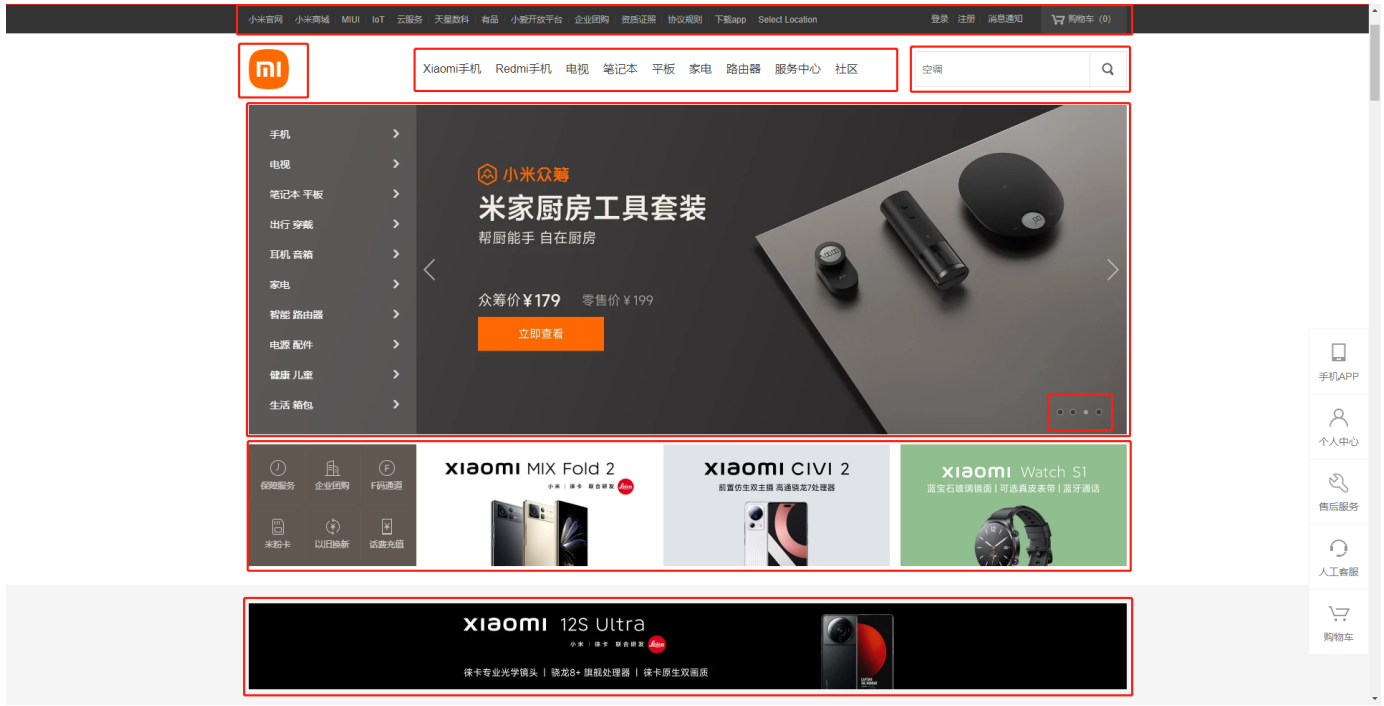
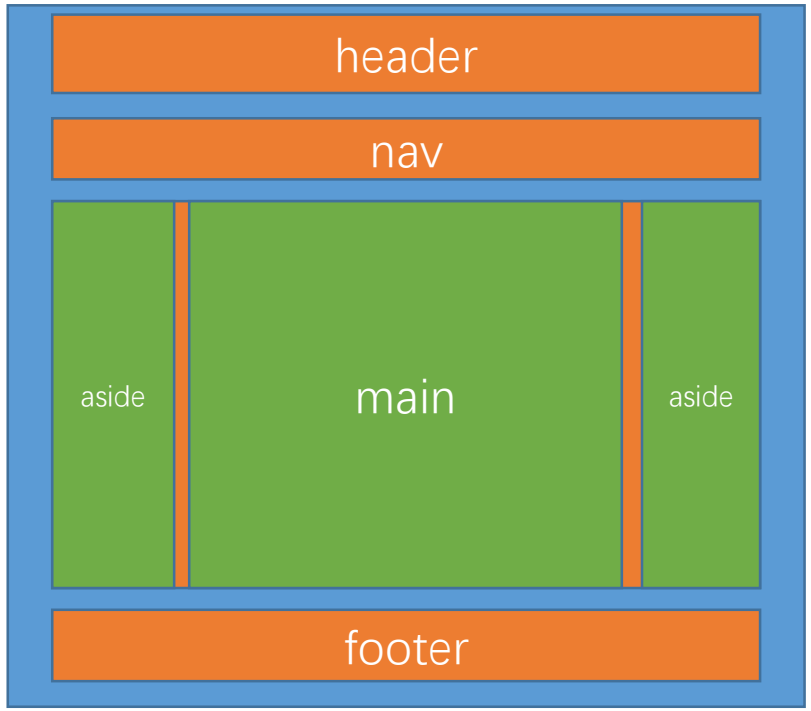
CSS定位与布局

盒子模型

CSS认为每个元素都是一个盒子，可以通过CSS样式定义元素的高、宽、边框、边距等，并将盒子定位在页面上的某个位置。页面布局就是通过大盒子嵌套小盒子，逐步细化完成的。完成元素定位和页面布局后，才涉及具体的设置元素的字体、文本等样式。





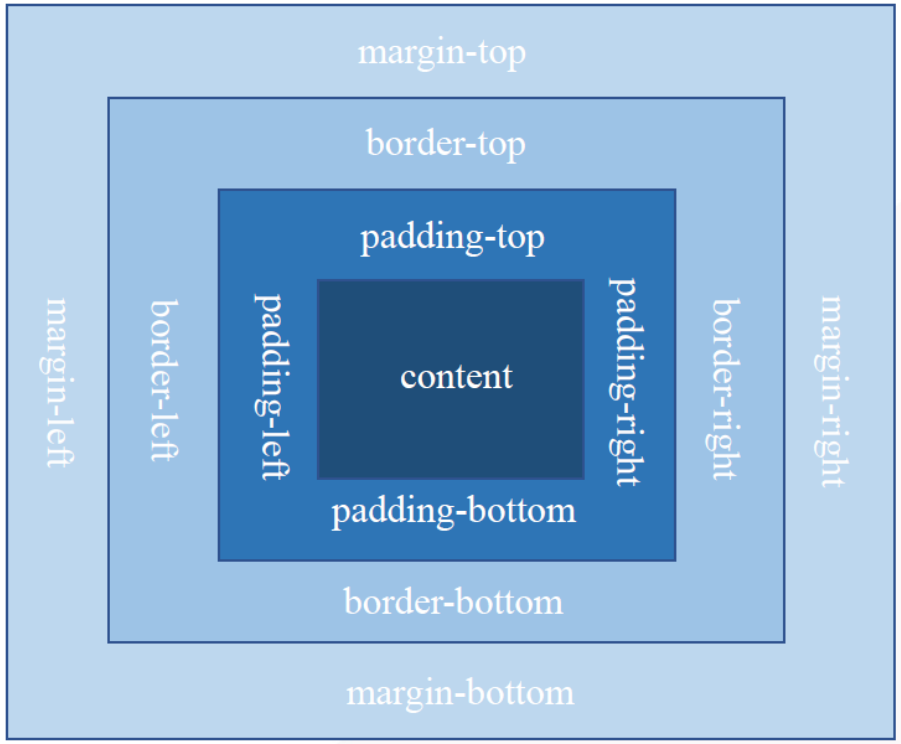
页面上一些区域、图片、导航、列表、段落等等，都可以是盒子

盒子模型

1、盒子模型组成

CSS 盒子模型本质上是一个盒子，封装周围的HTML 元素，它包括：边框、内边距、外边距和实际内容。

- content : 这个区域是用来显示内容，大小可以通过设置 width 和 height.
- padding: 包围在内容区域外部的空白区域；大小通过 padding 相关属性设置。
- border: 边框盒包裹内容和内边距。大小通过 border 相关属性设置。
- margin: 这是最外面的区域，是盒子和其他元素之间的空白区域。大小通过 margin 相关属性设置。



盒子模型

1、盒子模型组成

padding 和 border 再加上设置的宽高一起决定整个盒子的大小。

```
.box {
  width: 350px;
  height: 150px;
  margin: 25px;
  padding: 25px;
  border: 5px solid black;
}
```

- 如果使用标准盒模型，即content-box，那么实际占用空间的宽高分别为：宽度 = 410px (350 + 25 + 25 + 5 + 5)，高度 = 210px (150 + 25 + 25 + 5 + 5)。
- margin 不计入实际大小 —— 但它会影响盒子在页面所占空间，影响的是盒子外部空间。盒子的范围到边框为止 —— 不会延伸到 margin。

盒子模型

1、盒子模型组成

box-sizing属性 决定了盒子大小的计算方式:

① 默认取值为content-box，此时width、height指定内容的宽度和高度，border和padding不计算入width和height之内。

盒子的总宽度=width+padding+border

盒子的总高度=height+padding+border

以默认的content-box盒子进行页面布局时，如果已经定义好width和height，一旦添加border和padding，都会使盒子实际尺寸变大，影响布局。

盒子模型

1、盒子模型组成

默认浏览器会使用标准模型content-box。使用border-box模型需要通过box-sizing: border-box 来设置。取值为border-box，则盒子的width、height包含content和padding、border。使用这个模型，所有宽度都是可见宽度，所以内容宽度是width减去border和padding部分。

