

CSS盒模型与布局

窦连军 @八月虎baidu

北京乐美无限科技有限公司

CSS 布局基础

三个核心概念：

1. *CSS 盒模型*
2. *漂移 (Floating)*
3. *定位 (Positioning)*

上述三个概念决定着一个页面元素在屏幕上的具体位置。

来自日常生活中的智慧：盒模型



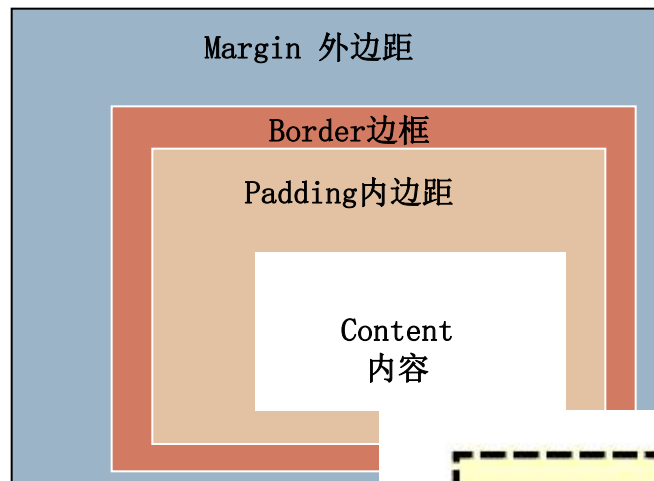
盒模型：外边距、边框、内边距、内容区



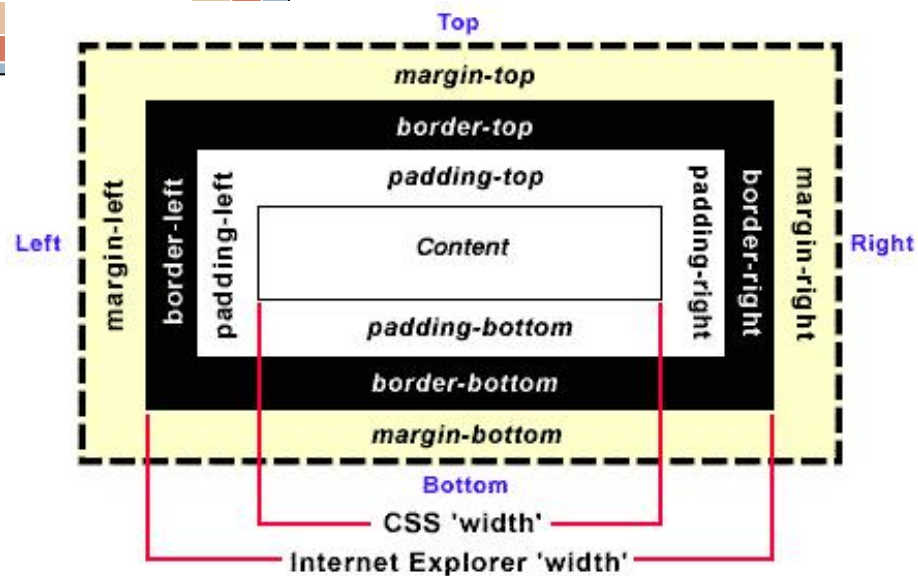
CSS盒模型

每个CSS盒子都划分为如下几个部分：

1. 外边距
2. 边框
3. 内边距
4. 内容区



margin, border 和 padding 属性都有四条边，每条边有对应的CSS属性，可以为其单独指定数值。



CSS盒模型

CSS中每个“块”元素都在一个“盒子”中，有外边距、内边距和边框，具体内容显示在内边距包裹的内容区中。

盒的宽度可以通过绝对单位（如：像素）来指定，也可以使用相对单位来指定：

px - 像素

em - 与字体相关，字体大小的倍数

percentage（百分比） - 宽度与盒所在的外部容器内容区有关

根元素是浏览器窗口。

盒模型各组成部分的作用

- 所有页面上的内容都包裹在一个“盒子”中。“盒子”由下面几部分构成：

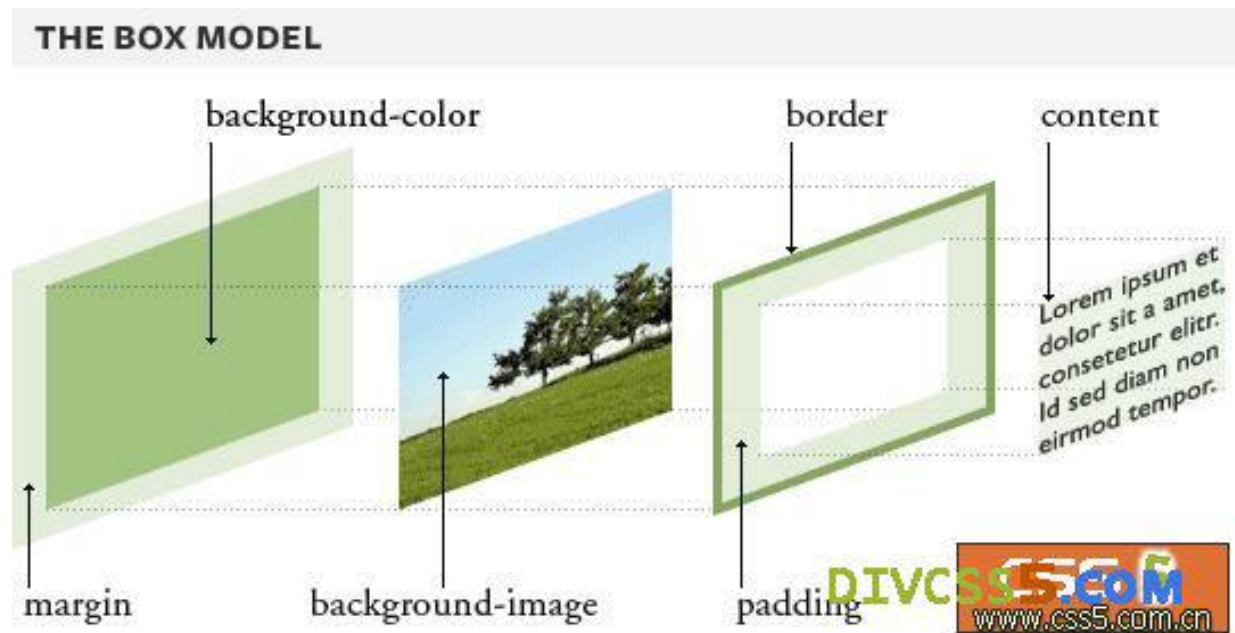
- 核心内容

- 核心内容之外围绕着内边距

- 内边距之外围绕着边框，边框勾画出了可视区域的轮廓

- 边框之外围绕着外边距，起到隔离区的作用，以将自己的盒子与其他盒子分开。

盒模型：透视图



即使有背景图，最好也同时指明背景色。

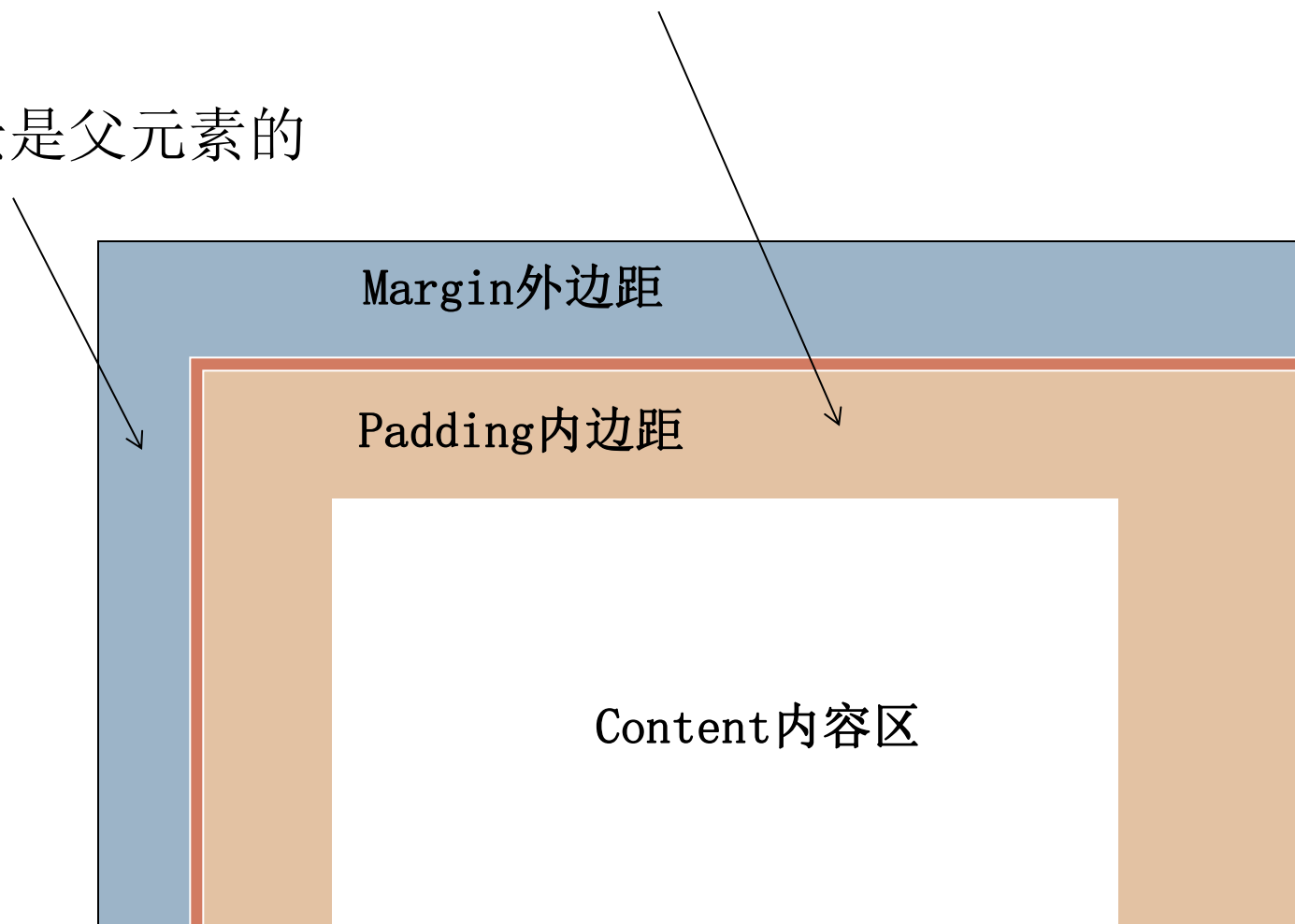
背景色位于外边距之内，覆盖范围包括边框线。

背景图以内边距左上角为坐标原点。但平铺和偏移时，可以延伸到边框线区。

外边距和内边距

内边距的背景是自己的（缺省透明的）

外边距的背景是父元素的



外边距和内边距

外边距

外边距定义了环绕在边框外侧的区域大小。

