盒模型：

Margin:

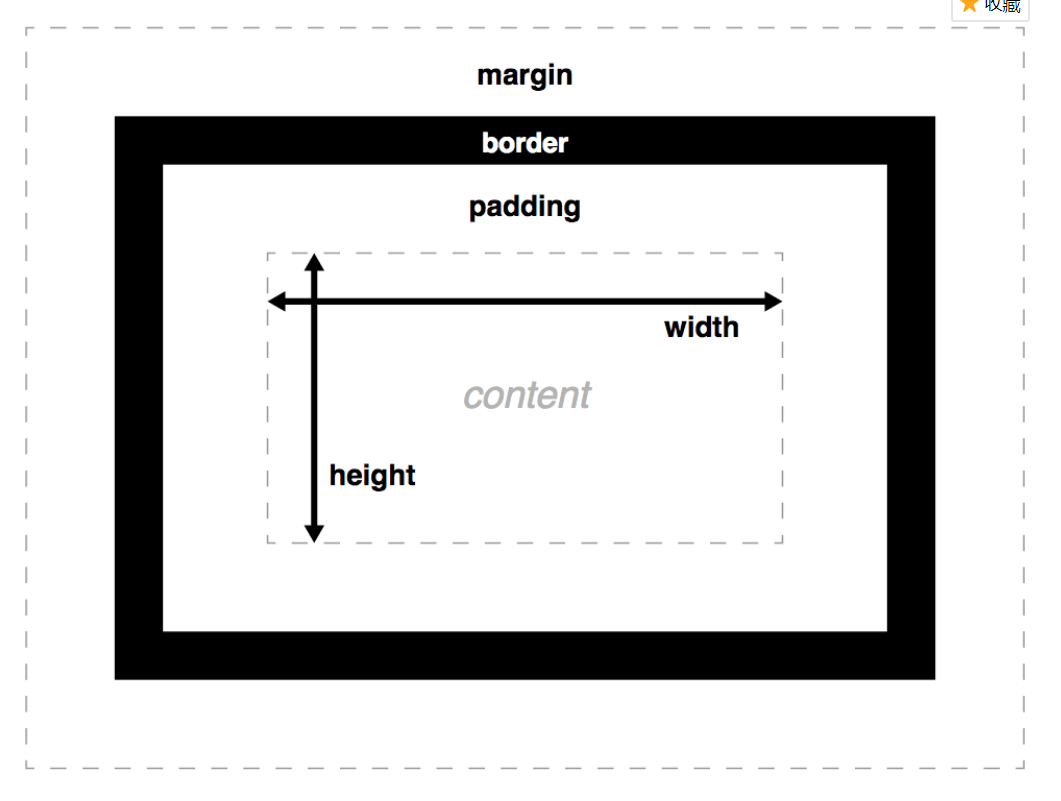
1. 可以改变容器的尺寸。
2. 元素尺寸：可视尺寸和占据尺寸。
3. Margin与可视尺寸：a.适用于没有设定width/height的普通block水平元素；b.只适用于水平方向尺寸。
4. 如何利用margin上述的特性？

一侧定宽的自适应布局：图片左浮动，跟随文字自然环绕效果。给<p>元素增加margin-left，可视尺寸减小，实现自适应效果。



1. Margin与占据尺寸：a.block/inline-block水平元素均适用；b.与有没有设定width/height值无关；c.适用于水平方向和垂直方向。
2. 如何利用margin上述的特性？

实现滚动容器内上下留白。



其他知识点：

1. 默认情况下background-color/background-image 延伸到了border的内边缘。该行为可以使用background-clip 属性来改变。

**background-clip  设置元素的背景（背景图片或颜色）是否延伸到边框下面。**

如果没有设置背景颜色或图片，那么这个属性只有在边框（border）设置为透明或半透明时才能看到视觉效果（查看 [border-style](https://developer.mozilla.org/zh-CN/docs/Web/CSS/border-style) 或 [border-image](https://developer.mozilla.org/zh-CN/docs/Web/CSS/border-image)），不然的话，这个属性造成的样式变化会被边框覆盖住。