盒子模型

2、border边框

为边框设置样式时，有大量的属性可以使用——有四个边框，每个边框都有样式、宽度和颜色。可以使用border属性一次设置所有四个边框的宽度、颜色和样式。

border: border-width border-style border-color

设置所有边的颜色、样式或宽度，可以使用以下单独属性：

- border-width
- border-style
- border-color

盒子模型

2、border边框

分别设置每边的宽度、颜色和样式，可以使用：

```
border-top: 1px solid red; /* 只设定上边框，  其余同理 */
border-bottom: 1px solid red;
border-left: 1px solid red;
border-right: 1px solid red;
```

盒子模型

3、内边距padding属性

padding 属性用于设置内边距，内边距位于边框和内容区域之间。与外边距不同，不能设置负值。

属性	作用
padding-left	左内边距
padding-right	右内边距
padding-top	上内边距
padding-bottom	下内边距

盒子模型

3、内边距padding属性

padding 属性（简写属性）可以有一到四个值。

值的个数	表达意思
padding: 5px;	1个值，代表上下左右都有5像素内边距;
padding: 5px 10px;	2个值，代表上下内边距是5像素　左右内边距是10像素;
padding: 5px 10px 20px;	3个值，代表上内边距5像素　左右内边距10像素　下内边距20像素;
padding: 5px 10px 20px 30px;	4个值，　上是5像素 右10像素 下20像素 左是30像素　顺时针

当我们给盒子指定 padding 值之后：

1、内容和边框之间有了距离，添加了内边距。

2、padding影响了盒子实际大小。

也就是说，如果盒子已经有了宽度和高度，此时再指定内边框，会撑大盒子。

解决：

如果要保证盒子跟效果图大小保持一致，那么让 width/height 减去多出来的内边距大小。

盒子模型

padding影响盒子

padding内边距可以撑开盒子。

因为每个导航栏里面的字数不一样多，可以不用给每个盒子设定宽度，直接添加padding最合适。

相关取值:

1. 上边框为3像素，颜色为 #ff8500
2. 下边框为1像素， 颜色为 #edeef0
3. 盒子高度为45像素，背景颜色为 #fcfcfc
4. 文字颜色为 #4c4c4c

盒子模型

4、外边距margin属性

margin 属性用于设置外边距，外边距是盒子周围一圈看不到的空间。它会把其他元素从盒子旁边推开。外边距属性值可以为正也可以为负。设置负值会导致和其他内容重叠。

属性	作用
margin-left	左外边距
margin-right	右外边距
margin-top	上外边距
margin-bottom	下外边距

margin 简写方式代表的意义跟 padding 完全一致

盒子模型

4、外边距margin属性

外边距可以让块级盒子**水平居中**，但是必须满足两个条件：

①盒子必须指定了宽度（width）。

②盒子左右的外边距都设置为 auto 。

```
.header{ width:960px; margin:0 auto;}
```

常见的写法，以下三种都可以：

- margin-left: auto; margin-right: auto;
- margin: auto;
- margin: 0 auto;

以上方法是让块级元素水平居中，行内元素或者行内块元素水平居中给其父元素添加 text-align:center 即可。

盒子模型

5、外边距合并

使用 margin 定义块元素的垂直外边距时，可能会出现外边距的合并。主要有两种情况:

1.相邻块元素垂直外边距的合并

2.嵌套块元素垂直外边距的塌陷

注意：垂直方向上相邻的元素才会有外边距合并的问题。

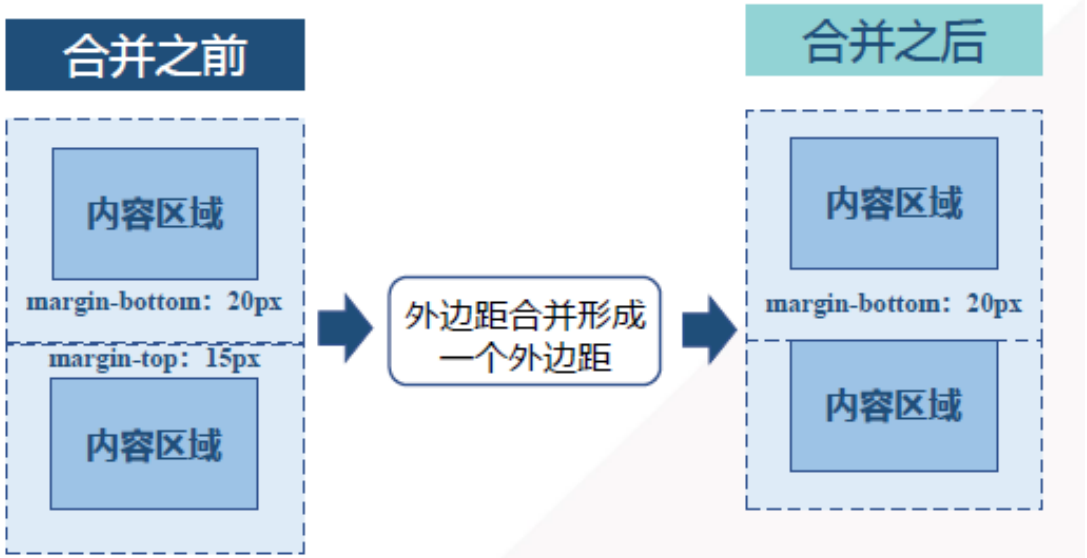
盒子模型

5、外边距合并

(1). 相邻块元素垂直外边距的合并

当上下相邻的两个块元素（兄弟关系）相遇时，如果上面的元素有下外边距 margin-bottom，下面的元素有 上外边距 margin-top ，则他们之间的垂直间距不是 margin-bottom 与 margin-top 之和。

取两个值中的 较大者这种现象被称为**相邻块元素垂直外边距的合并**。

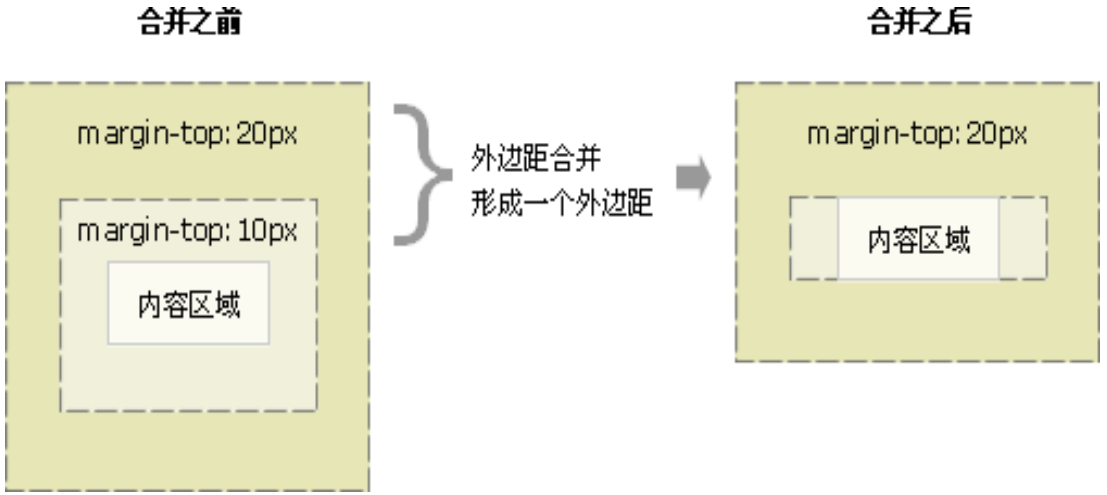


解决方案：

尽量只给一个盒子添加 margin 值。

盒子模型

5、外边距合并



(2). 父子相邻元素垂直外边距的塌陷

对于两个具有父子关系的块元素，父元素有上外边距同时子元素也有上外边距，此时父元素会塌陷相对较大的外边距值，而子元素没有实现上外边距效果。

解决方案：

①可以为父元素定义上边框border-top。

②改子元素的 margin-top 为父元素的 padding-top，同时为防止父元素变大，再设设置box-sizing:border-box。

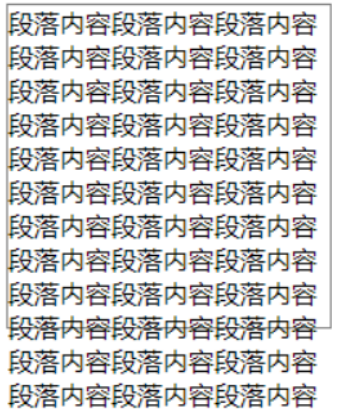
③可以为父元素设置overflow:hidden。

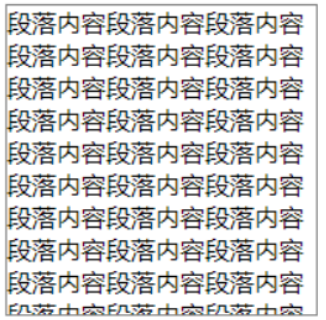
④给父或子元素添加固定定位或者绝对定位。

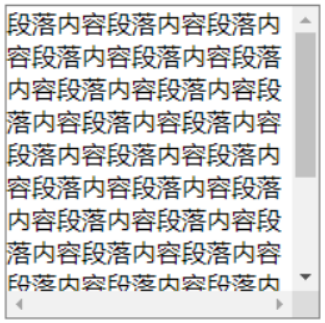
⑤给父元素或子元素设置display：inline-block

注意：父子元素间的相邻是指：父元素和第一个子元素，只有第一个子元素才会和父元素外边距合并。

盒子模型







6、overflow属性

overflow属性规定当内容溢出元素边框时如何显示，取值如下：

① 默认取值visible：内容不会被修剪，会呈现在元素框之外；

② 取值hidden：内容会被修剪，并且其余内容是不可见的；

③ 取值scroll：内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容。

④ auto：如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容。解决了scroll在不需要滚动条也会产生滚动条的问题