外边距属性可以为负值，用以强制实现两个区域的相互覆盖。

外边距属性可以影响其在父盒子中的位置。

外边距可以单独定义四个边，也可以使用快速定义方式。



外边距和内边距

内边距

内边距定义了边框和元素内容区之间的环绕带大小。

内边距不能是负值。

内边距影响内容区在边框线内的位置。

内边距可以单独定义四个边，也可以使用快速定义方式。

外边距和内边距: 外边距折叠

当2个或多个垂直关系的外边距在一起时，他们将叠加形成一个外边距。

这个新形成的外边距，其高度为2个外边距高度的最大值。

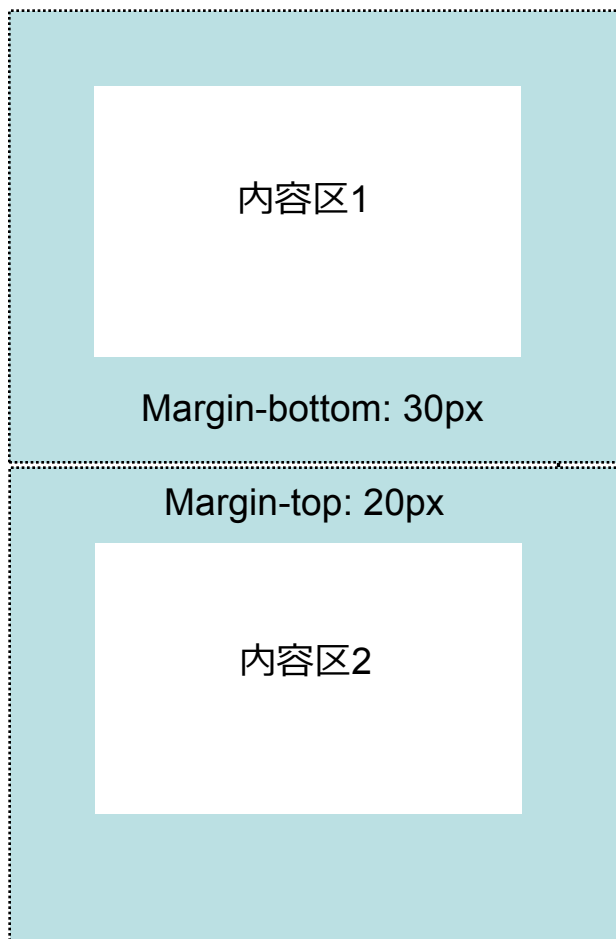
外边距叠加出现的场合：

2个或多个块元素，一个放在另一个之上。

或者，当一个块元素包裹在另一个块元素之内时。

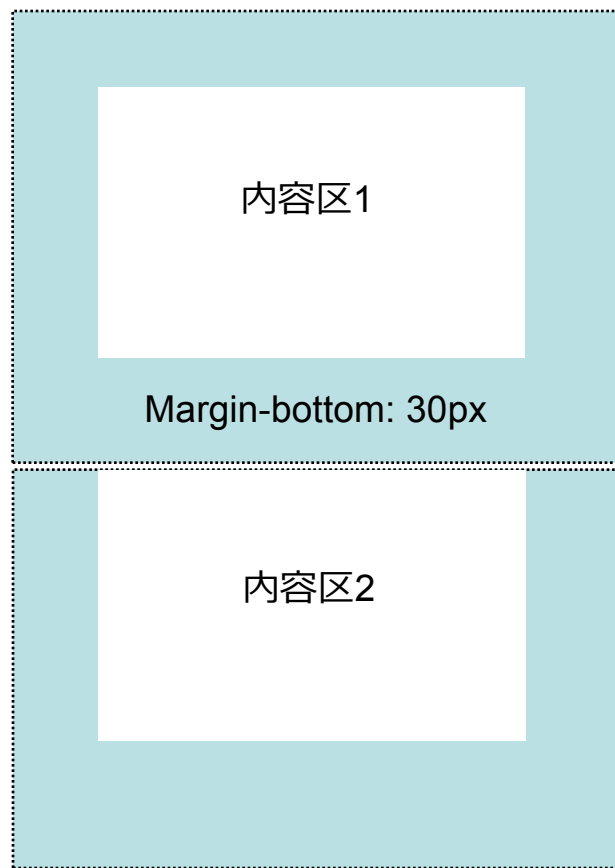
外边距折叠: 堆叠的元素

Before



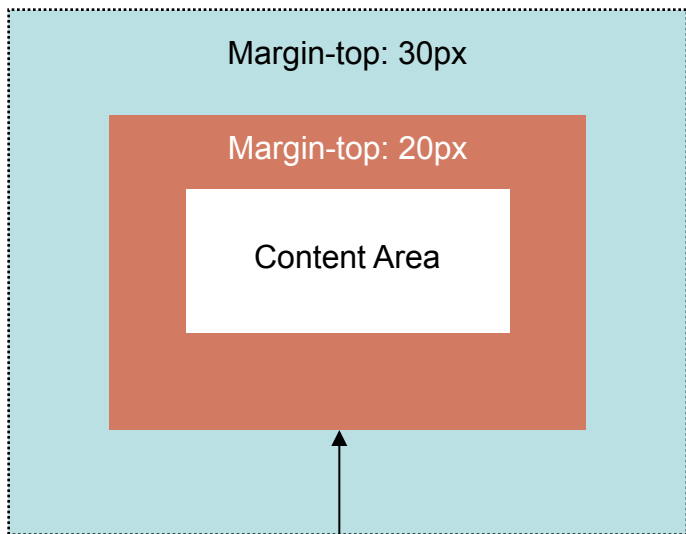
形成一个新的
外边距区

After



外边距折叠: 内置元素

折叠前

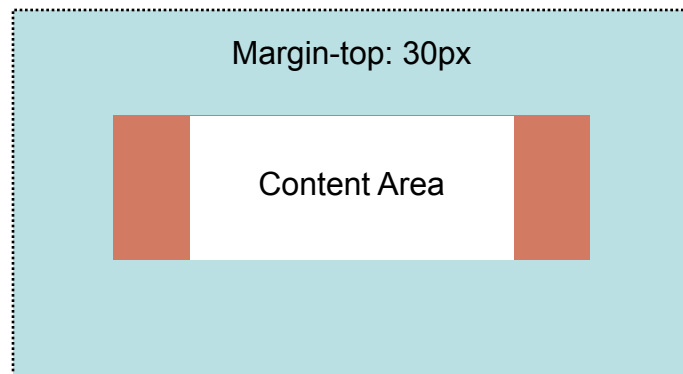


折叠形成一个单独的外边距

被包裹的块元素

外部的块元素

折叠后

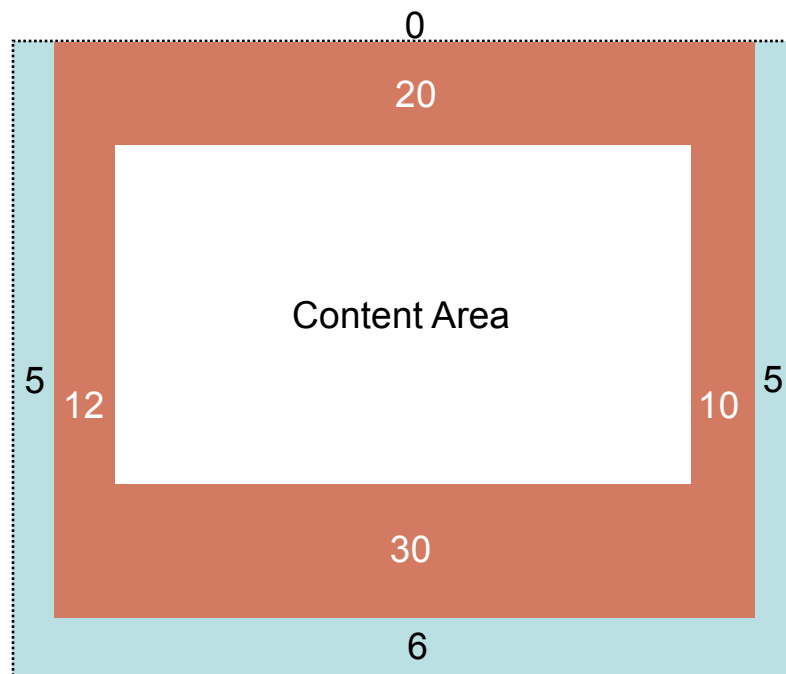


* Note: 仅仅出现在母子元素之间没有边框和内边距隔离时, 会出现这种情况。

CSS快捷方式: 外边距和内边距

对于外边距和内边距（以及其他），CSS提供了一种快捷写法来减少CSS代码书写的麻烦：

```
#container {  
    margin-top: 0;  
    margin-right: 5px;  
    margin-bottom: 6px;  
    margin-left: 5px;  
    padding-top: 20px;  
    padding-right: 10px;  
    padding-bottom: 30px;  
    padding-left: 12px;  
}
```