**语法**

background-clip: border-box

background-clip: padding-box

background-clip: content-box

background-clip: inherit

值

**border-box**

背景延伸到边框外沿（但是在边框之下）。

**padding-box**

边框下面没有背景，即背景延伸到内边距外沿。

**content-box**

背景裁剪到内容区 (content-box) 外沿。

**text**

背景被裁剪为文字的前景色(只有chrome支持)。

1. 如果content框变得比示例输出窗口大，它将从窗口溢出，此时会出现滚动条，你可以滚动窗口来查看盒子剩余的内容 。可以使用overflow属性来控制溢出。

overflow 属性指定当内容溢出其块级容器时,是否剪辑内容，显示滚动条或显示溢出的内容。

**语法**

值

visible

默认值。内容不会被修剪，会呈现在元素框之外。

hidden

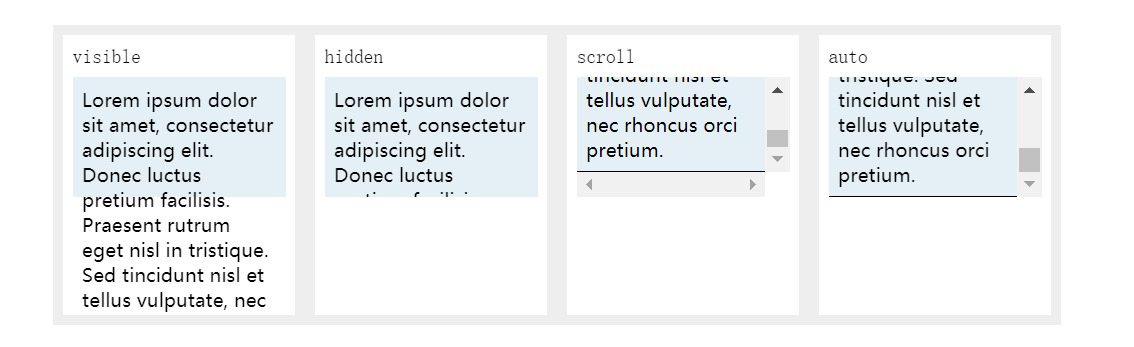
内容会被修剪，并且其余内容是不可见的。

scroll

内容会被修剪，并且浏览器会使用滚动条，无论内容是否会被裁减。这避免了在动态环境中滚动条的出现和消失问题。打印机会打印溢出的内容。

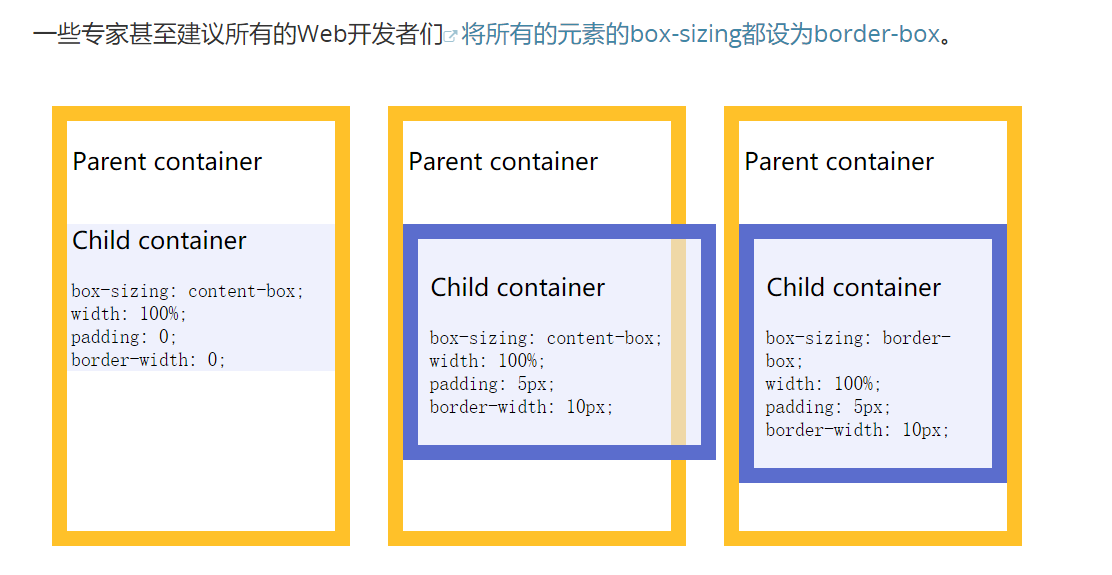
auto

取决于用户代理。浏览器，例如火狐，会在内容溢出时提供滚动条。



1. 框的高度不遵守百分比的长度；框的高度总是采用框内容的高度，除非指定一个绝对的高度（如：px 或者em），它会比在页面上默认是100%高度更实用。
2. border也忽略百分比宽度设置。
3. 框的总宽度是width, padding-right, padding-left, border-right, 以及 border-left 属性之和。

box-sizing 属性用于更改用于计算元素宽度和高度的默认的 CSS 盒子模型。



**值**

content-box

默认值，标准盒子模型。 width 与 height 只包括内容的宽和高， 不包括边框（border），内边距（padding），外边距（margin）。注意: 内边距, 边框 & 外边距 都在这个盒子的外部。 比如. 如果 .box {width: 350px}; 而且 {border: 10px solid black;} 那么在浏览器中的渲染的实际宽度将是370px;

