

css第03天

一、其他样式

1、圆角边框

原理



看看这个原理就懂了

在 CSS3 中，新增了圆角边框样式，这样我们的盒子就可以变圆角了。

border-radius 属性用于设置元素的外边框圆角。

语法：

```
border-radius:length;
```

- 参数值可以为**数值或百分比**的形式
- 如果是**正方形**，想要设置为一个圆，把**数值修改为高度或者宽度的一半即可**，或者直接写为**50%**
- 如果是**矩形** 设置为**高度的一半即可**

● 如果是个矩形，设置为高度的一半就可以做

新人福利

- 该属性是一个简写属性，可以跟四个值，分别代表**左上角、右上角、右下角、左下角**
- 分开写：border-top-left-radius、border-top-right-radius、border-bottom-right-radius 和 border-bottom-left-radius（太麻烦不建议）
- 四个数值

```
.radius {  
    width: 200px;  
    height: 200px;  
    border-radius: 10px 20px 30px 40px;  
}
```

3. 可以设置不同的圆角:



这个还可以设置成水滴形的

- **两个数值**

3. 可以设置不同的圆角:



对角线 还挺好看的

- 兼容性 ie9+ 浏览器支持, 但是不会影响页面布局, 可以放心使用

2、盒子阴影

CSS3 中新增了盒子阴影, 我们可以使用 box-shadow 属性为盒子添加阴影。

```
<title>盒子阴影</title>  
<style>  
    div {  
        width: 200px;  
        height: 200px;  
        background-color: #pink;  
        margin: 100px auto;  
        box-shadow: 10px 10px 10px 10px black  
    }  
</style>
```

Inset 可选。将外部阴影 (outset)

black

好鸭，坚持，加油



blur	可选。模糊距离。
spread	可选。阴影的尺寸。
color	可选。阴影的颜色。请参阅 CSS 颜色值。
inset	可选。将外部阴影 (outset) 改为内部阴影。

☆可以持续用这个css工具来研究作用

The screenshot shows the Chrome DevTools interface. In the Elements tab, there is a pink rectangular element with a black shadow. The Properties panel on the right shows the following CSS rules:

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div></div> == $0
  </body>
</html>
```

Styles tab (selected):

```
:hover .cls +
```

Computed tab (selected):

```
height: 200px;
background-color: pink;
margin: 10px auto;
box-shadow: 98px 10px 10px 10px black;
```

Properties tab (selected):

```
div {
  display: block;
}
```

调节第一个在水平位置上移动

第二个是上下移动

第三个模糊距离就是虚一点还是实一点

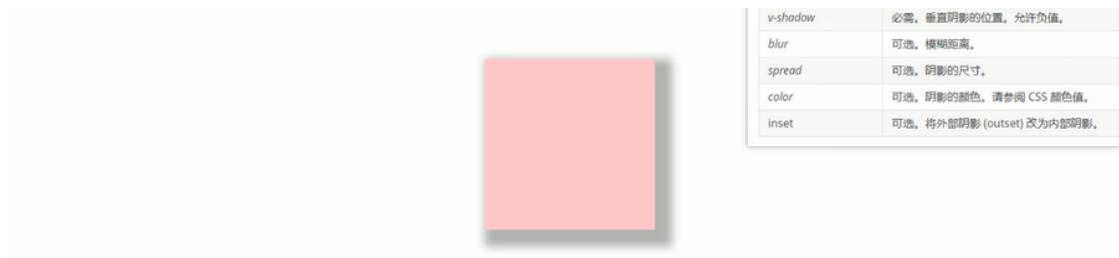
☆这个可以 其实真的也好看 我的网站可以写一个这个



v-shadow	必需。垂直阴影的位置。
blur	可选。模糊距离。
spread	可选。阴影的尺寸。
color	可选。阴影的颜色。请参阅 CSS 颜色值。
inset	可选。将外部阴影 (outset) 改为内部阴影。

第四个是影子大小

- 半透明效果



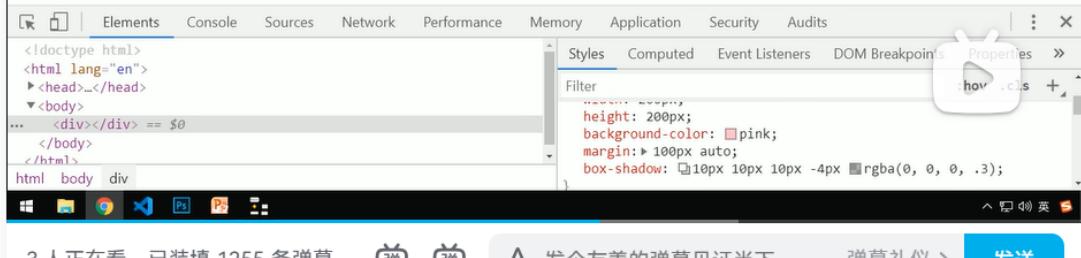
Elements tab showing the DOM structure:

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div></div> == $0
  </body>
</html>
```

Properties tab showing the styles applied to the selected element:

- height: 200px;
- background-color: pink;
- margin: 100px auto;
- box-shadow: 10px 10px 10px 10px rgba(0, 0, 0, .3);

使用rgba



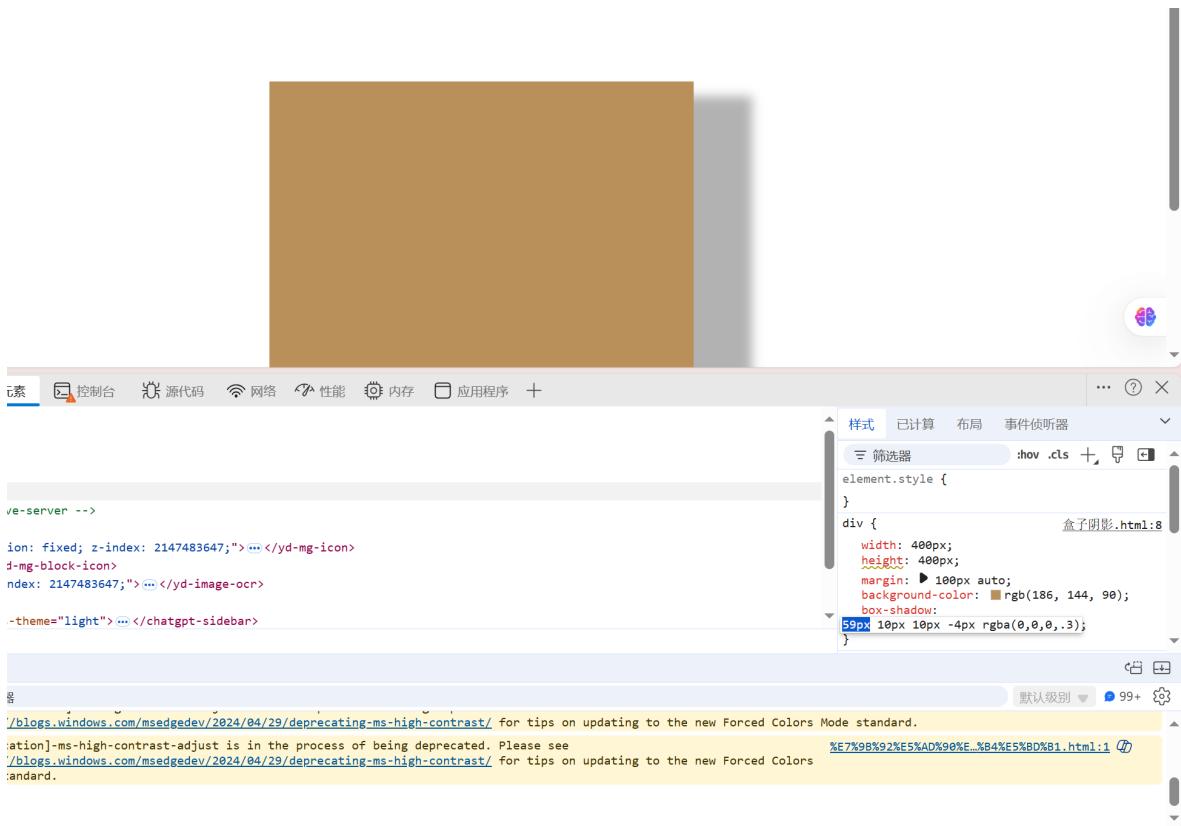
语法:

```
box-shadow: h-shadow v-shadow blur spread color inset;
```