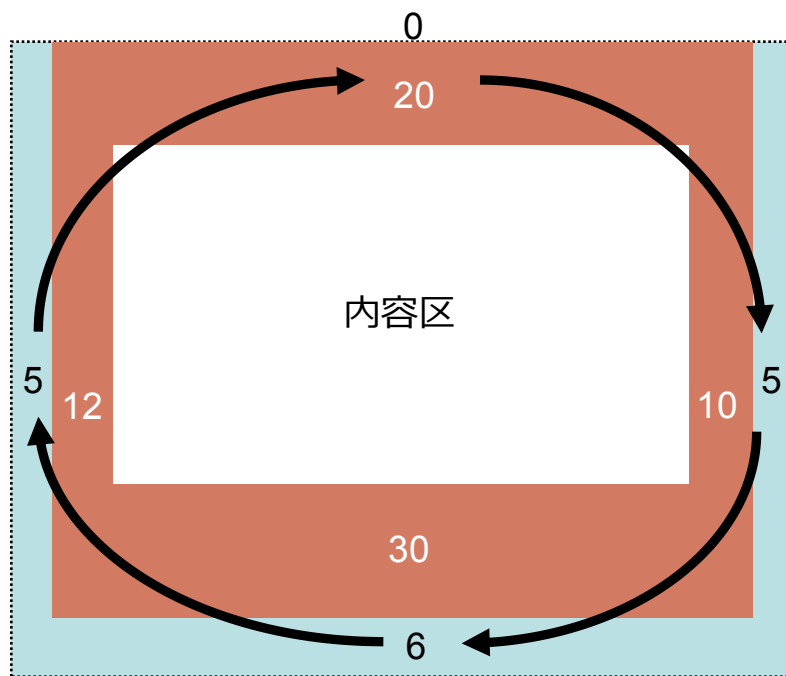
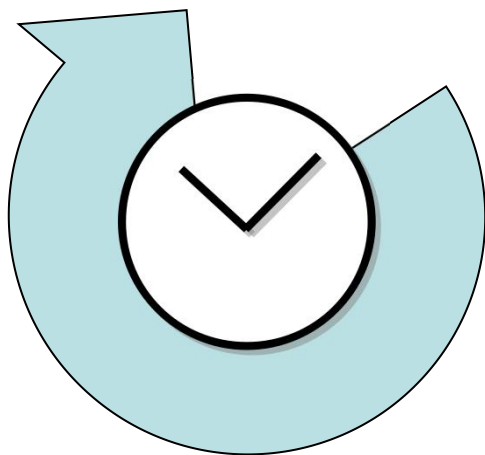


CSS快捷方式: 外边距和内边距

...这些写就显得简单多了:

```
#container {  
    padding: 20px 10px 30px 12px;  
    margin: 0px 5px 6px 5px;  
}
```

数值的顺序总是顺时针次序，
从顶部开始。

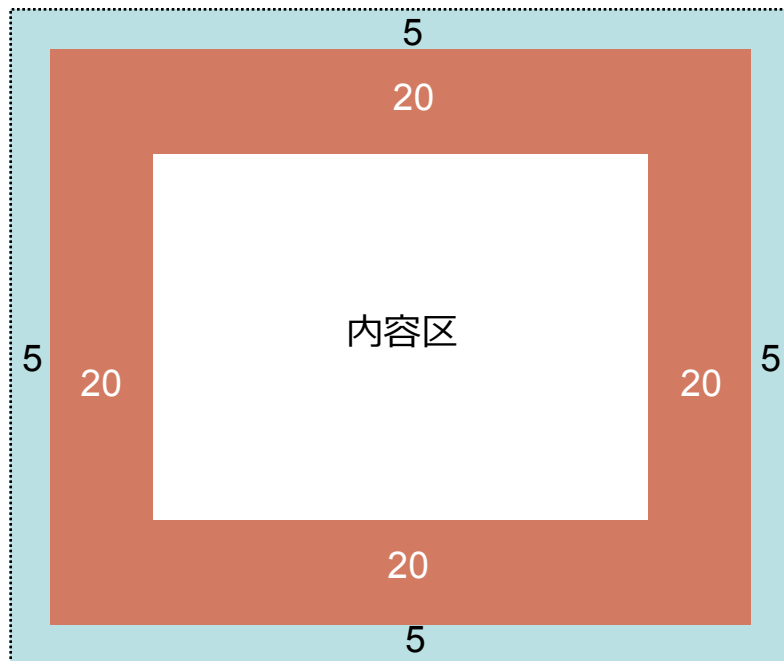


CSS快捷方式: 外边距和内边距

你可以只指定一个数值:

```
#container {  
    padding: 20px;  
    margin: 5px;  
}
```

该值将影响到4个边。



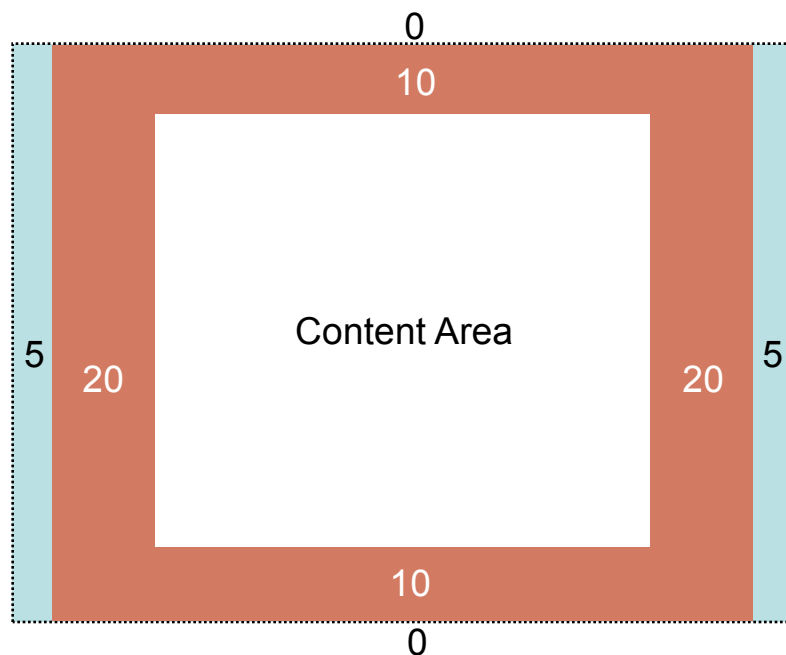
CSS快捷方式: 外边距和内边距

你可以指定两个数值:

```
#container {  
    padding: 10px 20px;  
    margin: 0px 5px;  
}
```

第一个值将影响到顶部和底部。

第二个值将影响左边和右边。



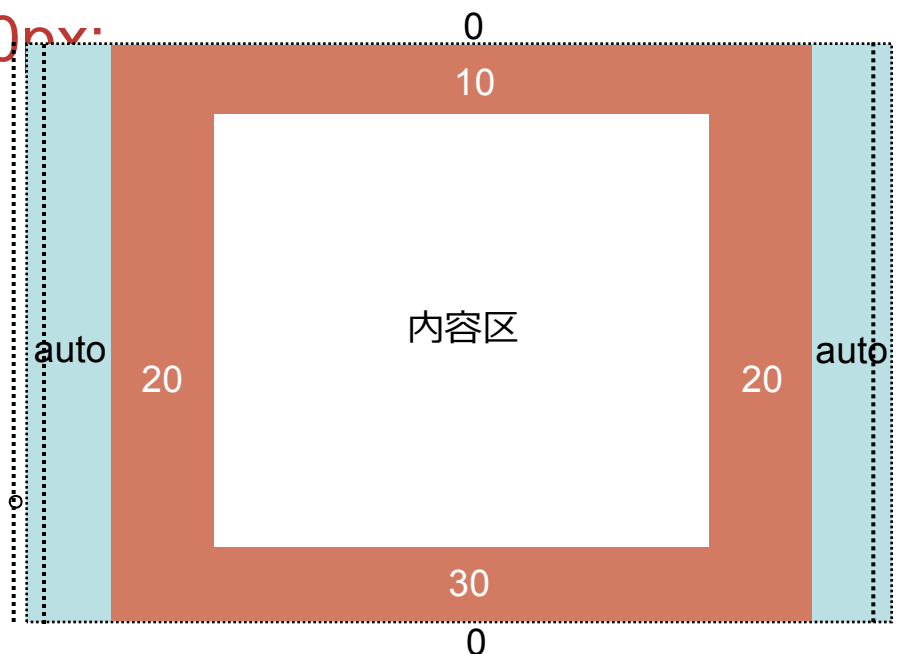
CSS 速记: Margin 和 Padding: auto

一个值得记住的数值: 'auto':

```
#container {  
  padding: 10px 20px 30px;  
  margin: 0px auto;  
}
```

通常用于左右的外边距,

auto 可以将 一个块元素居中。



子盒子在auto方式居中时, 必须设置自身的宽度。

用margin还是padding?

- 何时应当使用margin:

- 需要在border外侧添加空白时。
- 空白处不需要自己的背景（色），但需要父盒子的背景
- 上下相连的两个盒子之间的空白，需要相互抵消时。如15px + 20px的margin，将得到20px的空白。

- 何时应当用padding:

- 需要在border内测添加空白时。
- 空白处需要设置自身的背景（色）时。
- 上下相连的两个盒子之间的空白，希望等于两者之和时。如15px + 20px的padding，将得到35px的空白。

边框

边框可以应用到任何元素上。

边框总是位于内边距和外边距之间。

边框的值不能为负值。

如果边框没有指定**size**、**style**数值，则缺省值为“no-border”无边框。

边框可以定义四个边，也可以写到一行上。

边框

边框的核心属性有:

Width: *absolute* (绝对计量单位) (px, in, cm, or 'thin', 'medium', 'thick'), or *relative* (相对的计量单位, 如: ems)

Style: *dotted, dashed, solid, double, groove, ridge, inset, outset, hidden, etc*

Color: *'blue', 'red', #FF9900, etc*

你也可以通过设置背景图方式来设置“假”的边框。

内容区

Width of content + right-padding + left-padding +
right-border + left-border



$$200 + 20 + 20 + 10 + 10 = 280 \text{ px}$$

内容区（IE版本）

IE calculates the content box width as follows:

* **Content** = content + borders + padding



块盒子和行内盒子

Two types of boxes:

Block: Generated by HTML like P, DIV, Header tags, etc.

