



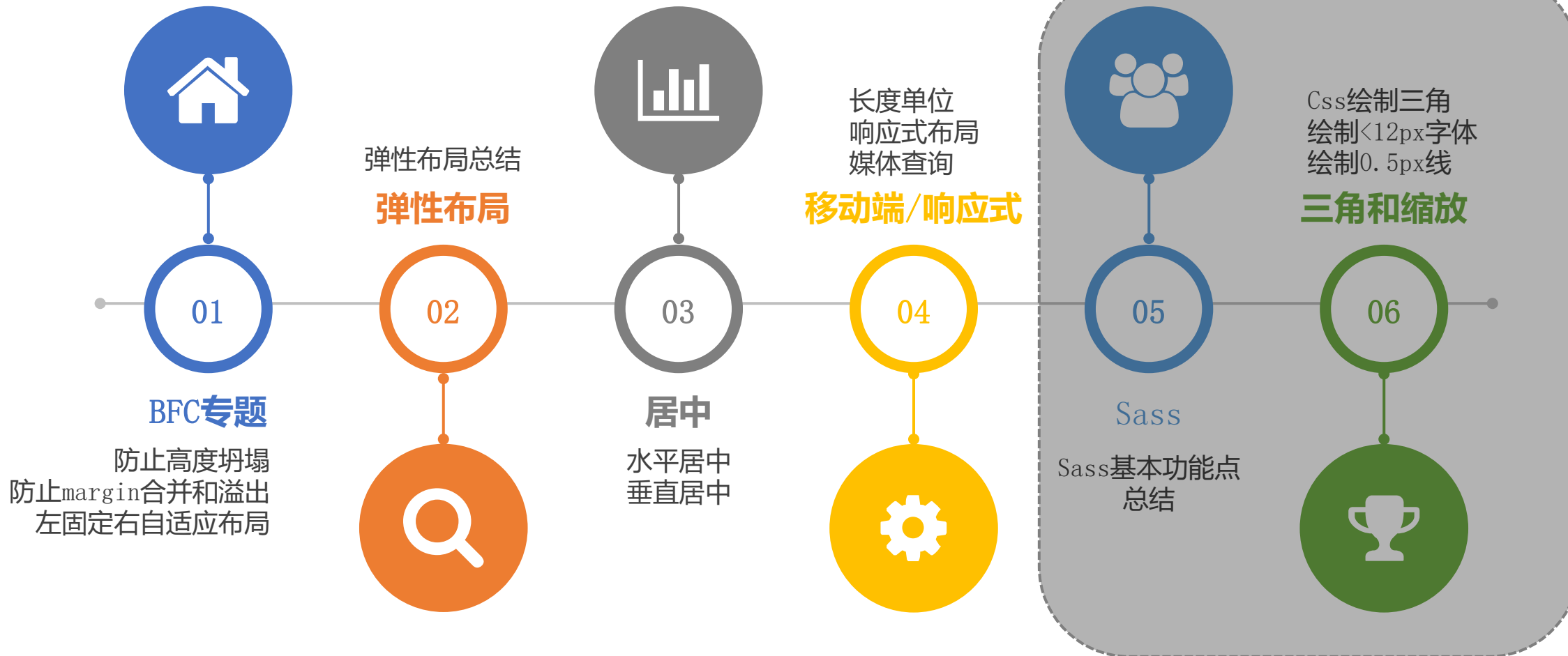
CSS高频笔试题精讲

Web学科教学总监

张东（东神）



高频笔试题包括





Q1: BFC专题



“

1. 防止高度坍塌

4种方案

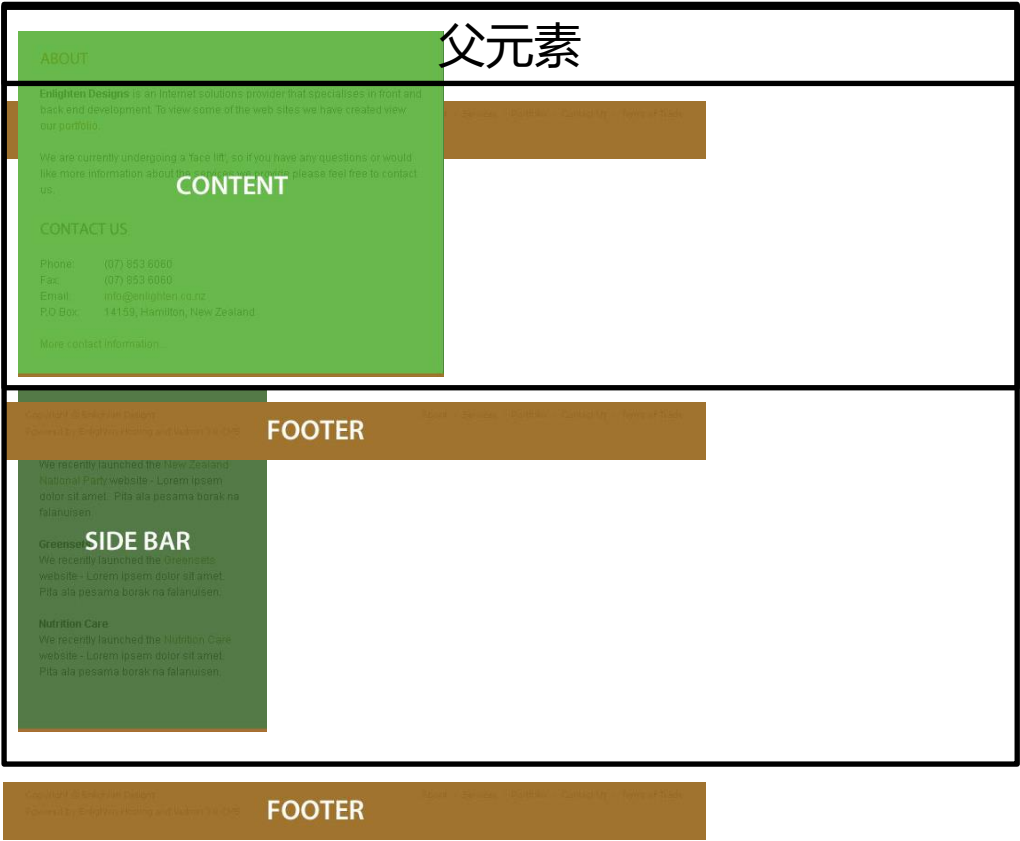
”





问题重现:

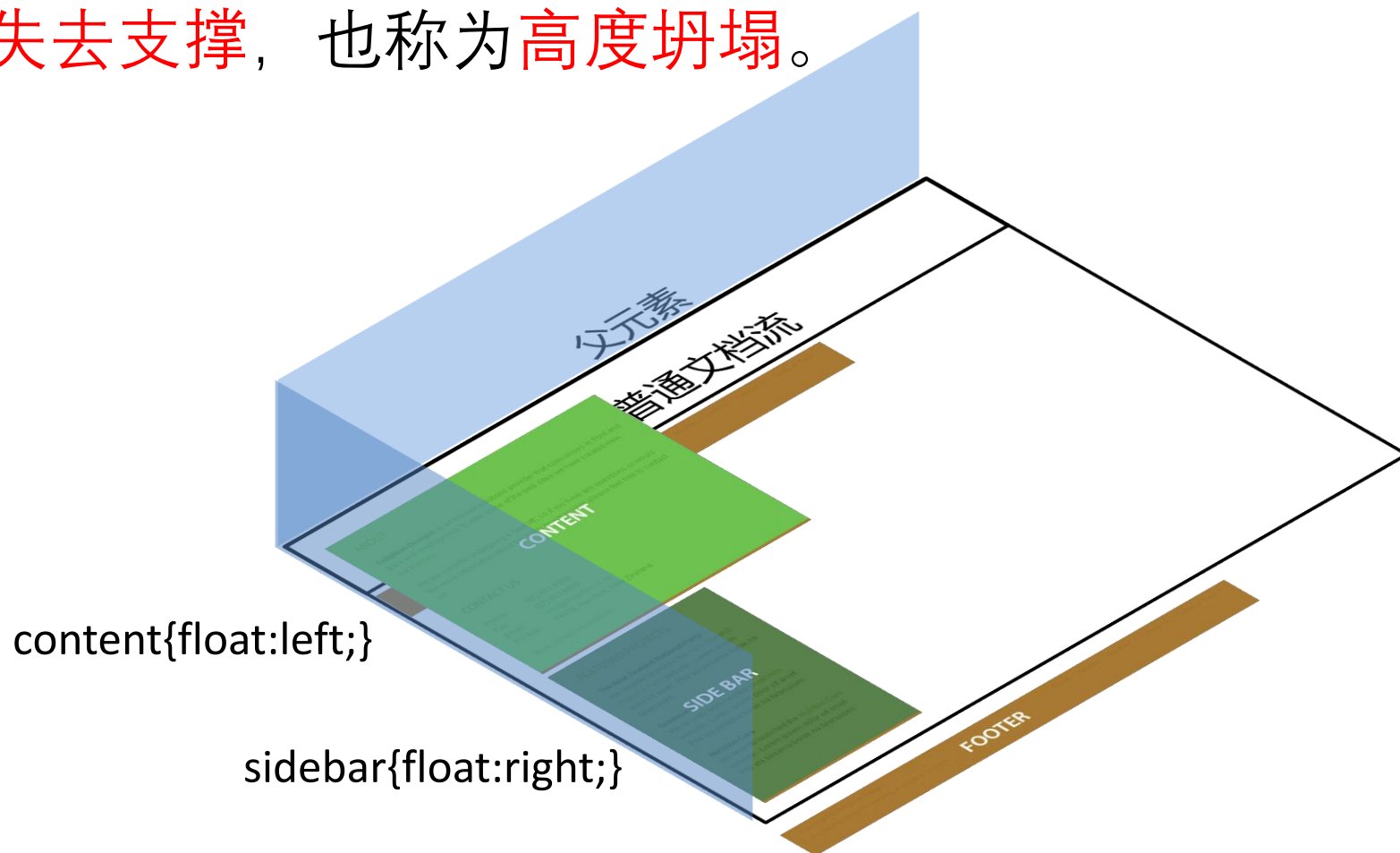
- 父元素的高度，都是由内部未浮动子元素的高度撑起的。



问题重现:

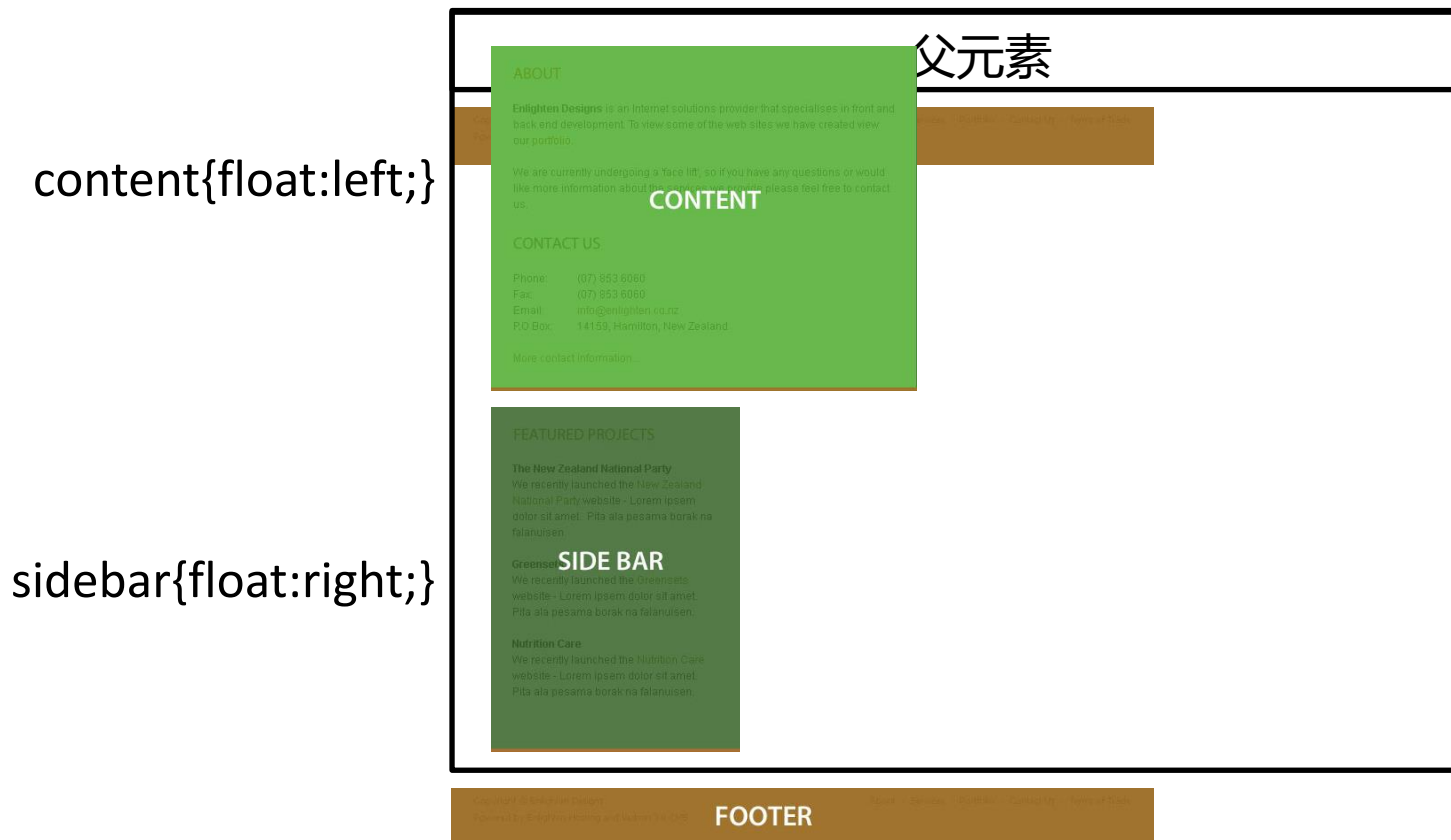


- 子元素浮动，还会对父元素造成影响。
- 如果子元素浮动起来，就不占用普通文档流的位置。父元素高度就会失去支撑，也称为高度坍塌。



问题重现:

- 俯视高度坍塌的效果

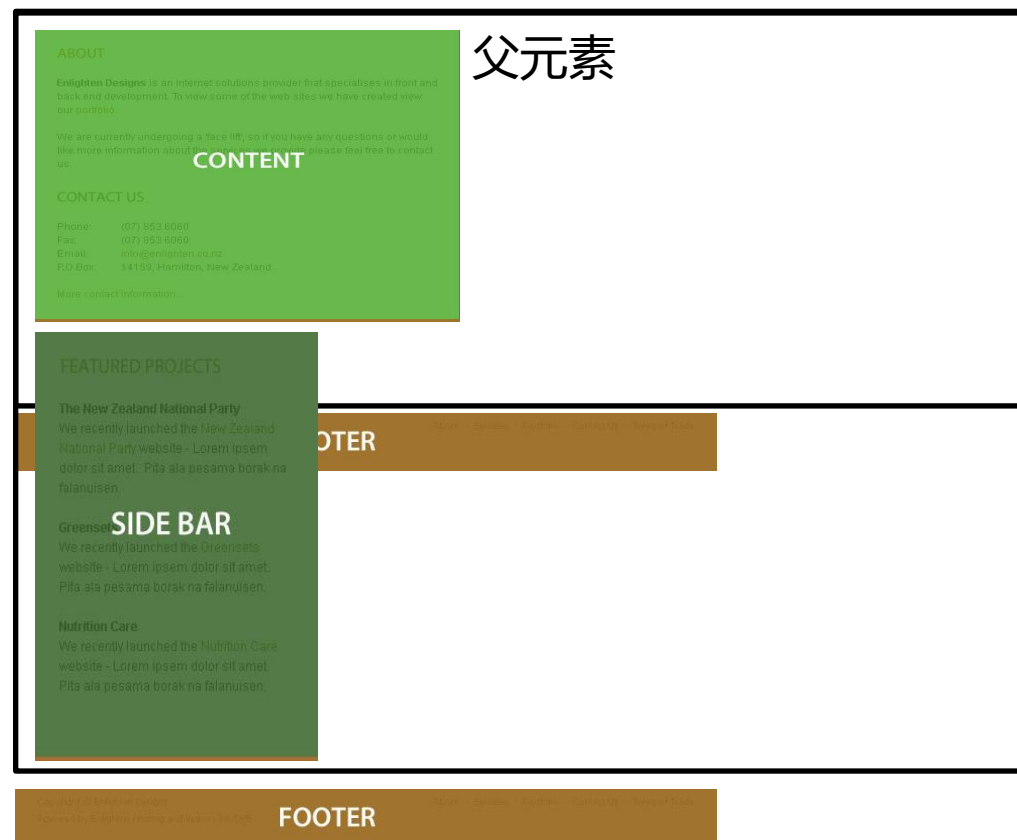


问题重现:



- 即使有部分元素留在普通文档流布局中支撑着父元素，如果浮动起来的元素高度高于留下的元素。那么浮动元素的高度会超出父元素边框，用户体验同样不好！

sidebar{float:right;}



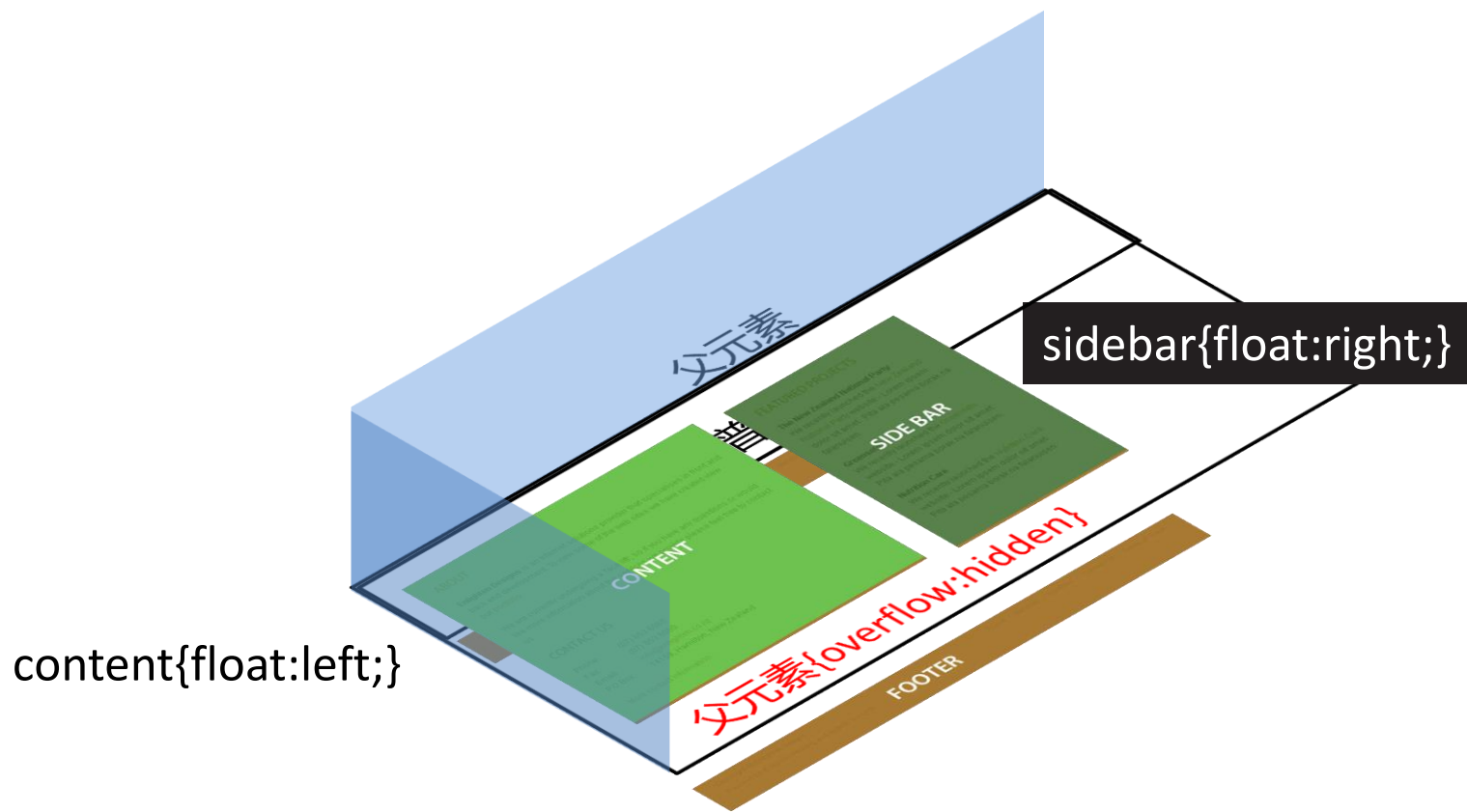
不好的解决



- ~~给父元素设置固定高度。~~
- 缺点: 多数情况下, 父元素高度由内容撑起, 很难提前固定父元素的高度。

解决: 防止高度坍塌, 4种方案:

- 方案一: 为父元素设置 `overflow:hidden` 属性。
- 原理: CSS中的 `overflow :hidden` 属性会强制要求父元素必须包裹住所有内部浮动的元素, 以及所有元素的margin范围。





解决: 防止高度坍塌, 4种方案:

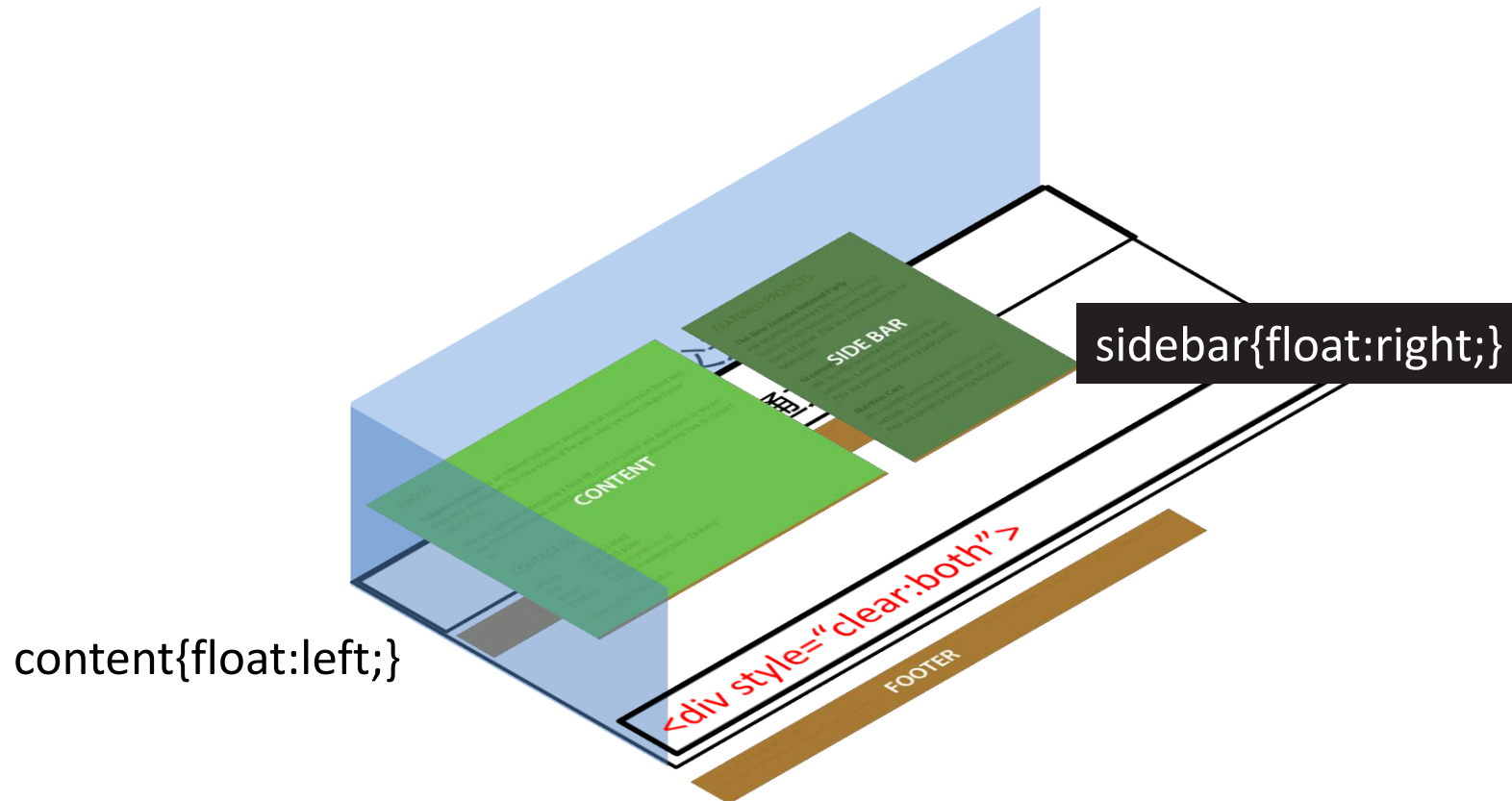
- 方案一: 为父元素设置 `overflow:hidden` 属性。
- 缺点, 如果刚好父元素有些超范围的子元素内容需要显示(比如, 个别 `position` 定位的子菜单项), 不想隐藏, 就会发生冲突。





解决：防止高度坍塌，4种方案：

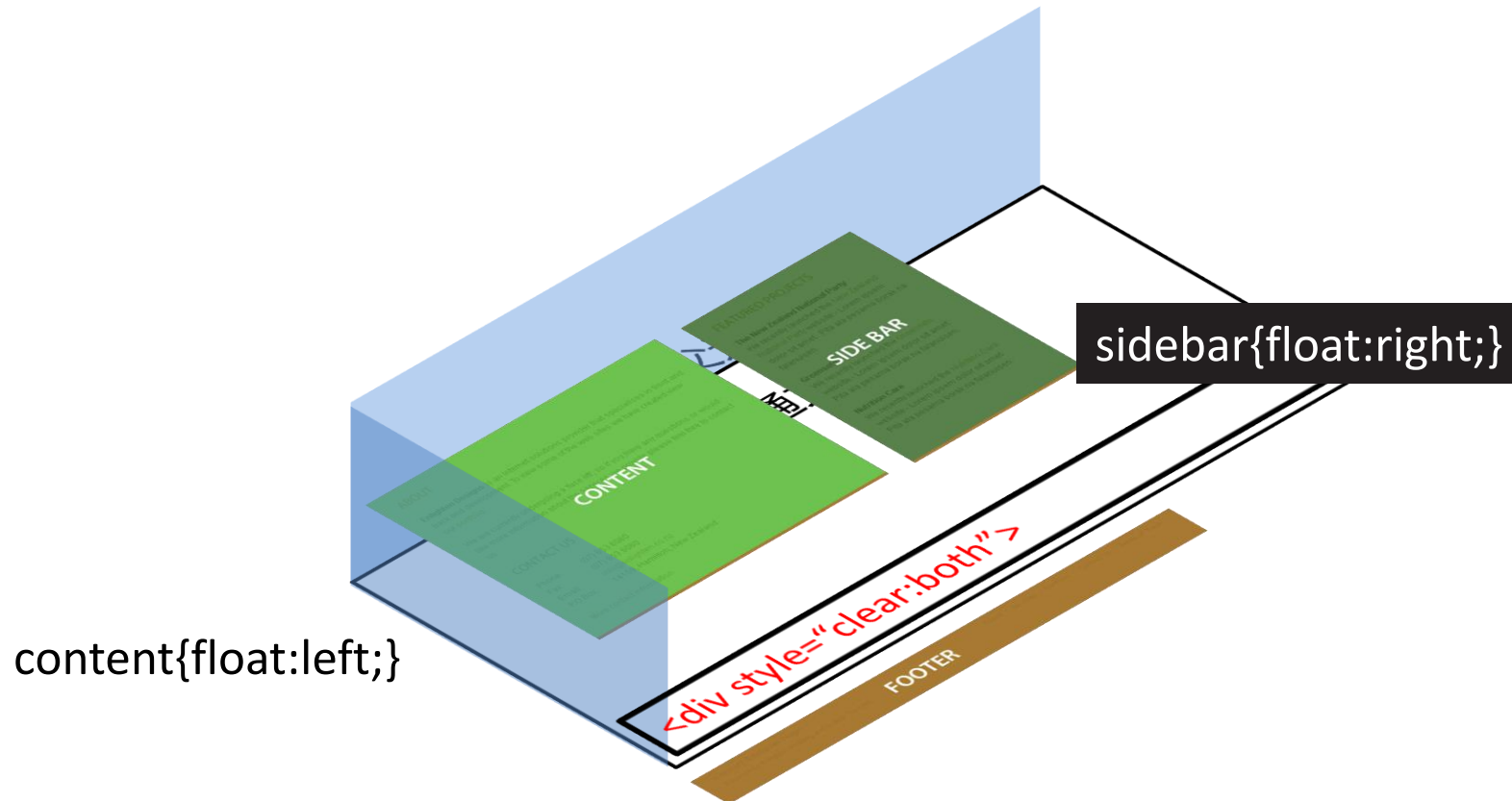
- 方案二：在父元素内的结尾追加一个空子元素（块级元素），并设置空子元素清除浮动影响（`clear:both`）。
- 原理：利用`clear:both`属性和父元素必须包含非浮动的元素两个原理





解决：防止高度坍塌，4种方案：

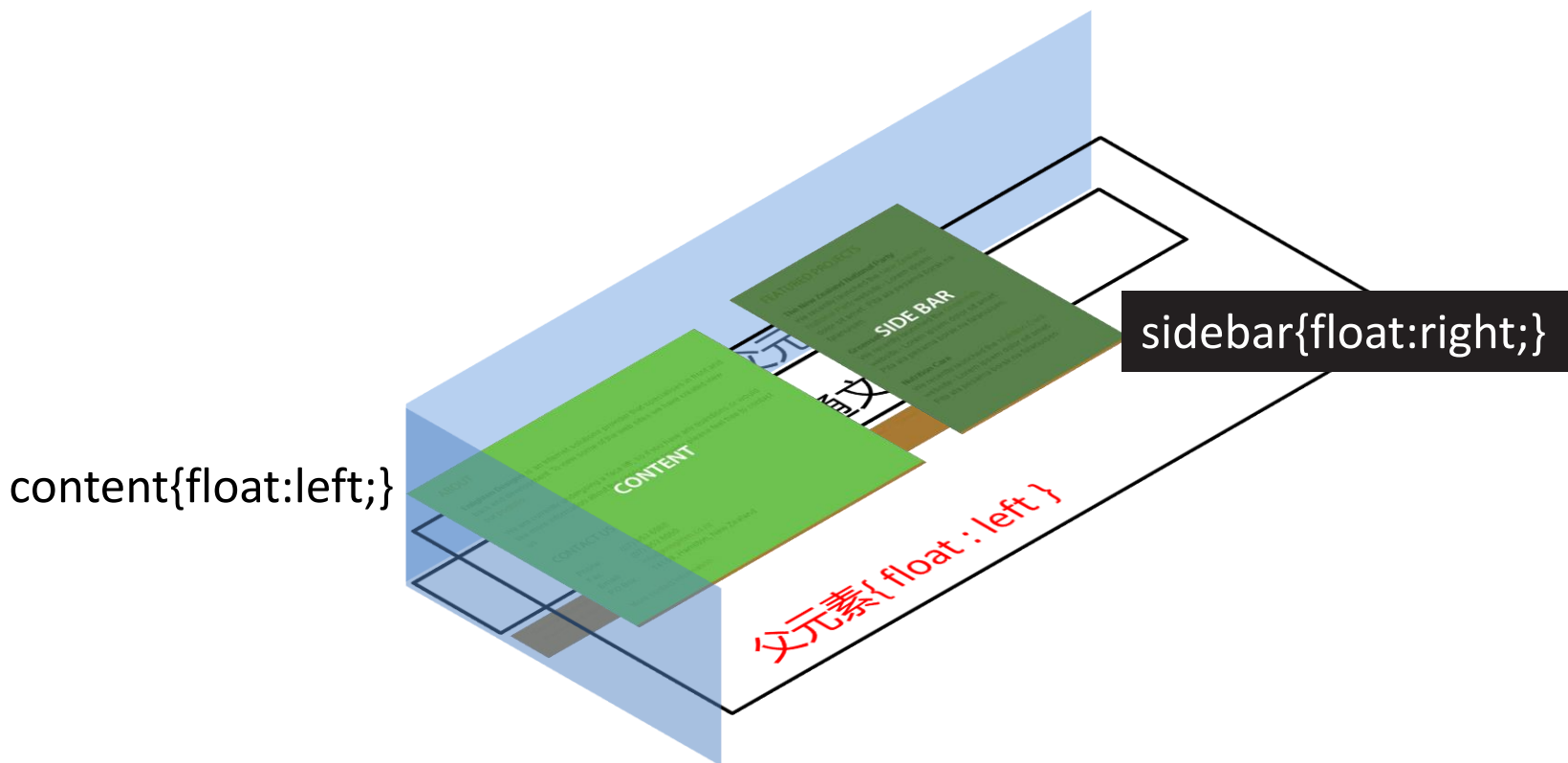
- 方案二：在父元素内的结尾追加一个空子元素（块级元素），并设置空子元素清除浮动影响（`clear:both`）。
- 缺点：无端多出一个无意义的看不见的空元素，影响选择器和查找元素。





解决：防止高度坍塌， 4种方案：

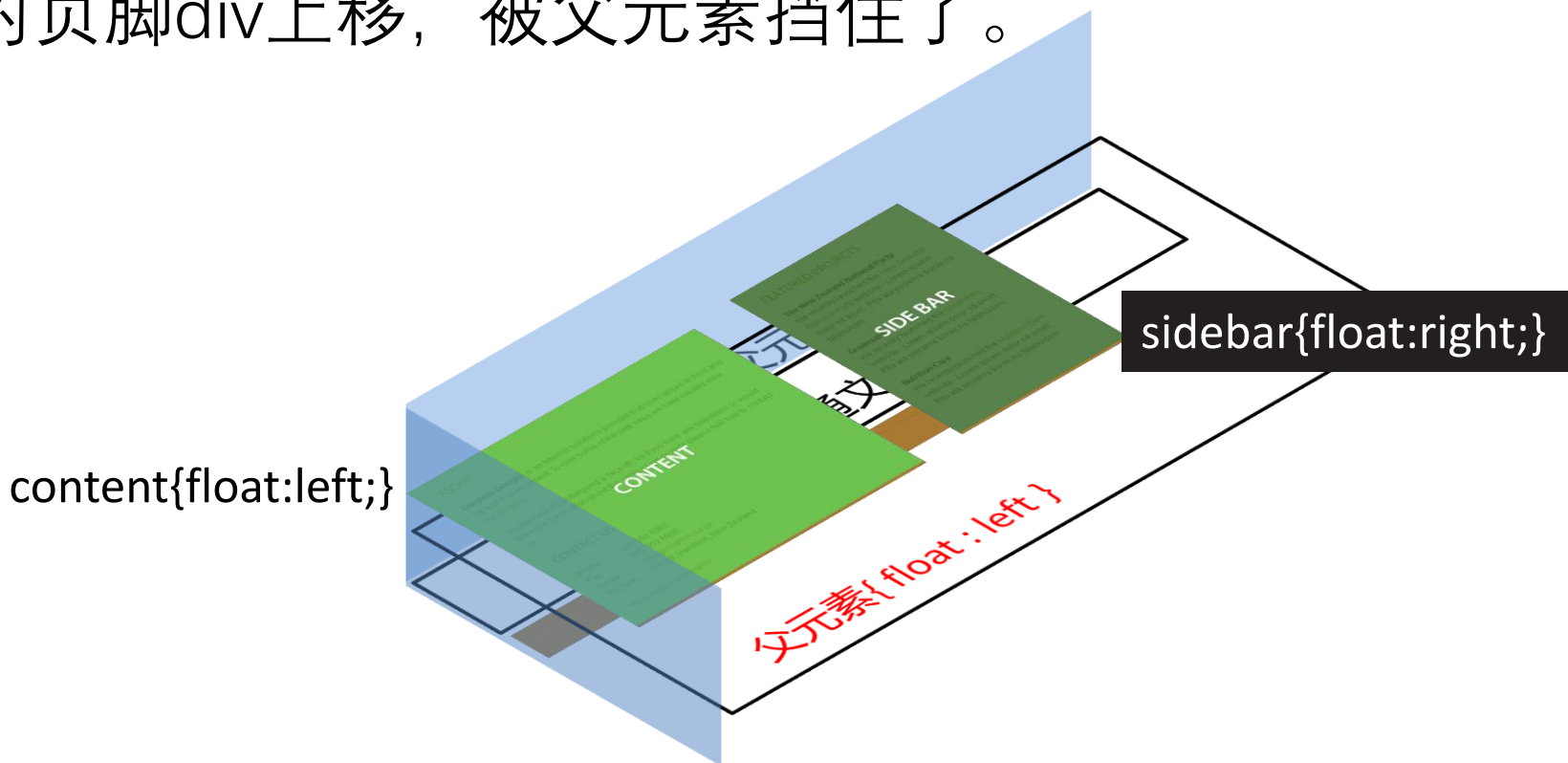
- 方案三：设置父元素也浮动。
- 原理：浮动属性也会强制父元素扩大到包含所有浮动的内部元素。





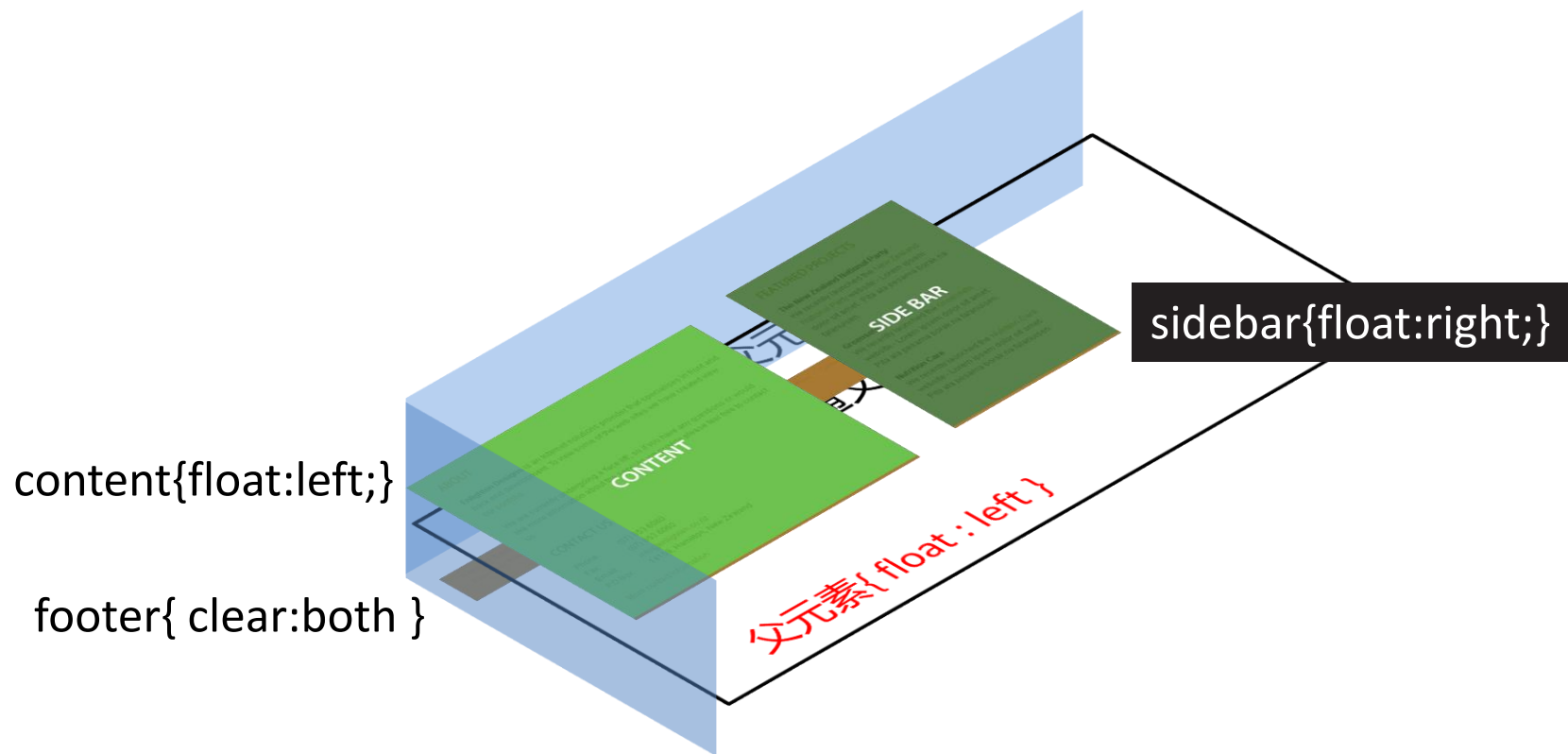
解决：防止高度坍塌，4种方案：

- 方案三：设置父元素也浮动。
- 缺点：会产生新的浮动影响。比如，父元素浮动，导致父元素之后平级的页脚div上移，被父元素挡住了。



解决: 防止高度坍塌, 4种方案:

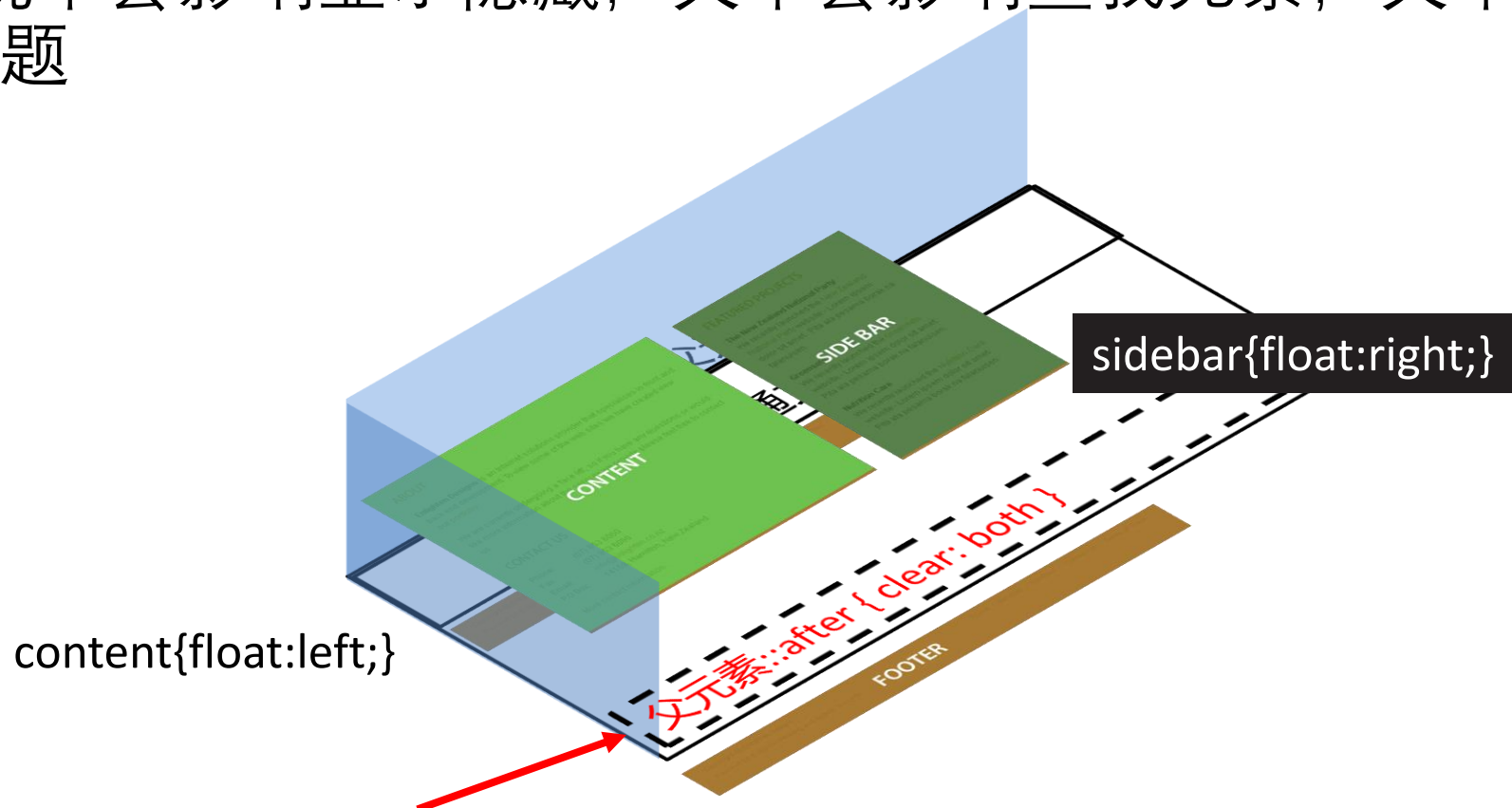
- 方案三: 设置父元素也浮动。
- 解决: 设置父元素之后的平级元素清除浮动 (clear:both) 。





解决: 防止高度坍塌, 4种方案:

- 完美解决: 为父元素末尾伪元素设置clear:both。
- 优点: 既不会影响显示隐藏, 又不会影响查找元素, 又不会产生新的浮动问题

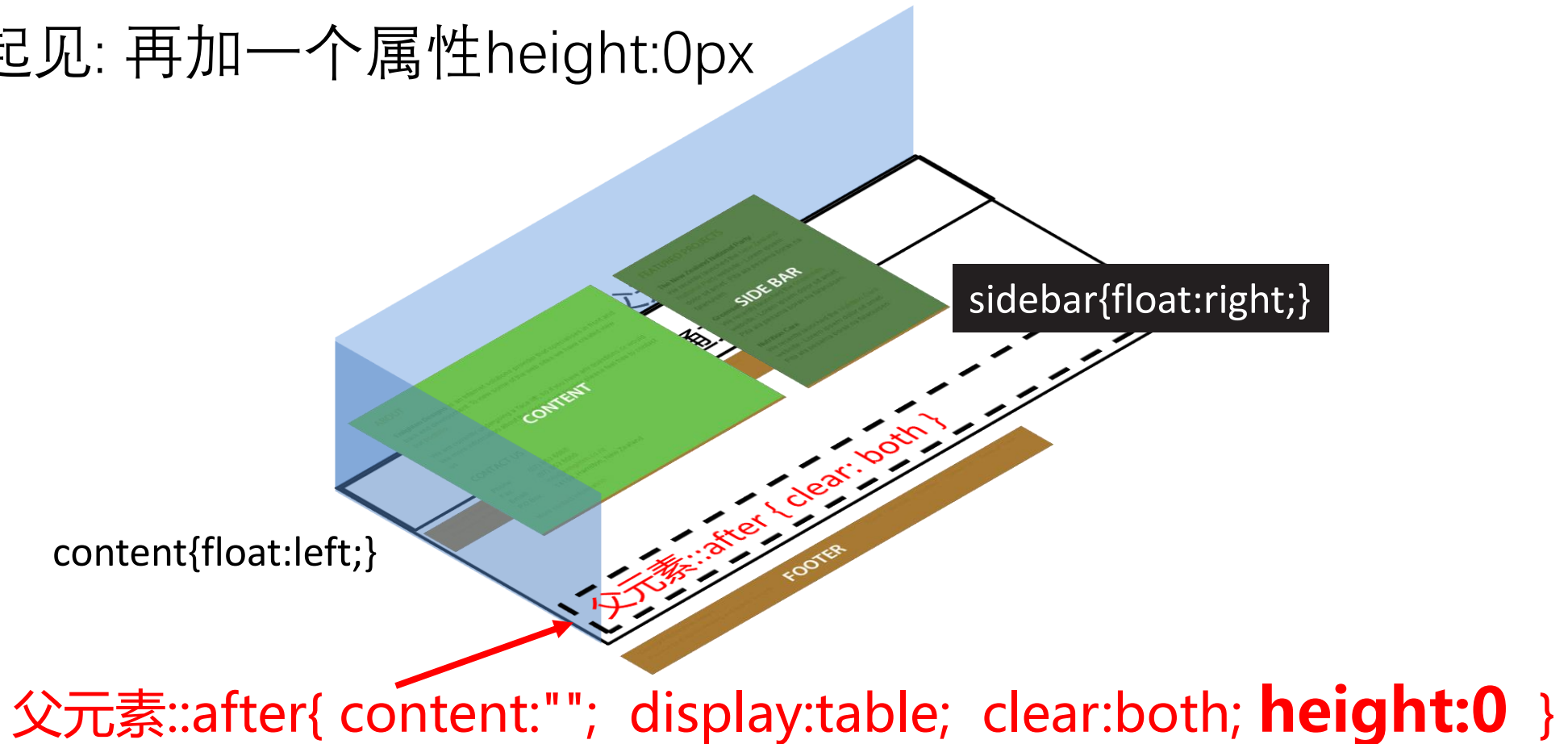


父元素::after{ content:""; display:table; clear:both; }

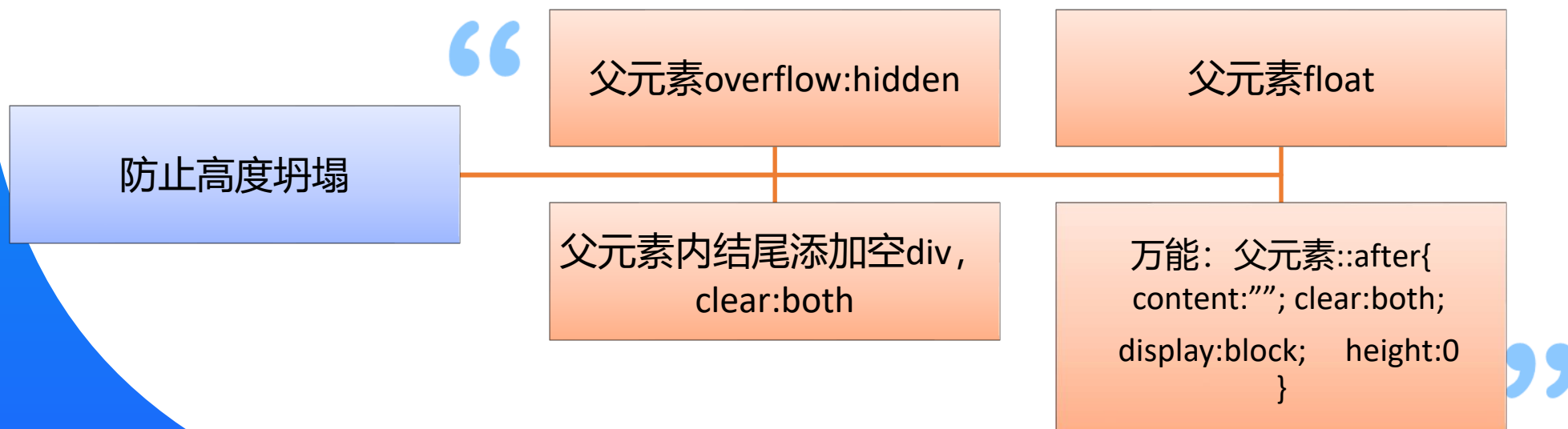


解决：防止高度坍塌，4种方案：

- 完美解决：为父元素末尾伪元素设置clear:both。
- 问题：个别浏览器，display:table，可能带默认高度
- 保险起见：再加一个属性height:0px



总结: 防止高度坍塌, 4种方式





“ 为什么overflow和float会强制父元素包裹浮动的子元素? ”





“

2. BFC

”



什么是BFC

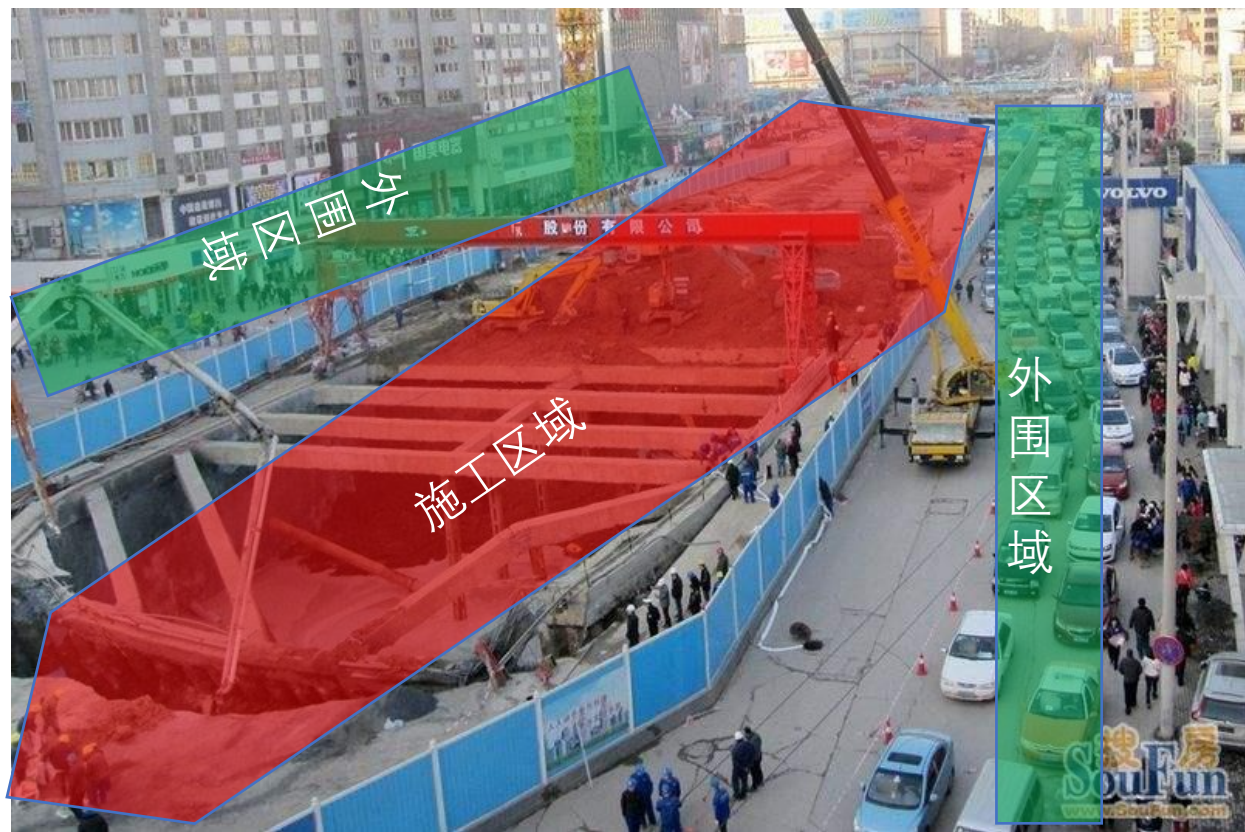


- BFC(Block formatting context)
- 直译为"块级格式化上下文"。
- 它是网页中一个**独立的渲染区域**（也成为formatting context）。
- 这个渲染区域**只有块级（Block）元素才能参与**。
- 它**规定了内部的块级元素如何布局**。
- BFC渲染区域内部如何布局，**与区域外部毫不相干**。
- **外部元素也不会影响BFC渲染区域内的元素**。

什么是BFC



- 简单说：BFC就是页面上的一个隔离的**独立渲染区域**。
- 区域**里面**的子元素**不会影响到外面**的元素。
- **外面**的元素**也不会影响到区域里面**的子元素。



2种渲染区域 (Formatting Context)



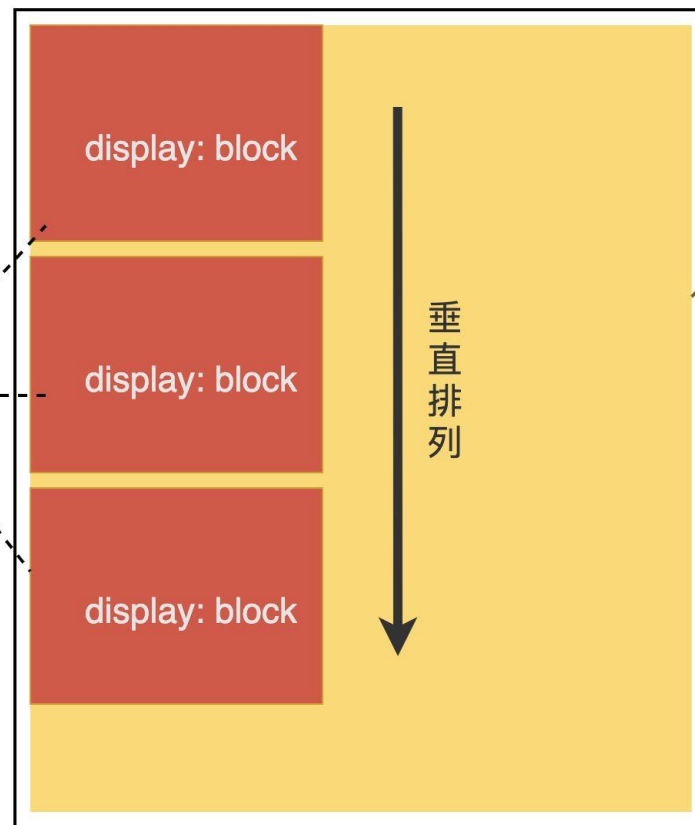
- 其实, css中有2种渲染区域:
- 块级元素渲染区域 和 行级元素渲染区域。

BFC(block formatting context)

- 块级元素渲染区域：所有display属性为 block, list-item, table 的元素，会生成块级元素渲染区域。
- 块级元素渲染区域内以BFC(block formatting context)方式渲染；

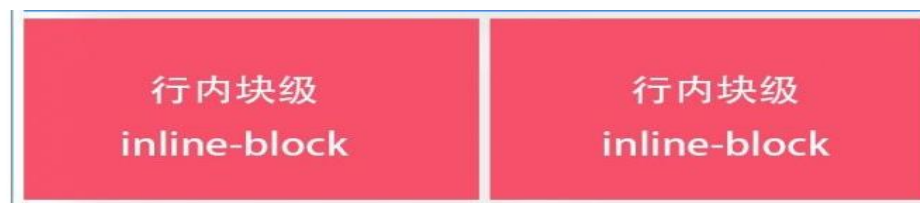
block-level box

本身會與 <body> 一同參與 <html> 建立的 BFC，故呈垂直排列。



IFC(inline formatting context)

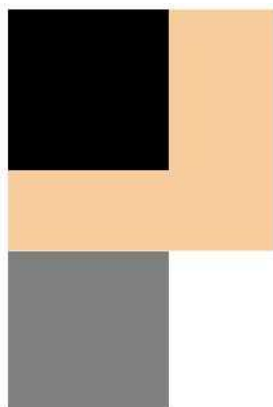
- 行级元素渲染区域：display 属性为 inline, inline-block, inline-table 的元素，会生成 行级元素渲染区域。
- 行级元素渲染区域内以IFC(inline formatting context) 方式渲染；



BFC的布局规则



- 默认，内部的块元素会在垂直方向，一个接一个地放置。每个块元素独占一行



The screenshot shows the browser's developer tools with the following content:

- Elements Panel:** Displays the HTML structure:

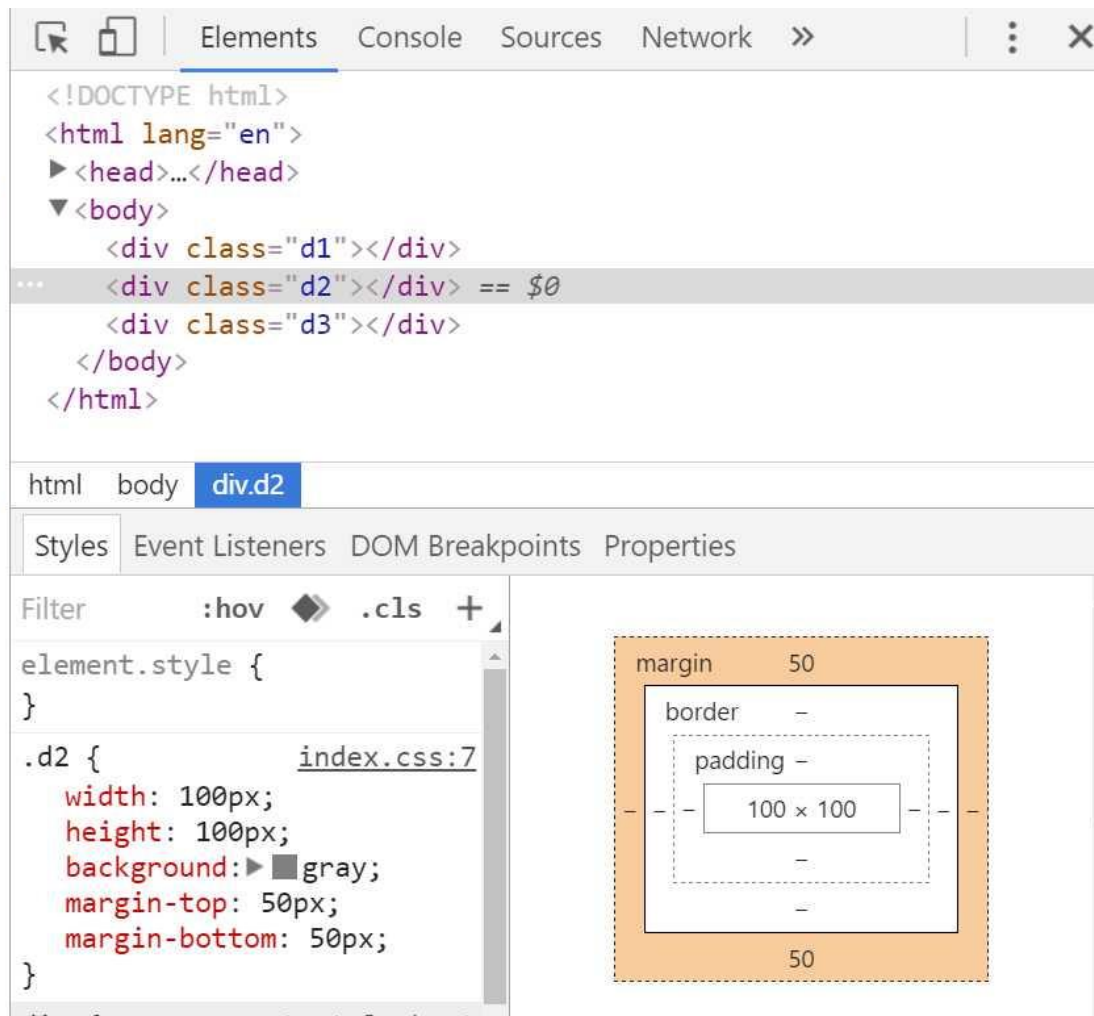
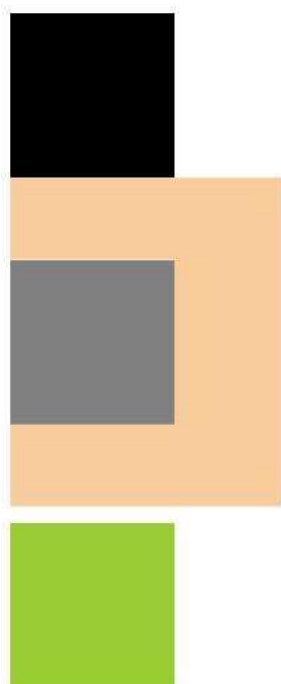
```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    ... <div class="d1"></div> == $0
    <div class="d2"></div>
    <div class="d3"></div>
  </body>
</html>
```
- Breadcrumbs:** html > body > **div.d1**
- Styles Panel:** Shows the CSS rules for the selected element.
 - Filter:** :hov, .cls
 - element.style:** { }
 - .d1 {** (from index.css:1)
 - width: 100px;
 - height: 100px;
 - background: black;
 - margin-bottom: 50px;
 - div {** (from user agent stylesheet)

- Diagram:** A visual representation of the box model for the selected element. It shows a central box labeled '100 x 100' with a dashed border. This is surrounded by a larger orange box representing the margin, with a label 'margin' and a value '50' indicating the bottom margin.