值	描述
h-shadow	必需。水平阴影的位置。允许负值。
v-shadow	必需。垂直阴影的位置。允许负值。
blur	可选。模糊距离。
spread	可选。阴影的尺寸。
color	可选。阴影的颜色。请参阅 CSS 颜色值。
inset	可选。将外部阴影 (outset) 改为内部阴影。

- 不可以写outset 默认的写上了就不显示了
- 盒子阴影不占用空间 不影响其他盒子的排列

☆样例



按上下键就可以在界面上进行调整了

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <style>  
        div {  
            width: 400px;  
            height: 400px;  
            margin: 100px auto;  
            /* 区分background-color和color */  
            background-color: #rgb(186, 144, 90);  
        }  
        div:hover {  
            box-shadow: 10px 10px 10px -4px #rgba(0,0,0,.3);  
        }  
    </style>  
</head>  
<body>  
    <div></div>  
</body>  
</html>
```

3、文字阴影

在 CSS3 中，我们可以使用 text-shadow 属性将阴影应用于文本。

语法：

```
text-shadow: h-shadow v-shadow blur color;
```

值	描述
h-shadow	必需。水平阴影的位置。允许负值。
v-shadow	必需。垂直阴影的位置。允许负值。
blur	可选。模糊的距离。
color	可选。阴影的颜色。参阅 CSS 颜色值 。



效果好稠 不要这个

☆☆☆二、浮动 (定位? 弹性盒子)

需要掌握

目标
TARGET

- ◆ 能够说出为什么需要浮动
- ◆ 能够说出浮动的排列特性
- ◆ 能够说出 3 种最常见的布局方式
- ◆ 能够说出为什么需要清除浮动
- ◆ 能够写出至少 2 种清除浮动的方法
- ◆ 能够利用 Photoshop 实现基本的切图
- ◆ 能够利用 Photoshop 插件实现切图
- ◆ 能够完成学成在线的页面布局

1、传统网页布局的三种方式

CSS 提供了三种传统布局方式(简单说，就是盒子如何进行排列顺序)：

- 普通流 (标准流)
- 浮动

- 定位

这三种布局方式都是用来摆放盒子的，盒子摆放到合适位置，布局自然就完成了。

注意：实际开发中，一个页面基本都包含了这三种布局方式（后面移动端学习新的布局方式）。

移动端的布局方式更多

2、标准流（普通流/文档流）

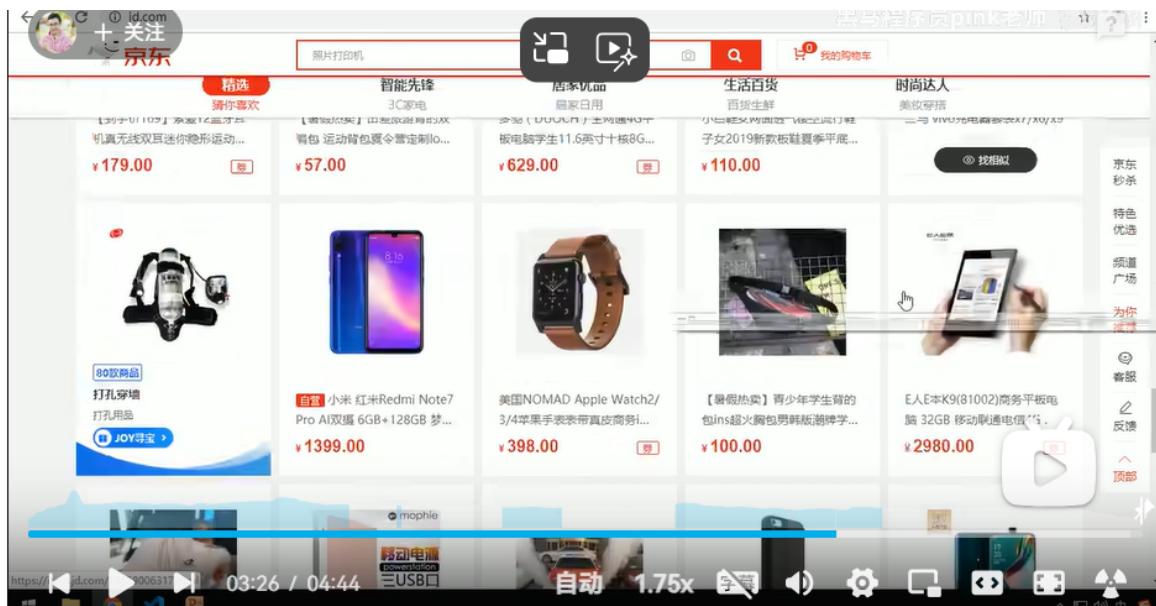
所谓的标准流：就是标签按照规定好默认方式排列

- 块级元素会**独占一行，从上向下顺序排列**。常用元素：div、hr、p、h1~h6、ul、ol、dl、form、table