Inline: Generated by actual content (images and text), and inline elements like the `` and `` tags.

Block boxes can contain other boxes within them.

```
<div id="box">
```

First line of text

```
<p>Paragraph of text</p>
```

```
</div>
```

Inline Block

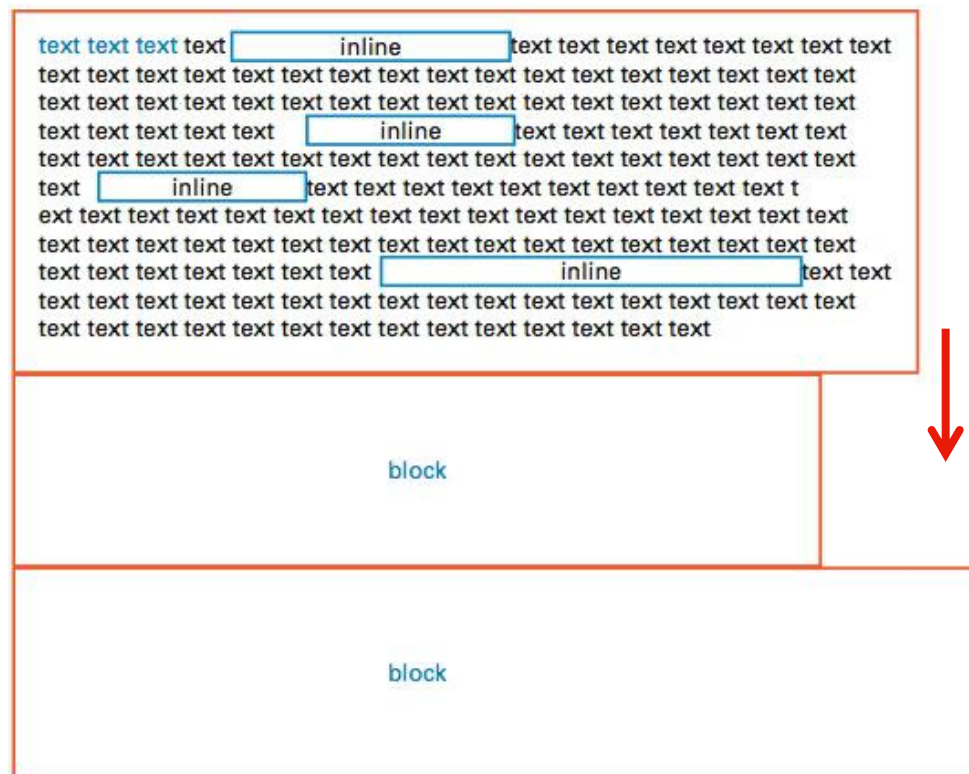
Containing Block

CSS: “Normal Flow（常规排版流程）”

CSS 为块元素定义的“盒子”们，它们是一个接一个排列的，一个在另一个之上，这样它们可以从上向下读。

在CSS中，这被称为“normal flow”常规排版流程。

(注意，CSS盒子对于行内元素 inline elements 是一个紧挨另一个横向排列的，视父块的盒子宽度，在盒内横向排列的元素，可自动换行。)



常规排版：Normal Flow

- 是浏览器的常规排版方式
- 默认情况下，所有元素从左边开始摆放，除非有特别设置。
- 块盒子从顶向下垂直排列在它们的父盒子中。
- 垂直排列时的外边距将发生折叠。
- 行内盒子从左向右排列。**但是...**

漂移

float:left, right, none;

一个漂移的“盒子”会先参考正常的排版流程，从该流程中取出后，尽量向版面的左侧或者右侧排版。

With the increase of laptop computer, traditional Desktop computer are slowly getting removed from the

market.



The ease of taking

laptop any ware with you is the biggest advantage of the Laptop computers. Laptop computers are still double as costly then desktop computers.

```
img {  
  float:left;  
}
```

With the increase of laptop computer, traditional Desktop computer are slowly getting removed from the



market. The ease of taking laptop any ware with you is the biggest advantage of the Laptop computers. Laptop computers are still double as costly then desktop computers.

同时漂移多个元素

漂移的盒子总是向左或向右进行布局，直到碰到父盒子的边缘或者碰到其他漂移盒子的边缘为止。

```
<ul>  
    <li>Home</li>  
    <li>Products</li>  
    <li>Services</li>  
    <li>Contact Us</li>  
</ul>
```

Home
Products
Services
Contact Us

应用下面的css样式之后：

```
li {  
    float:left;  
}
```

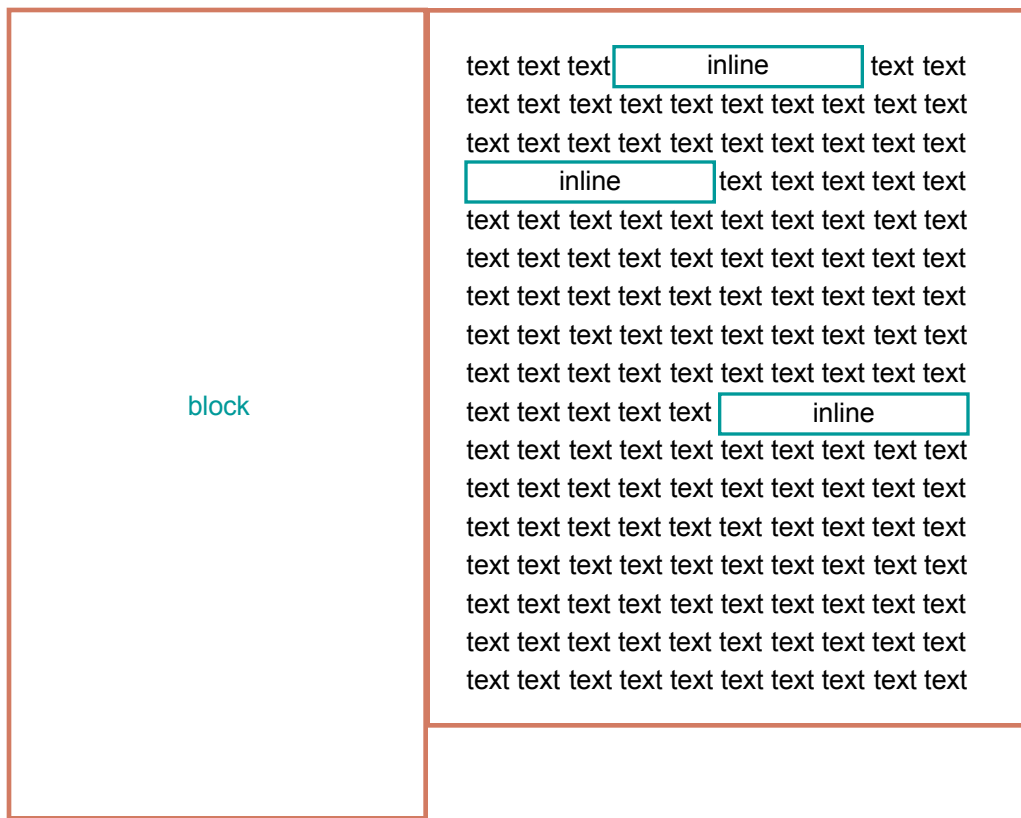
Home	Products	Services	Contact Us
------	----------	----------	------------

漂移: 对CSS的盒子进行固定

Float使元素在内容区中漂移到指定的一侧（左边或右边）。漂移的块盒子的顶部与后面的**行内元素**内容的顶部对齐，后面的其他内容将环绕在漂移元素的周围。

设置float 属性，可以将盒子从normal flow常规排版流程中脱离出来，可以向左或向右尽可能地移动。

使用 float，我们可以依次固定多个块元素。



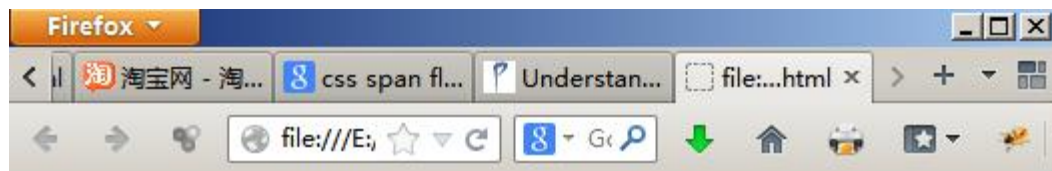
1. 所有, 不然, 实际宽度会尽的漂移盒子必须设置宽度可能的变窄。
2. 漂移盒子上下堆叠时的外边距不会发生折叠。

漂移示例1:

<p>

我是一个小的浮动块 我是正常的文本内容，我会围绕在前面代码中出现的浮动块周围，这段文字就是为了说明这个。zzz zz zzzzzz
zzzz zzzz zzzzz zzzzzzzz zzzzzzzz zzzzzz zz zzzzzz
zzzzzzzzzzzzzzzzzz