overflow属性在动画效果中经常使用，初始状态是溢出边框的部分隐藏，状态切换时再将其显示出来。

- overflow还有个作用是清除浮动的影响，具体说就是清除包含块内子元素的浮动。

盒子模型

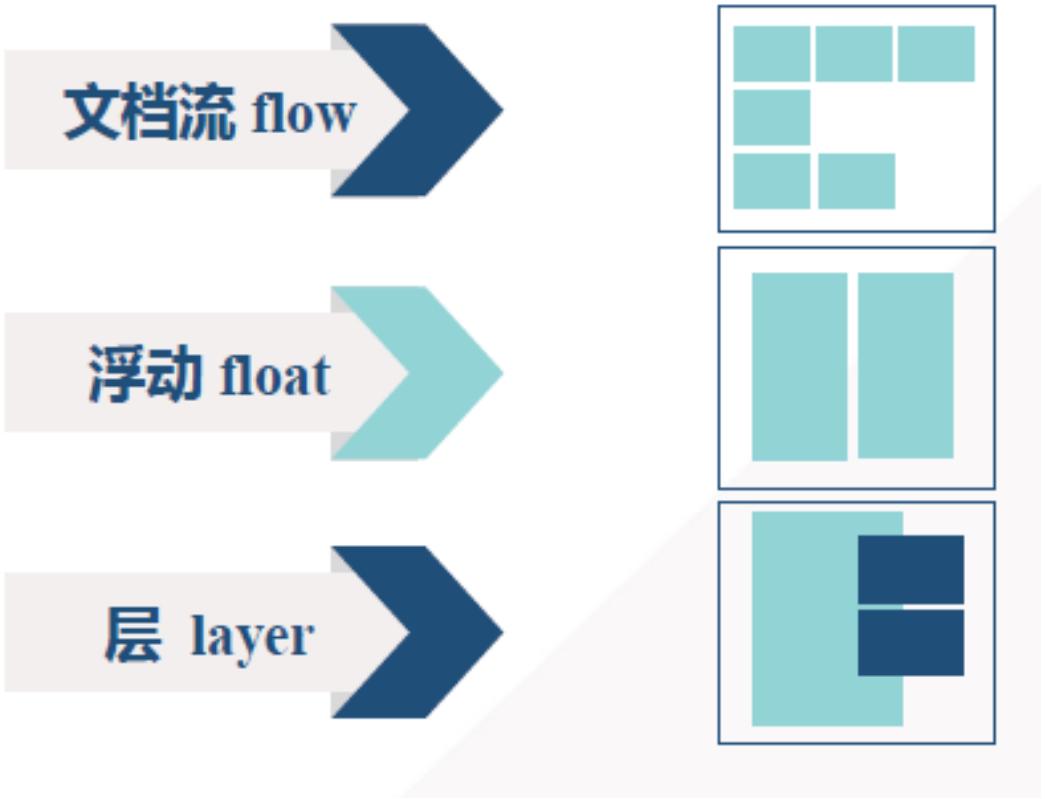
7、清除内外边距

网页元素很多都带有默认的内外边距，而且不同浏览器默认的也不一致。因此我们在布局前，首先要清除下网页元素的内外边距。

```
* {
    padding:0;        /* 清除内边距 */
    margin:0; /* 清除外边距 */
  }
```

CSS定位

分类：文档流定位、浮动定位、层定位



CSS定位

所谓的文档流: 就是指在不对页面进行任何布局控制时，浏览器默认的 HTML 布局方式。

1. **块级元素（block）**，会独占一行，从上向下顺序排列。width 和 height 属性可以发挥作用。内边距（padding），外边距（margin）和 边框（border）会将其他元素从当前盒子周围“推开”。

常用元素：div、hr、p、h1~h6、ul、ol、dl、form、table

2. **行内元素（inline）**，不会产生换行，从左到右顺序排列，碰到父元素边缘则自动换行。width 和 height 属性将不起作用。

常用元素：span、a、i、em 、strong等

以上都是文档流布局，文档流是最基本的布局方式。

CSS定位-文档流定位

- **将inline元素显示为block元素**

```
a{
    display:block;
}
```

inline元素 a 转换为 block 元素，从而使链接元素具有块状元素特点。

- **显示为inline元素**

display:inline;

CSS定位-文档流定位

- inline元素之间有一个间距问题

```
<style>
a{
    display: inline-block;
    background-color: lightskyblue;
  }
 </style>
<body>
 <a href=" https://lib.nankai.edu.cn ">nku library</a>
<a href="www.baidu.com">baidu</a>
</body>
```

CSS定位-文档流定位

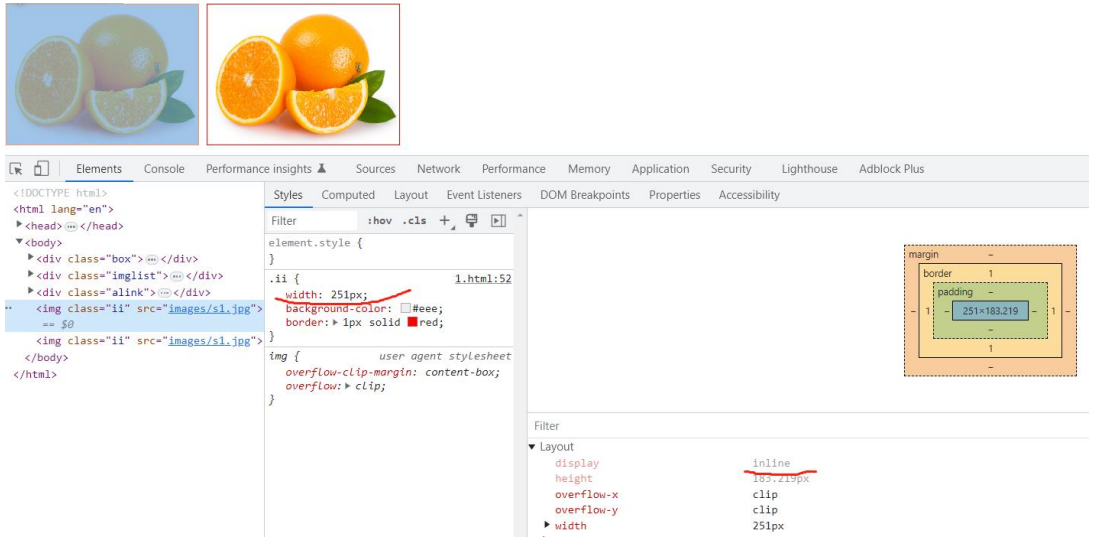
- inline-block元素

同时具备inline 元素、 block 元素的特点

- 元素同一行显示，除非内容到达父元素边缘，否则不会换行
- 而元素的height、width、margin、padding都可以设置

最典型的就是img元素。属于inline元素，但表现上更倾向于 inline-block。

通过给元素添加display:inline-block;来设置



相邻图片之间有空白间隙

HTML:

```
<div class="imglist">
  <img src="images/s1.jpg"/>
  <img src="images/s1.jpg"/>
  <img src="images/s1.jpg"/>
</div>
```

为图片所在的父元素设置 font-size:0; 的形式来解决

CSS:

```
<style>
    .imglist{
      text-align: center;
      font-size: 0; /*否则图间有间隙*/

    }
    .imglist img{
      height: 100px;
      width: 100px;
      margin-left: 5px;
      border: 1px solid #0cf;
      padding: 5px;
    }
  </style>
```

CSS定位-文档流定位


```
<ul>
    <li><a href="#">链接1</a></li>
    <li><a href="#">链接2</a></li>
    <li><a href="#">链接3</a></li>
</ul>
```

CSS定位-文档流定位

总结

元素分类：block、inline、inline-block

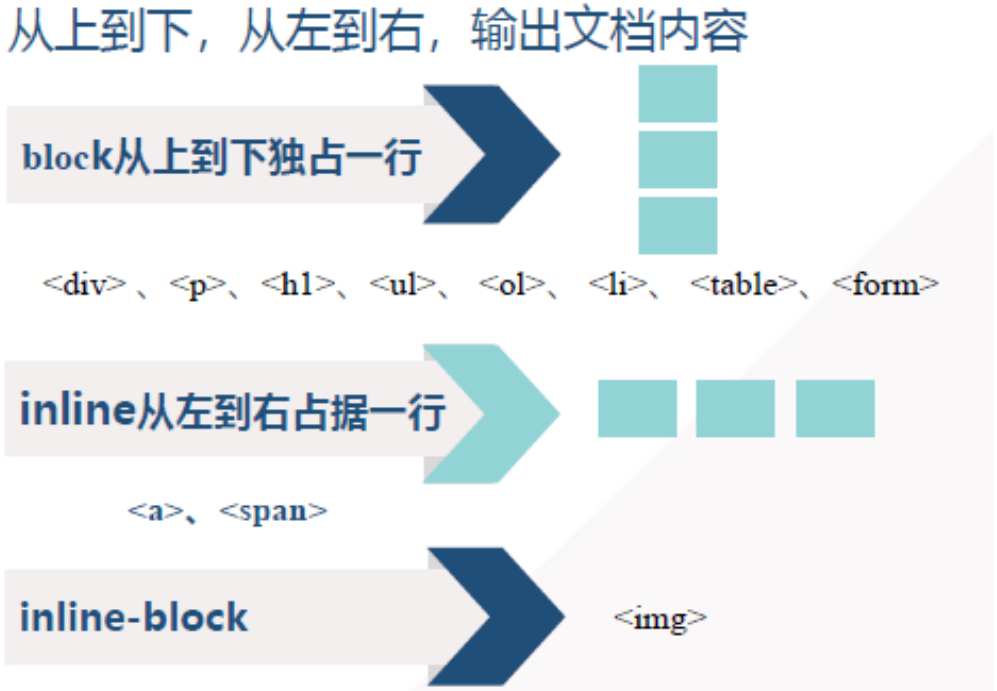
元素类型之间相互转换，通过display属性；

display:none; 元素不会被显示

display:block; 元素显示为block元素

display:inline; 元素显示为inline元素

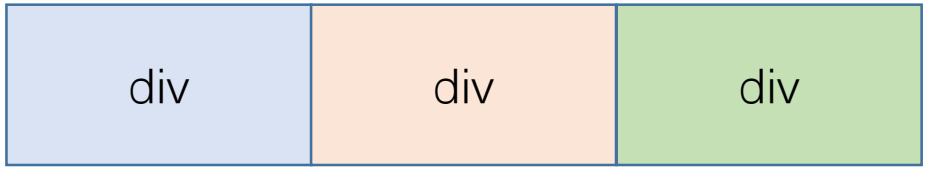
display:inline-block;元素显示为inline-block元素