BFC的布局规则



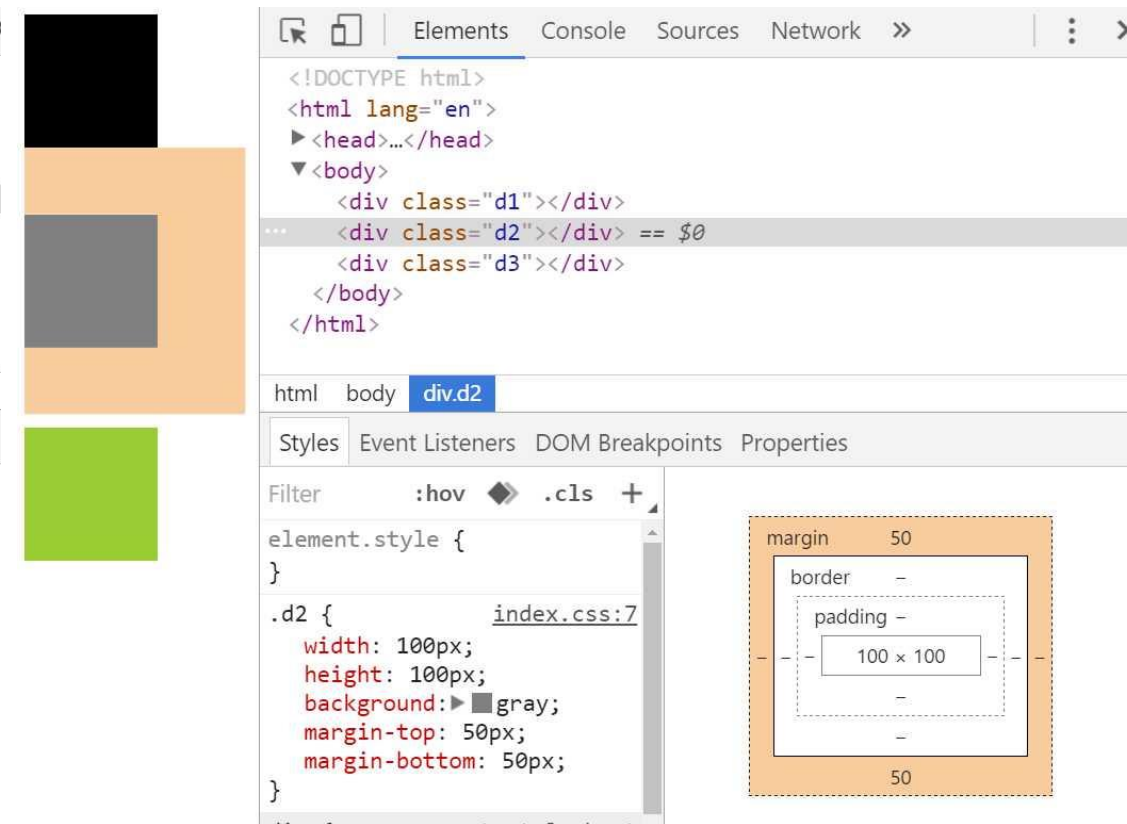
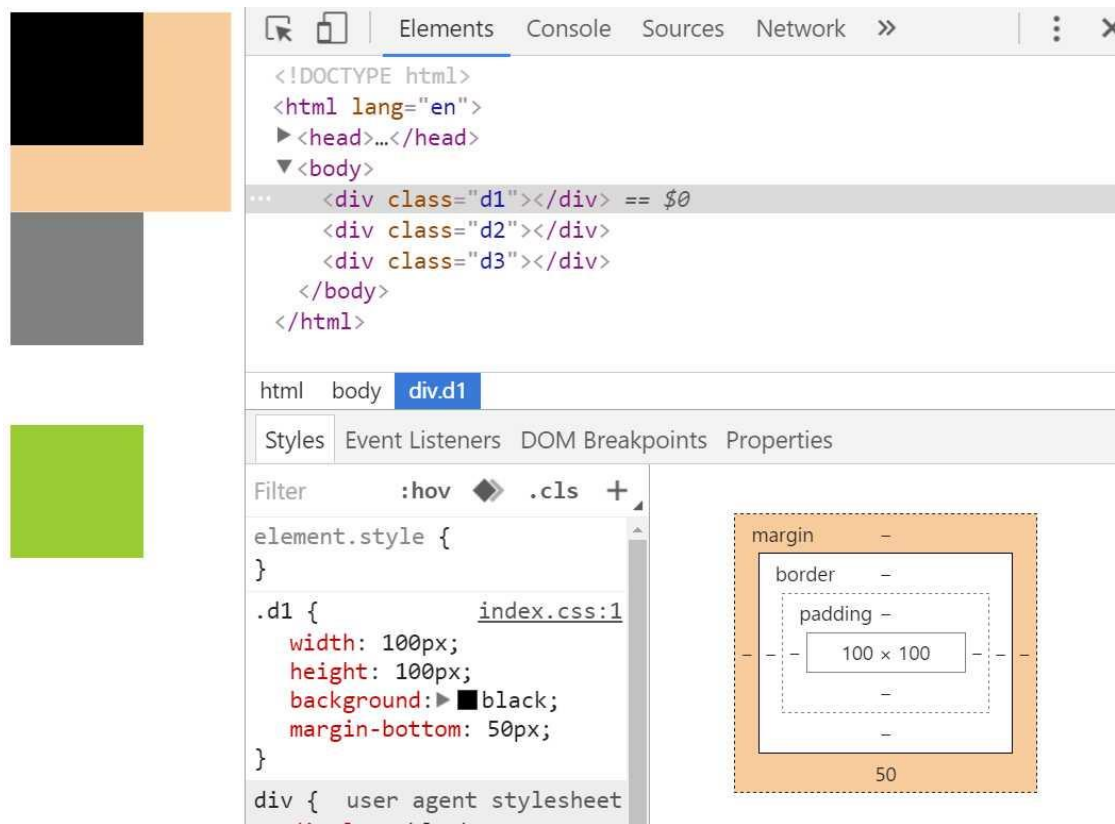
- 块元素垂直方向的总距离由边框内大小+margin共同决定。



BFC的布局规则



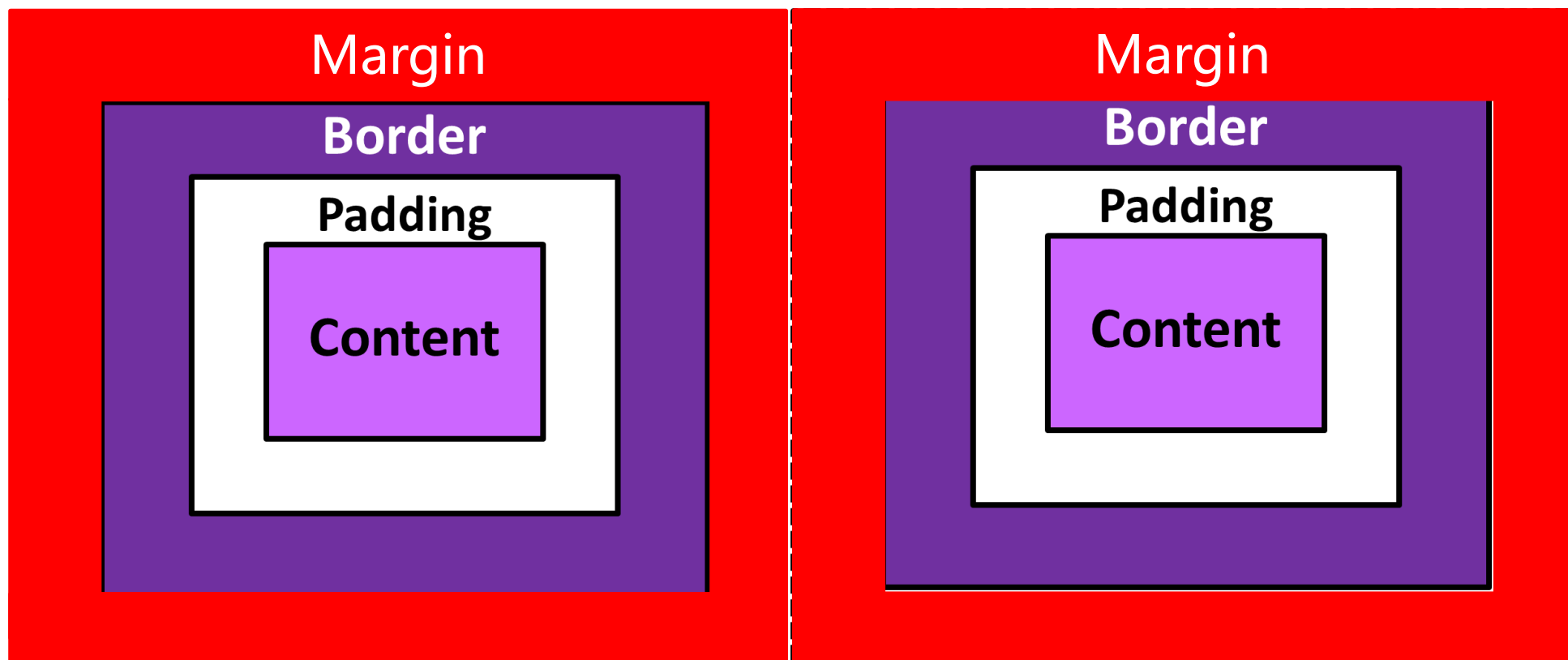
- 属于同一个BFC的两个相邻块元素在垂直方向上的margin会发生重叠/合并。但水平方向的margin不会



BFC的布局规则



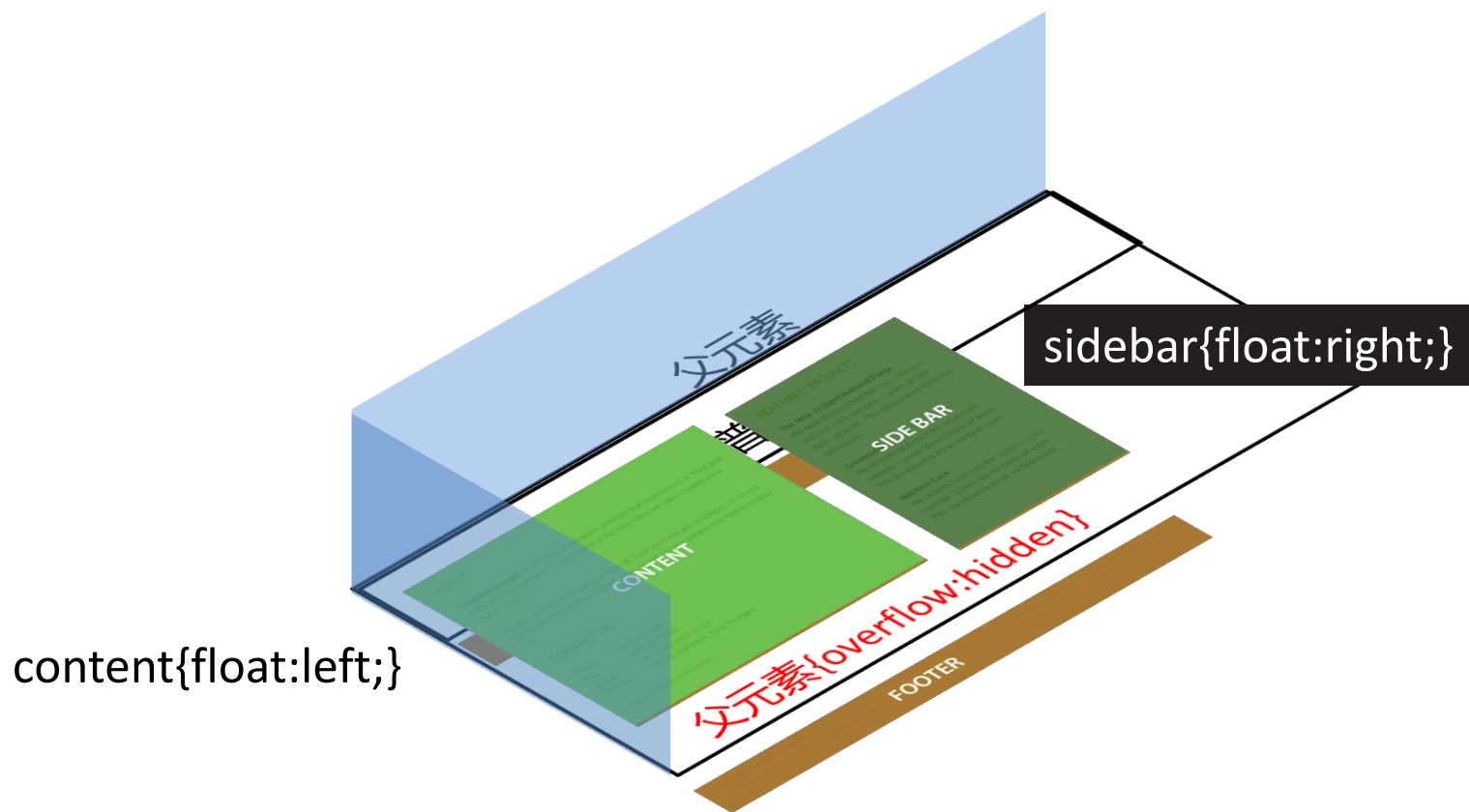
- 左侧BFC渲染区域的margin, 必须与右侧BFC渲染区域的margin相衔接, 不能出现重叠



BFC的布局规则



- 计算父元素BFC渲染区域的高度时，内部浮动元素的高度，都必须算在内。





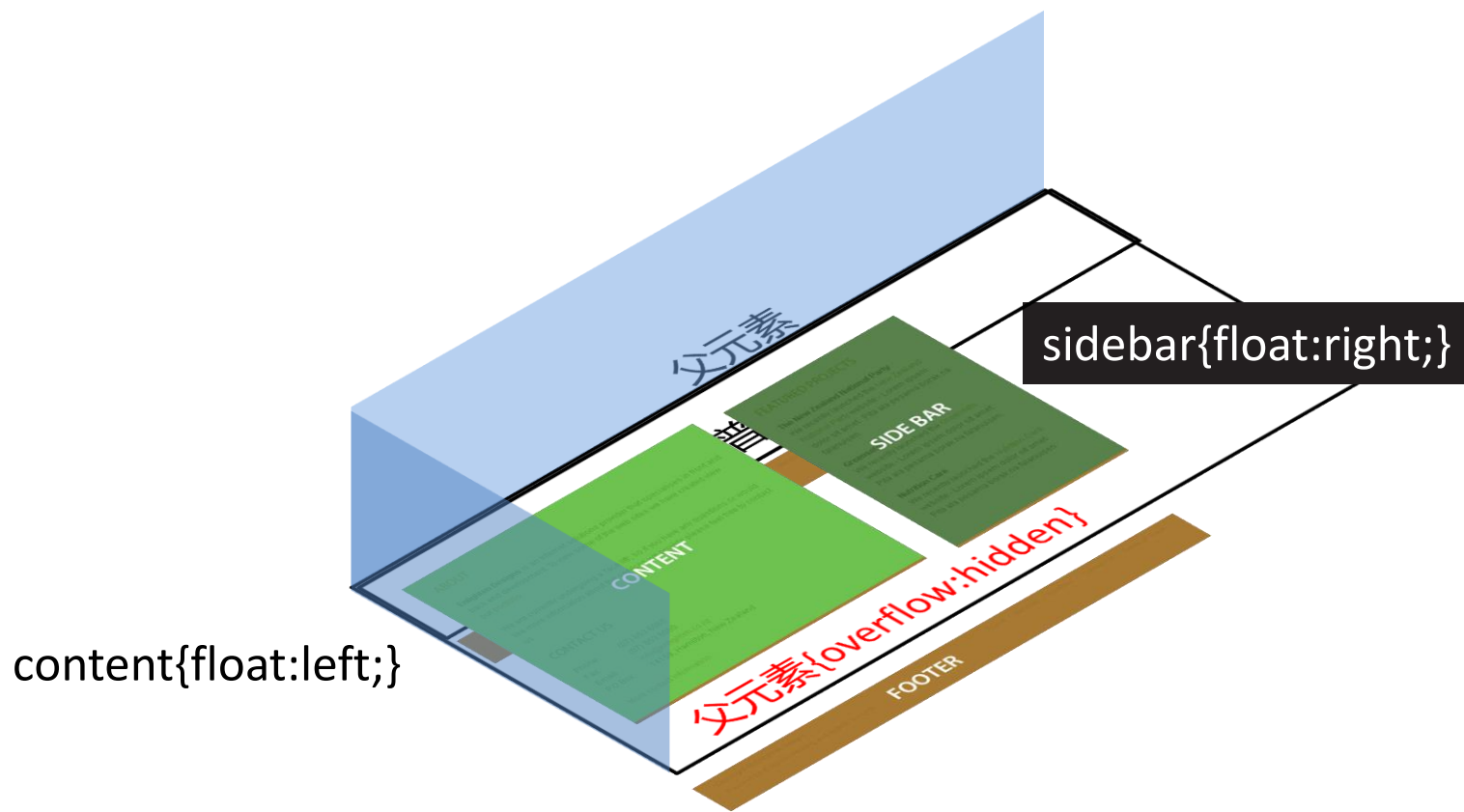
4种情况会形成BFC渲染区域

- float的值不是none
- position的值不是static或者relative。
- display的值是inline-block、table-cell、flex、table-caption或者inline-flex
- overflow的值不是visible

所以，形成BFC区域可以解决高度坍塌！



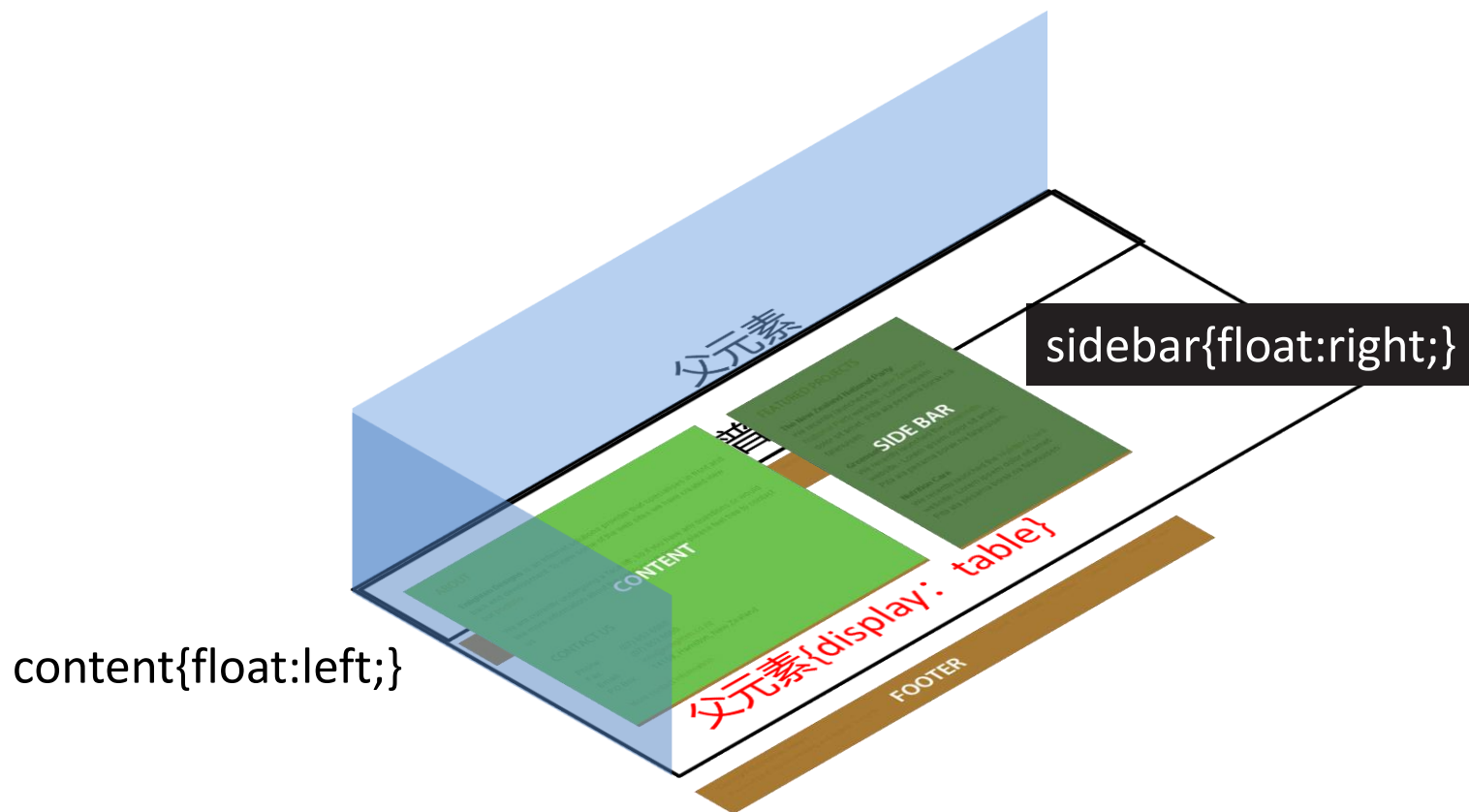
- 方案一:为父元素设置`overflow:hidden`属性（）。
- 原理:因为形成BFC区域，所以必须父元素必须包含内部float浮动元素