- 行内元素会**按照顺序，从左到右顺序排列**，碰到父元素边缘则自动换行。常用元素：span、a、i、em等

以上都是标准流布局，我们前面学习的就是标准流，标准流是最基本的布局方式。



这种就需要使用浮动



上右不动——定位

3、为什么需要浮动？

总结：有很多的布局效果，**标准流没有办法完成**，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找标准流，多个块级元素横向排列找浮动**。

使用行内块元素也可以 但是中间默认会有空白的缝隙

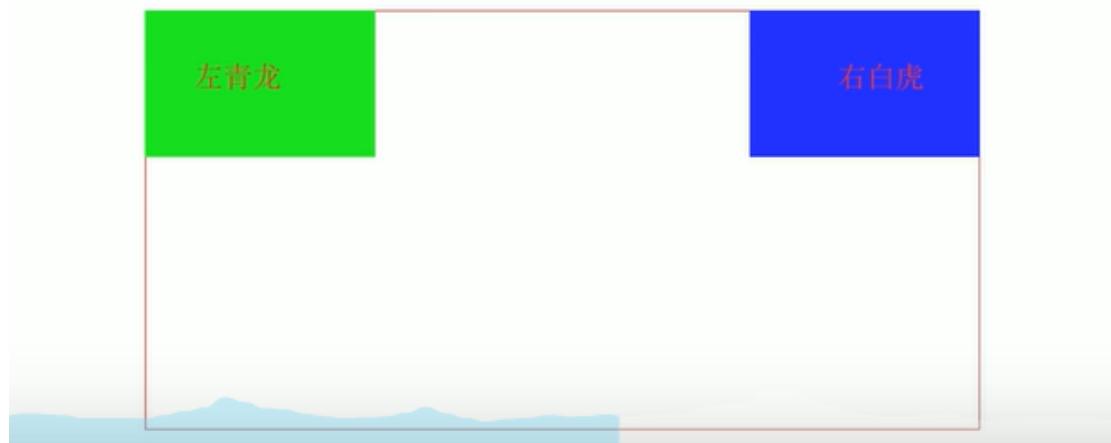


这个缝隙很麻烦 而且实际开发里经常没有缝隙 或者缝隙需要控制

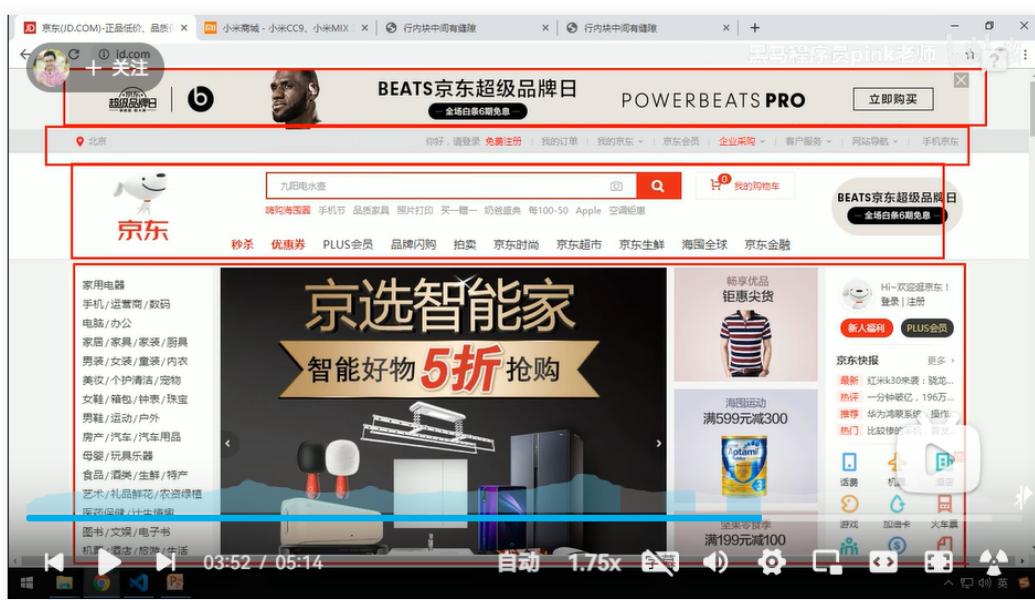
1.3 为什么需要浮动？

提问：我们用标准流能很方便的实现如下效果吗？

2. 如何实现两个盒子的左右对齐？



多个块级元素纵向排列找标准流



3 人正在看，已装填 1324 条弹幕



已关闭弹幕

弹幕礼仪 >

发送

多个块级元素横向排列找浮动



4、什么是浮动？

float 属性用于创建浮动框，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。**

形象解释：



step1 青龙left 移动到左边缘 页面



step2 白虎left 移动到紧靠白虎的浮动框



先看左浮还是右浮 然后找个边缘贴在一起 注意所有浮动都是紧紧贴着浮动的

- 注意所有浮动都是紧紧贴着浮动的!!!!!!

语法：

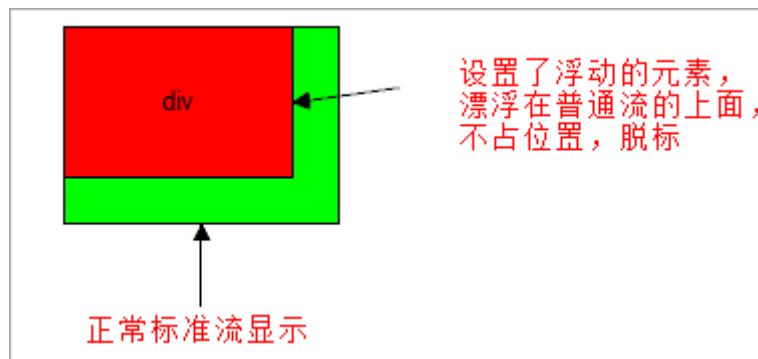
```
选择器 { float: 属性值; }
```

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

5、浮动特性(感觉像图层)

加了浮动之后的元素,会具有很多特性,需要我们掌握的.

1、浮动元素会脱离标准流(脱标：浮动的盒子不再保留原先的位置)



自写：

相当于浮动框单独看 和标准流不是一层了

比如浮动的已经飞在天上了 标准流的在底下

2. 浮动的盒子不再保留原先的位置

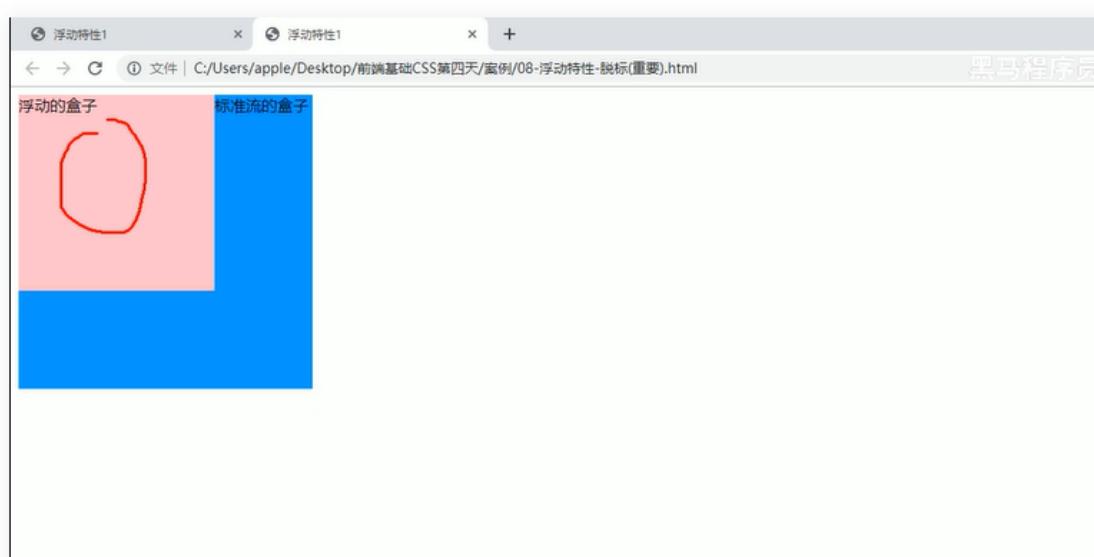
浮动元素没有了，恐怖如



除非忍不住不然不会点开弹幕的



等浮动的元素移开后 会有标准流去填补浮动走的元素的位置

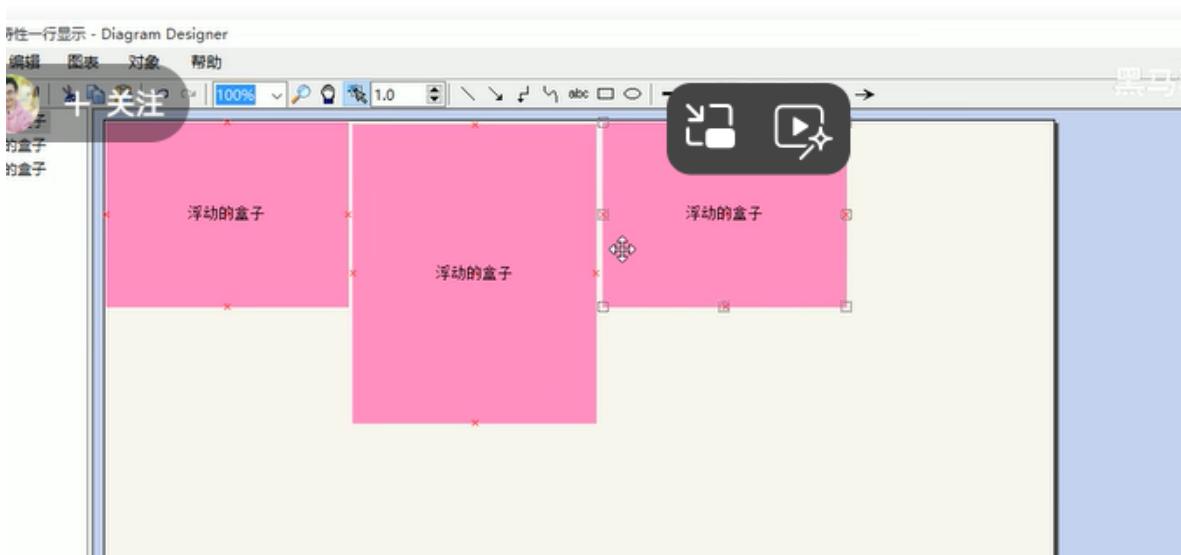


文字没有覆盖而是环绕

占据原来的位置实现叠加的效果

2、浮动的元素会一行内显示并且☆元素顶部对齐





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。

A screenshot of Visual Studio Code showing a code editor and a browser preview. The code editor has a file named "09-浮动特性-浮动盒.html" open, containing:

```
<html>
<head>
<title>浮动特性1</title>
<style>
    .box {
        width: 33.33333333333333%; // 33.33% of 100% = 33.33%
        border: 1px solid black;
        padding: 5px;
        margin: 5px;
        float: left;
    }
    .box1 { background-color: #f08080; }
    .box2 { background-color: #800080; }
    .box3 { background-color: #f0f0f0; }
    .box4 { background-color: #80c0ff; }
</style>

```

The browser preview shows four colored boxes (1, 2, 3, 4) floating next to each other. Box 4 is partially visible below the others. A red box highlights the gap between box 3 and box 4, which is circled in red, illustrating the lack of gaps between floating elements. A red arrow points from the code line `width: 33.33333333333333%;` to the highlighted gap.

3、☆浮动的元素会具有行内块元素的特性

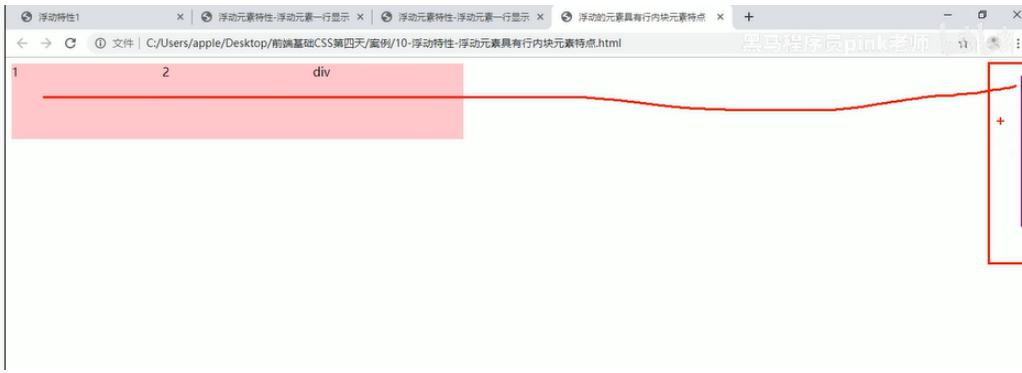
如果要设置宽高 并且需要flow

就不需要再单独设置为行内块了 已经具有行内块特点

浮动元素的大小根据内容来决定



标准流默认和父级一样宽



行内块元素的特点 加了浮动过后和内容一样宽了