CSS定位-浮动定位

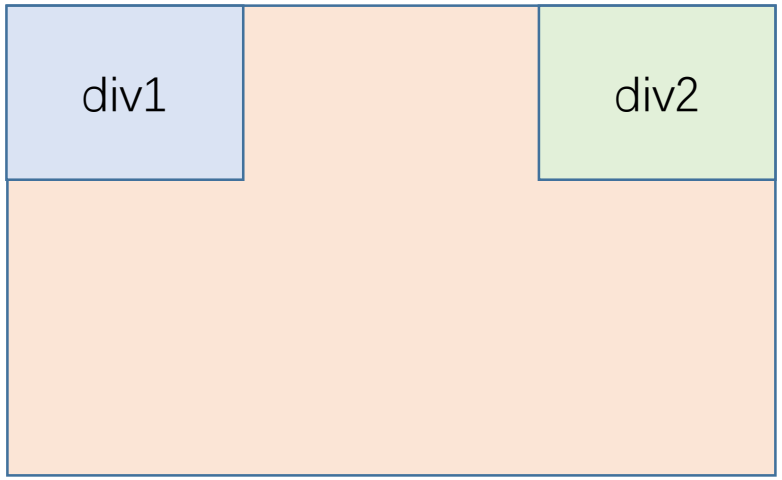
我们用标准流能很方便的实现如下效果吗?

1. 如何让多个块级元素(div)水平排列成一行?



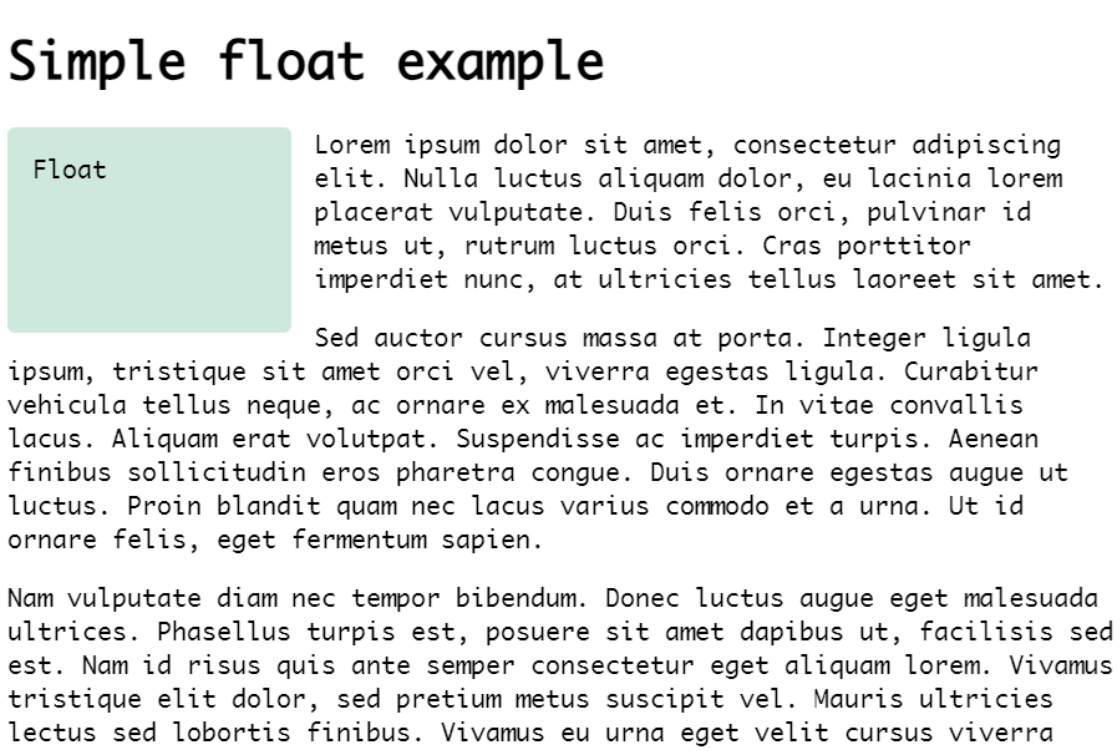
比较难，虽然转换为行内块元素可以实现一行显示，但是他们之间会有大的空白缝隙，很难控制。

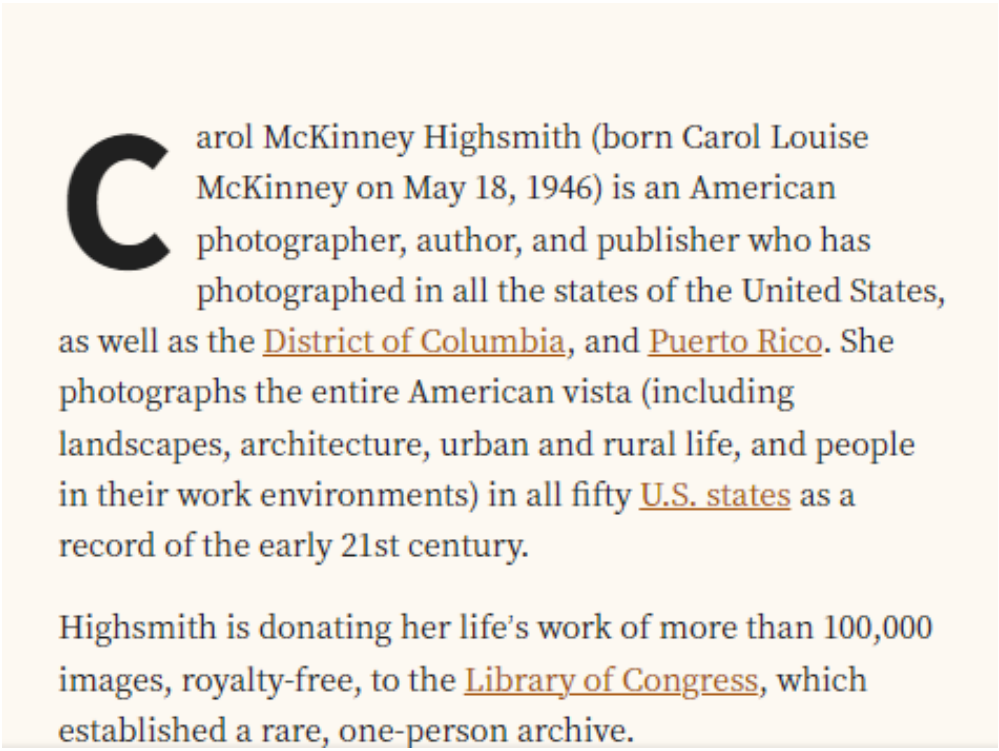
2. 如何实现两个盒子的左右对齐?



CSS定位-浮动定位

引入 float 属性是为了能让 Web 开发人员实现简单的布局，包括在一列文本中浮动的图像，文字环绕在它的左边或右边。任何东西都可以浮动，不仅仅是图像，所以我们也可以看见首字母下沉的应用场景。浮动成为在网页上创建多列布局的最常用方式之一。





CSS定位-浮动定位

有很多的布局效果，标准文档流没有办法完成，此时就可以利用浮动完成布局。因为浮动可以改变元素标签默认的排列方式。

浮动最典型的应用：可以让多个块级元素一行内排列显示。

网页布局第一准则：多个块级元素纵向排列使用标准文档流定位，多个块级元素横向排列使用浮动定位。浮动定位主要涉及两个属性，float和clear。

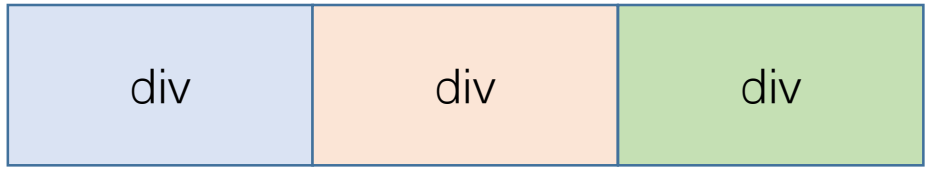
float属性用于设定浮动，创建浮动框，将其移动到一边，直到左边缘或右边缘触及包含块或另一个浮动框的边缘。语法：