所以，形成BFC区域可以解决高度坍塌！



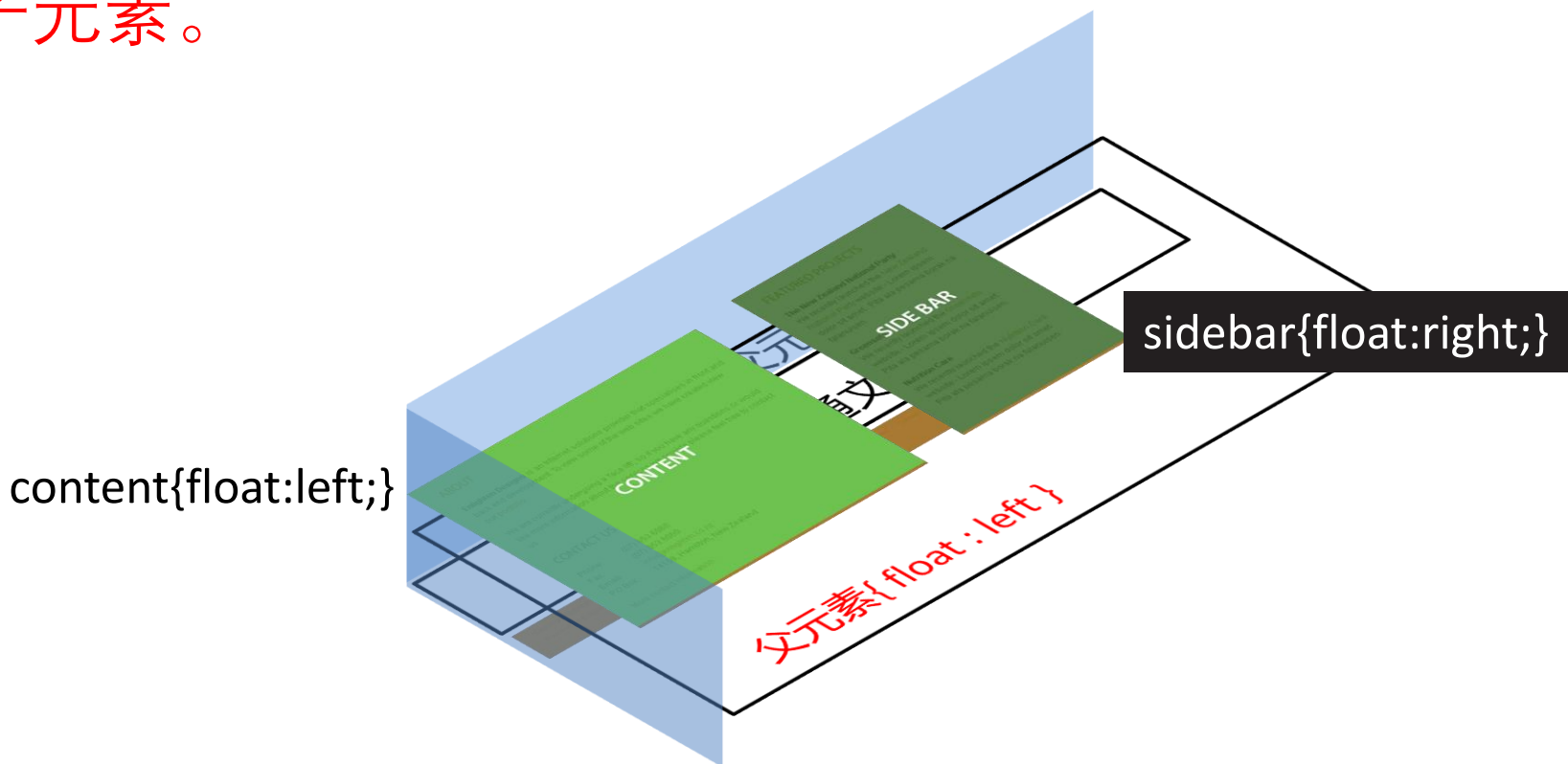
- 方案一:为父元素设置`overflow:hidden`属性 () 。
- 其实这里改成`display:table`，也可以。因为`display:table`也可以形成bfc区域。只不过，需要预防其他可能造成的新问题。



所以，形成BFC区域可以解决高度坍塌！



- 方案三：设置父元素也浮动。
- 原理：因为父元素float，也形成了BFC区域，必须包含内部float浮动的子元素。



总结: BFC (块级格式化上下文)

- “
- 独立渲染区域
 - 内部不影响外部,
外部也不能入侵影响内部
- ”



总结: 如何生成BFC区域, 4句话

- float的值不是none
- position的值不是static或者relative。
- display的值是
inline-block、table、table-cell、
flex、table-caption或inline-flex
- overflow的值不是visible



“

**3. BFC还可以解决
更多问题...**

”





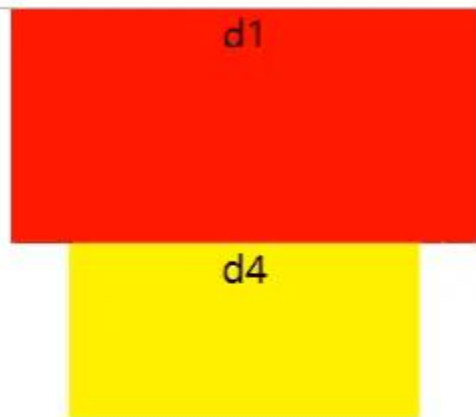
“ 3.1 避免垂直方向
margin合并 ”



问题重现:



- 垂直方向上，两个元素上下margin相遇时，两元素之间的总间距并不等于两个margin的和。而是等于最大的margin。
- 小的margin会被大的margin吞并。



d1的margin-bottom:0px

0px  50px

d4的margin-top:0px

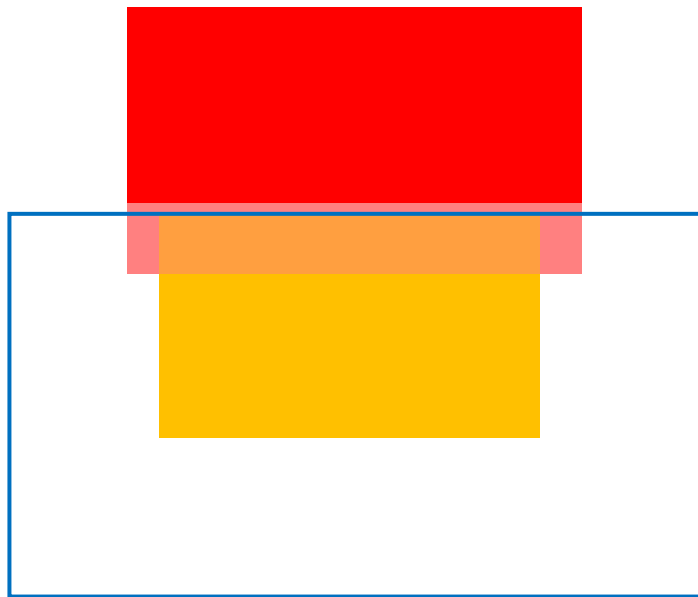
0px  50px



解决: 2步



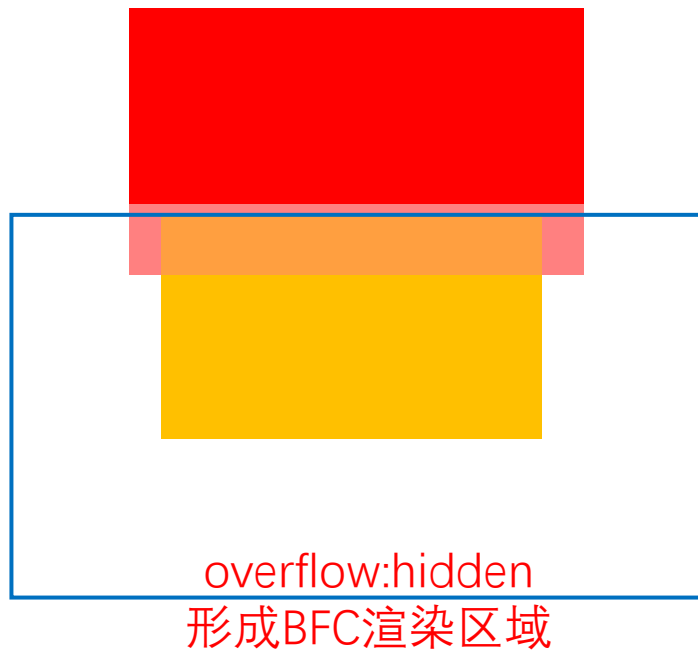
- 第一步: 用一个外围块元素包裹下方元素



解决: 2步



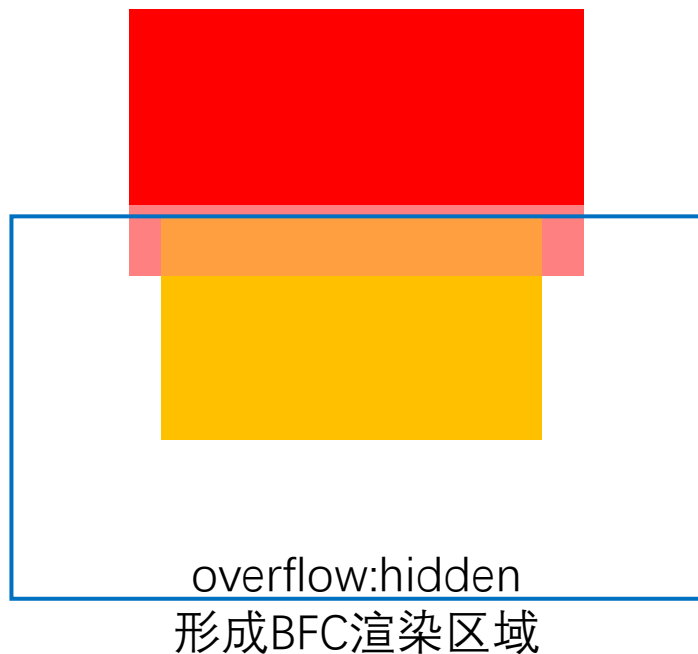
- 第二步: 设置新外层元素overflow:hidden



解决: 2步



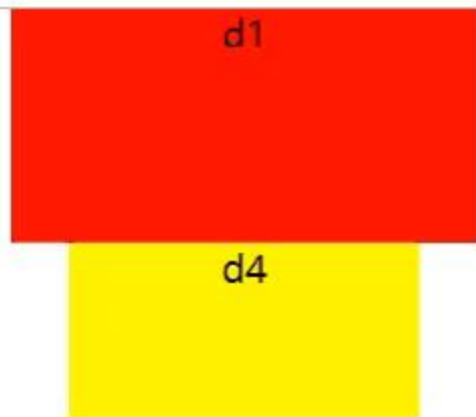
- 原理: 新外层元素, 变成一个BFC方式的渲染区域, 就必须包裹内部子元素及子元素的margin。
- 而且, 内部元素不能超出范围影响外部, 外部元素也不能进入BFC范围内, 影响内部。



解决: 2步



- 缺点: 为如果父元素中部分自由定位的子元素, 希望即使超出父元素范围, 也能显示时, 就冲突了。
- 解决: 第二步: 父元素::before{ content:""; display:table}
- 原理: display:table, 在子元素之前形成平级的bfc渲染区域。不允许子元素的margin进入::before范围内。
- 优点: 既不隐藏内容, 又不添加新元素, 又不影响高度。



d1的margin-bottom:0px

0px  50px

d4的margin-top:0px

0px  50px



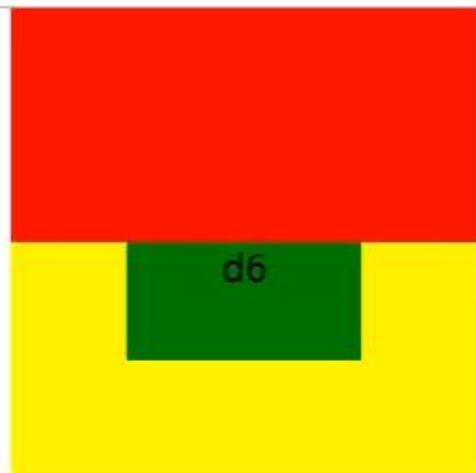
“ 3.2 避免垂直方向
margin溢出 ”



问题重现:



- 问题: 子元素设置margin-top, 会超出父元素上边的范围, 变成父元素的margin-top。
- 而实际上, 子元素与父元素之间, 依然是没有margin-top的
- 效果不是想要的。



d4的margin-top:0px

0px — 50px

d6的margin-top:0px

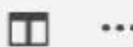
0px — 50px



5种解决:



- 1. 设置父元素overflow:hidden
- 原理: 父元素变成BFC渲染区域, 就必须包裹内层子元素的margin
- 缺点: 万一有的子元素, 即使溢出父元素, 也希望显示呢? 就会发生冲突。

<> margin.4.html ✕



```
21 #d4{
22     width:200px; height:100px;
23     margin:0 auto;
24     background-color:  yellow;
25     /* border:1px solid yellow; */
26 }
27 #d5{
28     position:absolute;
29     left:50%; margin-left:-50px;
30     top:100px;
31     width:100px;
32     background-color:  (0, 255, 0, 5);
```





5种解决:

- 2. 为父元素添加上边框，颜色设置为透明 (transparent)
- 原理: 这里不是bfc。而是因为边框本身可以阻隔margin溢出。
- 缺点: 边框会增大父元素的实际大小，导致布局错乱。

margin.4.html ×

```
21 #d4{
22     width:200px; height:100px;
23     margin:0 auto;
24     background-color:■yellow;
25     border:1px solid ■yellow;
26 }
27 #d5{
28     position:absolute;
29     left:50%; margin-left:-50px;
30     top:100px;
31     width:100px;
```

5种解决:



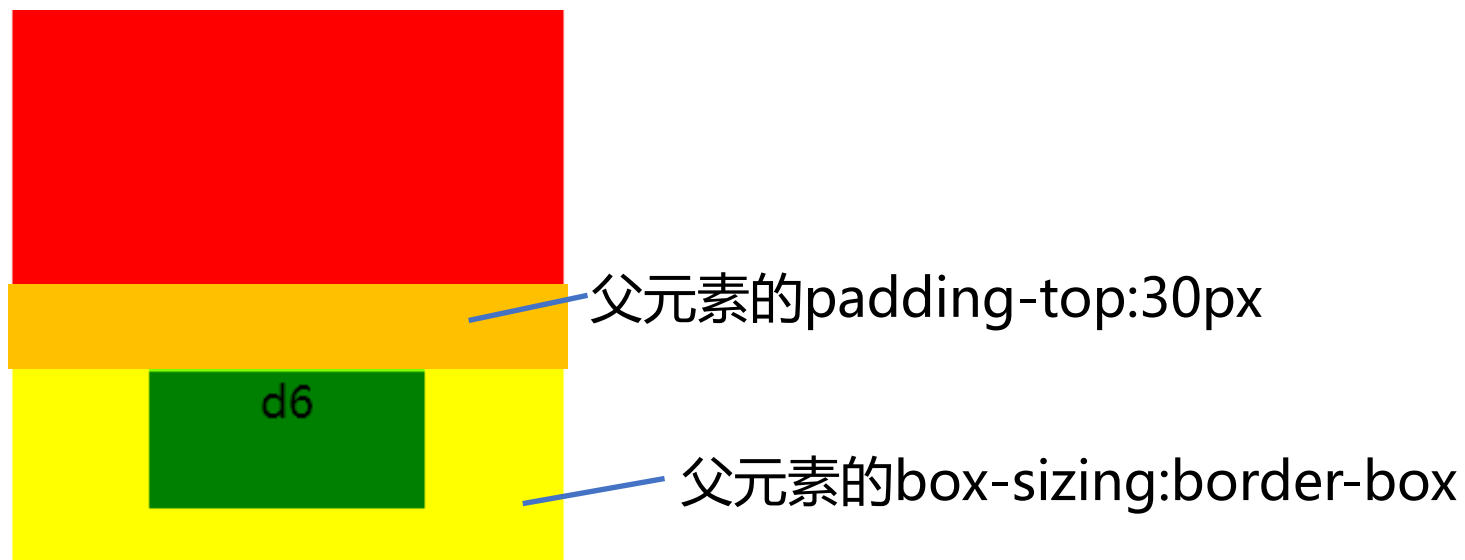
- 3. 用父元素的padding-top代替第一个子元素的margin-top
- 原理: 这里也不是bfc。而是因为padding本身可以阻隔margin溢出。



5种解决:



- 3.用父元素的padding-top代替第一个子元素的margin-top
- 缺点: 对父元素高度有影响。
- 解决: 可以设置父元素box-sizing:border-box。



5种解决:



- 4. 在父元素内第一个子元素之前添加一个空的<table></table>
- 原理: table的display属性默认相当于table, 所以形成小的bfc渲染区域。其他元素的margin不能进入table范围内。就阻隔了margin向上溢出。
- 优点: 空table元素没有大小, 不占用父元素控件。
- 缺点: 增加一个看不见的空元素, 干扰查找元素



资源管理器

打开的编辑器

✕ <> margin.4.html

4

▸ imgs

▸ videos

4.docx

4.pptx

<> boxModel.html

<> margin.1.html

<> margin.2.html

<> margin.3.html

<> margin.4.html

<> margin.html

<> test.html

▸ 大纲

<> margin.4.html ✕

43

<div id="d1"></div>

44

<div id="d3"></div>

45

<div id="d5"></div>

46

<div id="d4">

47

<div id="d6">d6</div>

48

</div>

49

<hr>

50

<h1>

51

d4的margin-top:<span
id="d4Margin">0px

52

0px<input id="rd4" type="range"



5种解决:



- 5. 最好的解决: 父元素::before{ content:""; display:table; }
- 优点:既不隐藏内容, 又不添加新元素, 又不影响高度。

<> margin.4.html ×

```
21 #d4{
22     width:200px; height:100px;
23     margin:0 auto;
24     background-color: yellow;
25     /* border:1px solid yellow; */
26     /* overflow:hidden; */
27 }
28 #d5{
29     position:absolute;
30     left:50%; margin-left:-50px;
31     top:100px;
```



“ 3.3 左定宽，
右自适应布局 ”



希望效果



左侧
定宽
侧边栏

右侧自适应浏览器宽度

2步:



- 第一步: 左边定宽元素左浮动: `.left{ float:left; width:固定宽 }`
- 第二步: 右边元素右浮动: `.right{ float:right; ...}`
- 问题: 右边元素虽然在右边了, 但是宽度无法自适应。



正确2步:

- 第一步: 左边定宽元素左浮动: `.left{ float:left; width:固定宽 }`
- 第二步: 右边元素不用右浮动, 而是`.right{overflow:hidden; ... }`
- 原理: 右边元素`overflow:hidden`后, 形成BFC渲染区域。左边的`float`元素就不能进入右边范围了。

左侧
定宽
侧边栏
`float:left`

右侧自适应浏览器宽度
`overflow:hidden`

总结: 解决垂直方向margin合并

“

- Step1: 添加父元素包裹下方元素

- Step2:

- 父元素overflow:hidden
- 父元素下第一个子元素前添加空<table>
- 父元素padding代替子元素margin
- 父元素+透明上边框
- 父元素::before{ content:""; display:table }

”



总结: 解决垂直方向margin溢出

“

- 父元素overflow:hidden
- 父元素下第一个子元素前添加空<table>
- 父元素padding代替子元素margin
- 父元素+透明上边框
- 父元素::before{ content:""; display:table }

”



总结: 左定宽, 右自适应布局



“

- Step1: 左: float:left
- Step2: 右: 右overflow:hidden

”





Q2: 弹性布局回顾



“

1. 弹性布局的概念

”





弹性布局 概念

- 现代Web开发要求，网页应该尽量适应不同显示设备的大小要求，灵活显示网页内容。
- 这就要求，网页内容要可以随显示设备的大小而动态调整布局。
- 前面学过的浮动定位float属性，的确可以实现根据显示设备大小，自动换行显示。但是，浮动定位float提供的可控制的属性太少了，以至于难于随心所欲的控制布局。

float浮动定位的问题:

- 想把多个块元素水平放在一行内, 要计算的要素太多。
- 一个元素总宽度= $\text{width} + \text{padding} * 2 + \text{border} * 2 + \text{margin} * 2$ 。
- 一行中多个元素都要按以上算法计算总宽度——太繁琐!