浮动的盒子的中间是没有缝隙的，是紧挨在一起的

6、浮动元素经常和标准流父级搭配使用

为了约束浮动元素位置, 我们网页布局一般采取的策略是:

先用标准流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. 符合网页布局第一准则

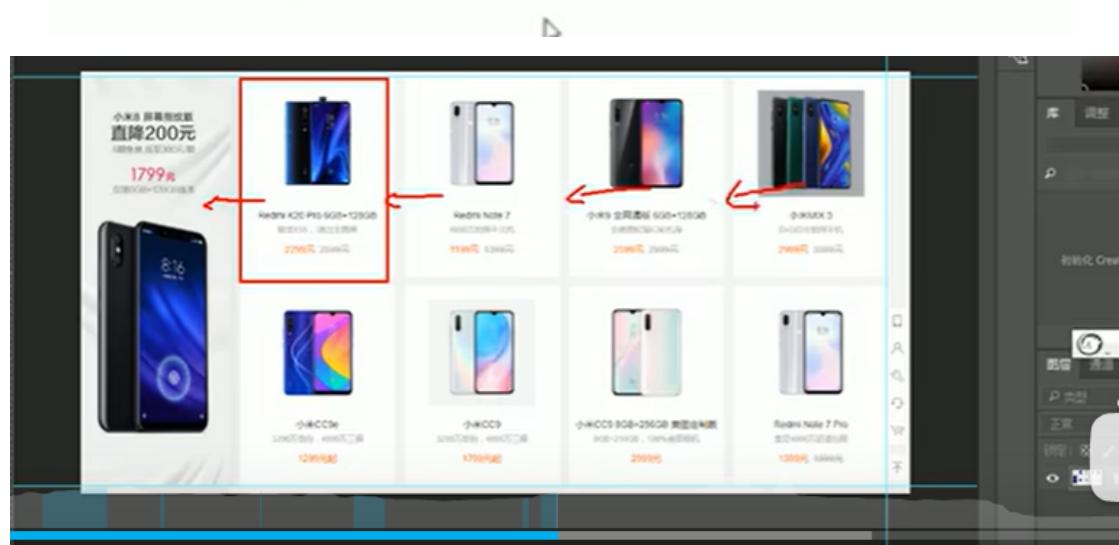


1.6 浮动元素经常和标准流父级搭配使用



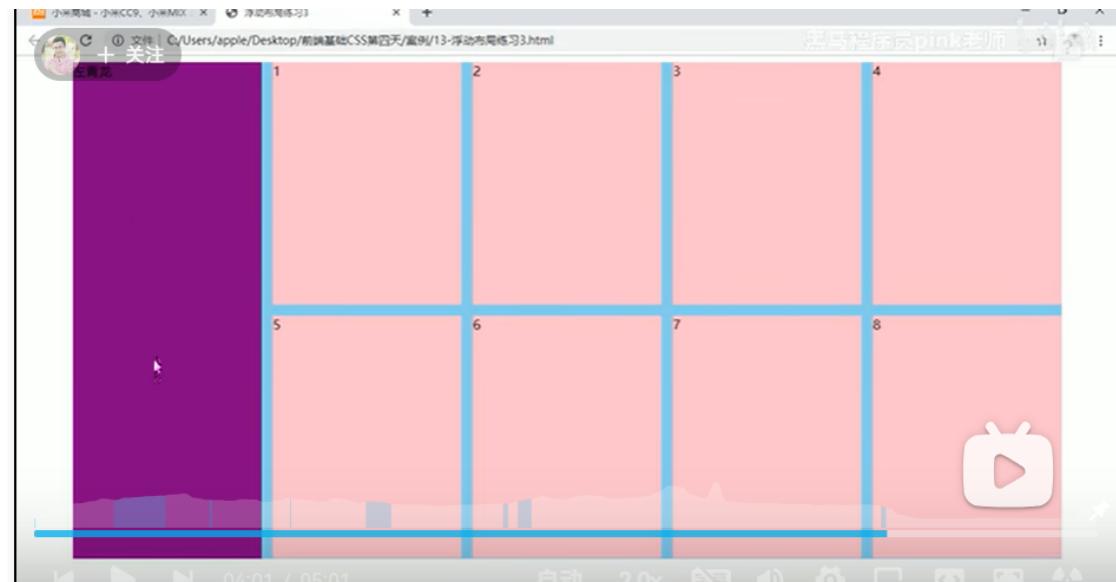
案例：小米布局案例

网页布局第二准则: 先设置盒子大小, 之后设置盒子的位置.



ul套li也可以

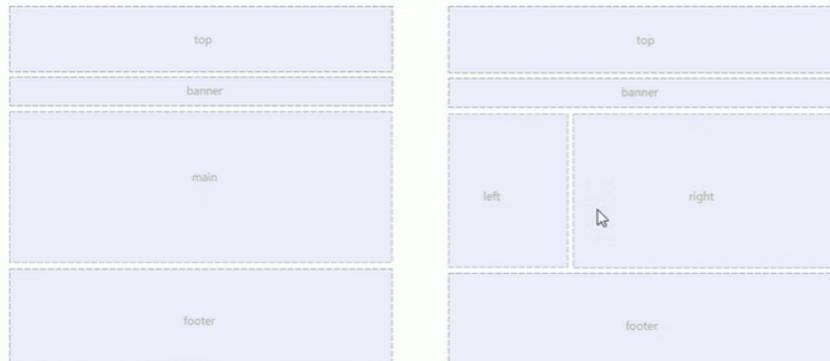
左边设置margin 这样方便全部间隔开

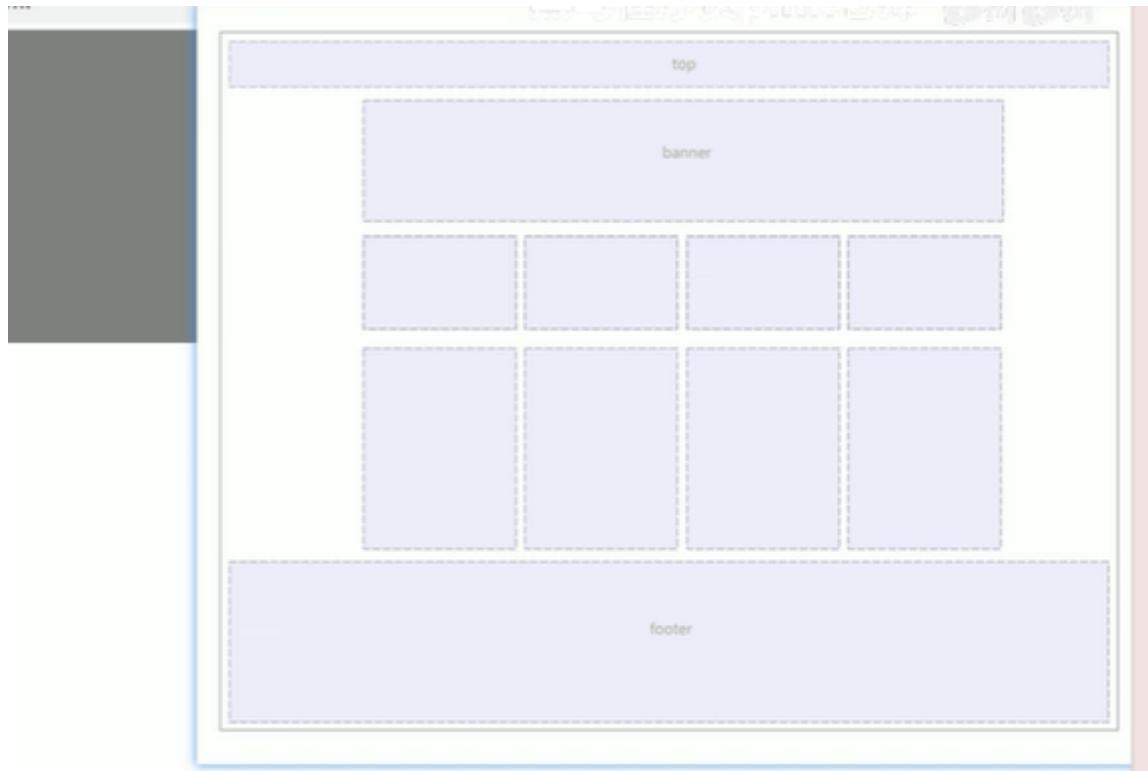


三、常见网页布局

2. 常见网页布局

2.1 常见网页布局

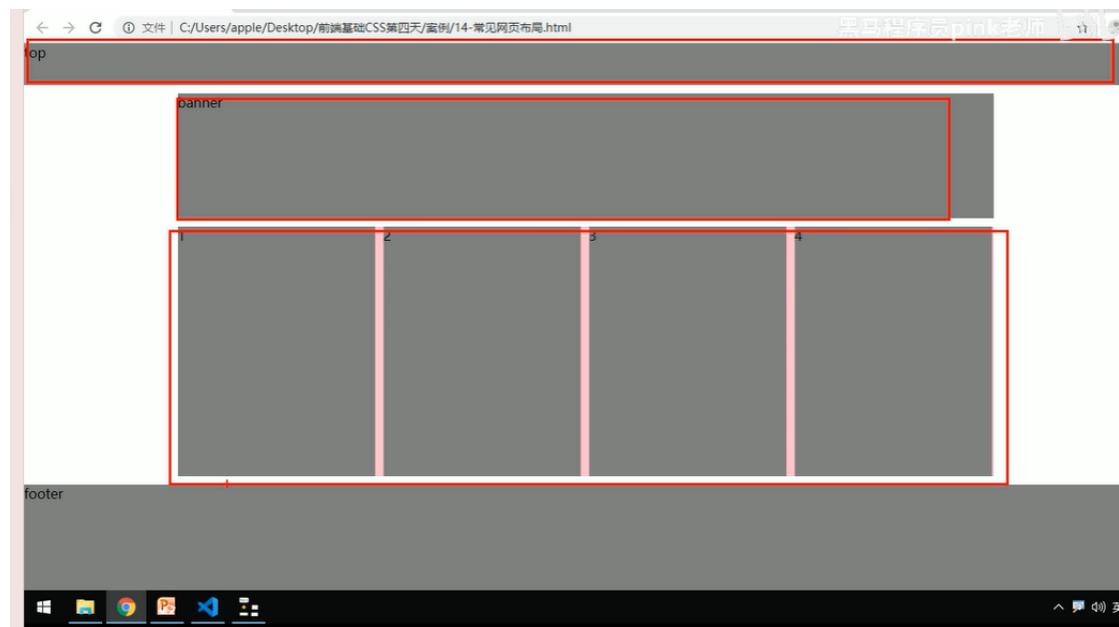




list-style:none可以去掉list前面的小黑点

可以先全部设置margin 如果最右边的因为margin而没有贴边超出格子了

单独用层叠性把那个改了就行



浮动布局注意点

1、浮动和标准流的父盒子搭配。

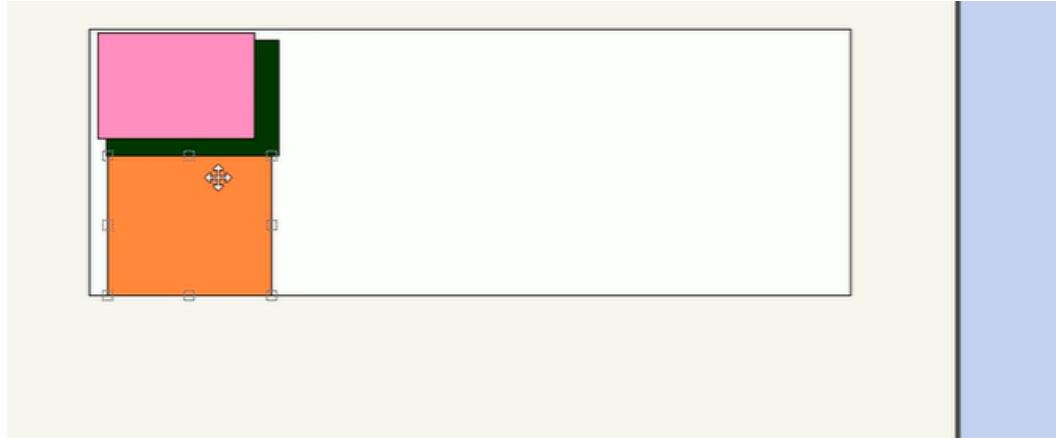
先用标准流的父元素排列上下位置,之后内部子元素采取浮动排列左右位置

2、一个元素浮动了,理论上其余的兄弟元素也要浮动。

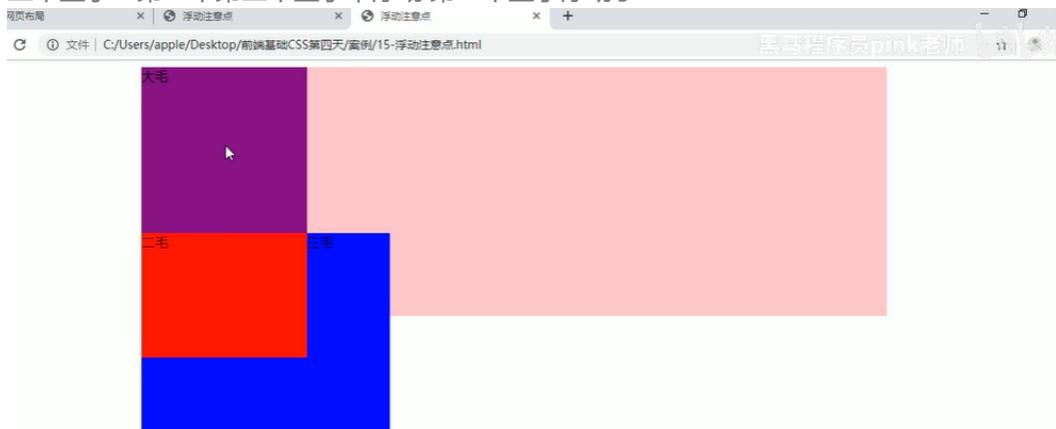
一个盒子里面有多个子盒子,如果其中一个盒子浮动了,其他兄弟也应该浮动,以防止引起问题。

一个案例:

1. 三个盒子：第一个浮动

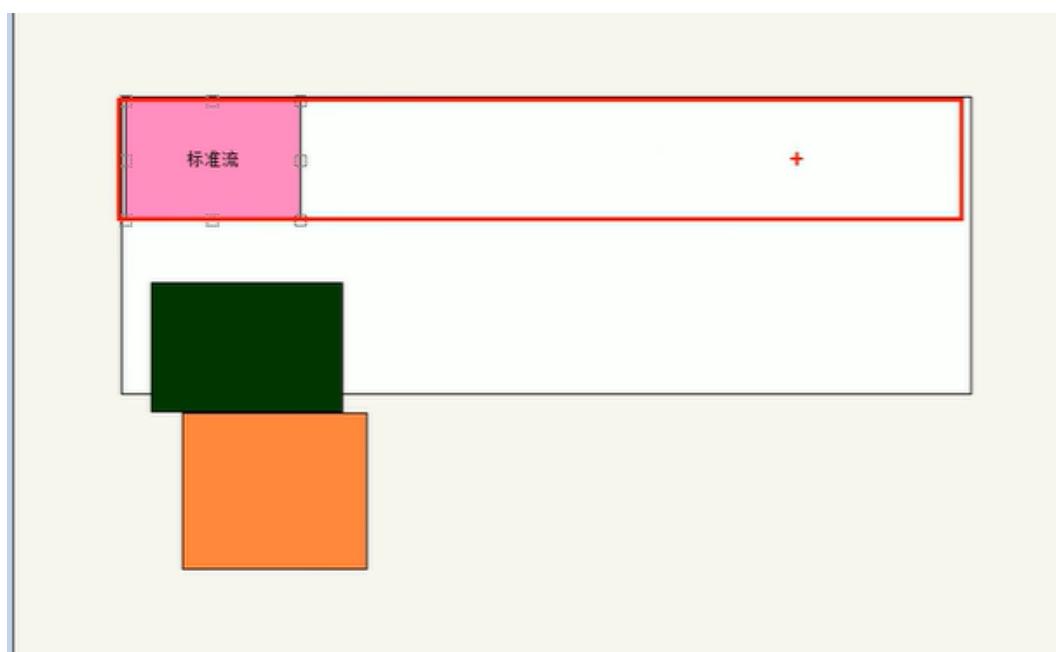


2. 三个盒子：第一个第三个盒子不浮动 第二个盒子浮动了