尺寸计算公式：width = 内容的宽度，height = 内容的高度。宽度和高度都不包含内容的边框（border）和内边距（padding）。

border-box

width 和 height 属性包括内容，内边距和边框，但不包括外边距。这是当文档处于 Quirks模式 时Internet Explorer使用的盒模型。注意，填充和边框将在盒子内 , 例如, .box {width: 350px; border: 10px solid black;} 导致在浏览器中呈现的宽度为350px的盒子。内容框不能为负，并且被分配到0，使得不可能使用border-box使元素消失。

这里的维度计算为：

width = border + padding + 内容的 width，

height = border + padding + 内容的 height。

1. 元素的盒子类型是由 display 属性指定的。

display类型：

|  |  |  |
| --- | --- | --- |
| **类型** | **定义** | **特点** |
| none | 将元素设置为none的时候既不会占据空间，也无法显示，相当于该元素不存在。 | 该属性可以用来改善重排与重绘，同时我也经常用它来做模态窗等效果。 |
| block | 盒子之前以及之后的内容出现在不同的行上 | 可以设置高度和宽度。  可以将一个行内元素转换成块级元素。 |
| inline | 它会与周围的文本和其它行内元素出现在同一行，并且其内容会像段落中的文本行一样，随着文本流换行 | 主要用来设置行内元素属性，设置了该属性之后设置高度、宽度都无效，同时text-align属性设置也无效，但是设置了line-height会让inline元素居中。可以将一个块级元素转换成行内元素。 |
| inline-block | 介于前两者之间 | 设置了inline-block属性的元素既具有block元素可以设置width和height属性的特性，又保持了inline元素不换行的特性。 |

网页布局：又称版式布局，是网页UI设计师将有限的视觉元素进行有机的排列组合，将理性的思维个性化的表现出来，是一种具有个人艺术特色的视觉传达方式。传达信息的同时有美感。

网页设计的特点（相对纸媒来说）：

1. 网页可以自适应宽度(百分比自适应宽度）。
2. 网页的长度理论上可以无限延长（高度自适应）

分栏布局（分列）：一般是头部，主体，底部。主体一般再根据需要分栏，一般两栏或者三栏，还有可能分为更多的栏目。

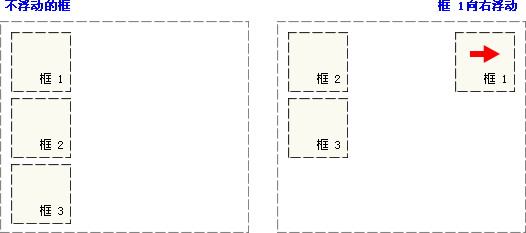
常见的分栏方式：一列，两列，三列和混合布局。

**Css浮动**（来自http://www.w3school.com.cn/css/css\_positioning\_floating.asp）

浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。

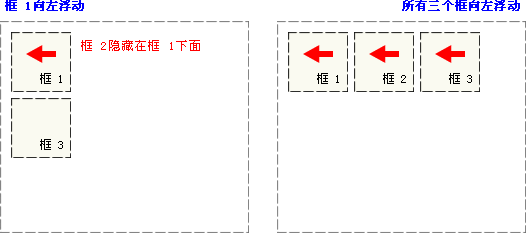
由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。

请看下图，当把框 1 向右浮动时，它脱离文档流并且向右移动，直到它的右边缘碰到包含框的右边缘：

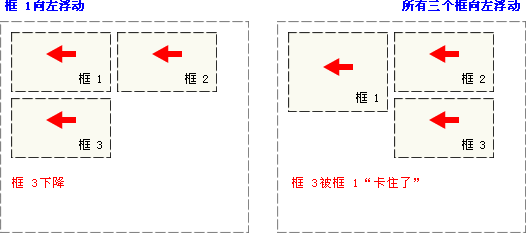


再请看下图，当框 1 向左浮动时，它脱离文档流并且向左移动，直到它的左边缘碰到包含框的左边缘。因为它不再处于文档流中，所以它不占据空间，实际上覆盖住了框 2，使框 2 从视图中消失。