弹性布局优点： 自动计算，自适应；

弹性布局可自动计算宽度、间距等，
使多个元素始终保持最初设计的样子。

弹性布局概念

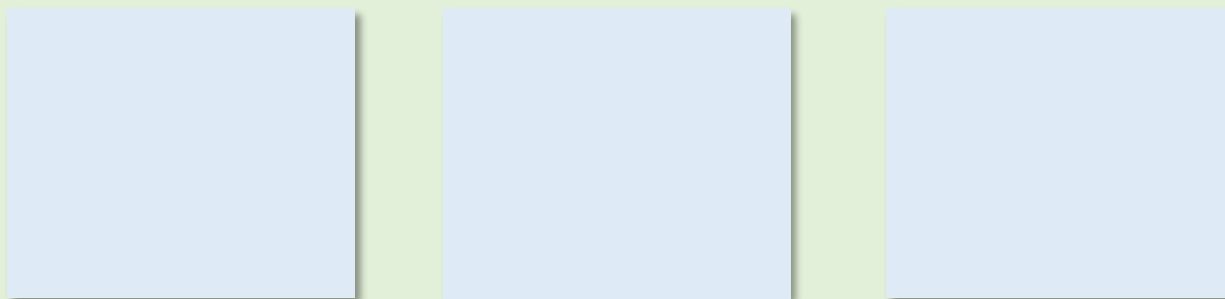
- 使用弹性布局前，先要了解一些概念。包括：
 - 弹性布局的容器
 - 弹性布局的项目
 - 主轴
 - 交叉轴

弹性布局概念



- 弹性布局的容器，简称“容器”，是指要实现布局效果的父元素。

弹性布局的容器



弹性布局概念



- 弹性布局的项目，简称“项目”，是指要实现布局效果的子元素，称为项目

弹性布局的容器

弹性布局的项目

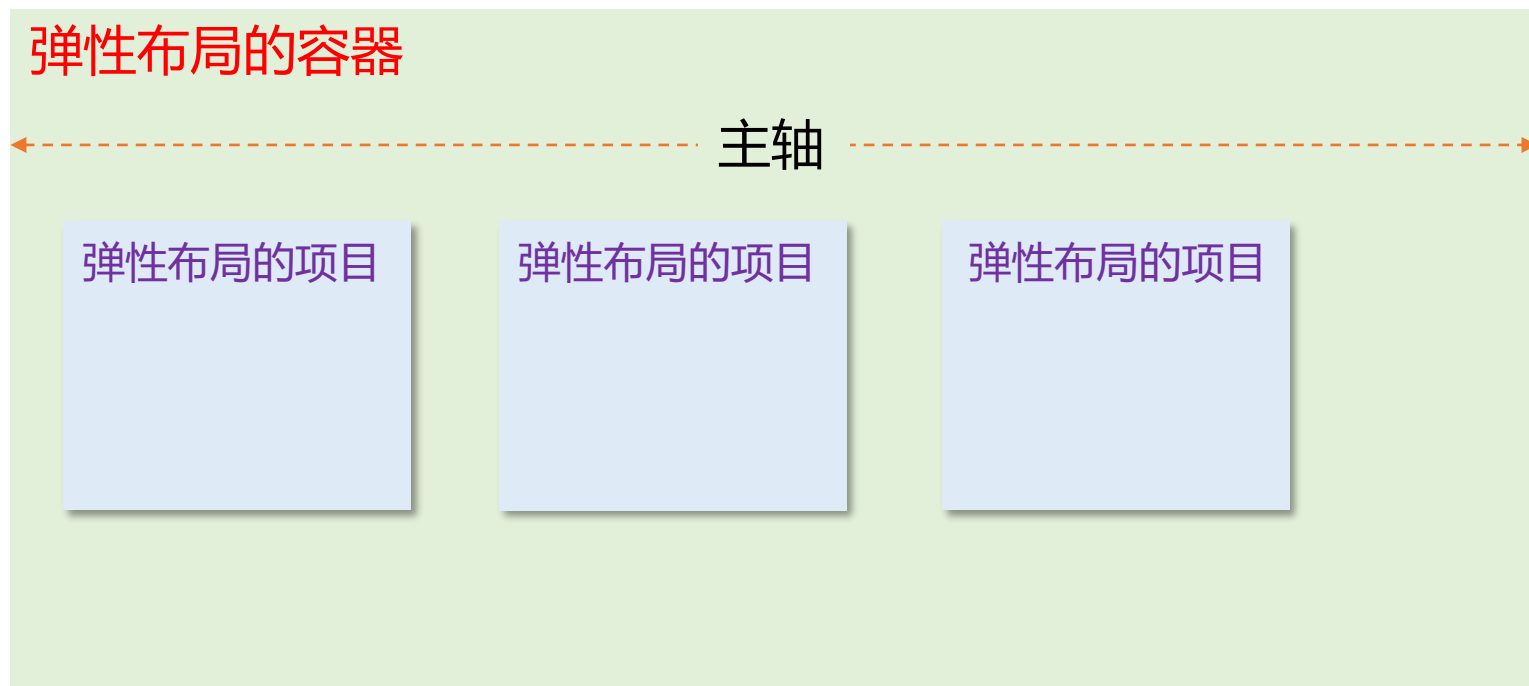
弹性布局的项目

弹性布局的项目

弹性布局概念



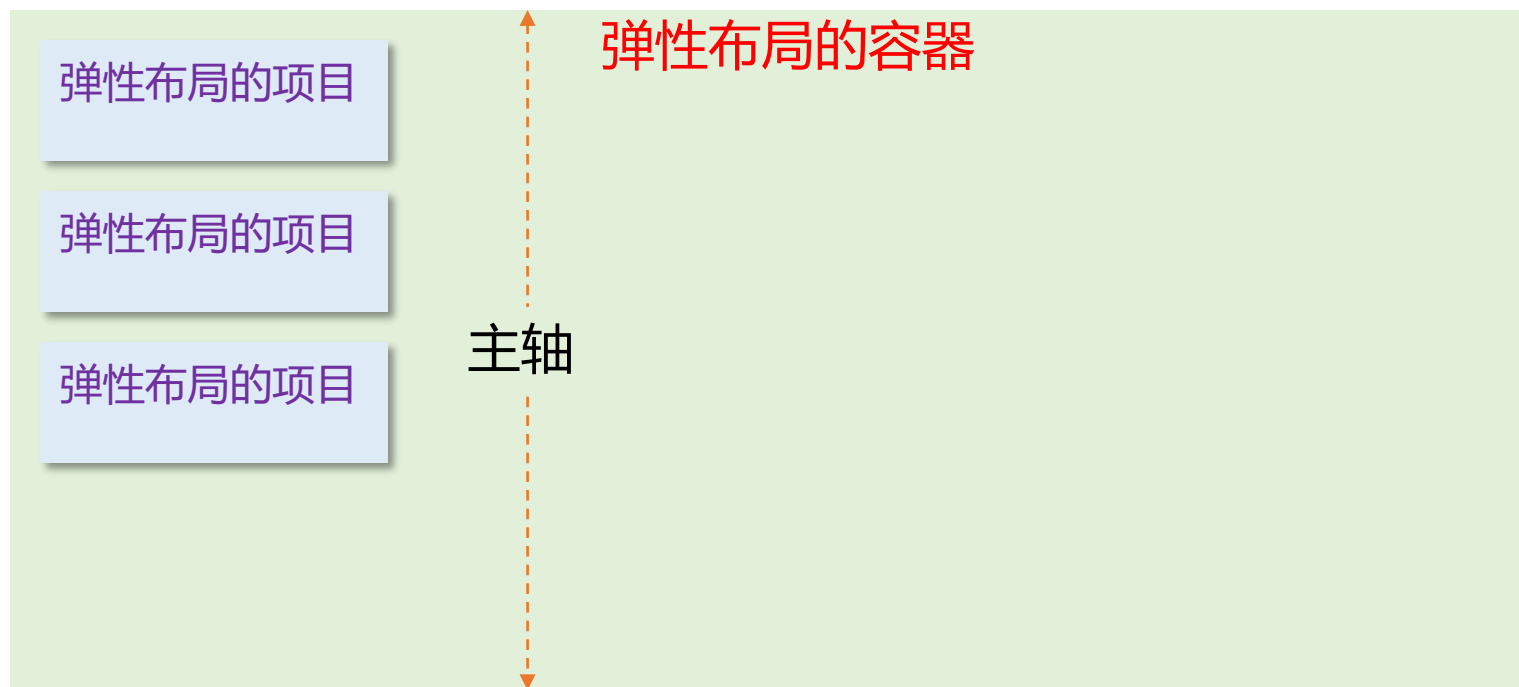
- 主轴，是指弹性布局的多个项目排列方向上的一根轴。
 - 如果弹性布局的多个项目按x轴排列，那么x轴就是主轴。



弹性布局概念



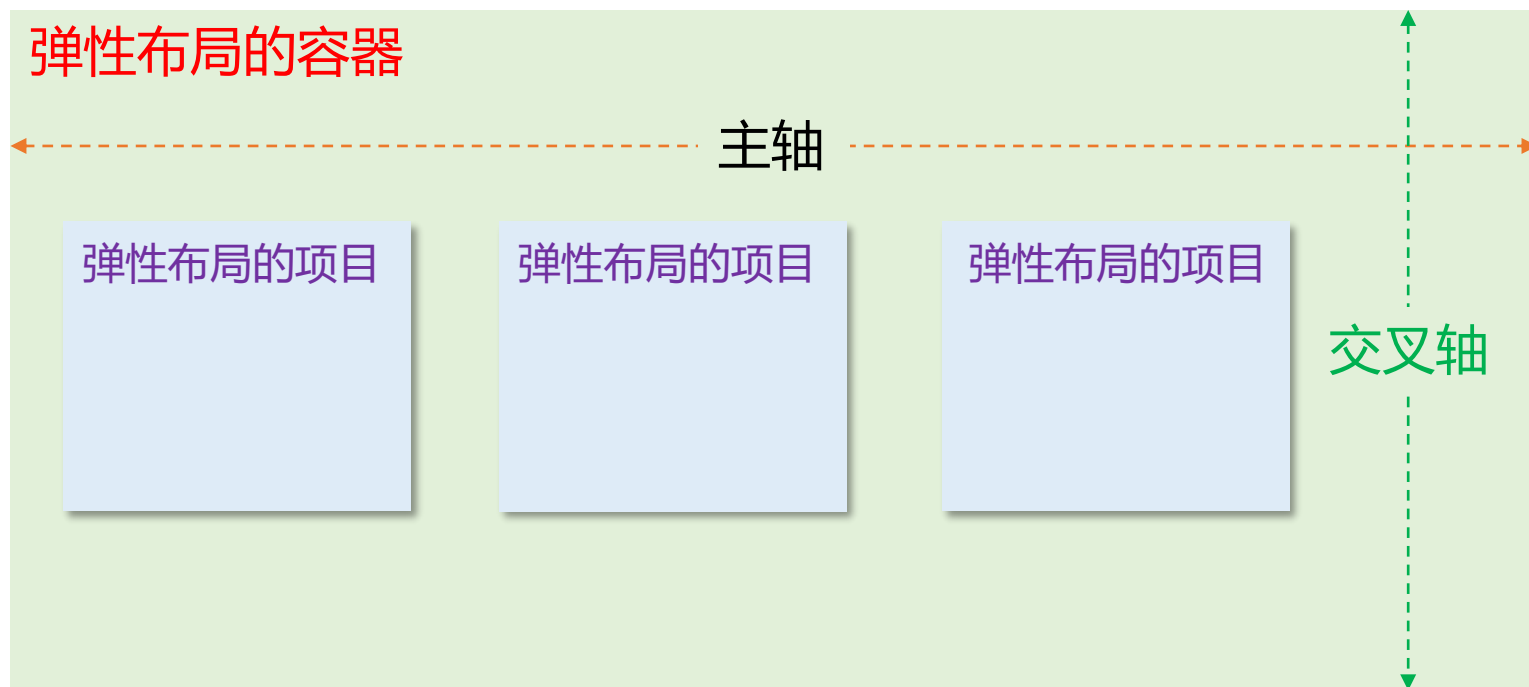
- 主轴，是指弹性布局的多个项目排列方向上的一根轴。
 - 如果弹性布局的多个项目按y轴排列，那么y轴就是主轴。



弹性布局概念



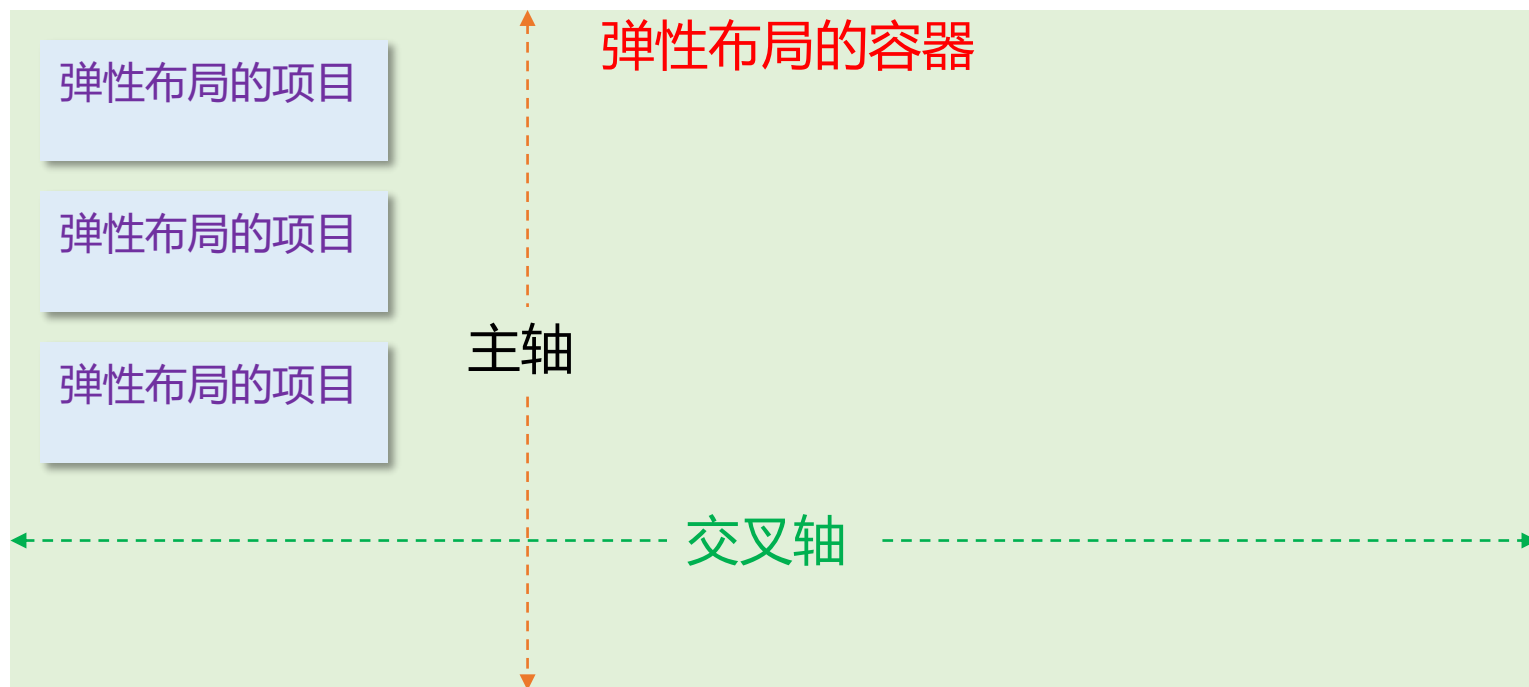
- 交叉轴，是指与主轴交叉的一根轴称为交叉轴
 - 如果主轴是x轴，那么y轴就是交叉轴。



弹性布局概念



- 交叉轴，是指与主轴交叉的一根轴称为交叉轴
 - 如果主轴是y轴，那么x轴就是交叉轴





“

2. 容器的属性

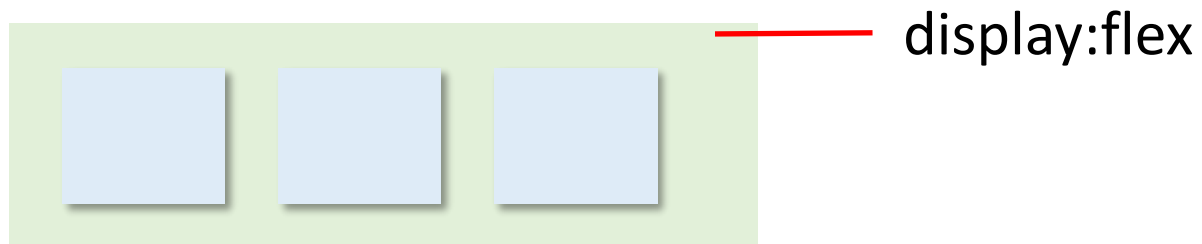
”



容器的属性



- 使用弹性布局，都要先使父元素变为弹性布局的容器。
 - 如果希望弹性布局的父元素独占布局中的一行，就要设置父元素的display属性为flex。



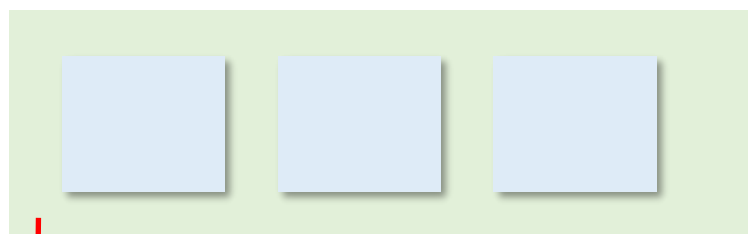
容器的属性



- 使用弹性布局，都要先使父元素变为弹性布局的容器。
 - 如果希望弹性布局的父元素显示为行内元素特征，与其它元素同在一行内，可设置父元素的display属性为inline-flex。



display:inline-flex



display: inline-flex

容器的属性

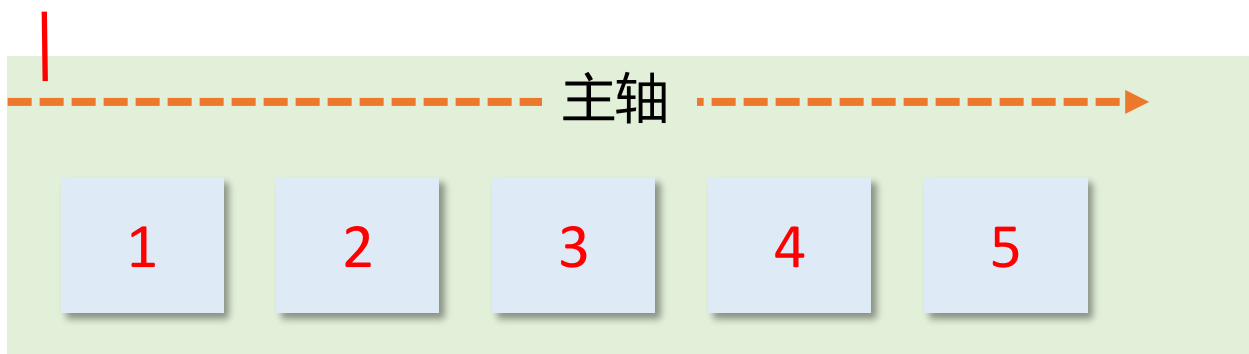
- 设置父元素为flex容器后，下一步就可以设置flex容器父元素的属性。
- 包括:
 - flex-direction属性
 - flex-wrap属性
 - flex-flow属性
 - justify-content属性
 - align-items属性



flex-direction属性

- flex-direction属性，可指定容器的主轴及其排列方向。
- 今后，只要希望设置弹性布局项目在容器内的排列方向时，就可设置容器的flex-direction属性
- 属性值包括：
 - row，默认值，即主轴是x轴，项目从最左端开始向右排列