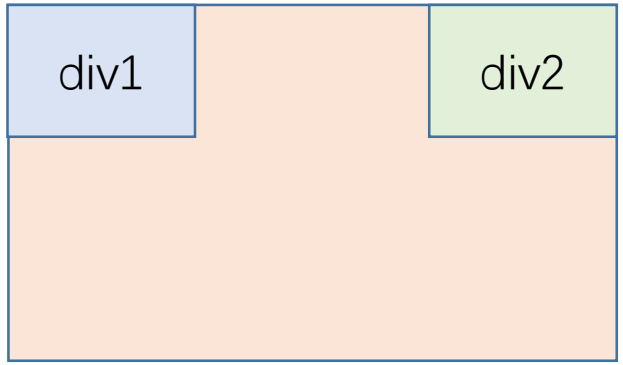
选择器 {float: none | left | right; }

属性值	描述
none	元素不浮动（**默认值**）
left	元素向**左**浮动
right	元素向**右**浮动

clear属性用于清除浮动元素的影响。

CSS定位-浮动定位






```
<style type="text/css">
    .box1,
    .box2,
    .box3 {
      float: left;
      width: 200px;
      height: 200px;
    }
    .box1 {
      border: 1px solid salmon;
      background-color: salmon;
    }
    .box2 {
      border: 1px solid lightblue;
      background-color: lightblue;
    }
    .box3 {
      border: 1px solid greenyellow;
      background-color: greenyellow;
    }
  </style>
```

CSS定位-浮动定位

加了浮动之后的元素，会具有很多特性:

1. 浮动元素会脱离标准文档流(脱标)

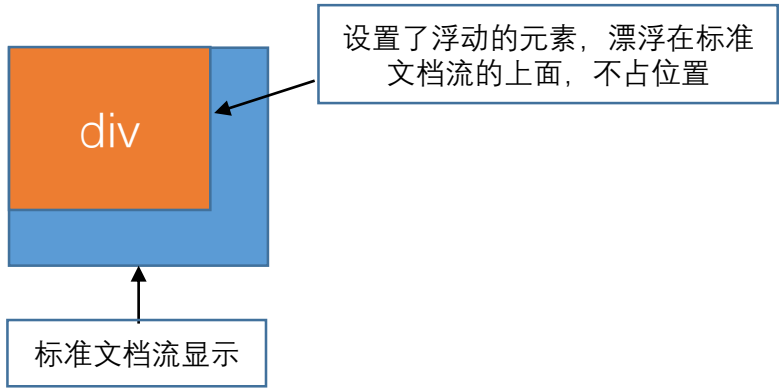
2. 浮动的元素会一行内显示并且元素顶部对齐

3. 浮动的元素会具有行内块元素的特性.

最重要特性：

脱离标准文档流的控制（浮） 移动到指定位置（动），（俗称脱标）

1. 浮动的盒子不再保留原先的位置



CSS定位-浮动定位

特性：

2. 如果多个盒子都设置了浮动，则它们会按照属性值一行内显示并且顶端对齐排列。

在一行显示，三个div都要设置浮动属性

注意： 浮动的元素是互相贴靠在一起的（不会有缝隙），如果父级宽度装不下这些浮动的盒子， 多出的盒子 会另起一行对齐。

3. 浮动元素会具有行内块元素特性。

任何元素都可以浮动。不管原先是什么模式的元素，添加浮动之后具有**行内块元素**相似的特性。

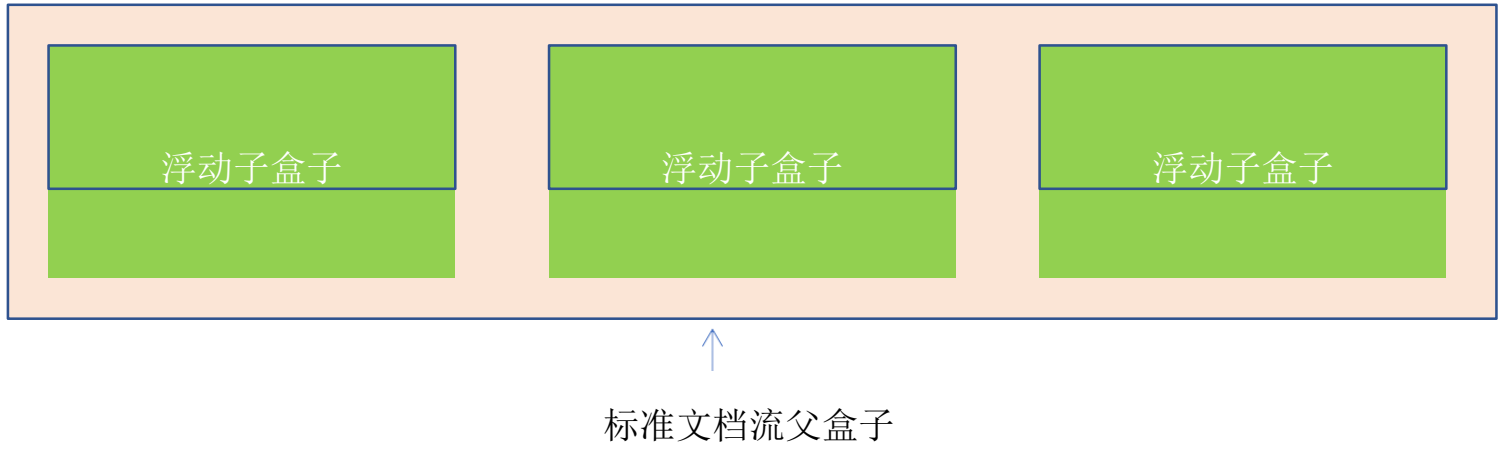
- 如果块级盒子没有设置宽度，默认宽度和父级一样宽，但是添加浮动后，它的大小根据内容来决定
- 浮动的盒子中间是没有缝隙的，是紧挨着一起的
- 行内元素同理

CSS定位-浮动定位

浮动元素经常和标准文档流父级搭配使用

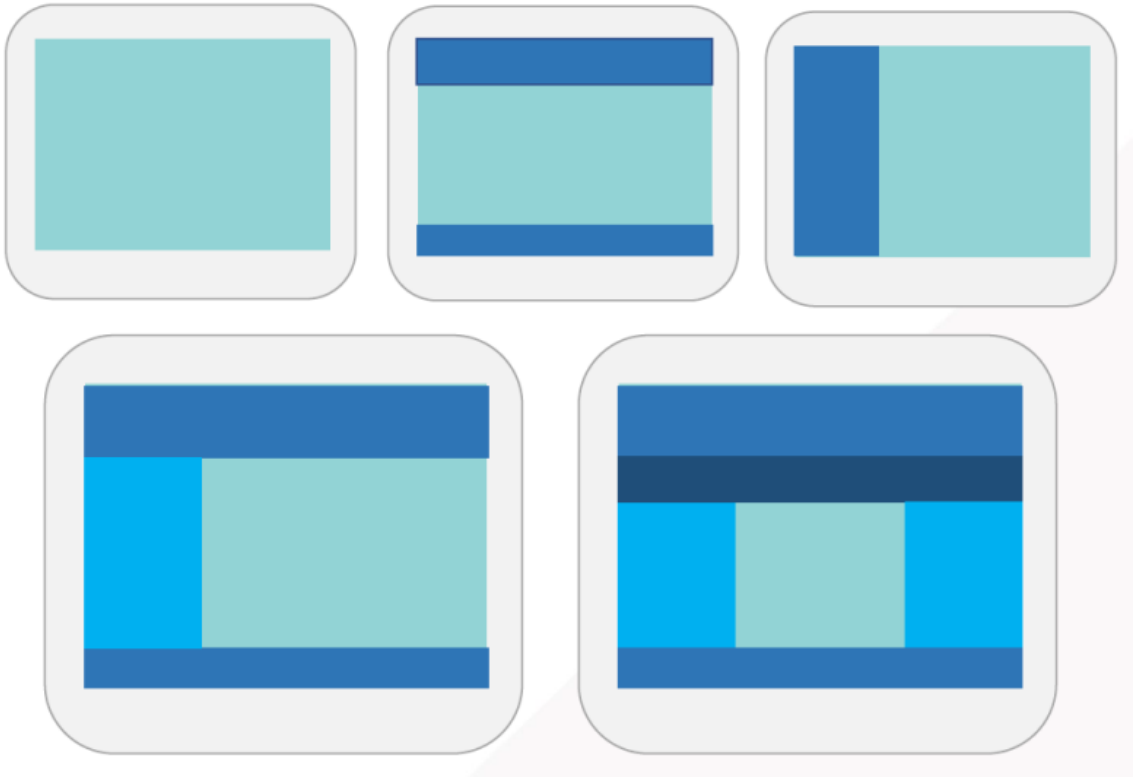
为了约束浮动元素位置, 我们网页布局一般采取的策略是:

先用标准文档流的父元素排列上下位置, 之后内部子元素再采取浮动排列左右位置. 符合网页布局第一准侧.