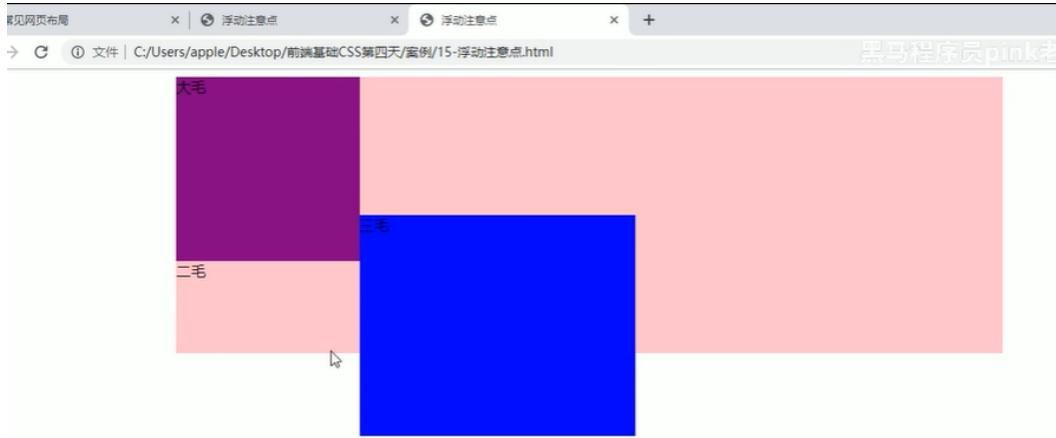


第二个盒子并没有压住第一个盒子

谁浮动谁才把位置让出来



3. 一三盒子浮动 二不浮动



二跑到一原来的位置去了然后三想和一紧挨但是

这里蓝色没有贴边的原因是紫色的高度大于粉色的高度，所以蓝色会贴紫色边显示。紫粉高度相同则不会遇到标准流的元素贴在那里了(弹幕)

反正浮动只影响后面的

字变了 环绕

类似图层但是也不完全是

浮动的盒子只会影响浮动盒子☆☆后面的标准流,不会影响前面的标准流.

四、清除浮动

1、为什么需要清除浮动？

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流盒子。

思考题：

我们前面浮动元素有一个标准流的父元素，他们有一个共同的特点，都是有高度的。

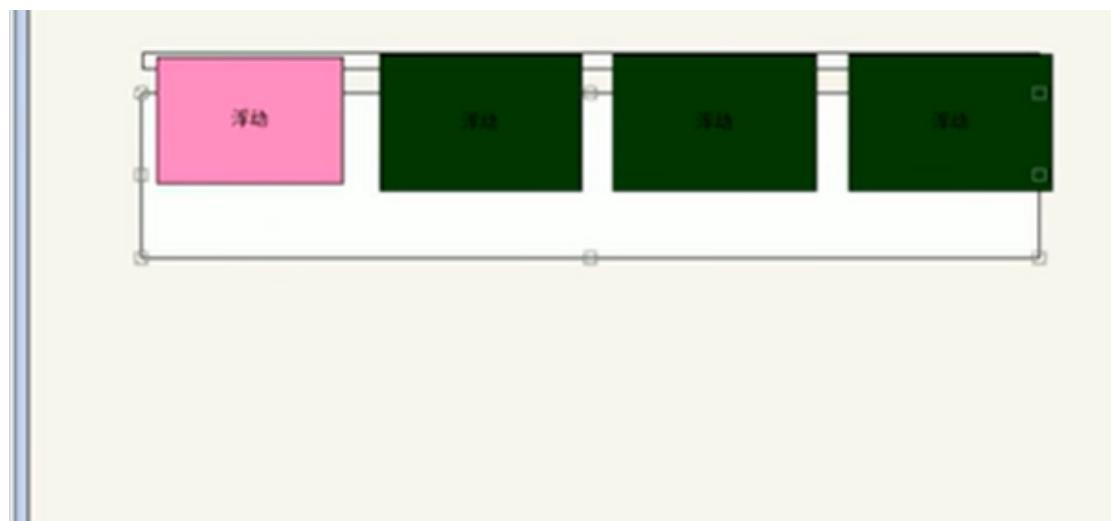
但是，所有的父盒子都必须有高度吗？



子盒子浮动

The diagram shows a container with three floating child elements. The first two children are green and blue boxes labeled "子盒子浮动". The third child is a light green box labeled "子盒子浮动，脱标，父盒子没有高度就为0，不会被撑开盒子". A horizontal bar labeled "下盒子移动到下侧" is positioned below the container.

如果不设置高度:



2、清除浮动本质

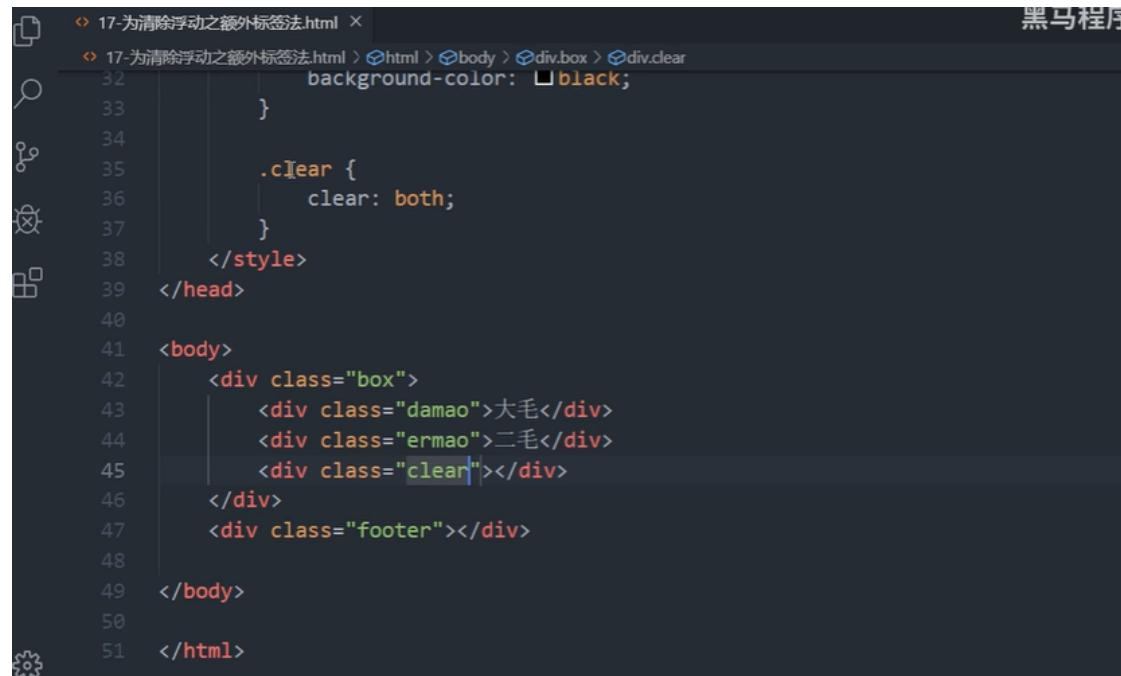
清除浮动的本质是清除浮动元素造成的影响：浮动的子标签无法撑开父盒子的高度

注意：

- 如果父盒子本身有高度，则不需要清除浮动
- 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，就不会影响下面的标准流了

3、清除浮动样式

语法：