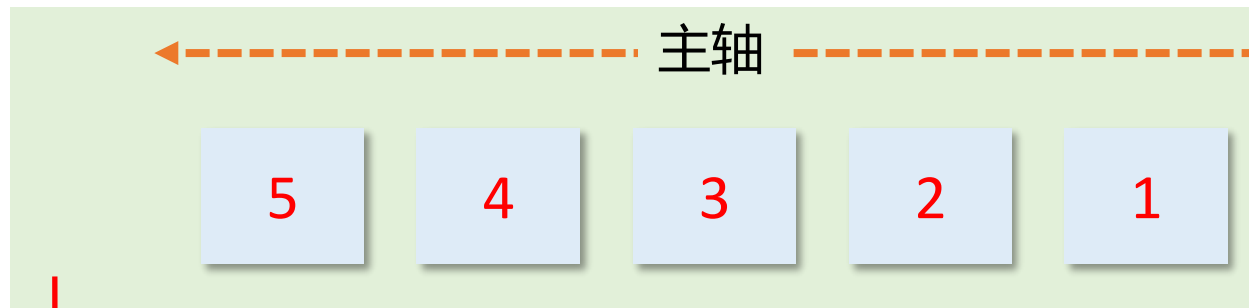
display:flex
flex-direction: row



flex-direction属性



- flex-direction属性，可指定容器的主轴及其排列方向。
- 今后，只要希望设置弹性布局项目在容器内的排列方向时，就可设置容器的flex-direction属性
- 属性值包括：
 - row-reverse，表示项目从最右侧开始，从右向左排列
 - 但是！注意项目的顺序在编写时虽然是12345，但是经过reverse反转后，实际排列的顺序变为54321



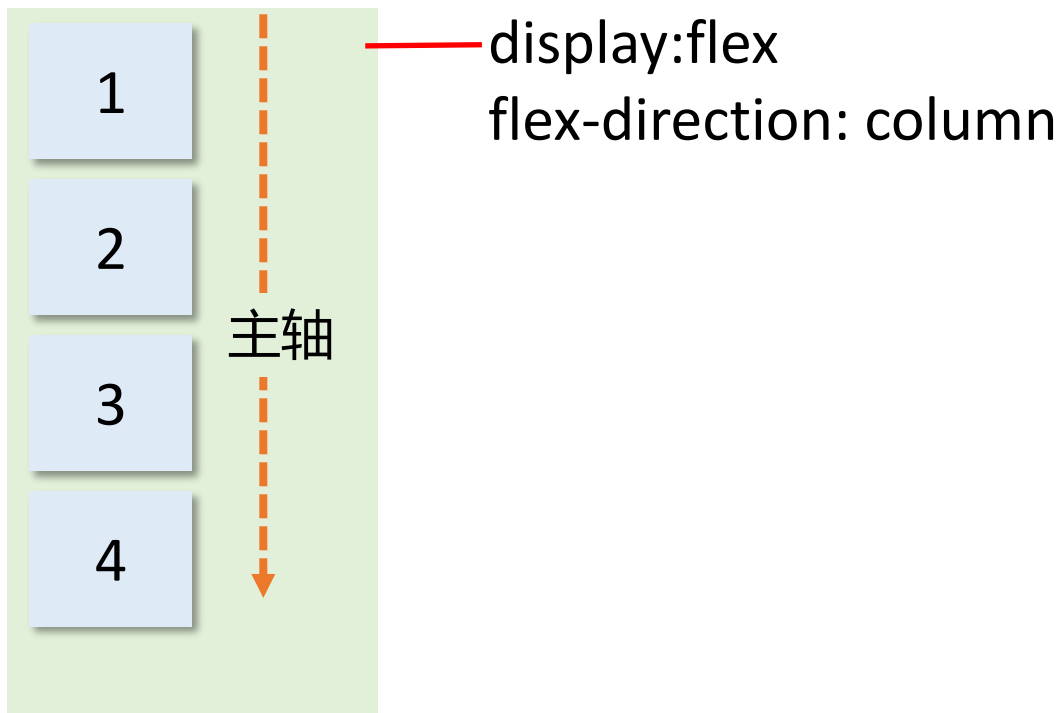
display: flex

flex-direction: row-reverse

flex-direction属性



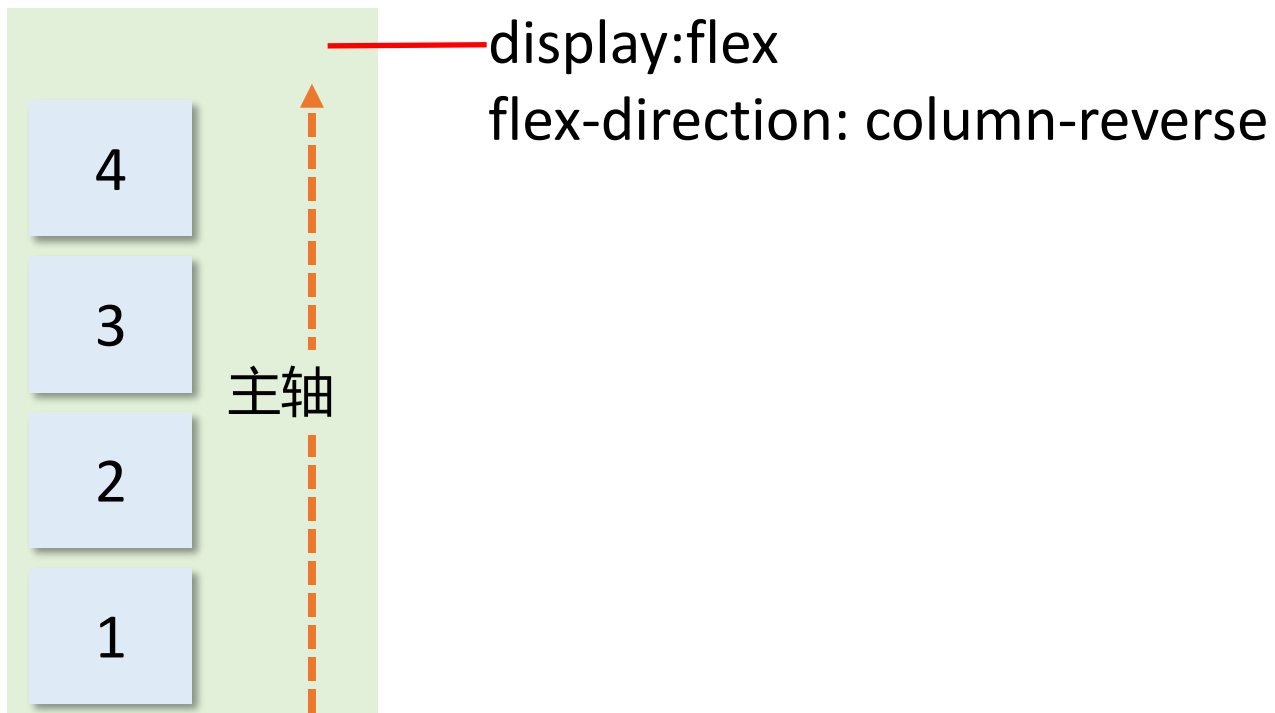
- flex-direction属性，可指定容器的主轴及其排列方向。
- 今后，只要希望设置弹性布局项目在容器内的排列方向时，就可设置容器的flex-direction属性
- 属性值包括：
 - column，主轴是y轴，项目从顶端开始，从上向下排列



flex-direction属性



- flex-direction属性，可指定容器的主轴及其排列方向。
- 今后，只要希望设置弹性布局项目在容器内的排列方向时，就可设置容器的flex-direction属性
- 属性值包括：
 - column-reverse, 主轴是y轴，项目从底部向上排列



flex-wrap属性

- flex-wrap属性，专门设置当一个主轴排列不下所有项目时，是否换行显示。
- 属性值包括：
 - nowrap，默认值，表示空间不够时，也不换行，项目自动缩小；比如，移动端网页的底部有一个占满整个宽度的导航条，并且导航条内的弹性布局项目

screen width: 1280px

1rem=16px

微信

通讯录

发现

我

🔍 📄 Elements Console Sources Network Performance Memory Application >> ⋮ ✕

```
<!doctype html>
<html lang="en">
  <head>...</head>
  <body>
    <p>...</p>
    <p>1rem=16px</p>
    <ul>
      <li>微信</li> == $0
      <li>通讯录</li>
      <li>发现</li>
      <li>我</li>
    </ul>
```

Styles Computed Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls +

element.style {
}

li {
 width: 25%;
 border: .1rem solid #aaa;
 text-align: center;
}

* {
 margin: 0;
}

5.7.1.1.html:15
5.7.1.1.html:9

flex-wrap属性

- flex-wrap属性，专门设置当一个主轴排列不下所有项目时，是否换行显示。
- 属性值包括：
 - wrap，表示当内容放不下时应该换行显示



资源管理器

▼ 打开的编辑器

× <> 5.7.1.1.html

▼ 7

> videos

<> 5.7.1.1.html

7.doc

≡ 7.pptx

> 大纲

<> 5.7.1.1.html ×

<> 5.7.1.1.html > html > head > style > ul

```
8      <style>
9          *{margin:0; padding:0;}
10         p{text-align:center; padding:8px 0}
11         ul{
12             list-style:none;
13             display: flex;|
14         }
15         li{
16             width:25%;
17             border:.1rem solid #aaa;
18             text-align:center;
19         }
20         li+li{
```

flex-flow属性

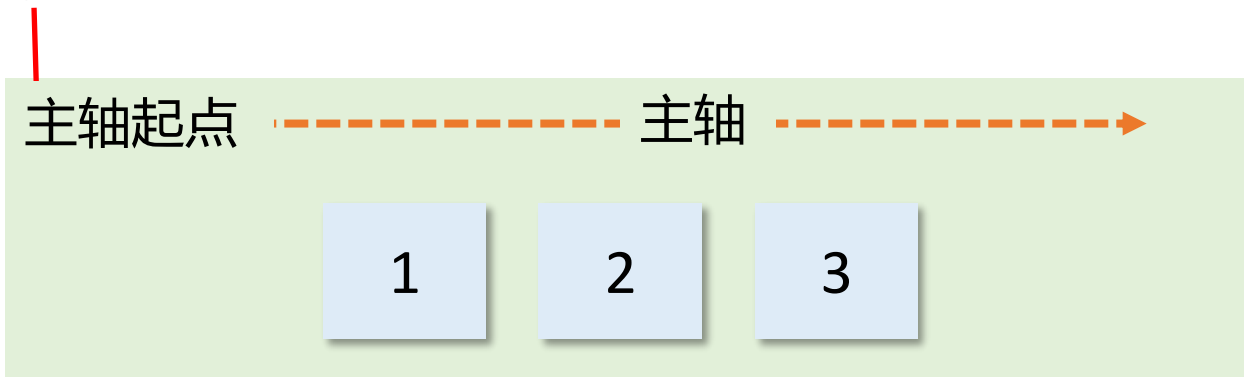
- flex-flow属性，是简写形式。
- 一个属性可同时设置flex-direction和flex-wrap两个属性
- 语法格式：
 - flex-flow: flex-direction flex-wrap
 - 其中，两个值之间用空格分割。
- 比如：“flex-flow: row nowrap”默认值，表示主轴是x轴，起点在左端，并且空间不够时，也不换行，项目自动缩小。

justify-content属性



- justify-content属性，专门定义项目在主轴上的对齐方式。
- 其取值可以是：
 - flex-start，表示以主轴的起点对齐

```
display:flex  
flex-direction: row  
justify-content: flex-start
```

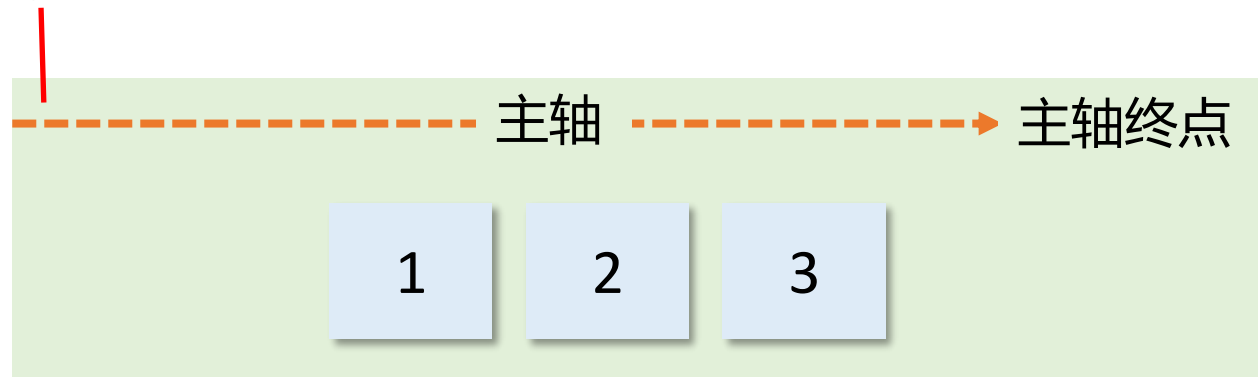


justify-content属性



- justify-content属性，专门定义项目在主轴上的对齐方式。
- 其取值可以是：
 - flex-end，表示以主轴的终点对齐

display: flex
flex-direction: row
justify-content: flex-end

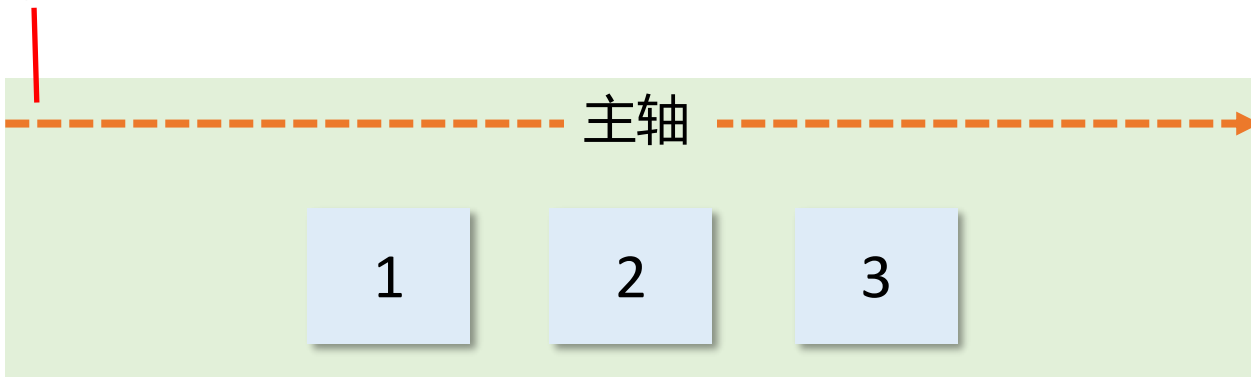


justify-content属性



- justify-content属性，专门定义项目在主轴上的对齐方式。
- 其取值可以是：
 - center，表示在主轴上居中对齐

display: flex
flex-direction: row
justify-content: center

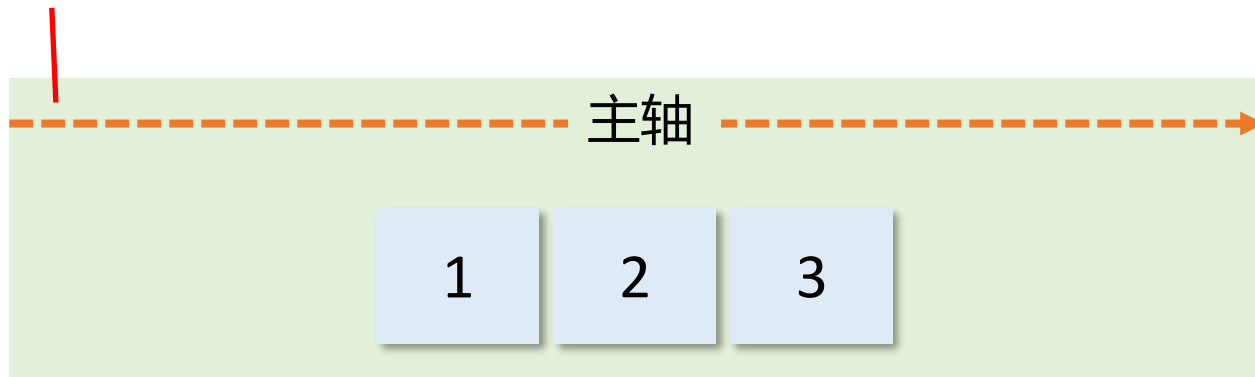


justify-content属性



- justify-content属性，专门定义项目在主轴上的对齐方式。
- 其取值可以是：
 - space-between，表示两端对齐

display:flex
flex-direction: row
justify-content: space-between

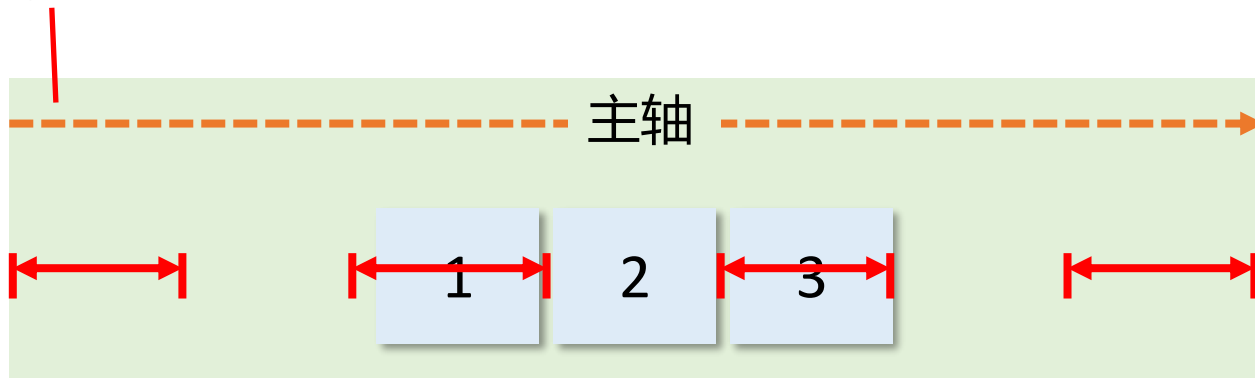


justify-content属性



- justify-content属性，专门定义项目在主轴上的对齐方式。
- 一个属性可同时设置flex-direction和flex-wrap两个属性
- 其取值可以是：
 - space-around，表示每个项目两端间距相同