CSS定位-浮动定位

常见网页布局



应用浮动定位可以完成上边的布局

CSS定位-浮动定位

浮动布局注意点

1. 浮动和标准文档流的父盒子搭配。

先用标准文档流的父元素排列上下位置, 之后内部子元素采取浮动排列左右位置

2. 一个元素浮动了，理论上其余的兄弟元素也要浮动。

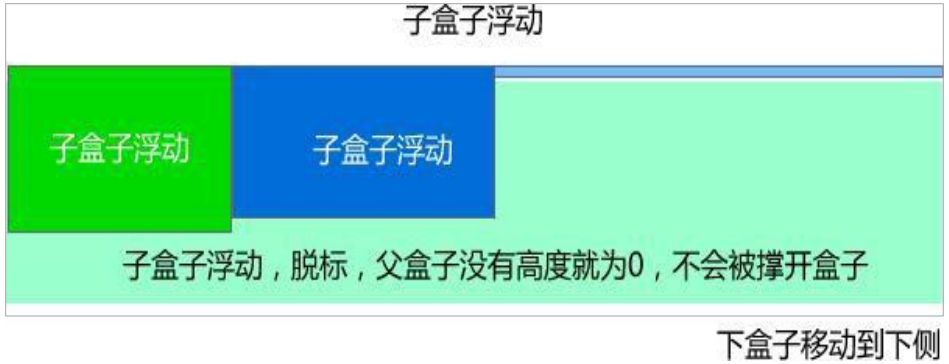
一个盒子里面有多个子盒子，如果其中一个盒子浮动了，那么其他兄弟也应该浮动，以防止引起问题。

浮动的盒子只会影响浮动盒子后面的标准文档流，不会影响前面的标准文档流.

CSS定位-浮动定位

清除浮动

由于父级盒子很多情况下，不方便给高度，但是子盒子浮动又不占有位置，最后父级盒子高度为 0 时，就会影响下面的标准文档流盒子。



- 由于浮动元素不再占用原标准文档流的位置，所以浮动元素后面的元素排版就会被影响。

清除浮动本质

- 是清除浮动元素对后边元素排列造成的影响。
- 如果父盒子本身有高度，那么不需要清除浮动。
- 清除浮动之后，父级就会根据浮动的子盒子自动检测高度。父级有了高度，就不会影响下面元素的标准文档流定位了

CSS定位-浮动定位

清除浮动

语法:

选择器{clear:属性值;}

属性值	描述
left	不允许左侧有浮动元素（清除左侧浮动的影响）
right	不允许右侧有浮动元素（清除右侧浮动的影响）
both	同时清除左右两侧浮动的影响

实际开发中， 几乎只用 clear: both;

把父盒子里浮动的盒子圈到里面，让父盒子闭合出口和入口不让他们出来影响其他元素。

CSS定位-浮动定位

清除浮动方法

方法一：使用带clear属性的空元素，

在浮动元素后使用一个空元素如 ，并在CSS中赋予.clear{clear:both;}设置即可清理浮动。

方法二：使用CSS的overflow属性，

给浮动元素的父容器添加overflow:hidden;或overflow:auto;可以清除浮动。在添加overflow属性后，浮动元素又回到了容器层，把容器高度撑起，达到了清理浮动的效果。

方法三：使用CSS的::after伪元素，

给浮动元素的容器添加一个clearfix的类名，然后给这个类添加一个::after伪元素实现元素末尾添加一个看不见的块元素清理浮动。

CSS定位-浮动定位

清除浮动方法

方法三：使用CSS的:after伪元素，

给浮动元素的容器添加一个clearfix的类名，然后给这个类添加一个:after伪元素实现元素末尾添加一个看不见的块元素清理浮动。

```
<div class="news clearfix">
    <img src="news-pic.jpg" />
    <p>some text</p>
</div>
```

```
.news {
  background-color: gray;
  border: solid 1px black;
  }

.news img {
  float: left;
  }

.news p {
  float: right;
  }
```

```
.clearfix::after{
  content: “ ";
  display: block;
  height: 0;
  clear: both;
  visibility: hidden;
  }
```

CSS 属性书写顺序(重点)

建议遵循以下顺序：

1.布局定位属性：display / position / float / clear / visibility / overflow（建议 display 第一个写，毕竟关系到模式）

2.自身属性：width / height / margin / padding / border / background

3.文本属性：color / font / text-decoration / text-align / vertical-align / white- space / break-word

4.其他属性（CSS3）：content / cursor / border-radius / box-shadow / text-shadow / background:linear-gradient …

```
.jdc {
    display: block;
    position: relative;
    float: left;
    width: 100px;
    height: 100px;
    margin: 0 10px;
    padding: 20px 0;
    font-family: Arial, 'Helvetica Neue', Helvetica, sans-serif;
    color: #333;
    background: rgba(0,0,0,.5);
    border-radius: 10px;
}
```

CSS定位-层定位

定位允许从正常的文档流布局中取出元素，并使元素具有不同的行为，例如放在另一个元素的上面，或者始终保持在浏览器视窗内的固定位置。

实现上述效果，可以使用层定位，position属性。

应用场景：

1、某个元素可以自由的在一个盒子内移动位置，并且压住其他盒子。

2、某个盒子可以固定在屏幕中某个位置。

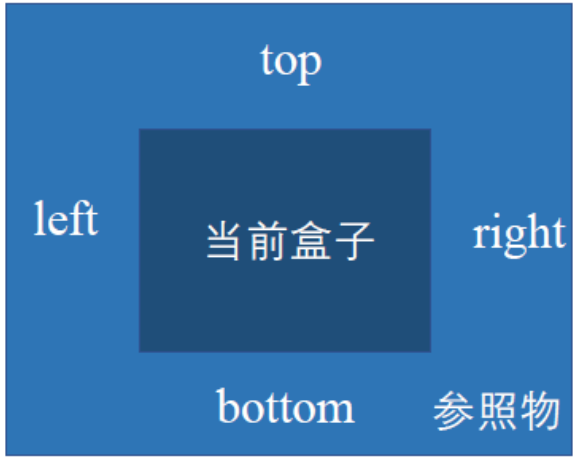
CSS定位-层定位

position属性（相对于谁定位）

1. static　　static表示“将元素放入它在文档布局流中的正常位置”，无其他特殊含义。
2. fixed　　固定定位
3. relative　　相对定位
4. absolute　　绝对定位

通过以下属性定位（位置在哪里）

top right bottom left来精确指定要将定位元素移动到的位置



CSS定位-层定位

某个元素可以自由的在一个盒子内移动位置，并且压住其他盒子。也就是
元素开始重叠，什么决定哪些元素出现在其他元素的顶部?

此时，可以使用 z-index 来控制元素的前后次序 (z轴)

语法：
选择器{z-index:1;}

- 数值可以是正整数、负整数或 0, 默认值是 auto，数值越大，元素越靠上
- 如果属性值相同，则按照书写顺序，后来居上
- z-index 只接受无单位数值
- 只有定位的元素才有 z-index 属性

CSS定位-层定位

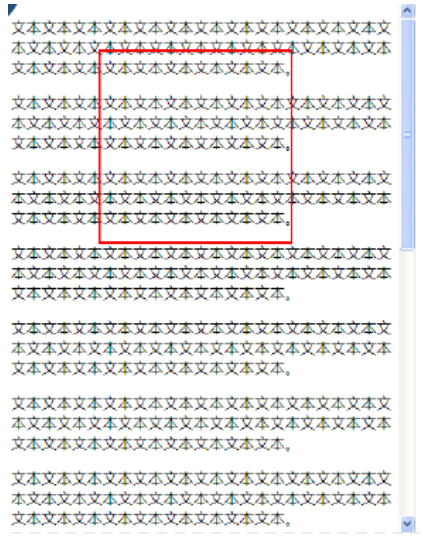
1、静态定位position: static

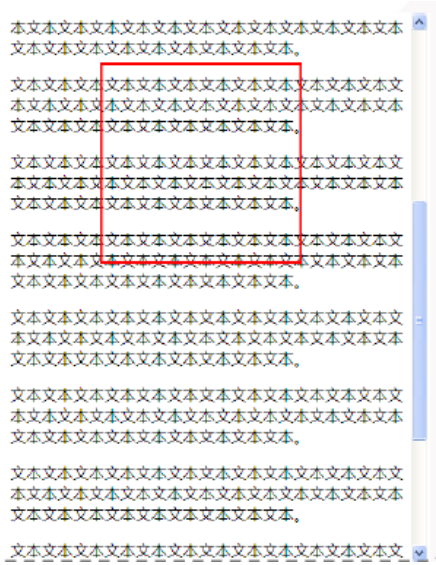
静态定位是元素的默认定位方式，无定位的意思。

- 静态定位按照标准文档流特性摆放位置，它没有边偏移
- 静态定位在布局时很少用到

2、固定定位position: fixed

固定定位固定元素则是相对于浏览器窗口本身。也就是说我们可以创建固定的有用的 UI 项目，如持久导航菜单。

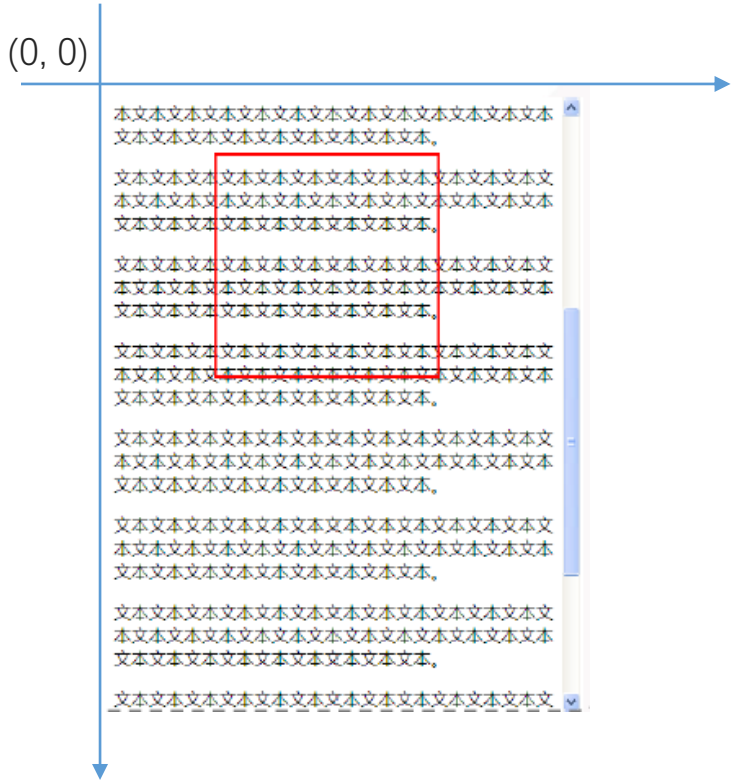




CSS定位-层定位

```
<style>
    .box {
      position: fixed;
      width: 200px;
      height: 200px;
      border: 1px solid red;

      top: 50px;
      left: 100px;
    }
  </style>
```