</p>

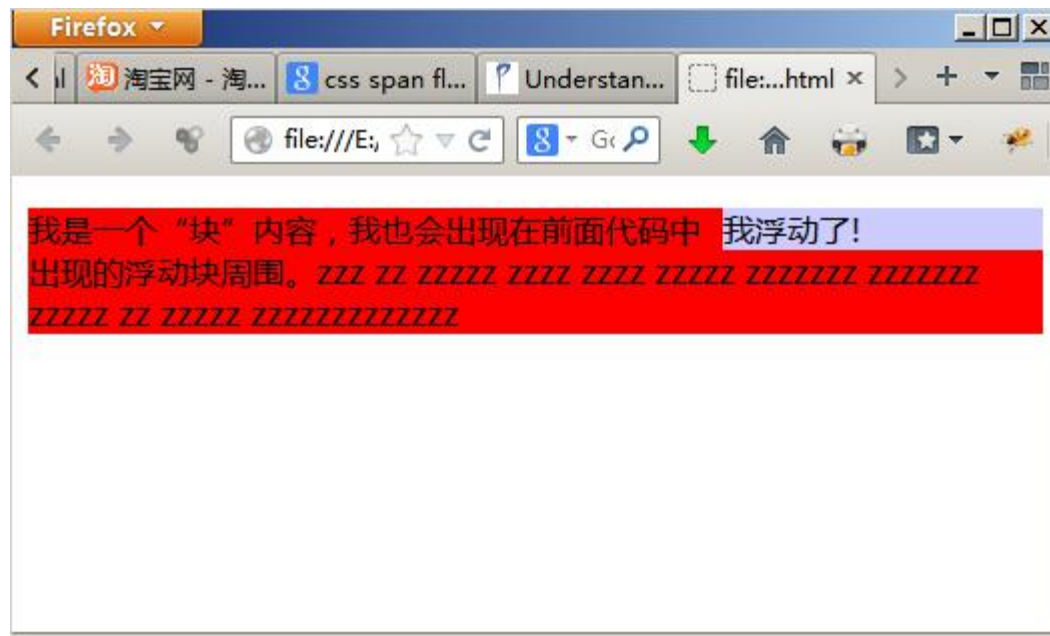


我是正常的文本内容，我会围绕在前面代码中出现 我是一个小的浮动块的浮动块周围，这段文字就是为了说明这个。zzz zz zzzzz zzzz zzzz
zzzz zzzzzzz zzzzzzz zzzzz zz zzzzz zzzzzzzzzzzzzzzzzzz

漂移示例2:

```
<p style="float:right; margin: 0px; background:#ccf;  
width:10em">我浮动了!</p>
```

```
<p style="background-color: red;">我是一个“块”内容，我也  
会出现在前面代码中出现的浮动块周围。 zzz zz zzzzzz zzzzz  
zzzz zzzzzz zzzzzzzz zzzzzzzz zzzzzz zz zzzzzz zzzzzzzzzzzzzzzzzzz  
</p>
```



Float 值

Float 值有三个选项:

float: left;

float: right;

float: none;

block

text text text inline text text
text text text text text text text text text text text text
text text text text text text text text text text text text
inline text text text text text text
text text text text text text text text text text text text
text text text text text text text text text text text text
text text text text text text text text text text text text
text text text text text text text text text text text text
text text text text text text text text text text text text
text text text text text text text text text text text text
text text text text text text text text text text text text
text text text text text text text text text text text text
text text text text text text text text text text text text
text text text text text text text text text text text text

The image displays a grid of 48 rectangular text boxes, organized into 8 horizontal rows and 6 vertical columns. Every box contains the identical black text "text text". Two specific boxes are distinguished by colored borders: one in the second row from the top, third column from the left, which features a thick red border; and another in the fourth row from the top, first column from the left, which features a thick blue border.



Float注意事项:

可以通过设置浮动元素的margin来操纵自己与外围文本之间的间隔。

```
float:left;  
width:7em;  
margin-  
right:1em;  
margin-  
bottom:1em;
```

I'm 'normal' content, and I wrap around any floats which appear before me in source code. Here's a bunch of text just to ensure that you can see my wrapping around the float. Here's a bunch of text just to ensure that you can see my wrapping around the float. Here's a bunch of text just to ensure that you can see my wrapping around the float. Here's a bunch of text just to ensure that you can see my wrapping around the float. Here's a bunch of text just to ensure that you can see my wrapping around the float.

Float注意事项:

块元素在浮动元素的后面，只有块元素的“行内元素内容”会围绕浮动元素，块元素的边框和背景仍然在浮动元素的后面。

浮动元素相当于后面紧跟的块元素内容的“列”。

Hello World

Notice how the background color of this paragraph continues as if the float weren't there, despite the fact that the content of this paragraph wraps around the float and its margins. See also how the and the bottom border of the header above also behaves similarly. Whether you think this is Good or Bad, it's the way the specifications say it's supposed to behave.

```
margin:0 2em
```

This float is tall to show that borders and backgrounds go under floats.

Float元素注意事项:

可以使用块元素和“浮动元素”来进行内容“分列”布局。

只需要设置好后面块元素的margin边距，令其足够宽即可。

```
width:8em
```

This float is
being used as a
column.

This paragraph has `margin-left:9em` to ensure it stays to the right of the floated 'column'. (I'm using `9em` and not `8em` to provide some whitespace between the two columns.) Note also how the background of this paragraph doesn't go under the column, because of the margin. And finally, note that this paragraph has gotten past the end of the float, but isn't wrapping underneath it.

Float注意事项:

若浮动元素不声明宽度，则其宽度趋向于0，即压缩到其内容能承受的最小宽度。

Inline元素在漂移时，可以设置盒子的高度和宽度。

Inline元素漂移时，其盒子的高度由父盒子的line-height属性决定。

block元素漂移时，其盒子的高度由height决定。

Float使用注意事项

- 尽量使排列对象的高度保持一致，否则页面就会出错。例：

```
.mokuai { width:90px; padding:12px 23px 0 0; float:left}
```

新软速递 更多 >



Scopy
摄影图像



Kinetic -...
养生保健



全国公交...
旅行



Honk
导航



You Gotta...



Let's cre...



ComicGlas...
实用工具

高度不一，产生错位！



Infinicam
摄影图像



Akinator
娱乐



Eye Illus...
娱乐



360 Panorama
摄影图像



PingChat!...
社交网络

Float使用注意事项

- 加上height属性来做配合，解决排列错位问题。

```
.mokuai { width:90px; padding:12px 23px 0 0; float:left; height:116px; }
```

新软速递 更多 >



Scopy
摄影图像



Kinetic -...
养生保健



全国公交...
旅行



Honk
导航



You Gotta...



Let's cre...



ComicGlas...
实用工具



Infinicam
摄影图像



Akinator
娱乐



Eye Illus...
娱乐



360 Panorama
摄影图像



PingChat!...
社交网络

重建Normal Flow排版: “Clear”

重建 “normal flow”排版过程，我们可以使用 clear属性。

clear 属性有三个值选项：

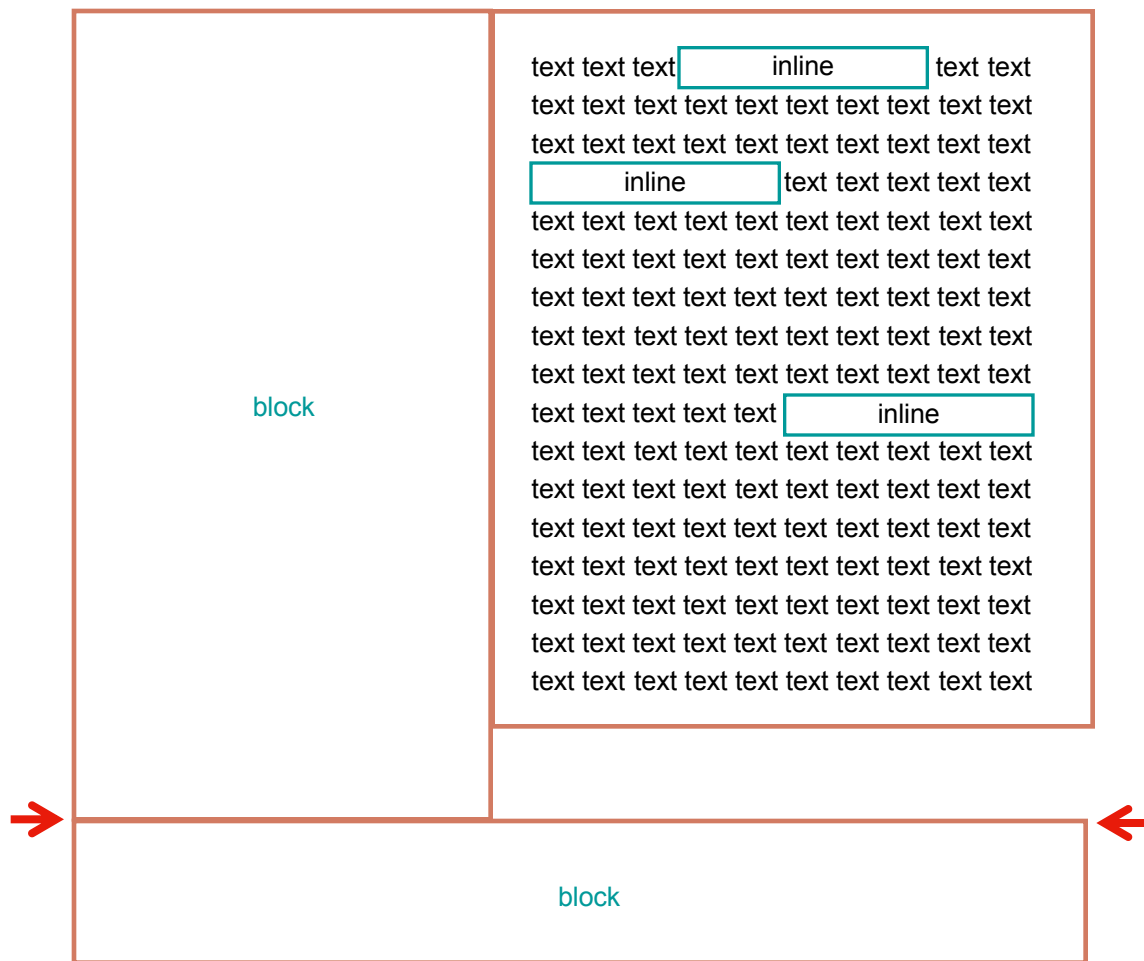
clear: left;

clear: right;

clear: both;

这些指明了元素盒子的那一边应该不能出现float相邻元素。

起到了将后继的元素内容强制下压到前面浮动元素的下方。



重建Normal Flow排版: “Clear”

- clear:left 的含义:

—排版流程向下偏移，直到我的左侧再也没有浮动的元素，然后才开始下一个内容元素的排版。

```
margin-  
right:1em;  
margin-  
bottom:1em;  
This float is tall to  
show what happens  
when you have two  
paragraphs next to  
it.
```

This is paragraph number 1.