```
17-为清除浮动之额外标签法.html
<html>
  <head>
    <style>
      .box {
        background-color: black;
      }
      .clear {
        clear: both;
      }
    </style>
  </head>
  <body>
    <div class="box">
      <div class="damao">大毛</div>
      <div class="ermao">二毛</div>
      <div class="clear"></div>
    </div>
    <div class="footer"></div>
  </body>
</html>
```

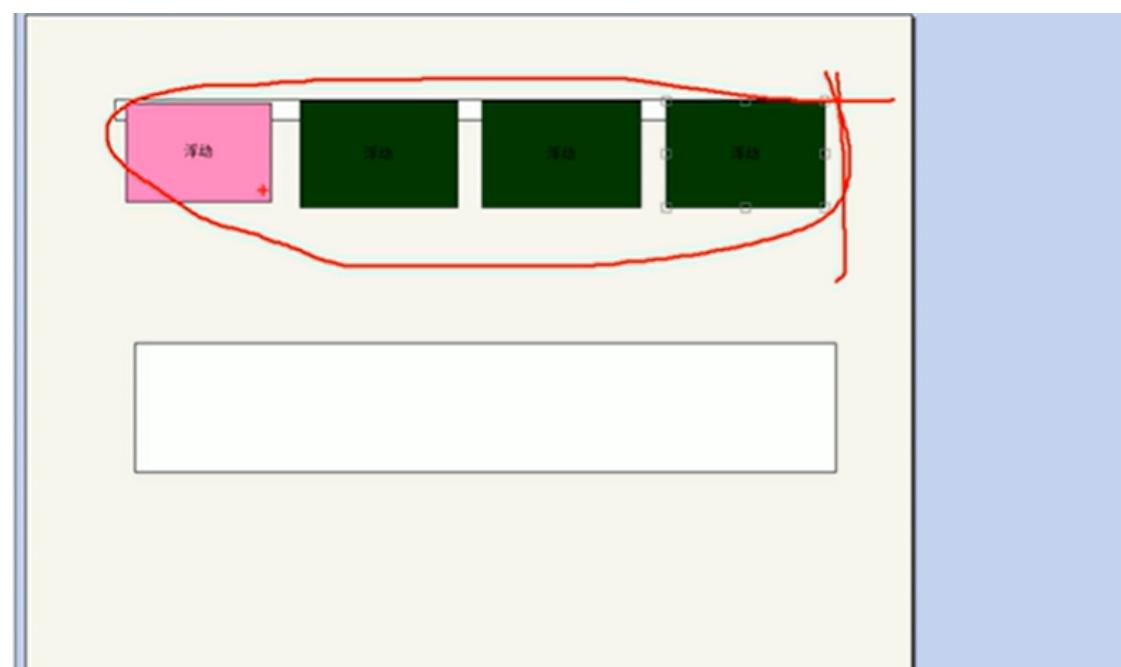
注意写法

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

我们实际工作中，几乎只用 **clear: both;**

清除浮动的策略是：闭合浮动.

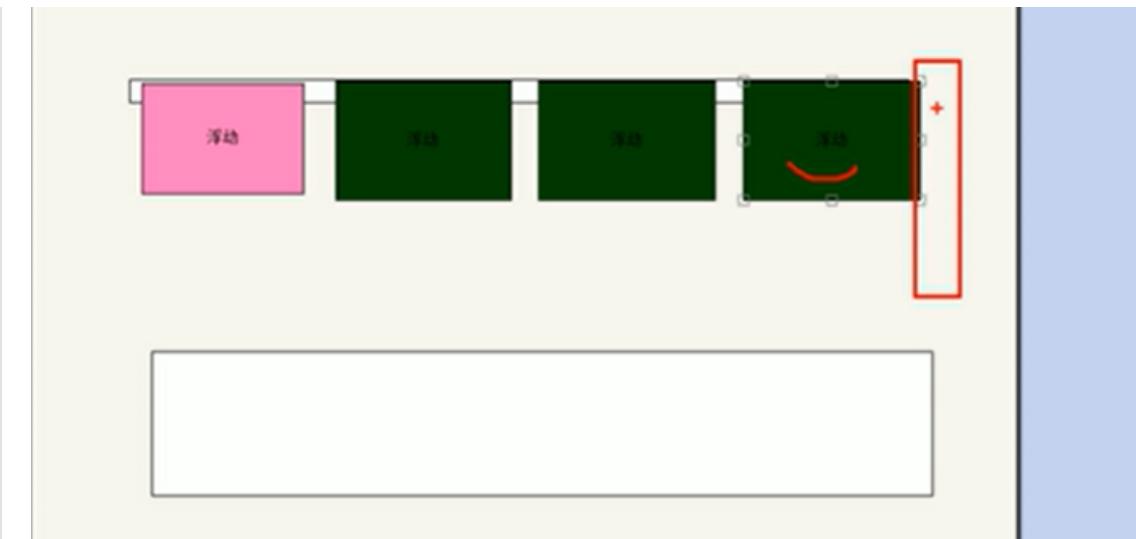


把浮动的影响闭合在父元素里

4、清除浮动的多种方式

4.1、额外标签法

额外标签法也称为隔墙法，是 W3C 推荐的做法。

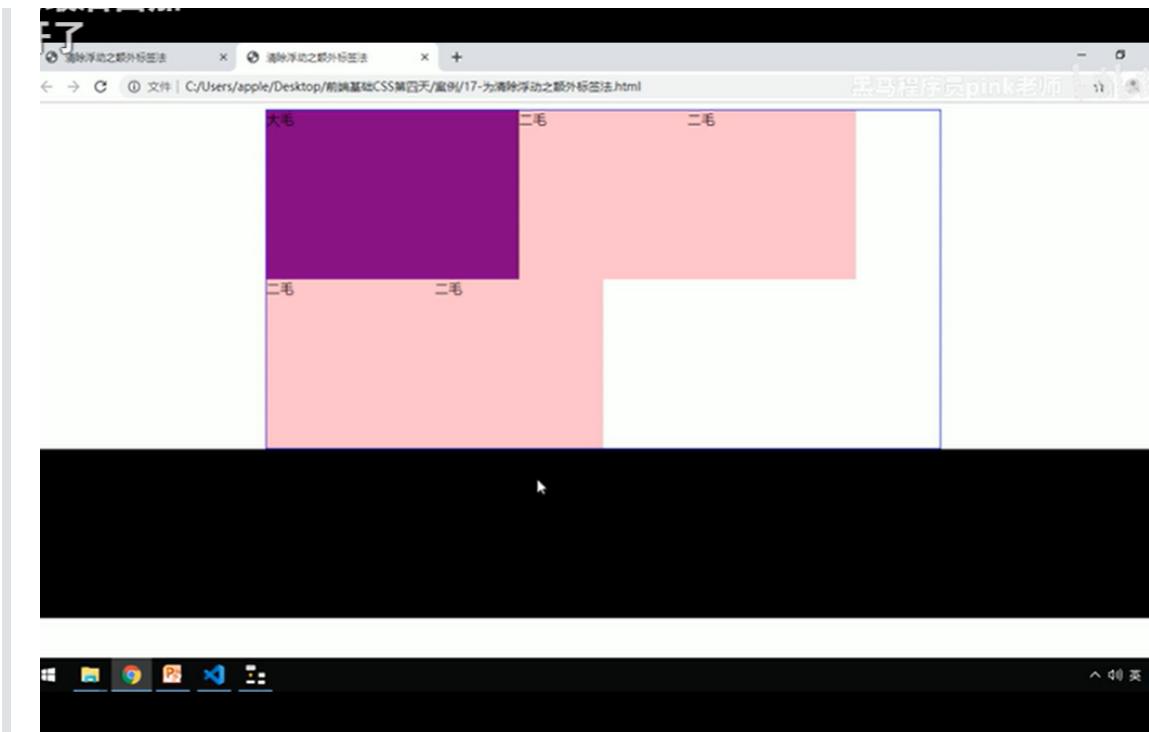


当成一堵墙堵上