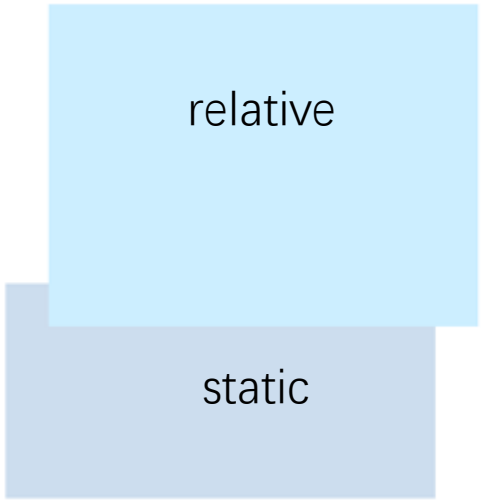
CSS定位-层定位

两侧固定的广告

CSS定位-层定位

3、相对定位position:relative

定位为relative 的元素脱离标准文档流，但其在文档流中的原位置依然存在

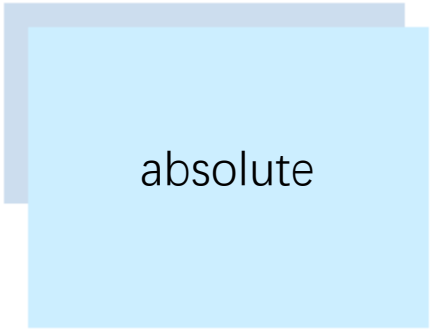



```
<style>
    .box-relative {
      position: relative;
      top: 20px;
      left: 20px;
      width: 200px;
      height: 150px;
      background-color: #cef;
    }
    .box-static {
      width: 200px;
      height: 100px;
      background-color: #cde;
    }
  </style>
</head>
<body>
  <div class="box-relative"></div>
  <div class="box-static"></div>
</body>
```

CSS定位-层定位

4、绝对定位position:absolute

定位为absolute 的元素的层脱离标准文档流，但与relative 的区别是其在正常文档流中的 原位置不再存在




```
<style>
    .box-absolute {
      position: absolute;
      top: 20px;
      left: 20px;
      width: 200px;
      height: 150px;
      background-color: #cef;
    }
    .box-static {
      width: 200px;
      height: 100px;
      background-color: #cde;
    }
  </style>
</head>
<body>
  <div class="box-absolute"></div>
  <div class="box-static"></div>
</body>
```

CSS定位-层定位

5、粘性定位position:sticky

sticky定位基本上是相对位置和固定位置的混合体，它允许被定位的元素表现的像相对定位一样，直到它滚动到某个阈值点为止（例如，从距离浏览器视口顶部30px，左侧30px起），此后它就变得固定了。例如，它可用于使导航栏随页面滚动直到特定点，然后粘贴在页面顶部。

```
.positioned {
    position: sticky;
    top: 30px;
    left: 30px;
}
```

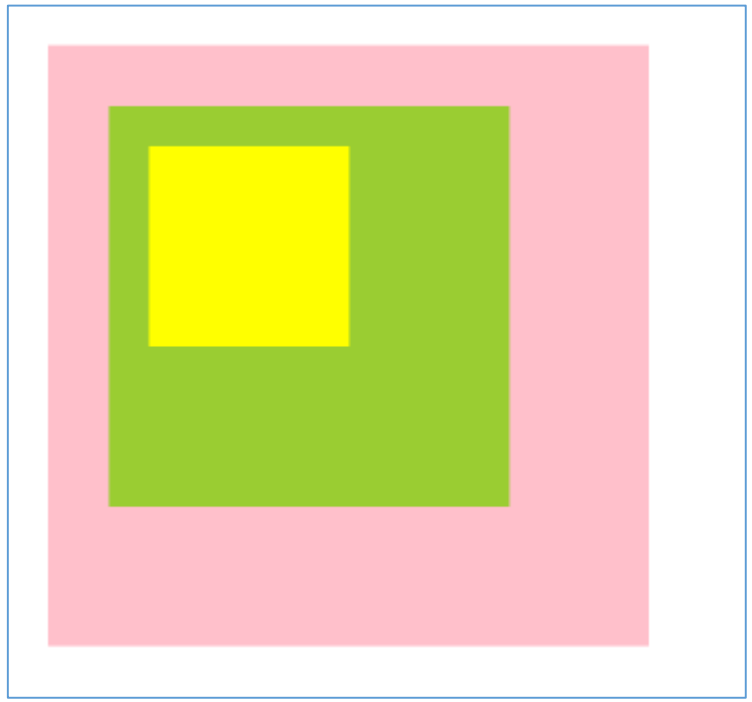
CSS定位-层定位

定位上下文---相对定位position:relative

relative定位的层总是相对于其直接父元素， 无论其父元素是什么定位方式。

```
.box1 {
    position: absolute;
    background-color: pink;
    width: 300px;
    height: 300px;
    top: 20px;
    left: 20px;
  }
  .box2 {
    position: static;
    background-color: yellowgreen;
    width: 200px;
    height: 200px;
    margin-top: 30px;
    margin-left: 30px;
  }
  .box3 {
    position: relative;
    background-color: yellow;
    width: 100px;
    height: 100px;
    top: 20px;
    left: 20px;
  }
```

```
<div class="box1">
    <div class="box2">
      <div class="box3"></div>
    </div>
  </div>
```



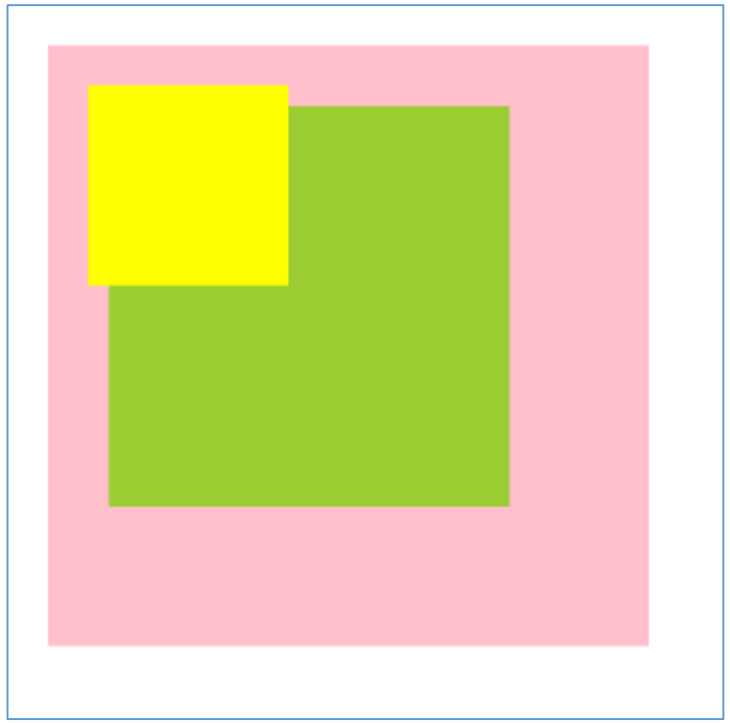
CSS定位-层定位

定位上下文---绝对定位position:absolute

对于absolute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。

```
<div class="box1">
    <div class="box2">
      <div class="box3"></div>
    </div>
  </div>
```

```
.box1 {
    position: absolute;
    background-color: pink;
    width: 300px;
    height: 300px;
    top: 20px;
    left: 20px;
  }
  .box2 {
    position: static;
    background-color: yellowgreen;
    width: 200px;
    height: 200px;
    margin-top: 30px;
    margin-left: 30px;
  }
  .box3 {
    position: absolute;
    background-color: yellow;
    width: 100px;
    height: 100px;
    top: 20px;
    left: 20px;
  }
```