This is (unstyled) paragraph number 2.

```
margin-  
right:1em;  
margin-  
bottom:1em;  
This float is tall to  
show what happens  
when you have two  
paragraphs next to  
it.
```

This is paragraph number 1.

This paragraph has `clear:left` applied to it, and starts after the float is done.

重建Normal Flow排版: “Clear”

- 用clear来防止容器元素的内容溢出。

```
margin-right:1em;
```

Here's a bunch more content to make this float extra tall, flowing outside the box that holds this example

This is all the content that will be held by this container.

另外一种方案是：容器元素加属性overflow:auto

```
<p style="float:left; ..."> ... </p>
<p style="margin-bottom:1em">This is paragraph number 1.</p>
<p>This is (unstyled) paragraph number 2.</p>
<div style="clear:left"> </div>
```

```
margin-right:1em
```

This float is extra tall with extra content to show what happens when you have two paragraphs next to it.

This is paragraph number 1.

Look! The float is held within the container by the empty `<div>` placed after this paragraph!

因为加了一个空的div，使含漂移元素的父盒子高度增加了，避免了漂移元素内容的溢出

躲避漂移

clear:both ;

或者

```
<style type="text/css">
```

```
.clearfix:after {
```

```
    content: "."; display: block; height: 0; clear: both; visibility: hidden;
```

```
}
```

```
.clearfix {
```

```
    display: inline-block;
```

```
} /* for IE/Mac */
```

```
</style>
```

```
<!-- if IE -->
```

Float作业

- 1. 做菜单导航
- 2. 做图文混排
- 3. 做分栏目布局
- 4. 做商品列表

CSS 定位

CSS布局中的第三个核心概念，其他两个是‘box model（盒模型）’和‘float(漂移)’, 是 positioning（定位）。

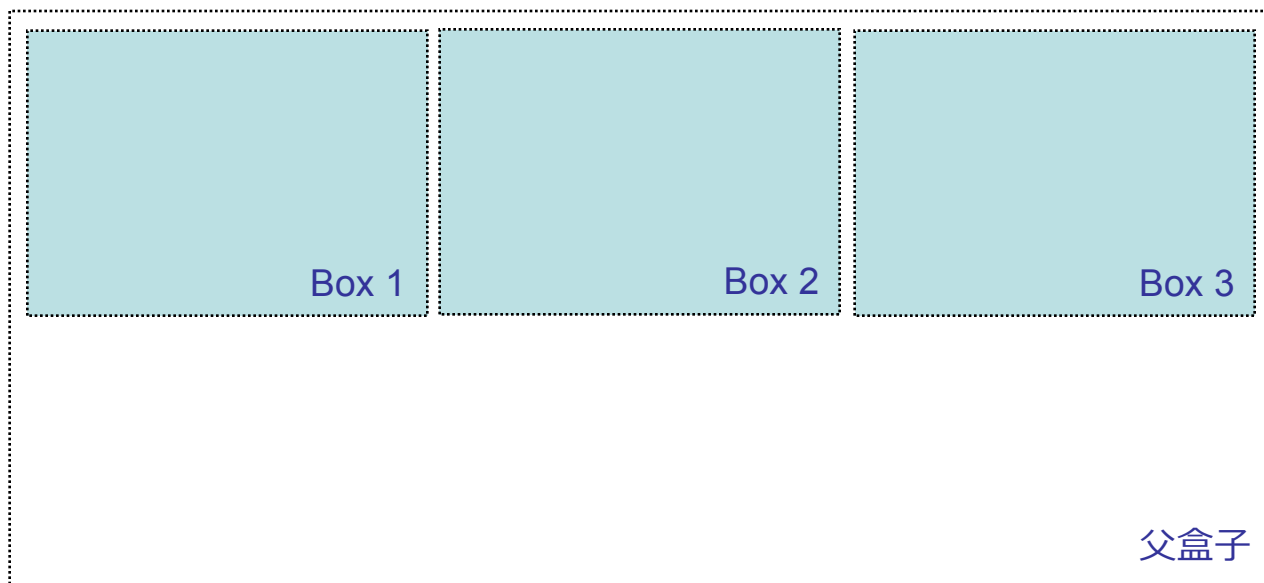
有四种定位方式：

- **Static Positioning**（静态定位、常规排版）
- **Relative Positioning**（相对定位）
- **Absolute Positioning**（绝对定位）
- **Fixed Positioning**（固定定位）

非常重要：理解这四种方式之间的不同。

CSS Positioning: 相对定位 (Relative Positioning)

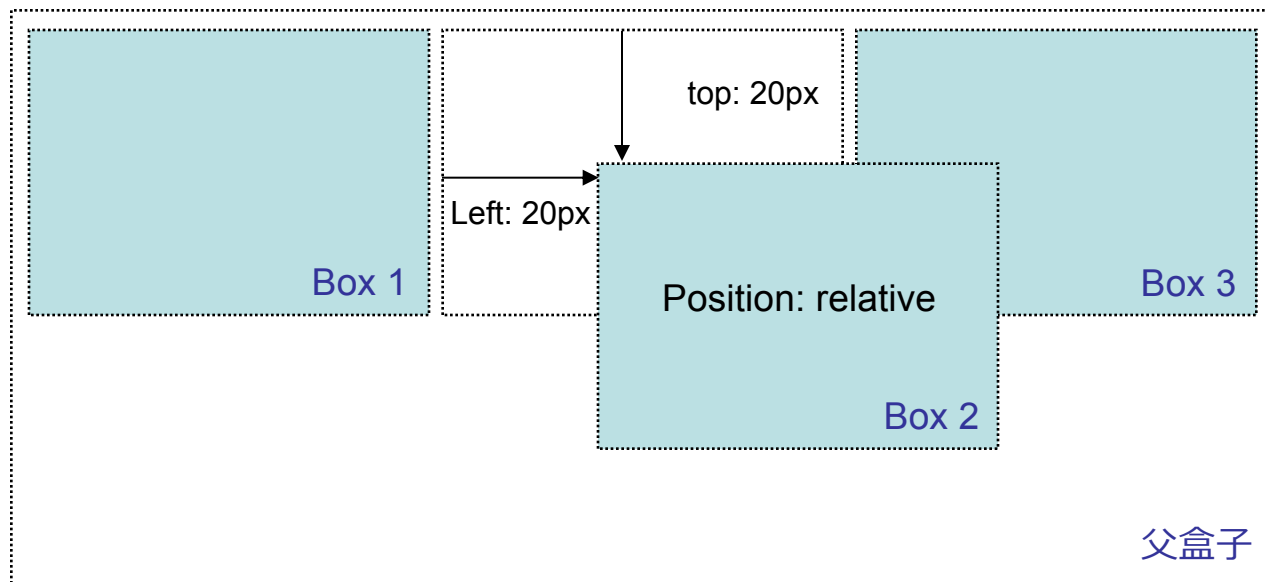
相对定位的元素，是相当于normal flow正常排版位置而言。



CSS Positioning: 相对定位 (Relative Positioning)

本例中, **box 2** 左上角偏离20px. 结果是其偏离了正常排版位置点的20px。Box 2 可能覆盖掉其他的box盒子, 但其他的盒子仍然能够识别Box2的原来的位置, 而排版下去。

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



CSS Positioning: 绝对定位(Absolute Positioning)

绝对定位是完全脱离了正常排版的流程，只与其父盒子的定位有关。父盒子的内边距padding左上边缘点(也即边框内侧)为坐标参考原点（A点），子盒子的外边距margin的左上边缘点作为偏移点（B点）。

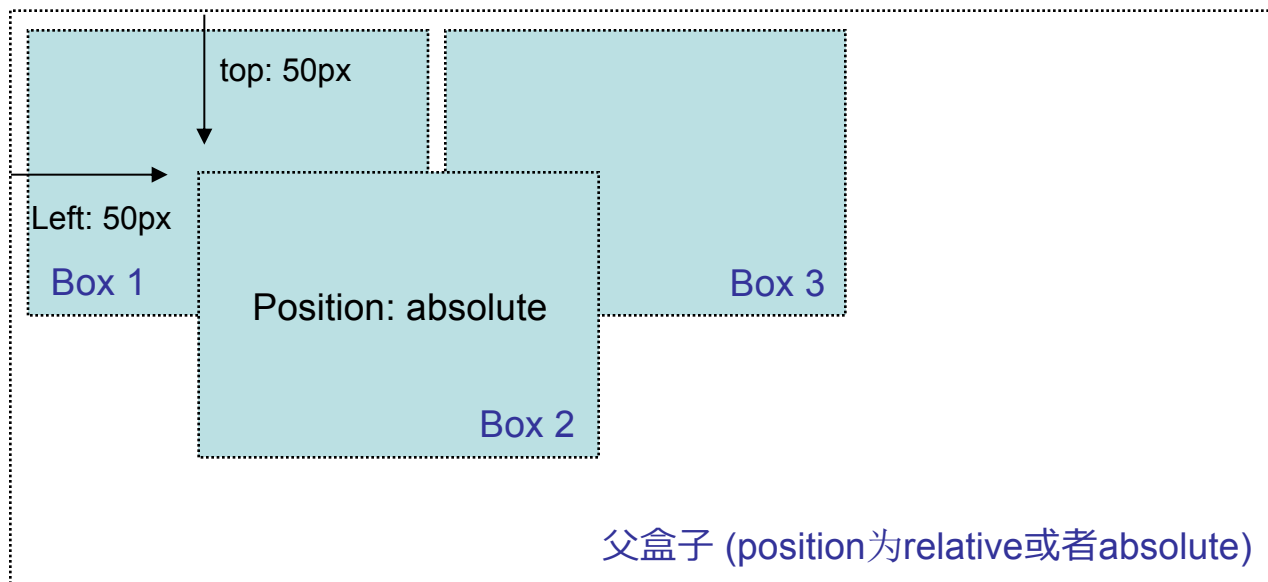
如果没有父盒子，则其定位只与初始的块相关，一般指浏览器窗口（body元素）。



CSS Positioning: 绝对定位 (Absolute Positioning)