```
14
15      .clear {
16          clear: both;
17      }
18  
```

```
</style>
19 </head>

20
21 <body>
22     <div class="box">
23         <div class="damao">大毛</div>
24         <div class="ermao">二毛</div>
25         <div class="clear"></div>
26     </div>
27     <div class="footer"></div>
28
29 </body>
```



使用方式：

额外标签法会在浮动元素末尾添加一个空的标签。

例如 `<div style="clear:both"></div>`, 或者其他标签（如`
`等）。

优点：通俗易懂，书写方便

缺点：添加许多无意义的标签，结构化较差

注意：要求这个新的空标签必须是块级元素。

总结：

1、清除浮动本质是？

清除浮动的本质是清除浮动元素脱离标准流造成的影响

2、清除浮动策略是？

闭合浮动. 只让浮动在父盒子内部影响,不影响父盒子外面的其他盒子.

3、额外标签法？

隔墙法, 就是在最后一个浮动的子元素后面添

4、加一个额外标签, 添加 清除浮动样式.

实际工作可能会遇到,但是不常用

4.2、☆父级添加 overflow 属性

可以给父级添加 **overflow** 属性，将其属性值设置为 **hidden**、**auto** 或 **scroll**。

```
<style>
    .box {
        overflow: hidden;
        width: 800px;
        border: 1px solid blue;
        margin: 0 auto;
    }

    .damao {
        float: left;
        width: 300px;
        height: 200px;
        background-color: purple;
    }

```

外边距合并也是用overflow:hidden; 来处理的 溢出隐藏

码



会把外部的溢出的直接去掉

例如：

```
overflow:hidden | auto | scroll;
```

优点：代码简洁

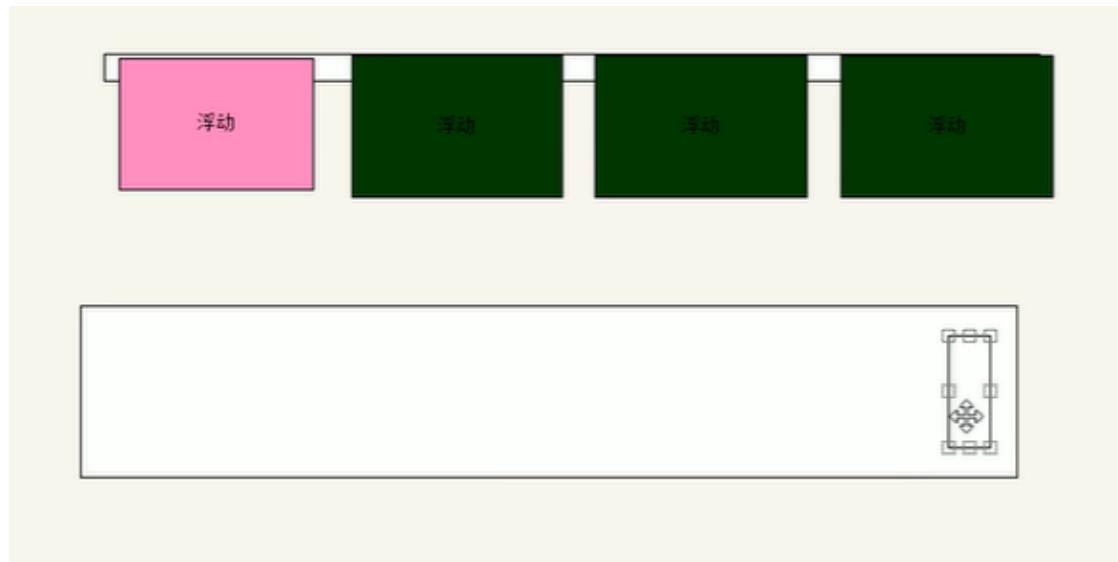
缺点：无法显示溢出的部分

注意：是给父元素添加代码

flex布局也是给父元素设置

4.3、☆父级添加after伪元素

学习了伪元素才知道 核心语句是clear both



新增了一个清除浮动的盒子after在后面

:after 方式是额外标签法的升级版。给父元素添加：

```
.clearfix:after {
  content: "";
  display: block;
  height: 0;
  clear: both;
  visibility: hidden;
}
.clearfix /* IE6、7 专有 */
*zoom: 1;
}
```

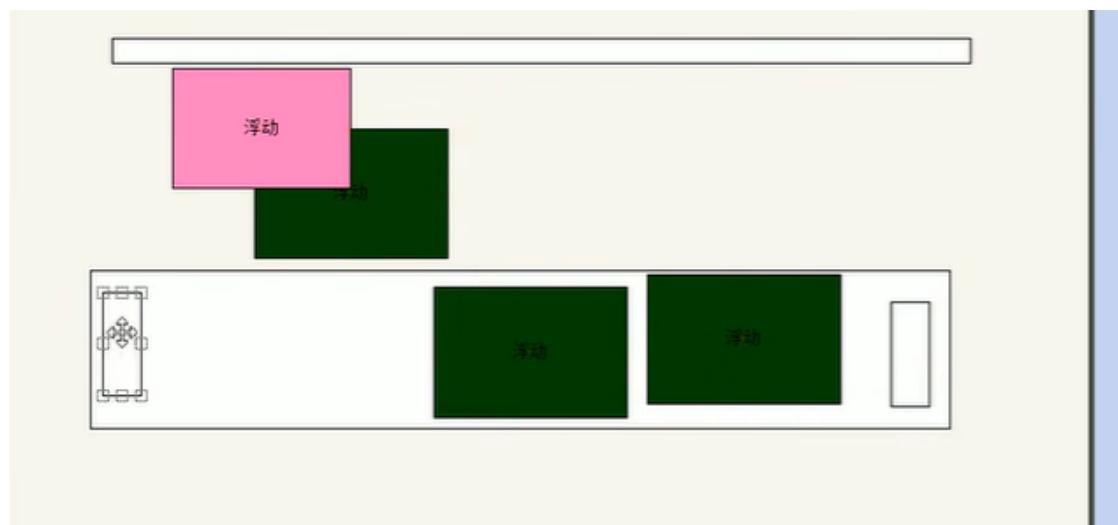
优点：没有增加标签，结构更简单

缺点：照顾低版本浏览器

代表网站：百度、淘宝网、网易等

4.4、☆父级添加双伪元素

给父元素添加



```
.clearfix:before, .clearfix:after {  
    content: "";  
    display: table;  
}  
.clearfix:after {  
    clear: both;  
}  
.clearfix {  
    *zoom: 1;  
}
```

优点：代码更简洁

缺点：照顾低版本浏览器

代表网站：小米、腾讯等

总结

为什么需要清除浮动？

1. 父级没高度。
2. 子盒子浮动了。
3. 影响下面布局了，我们就应该清除浮动了。

清除浮动的方式	优点	缺点
额外标签法（隔墙法）	通俗易懂，书写方便	添加许多无意义的标签，结构化较差。
父级overflow:hidden;	书写简单	溢出隐藏
父级after伪元素	结构语义化正确	由于IE6-7不支持:after，兼容性问题
父级双伪元素	结构语义化正确	由于IE6-7不支持:after，兼容性问题

五、（没听这个 可以补一下）PS 切图

1、图层切图

最简单的切图方式：右击图层 → 导出 → 切片。

2、切片切图

2.1、利用切片选中图片

利用切片工具手动划出

2.2、导出选中的图片

文件菜单 → 存储为 web 设备所用格式 → 选择我们要的图片格式 → 存储。

3、PS插件切图

Cutterman 是一款运行在 Photoshop 中的插件，能够自动将你需要的图层进行输出，以替代传统的手工“导出 web 所用格式”以及使用切片工具进行挨个切图的繁琐流程。

官网: <http://www.cuttermen.cn/zh/cuttermen>

注意: Cuttermen 插件要求你的 PS 必须是完整版, 不能是绿色版, 所以大家需要安装完整版本。