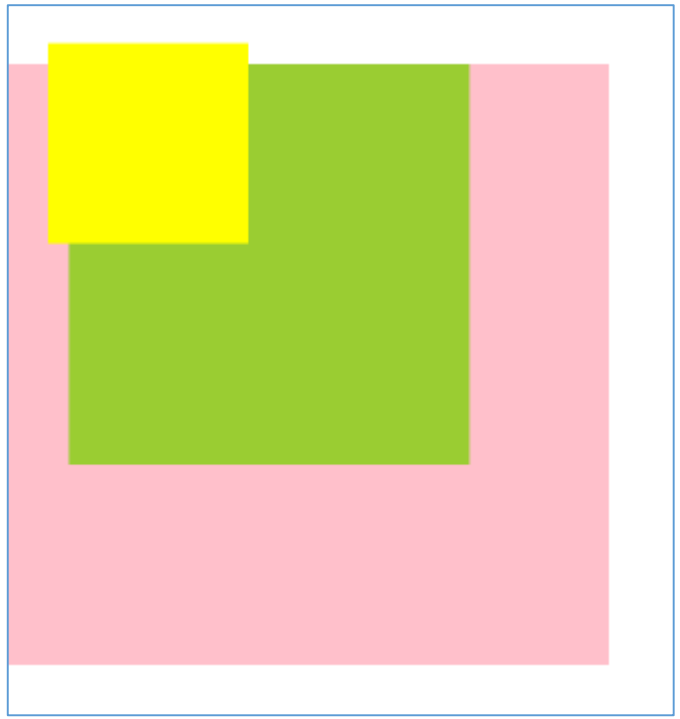
CSS定位-层定位

定位上下文---绝对定位position:absolute

对于absolute定位的层,如果其父层中都未定义absolute或relative，则绝对定位元素根据浏览器视口来定位，即相对于body来进行定位。

```
.box1{
      position:absolute;//改为static
      top:20px;
      left:20px;
}
```

box3（黄色矩形块元素）则相对于body进行定位偏移



CSS定位-层定位

值	语义	属性
static	静态定位	没有定位，元素出现在正常的标准文档流中 top, bottom, left, right, z-index 无效
fixed	固定定位	相对于浏览器窗口 进行定位 top, bottom, left, right, z-index 有效
relative	相对定位	相对于其直接父元素 进行定位 top, bottom, left, right, z-index 有效
absolute	绝对定位	相对于static 定位以外的第一个父元素进行定位 top, bottom, left, right, z-index 有效
sticky	粘性定位	允许被定位的元素表现的像相对定位一样，直到它滚动到某个阈值点为止，然后表现的像固定定位

CSS定位-层定位

父元素box1　　position:relative;

子元素box2　　position:absolute;

子元素box2　　top、 bottom 、 left 、 right 相对于父元素来进行偏移定位

```
<div class="box1">
    <div class ="box2">
    </div>
</div>
```

```
.box1{
    position:relative
}
.box2{
    position:absolute
}
```

CSS定位-层定位

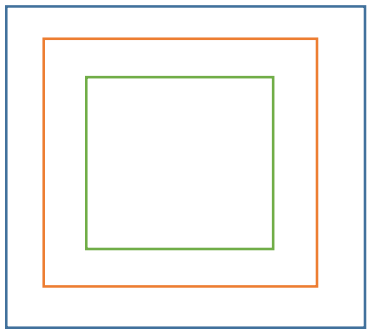
子绝父相

子级是绝对定位的话，父级要用相对定位

① 子级绝对定位，不会占有位置，可以放到父盒子里面的任何一个地方，不会影响其他的兄弟盒子。
② 父盒子需要加定位限制子盒子在父盒子内的显示。
③ 父盒子在布局时，需要占有位置，因此父盒子只能是相对定位。

相对定位经常用来作为绝对定位的父级。

当然，子绝父相不是永远不变的，如果父元素不需要占有位置，子绝父绝也会遇到。



CSS定位-层定位

子绝父相：relative+absolute

```
<div class="box1">
    <img src =“coffee.jpg”
    <div class="box2">
        一起享受咖啡带来的温暖吧
    </div>
</div>
```


```
div{
    border:1px solid red;
    color: #fff;
}
.box1{
    width:170px;
    height:190px;
    position:relative;
}
.box2{
    width:99%;
    position:absolute;
    bottom:0;
}
```

CSS定位-层定位

相对定位vs 绝对定位

	相对定位	绝对定位
position取值	relative	absolute
文档流中原位置	保留	不保留
定位参照物	直接父元素	非static的父元素

绝对定位的盒子水平居中:

加了绝对定位的盒子不能通过 margin:0 auto 水平居中，但是可以通过以下计算方法实现水平居中。

① left: 50%;：让盒子的左侧移动到父级元素的水平中心位置。

② margin-left: -100px;：让盒子向左移动自身宽度的一半（假设盒子宽度为200px）。

垂直居中同理。top:50%;margin-top:-100px;

总结

通过盒子模型，清楚知道大部分html标签是一个盒子。

通过CSS浮动、定位属性可以让每个盒子排列布局构成网页。

一个完整的网页，是标准文档流、浮动、定位一起，依据各自特性完成布局。

1. 标准文档流

可以让盒子上下排列或者左右排列，垂直的块级盒子显示就用标准文档流布局。

2. 浮动

可以让多个块级元素一行显示或者左右对齐盒子，多个块级盒子水平显示就用浮动布局。

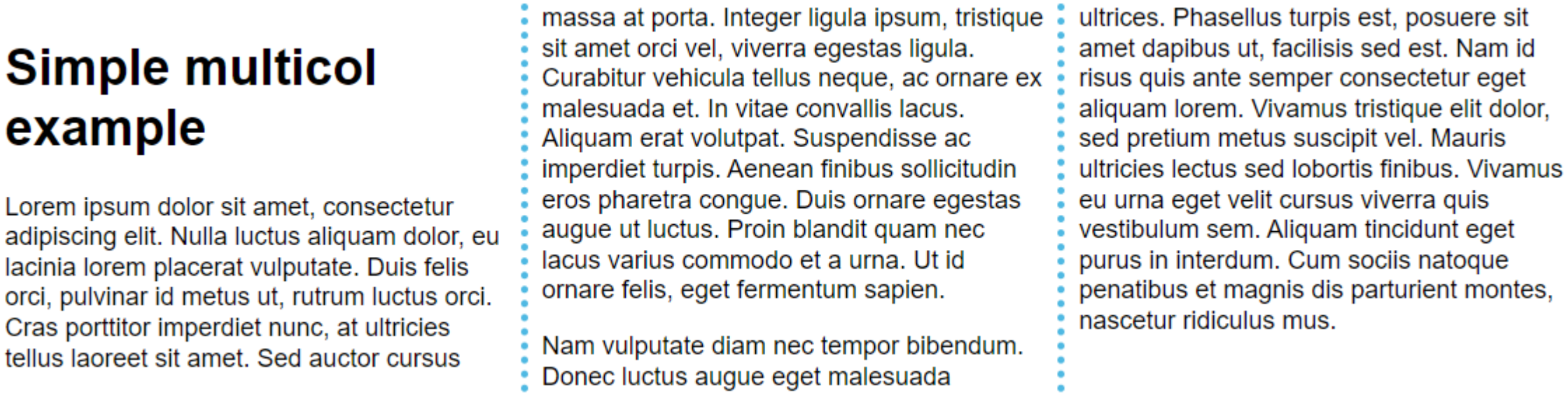
3. 定位

定位最大的特点是有层叠的概念，就是可以让多个盒子前后叠压来显示。如果元素自由在某个盒子内移动就用定位布局。

多栏式布局

- 多栏布局声明提供了一种多列组织内容的方式
- column-count 将创建指定数量的列
- column-width: 200px;浏览器将按照指定的宽度尽可能多的创建列;
- 无法单独的给某一列设定样式。
- 改变样式可以使用 column-gap 改变列间间隙。可以用 column-rule 在列间加入一条分割线。

多栏式布局




```
.container {
    column-count: 3;
    column-gap: 20px;
    column-rule: 4px dotted rgb(79, 185, 227);
  }
```

元素的显示与隐藏

类似网站广告，当我们点击关闭就不见了，但是我们重新刷新页面，会重新出现!

本质：让一个元素在页面中隐藏或者显示出来。

1. display属性

display 属性用于设置一个元素应如何显示。

display: none ； 隐藏对象

display：block ； 除了转换为块级元素之外，同时还有显示元素的意思

display 隐藏元素后，不再占有原来的位置。

2. visibility 可见性

visibility 属性用于指定一个元素应可见还是隐藏。

visibility：visible ; 元素可见

visibility：hidden; 元素隐藏

visibility 隐藏元素后，继续占有原来的位置。

如果隐藏元素想要原来位置， 就用 visibility：hidden

如果隐藏元素不想要原来位置， 就用 display：none (用处更多)

元素的显示与隐藏

3. overflow属性

overflow 属性指定了如果内容溢出一个元素的框（超过其指定高度及宽度） 时，会发生什么。

属性值	描述
visible	不剪切内容也不添加滚动条
hidden	不显示超过对象尺寸的内容，超出的部分隐藏掉
scroll	不管超出内容否，总是显示滚动条
auto	超出自动显示滚动条，不超出不显示滚动条

一般情况下，我们都不想让溢出的内容显示出来，因为溢出的部分会影响布局。

但是如果有定位的盒子， 请慎用overflow:hidden 因为它会隐藏多余的部分。

1.display：显示隐藏元素，但是不保留位置

2.visibility：显示隐藏元素，但是保留原来的位置

3.overflow：溢出显示隐藏，但是只是对于溢出的部分处理

元素的显示与隐藏

鼠标经过显示遮罩，鼠标移走隐藏遮罩

1. 练习元素的显示与隐藏
2. 练习元素的定位

核心原理： 原先半透明的黑色遮罩看不见， 鼠标经过大盒子，就显示出来。

遮罩的盒子不占有位置，就需要用绝对定位 和 display 配合使用。

元素的显示与隐藏

```
<style>
    .panda {
        position: relative;
        width: 444px;
        height: 320px;
        background-color: pink;
        margin: 30px auto;
    }

    .panda img {
        width: 100%;
        height: 100%;
    }

    .mask {
        /* 隐藏遮罩层 */
        display: none;
        position: absolute;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background: rgba(0, 0, 0, .4) url(images/arr.png) no-repeat center;
    }

    /* 当我们鼠标经过了panda这个盒子，就让里面遮罩层显示出来 */
    .panda:hover .mask {
        /* 而是显示元素 */
        display: block;
    }
</style>
```

```
<div class=“panda">
        <div class="mask"></div>
        <img src="images/panda.jpg" alt="">
    </div>
```