

- （1）与浮动类似，块元素添加绝对定位后，默认宽与内容的宽相同。
- （2）嵌套结构中的绝对定位。当父元素或祖先元素中有相对定位或绝对定位时，子元素的绝对定位将相对于父元素或祖先元素进行定位。当父元素或祖先元素都没有定位属性时，子元素将相对于浏览器窗口进行偏移。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CSS 定位</title>
<style>
      #box1{ width:50px; height:50px; background:red;
         margin-left:50px; position:relative; }
      #box2{ width:50px; height:50px; background:green;
         position:absolute; top:50px; left:50px; }
</style>
</head>
<body>
<div id="box1">
     <div id="box2"></div>
</div>
</body>
</body>
</html>
```

7.2.4 固定定位

- 固定定位通过将position属性值设置为fixed来实现，固定定位与绝对定位类似，也是脱离文档流。二者的不同点是当元素的position属性设置为fixed时，元素将被固定，即不会随着滚动条的拖动而改变位置。在视野中，固定定位的元素的位置不会改变。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CSS 定位</title>
<style>
      body{ height:1000px;}
      #box1{ width:50px; height:50px; background:red; }
      #box2{ width:50px; height:50px; background:green;
         position:fixed; top:50px; left:50px; }
      #box3{ width:50px; height:50px; background:blue; }
</style>
</head>
<body>
<div id="box1"></div>
<div id="box2"></div>
<div id="box3"></div>
</body>
</body>
</html>
```

7.2.4 固定定位

- 固定定位跟绝对定位还有一个不同点是固定定位永远都是相对浏览器窗口左上角进行偏移，网页中的回到顶部按钮就是用固定定位实现的。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CSS 定位</title>
<style>
      body{ height:1000px;}
      #top{ width:35px; background:#bbbbbb; color:white;
          font-size:12px; text-align:center;
          letter-spacing:1px; line-height:16px;
          position:fixed; bottom:4px; right:4px; }
</style>
</head>
<body>
<div id="top">回到顶部</div>
</body>
</body>
</html>
```

7.2.5 定位的层级

- 当多个元素添加定位操作时，可能会出现叠加情况，即在默认情况下后输出的HTML结构层级就会越高。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CSS 定位</title>
<style>
      #box1{ width:50px; height:50px; background:red;
         position:absolute; left:0; top:0;  }
      #box2{ width:50px; height:50px; background:green;
         position:absolute; left:25px; top:25px; }
</style>
</head>
<body>
<div id="box1"></div>
<div id="box2"></div>
</body>
</body>
</html>
```

7.2.5 定位的层级

- 定位层级与定位属性配套使用，用于调节层级的z-index属性，其属性值用数字表示，数字越大，层级越高。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CSS 定位</title>
<style>
      #box1{ width:50px; height:50px; background:red;
         position:absolute; left:0; top:0; z-index:2;  }
      #box2{ width:50px; height:50px; background:green;
        position:absolute; left:25px; top:25px; z-index:1; }
</style>
</head>
<body>
<div id="box1"></div>
<div id="box2"></div>
</body>
</body>
</html>
```

本章小结

- 本章首先介绍了CSS中元素的浮动、浮动的方向及所呈现的效果，以及清除浮动的方法，然后讲解了CSS中元素的定位及几种常见的定位模式。通过本章的学习，能够掌握浮动的使用方法和以定位的方式对网页进行布局的操作，以及能够掌握清除浮动的几种常用方法，为后续学习网页布局相关知识打下基础。

千锋教育™
WWW.QFEDU.COM

THANK YOU