绝对定位的盒子，该父盒子中的其他盒子排版时，将不考虑该盒子，将对该盒子视而不见。

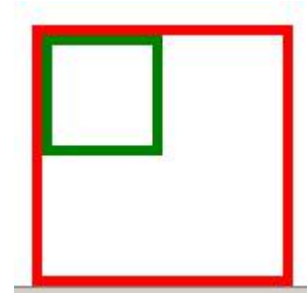
```
#myBox {  
  position: absolute;  
  left: 50px;  
  top: 50px;  
}
```



CSS Positioning: 带父节点的绝对定位 (Absolute Positioning)

```
#f1 {  
  position: relative; /* or absolute */  
  top: 100px;  
  left: 100px ;  
  width: 100px ;  
  height: 100px ;  
  border: solid 5px red;  
  padding: 10px;  
}
```

```
<div id="f1">  
  <div id="f2">  
  
  </div>  
</div>
```



```
#f2 {  
  position: absolute;  
  top: 0px;  
  left: 0px;  
  width: 50% ;  
  height: 50px ;  
  border: solid 5px green ;  
}
```

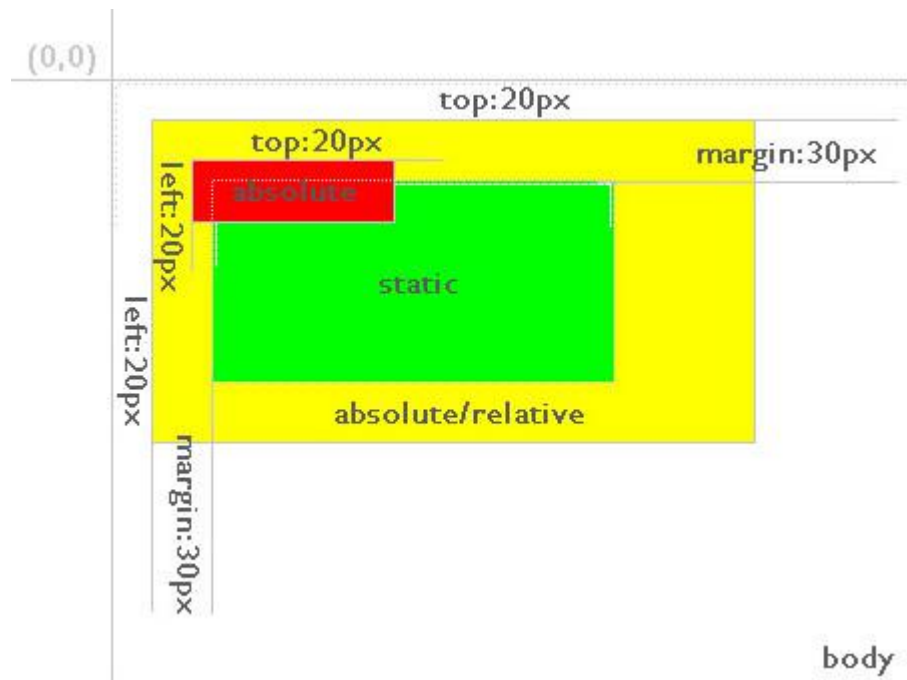
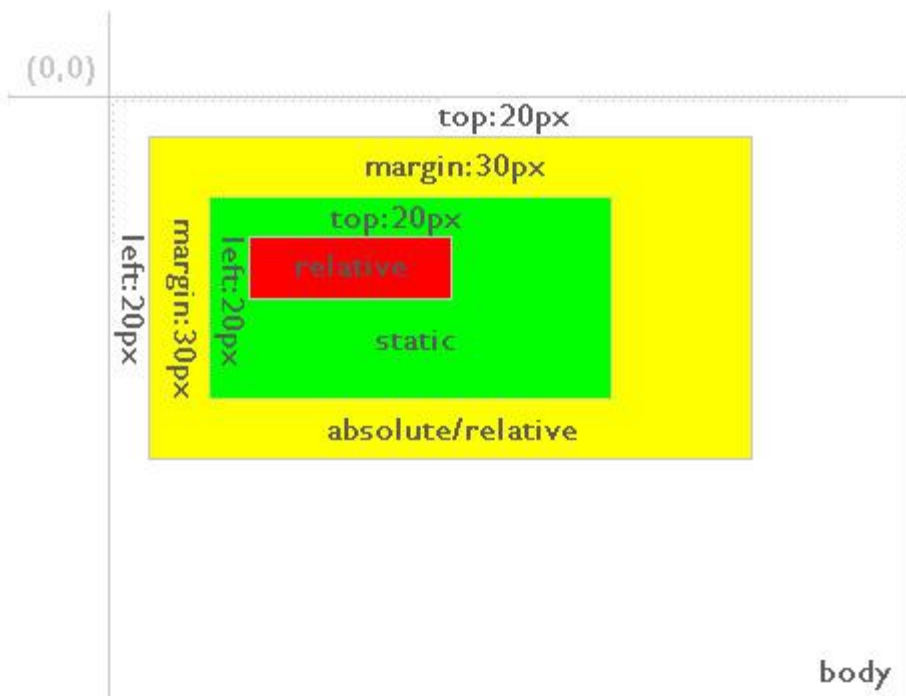
示例:

当父节点的position为relative或者absolute时，Left、top等是相对于父元素的边框线内侧开始计算的。

百分比指相对于参考定位对象，非父元素

CSS Positioning: 相对定位 vs 绝对定位

- **Relative** 相对定位的层总是相对于其原有位置，无论其父元素是何种定位方式。
- **Absolute** 绝对定位的层总是相对于其最近的定义为**absolute**或**relative**的父层，而这个父层并不一定是其直接父层。如果其父层中都未定义**absolute**或**relative**，则其将相对**body**进行定位。



CSS Positioning: 固定定位（Fixed Positioning）

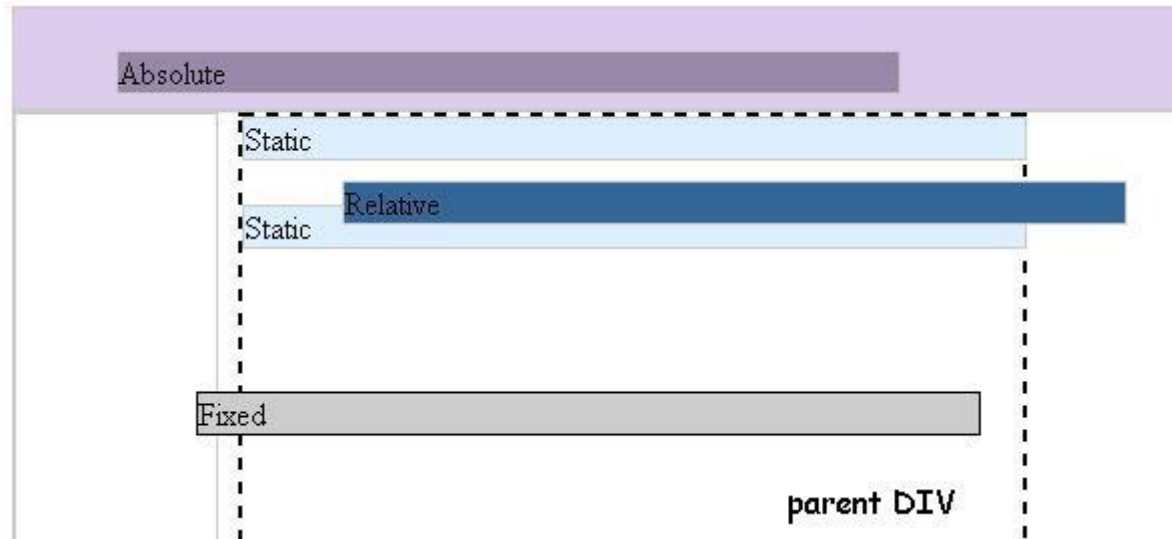
Fixed Positioning固定位置是Absolute Positioning 的一个特例。

其允许浮动元素始终固定在浏览器窗口（非网页）的固定位置，不管内容如何滚动，位置都不变。

固定定位的元素，其百分比的宽度参考的不是其父元素，是网页内容的宽度，即：<html>标记的宽度。

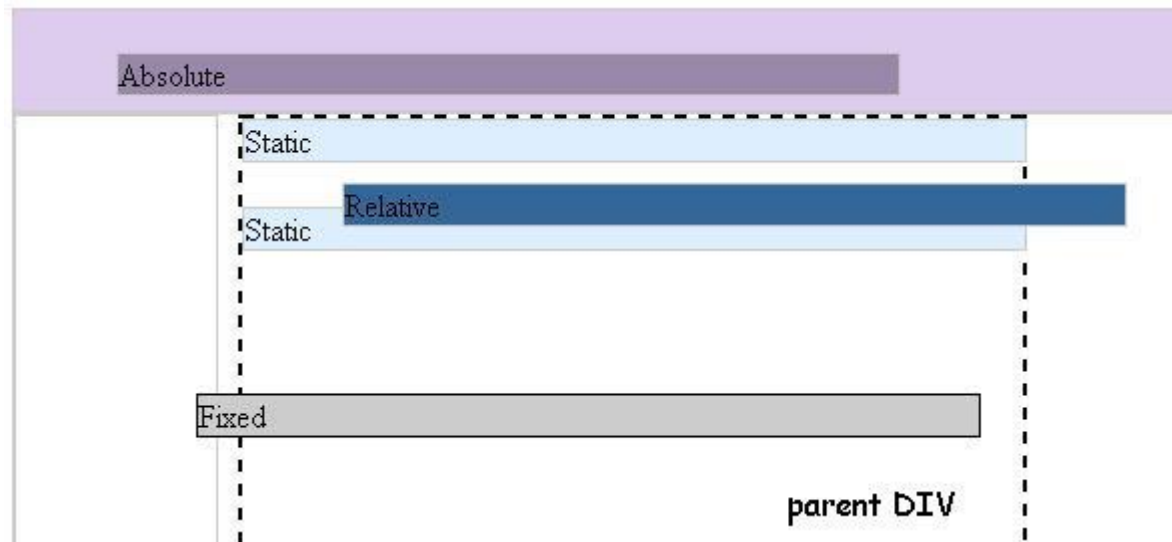
Positioning – static (静态)

position: **static**; (缺省值) 元素采用正常排版布局(忽略 top, bottom, left, right, 或 z-index 的属性声明)