如果把所有三个框都向左移动，那么框 1 向左浮动直到碰到包含框，另外两个框向左浮动直到碰到前一个浮动框。

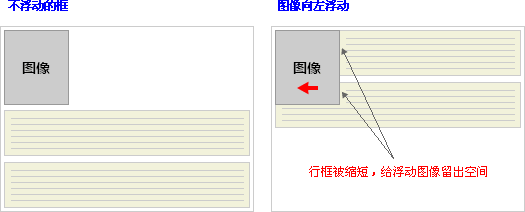


如下图所示，如果包含框太窄，无法容纳水平排列的三个浮动元素，那么其它浮动块向下移动，直到有足够的空间。如果浮动元素的高度不同，那么当它们向下移动时可能被其它浮动元素“卡住”：



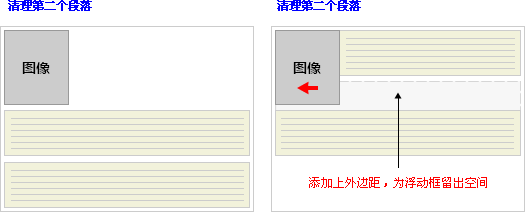
**行框和清理**

浮动框旁边的行框被缩短，从而给浮动框留出空间，行框围绕浮动框。因此，创建浮动框可以使文本围绕图像：



要想阻止行框围绕浮动框，需要对该框应用 clear 属性。clear 属性的值可以是 left、right、both 或 none，它表示框的哪些边不应该挨着浮动框。

为了实现这种效果，在被清理的元素的上外边距上添加足够的空间，使元素的顶边缘垂直下降到浮动框下面：



这是一个有用的工具，它让周围的元素为浮动元素留出空间。

让我们更详细地看看浮动和清理。假设希望让一个图片浮动到文本块的左边，并且希望这幅图片和文本包含在另一个具有背景颜色和边框的元素中。您可能编写下面的代码：

.news {

background-color: gray;

border: solid 1px black;

}

.news img {

float: left;

}

.news p {

float: right;

}

<div class="news">

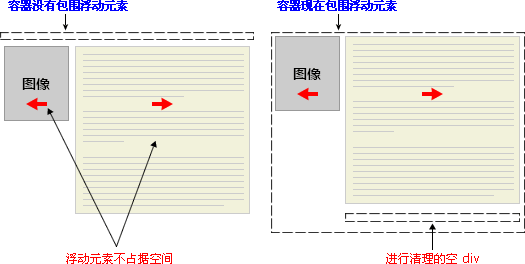
<img src="news-pic.jpg" />

<p>some text</p>

</div>

这种情况下，出现了一个问题。因为浮动元素脱离了文档流，所以包围图片和文本的 div 不占据空间。

如何让包围元素在视觉上包围浮动元素呢？需要在这个元素中的某个地方应用 clear：



不幸的是出现了一个新的问题，由于没有现有的元素可以应用清理，所以我们只能添加一个空元素并且清理它。

.news {

background-color: gray;

border: solid 1px black;

}

.news img {

float: left;

}

.news p {

float: right;

}

.clear {

clear: both;

}

<div class="news">

<img src="news-pic.jpg" />

<p>some text</p>

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

</div>

这样可以实现我们希望的效果，但是需要添加多余的代码。常常有元素可以应用 clear，但是有时候不得不为了进行布局而添加无意义的标记。

不过我们还有另一种办法，那就是对容器 div 进行浮动：

.news {

background-color: gray;

border: solid 1px black;

float: left;

}

.news img {

float: left;

}

.news p {

float: right;

}

<div class="news">

<img src="news-pic.jpg" />

<p>some text</p>

</div>

这样会得到我们希望的效果。不幸的是，下一个元素会受到这个浮动元素的影响。为了解决这个问题，有些人选择对布局中的所有东西进行浮动，然后使用适当的有意义的元素（常常是站点的页脚）对这些浮动进行清理。这有助于减少或消除不必要的标记。

**清除浮动两大基本方法：**

1. **Clear**

脚底加入clear：both；

两种形式：①HTML block水平元素底部添加<div class=”clear”></div>

② css after伪元素底部生成 - .clearfix:after{ }

更好的方法：.clearfix:after{content:‘’；display:table; clear:both; }

.clearfix:after{\*zoom:1 }

1. **父元素BFC/haslayout**

float：left/right

position：absolute/fixed

overflow:hidden/scroll (IE7+)

display:inline-block/table-cell (IE8+)

width/height/zoom:1/…(IE6/IE7)