Positioning – absolute绝对定位

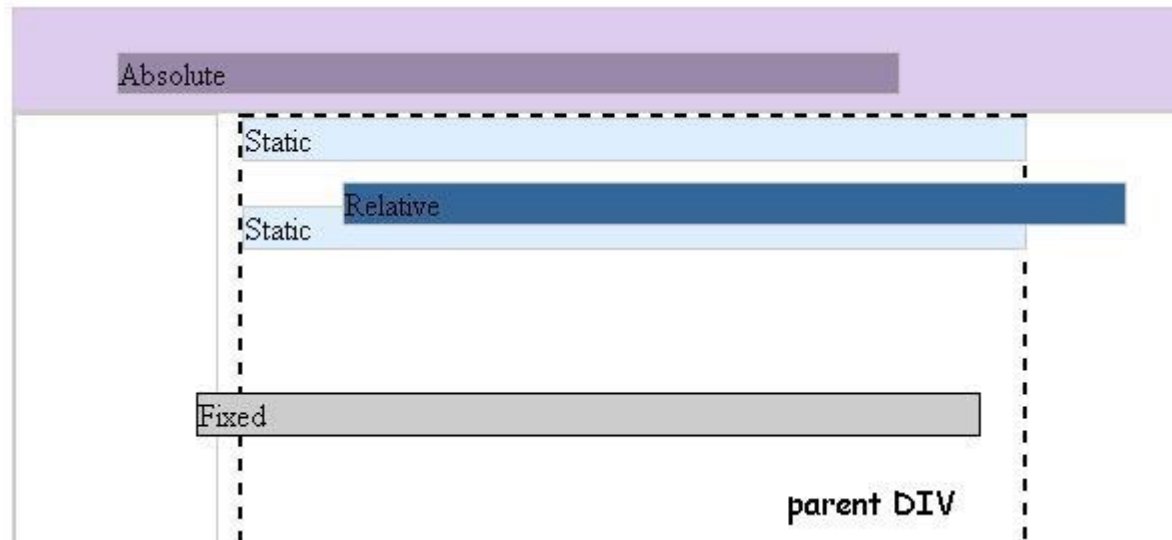
`position: absolute;` 生成一个绝对定位元素, 其位置基于 **第一个非static (常规) 布局的祖先元素** (如果没有这样的祖先元素, 那么将相对于HTML文档的BODY元素). 使用 `bottom`, `right`, `top` 和 `left` 属性来定位。



Positioning – fixed固定定位

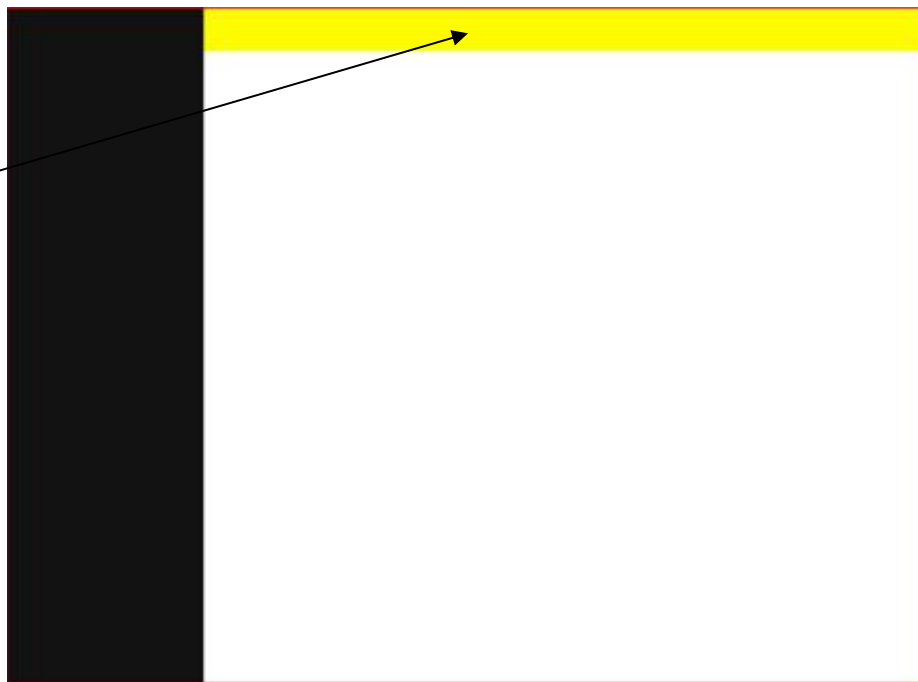
`position:fixed`; 生成一个绝对定位的元素, 其定位是基于 **浏览器窗口**, 而非页面窗口, 因此, 当 **页面滚动时该元素的屏幕位置不变**. 使用 `bottom`, `right`, `top` 和 `left` 属性来放置元素。

固定定位的width的百分比是相对于网页内容（即<html>元素）的宽度。



Positioning – fixed固定定位

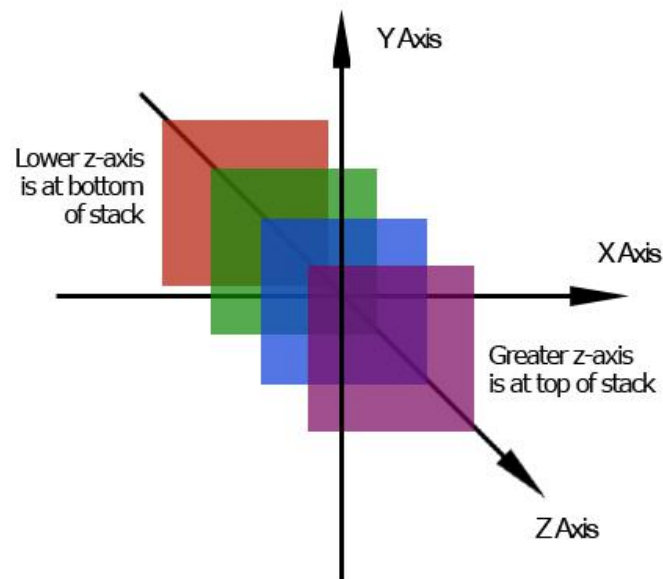
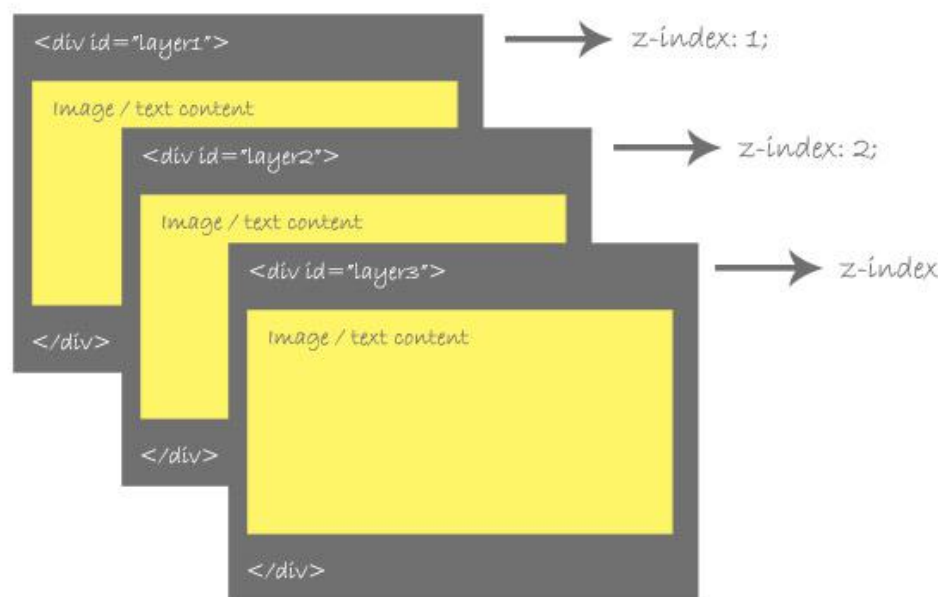
```
.top-bar {  
  position: fixed;  
  top:0;  
  left:250px;  
  right:0;  
  height:54px;  
  background:#090909;  
  z-index:1000;  
}
```



通过指定偏移，自动调整内容区尺寸。


z-index

- The z-index 属性表述了元素的 **堆叠次序**.
- 有着更大次序号的元素总是显示在其他元素之前.
- 仅应用在 **定位元素** (position:absolute, position:relative, or position:fixed).



Floats & Positioning

总结:

 **Floats** 将盒子从正常排版流程中拿出来，然后左右漂移到父盒子的最远的边缘。

Positioning:

- 1、static：默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。
- 2、relative：生成相对定位的元素，通过top,bottom,left,right的设置相对于其正常位置进行定位。可通过z-index进行层次分级。
- 3、absolute：生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。可通过z-index进行层次分级。
- 4、fixed：生成绝对定位的元素，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。可通过z-index进行层次分级。

参考

- <http://zh.learnlayout.com/>