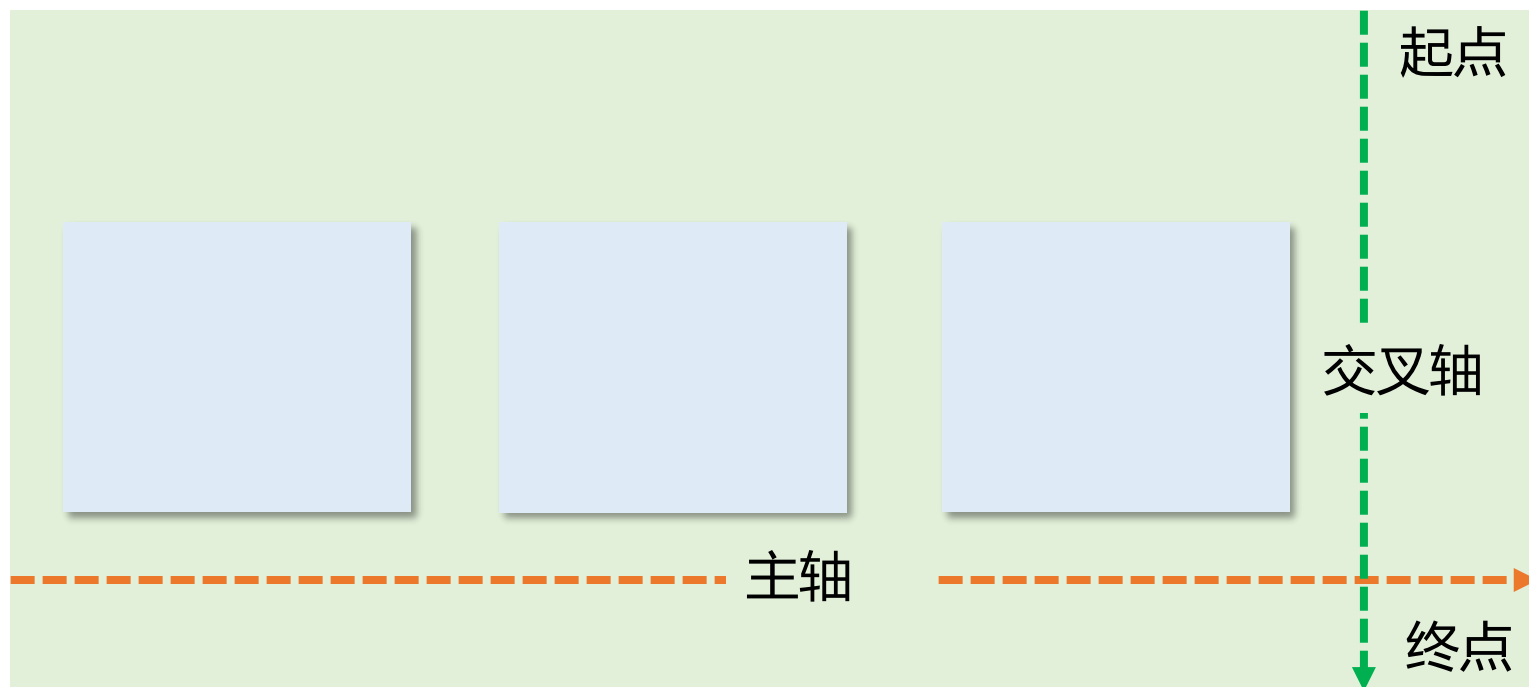
```
display:flex  
flex-direction: row  
justify-content: space-around
```



align-items属性



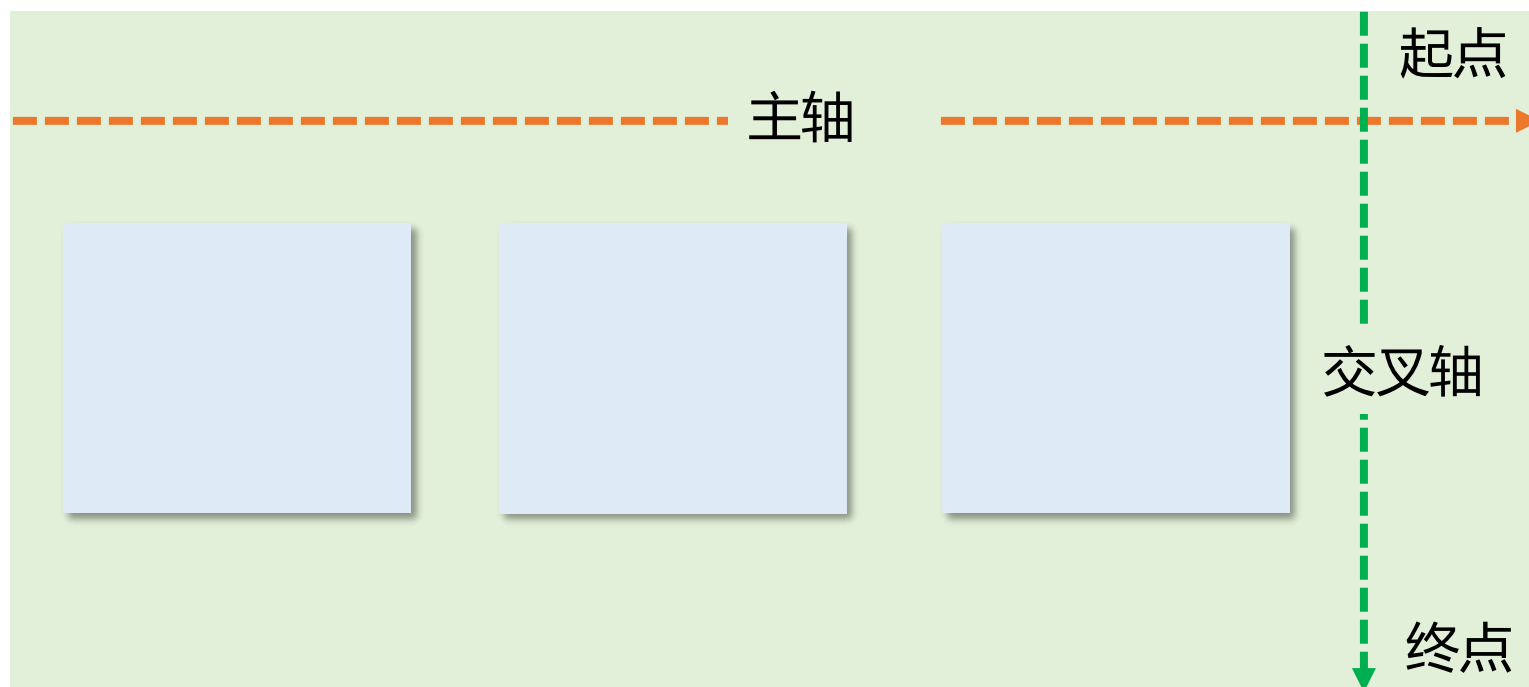
- align-items属性，专门定义所有项目在交叉轴上的统一对齐方式。
- 其取值可以是：
 - flex-start，表示让项目以交叉轴的起点方向对齐



align-items属性



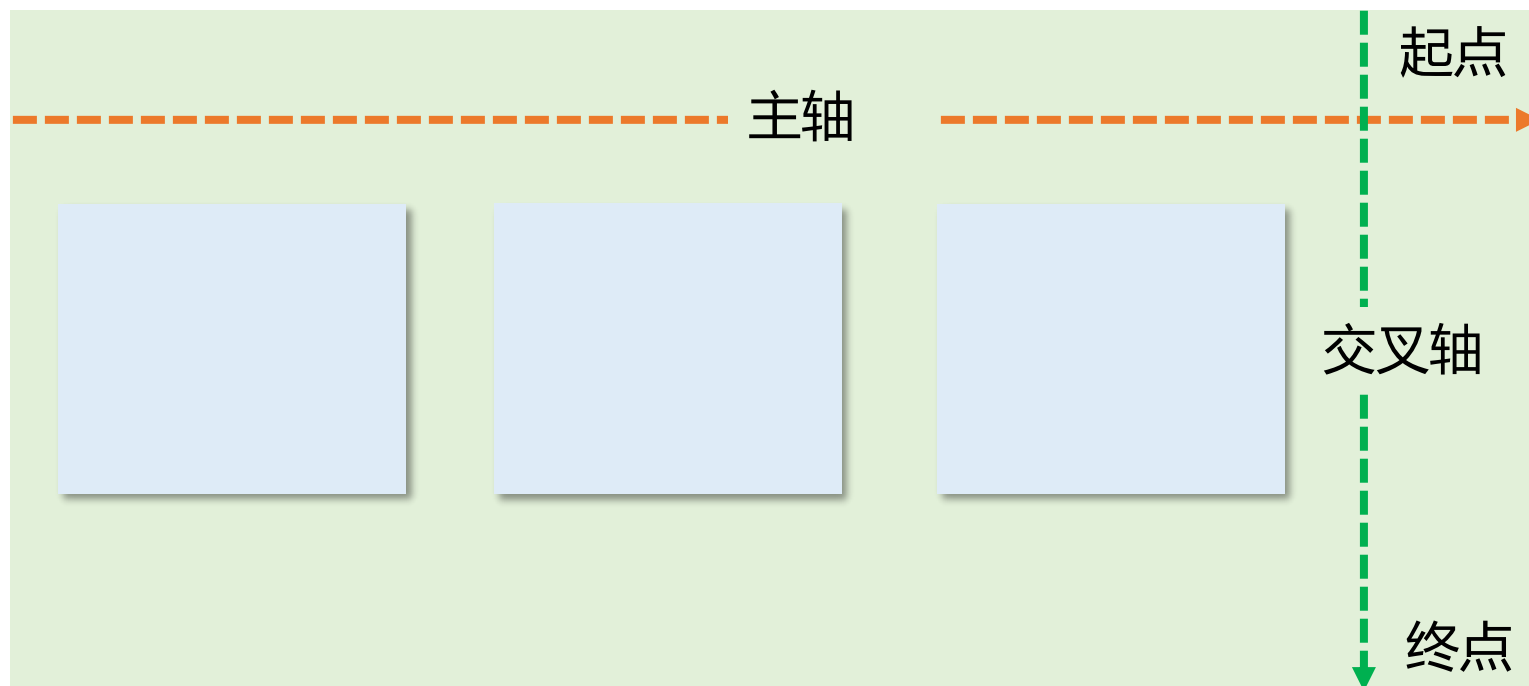
- align-items属性，专门定义所有项目在交叉轴上的统一对齐方式。
- 其取值可以是：
 - flex-end，表示让项目以交叉轴的终点方向对齐



align-items属性



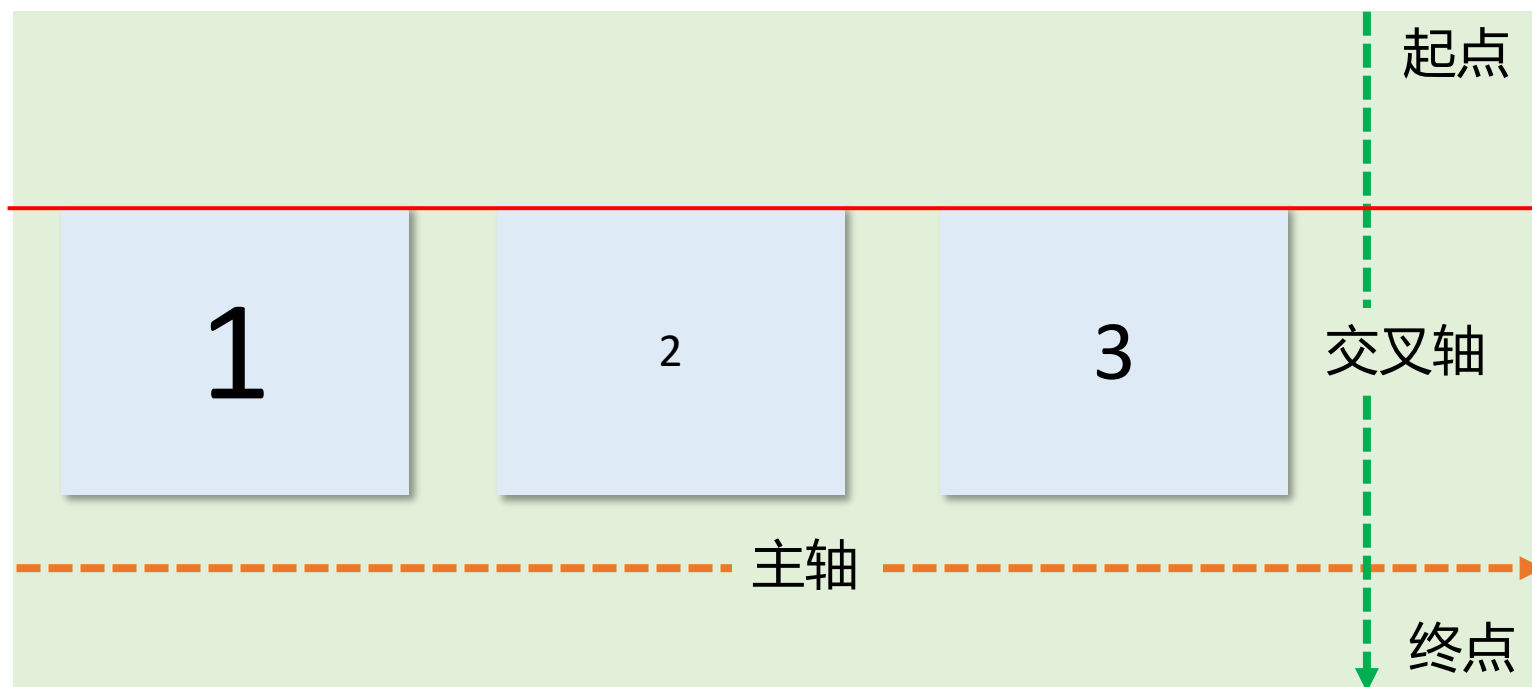
- align-items属性，专门定义所有项目在交叉轴上的统一对齐方式。
- 其取值可以是：
 - center，表示让项目以交叉轴的中线居中对齐



align-items属性



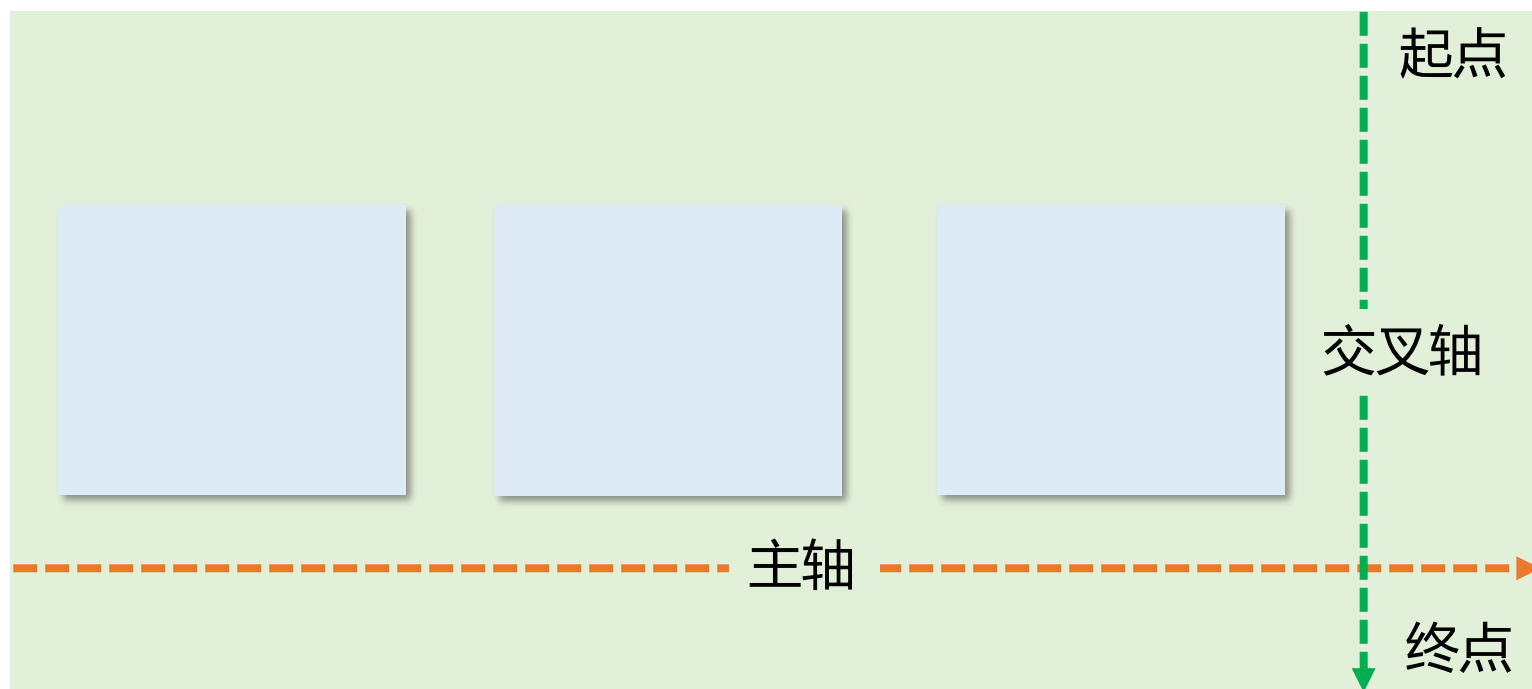
- align-items属性，专门定义所有项目在交叉轴上的统一对齐方式。
- 其取值可以是：
 - baseline，表示让项目以交叉轴的基线对齐



align-items属性



- align-items属性，专门定义所有项目在交叉轴上的统一对齐方式。
- 其取值可以是：
 - stretch，表示如果项目未设置尺寸，就让项目在交叉轴上占满所有空间。





“

3. 项目属性

”



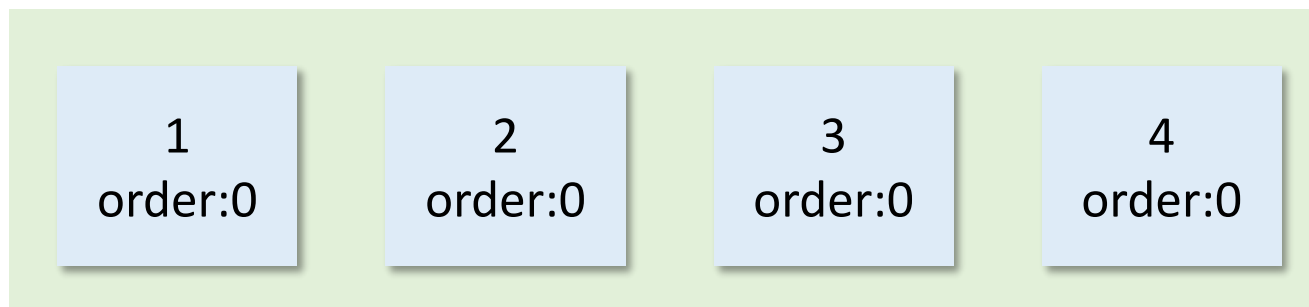
项目属性

- 除了可以对父容器设置属性外，还可对项目设置属性。
- 该组属性只能设置在项目元素上，只控制一个项目，不影响容器以及其他项目的效果。
- 今后，只要个别项目的样式与大部分项目有所不同时就可为这个特立独行的项目设置项目属性。

order属性



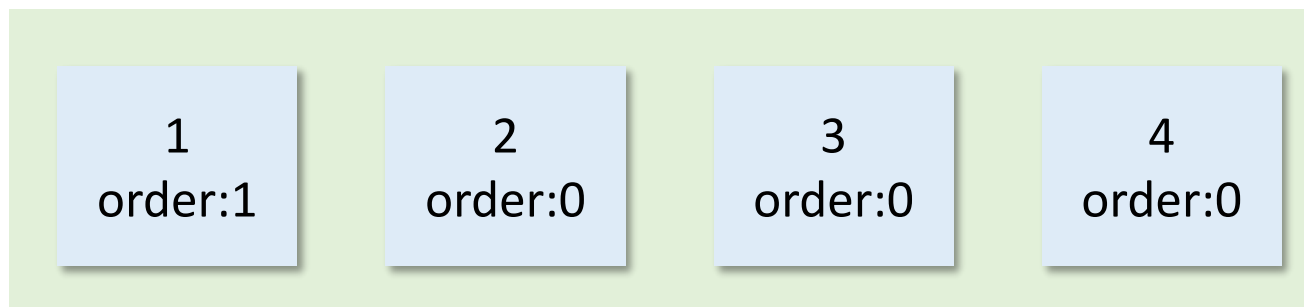
- order属性，专门定义项目的排列顺序。
- 其值为整数数字，无需单位。
- 值越小，越靠近起点，默认值是0。



order属性



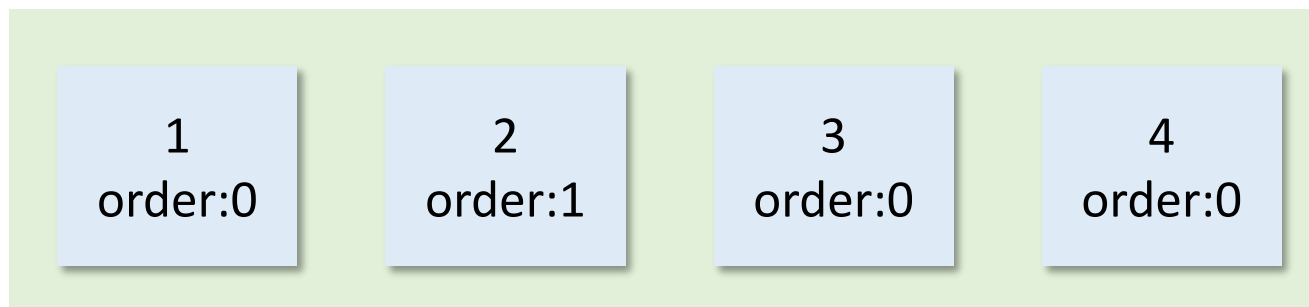
- order属性，专门定义项目的排列顺序。
- 其值为整数数字，无需单位。
- 值越小，越靠近起点，默认值是0。



order属性



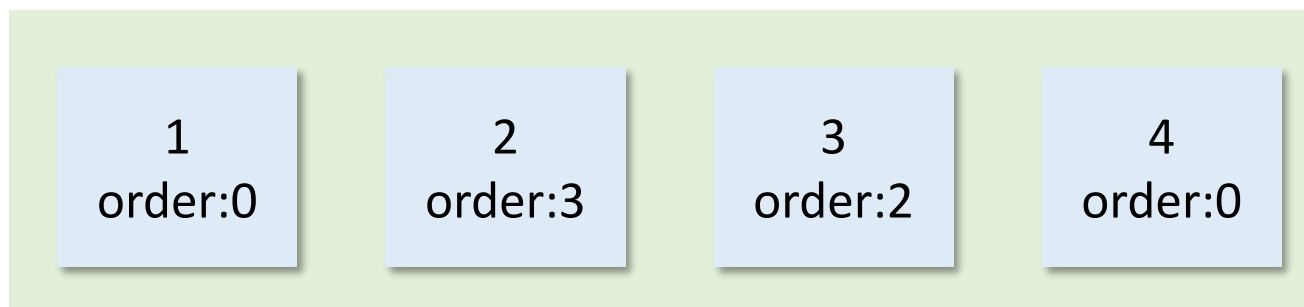
- order属性，专门定义项目的排列顺序。
- 其值为整数数字，无需单位。
- 值越小，越靠近起点，默认值是0。



order属性



- order属性，专门定义项目的排列顺序。
- 其值为整数数字，无需单位。
- 值越小，越靠近起点，默认值是0。



flex-grow属性



- flex-grow属性，专门定义项目的放大比例。
- 如果容器有足够的空间，项目可以放大。
- 其值为整数数字，无需单位。
- 默认情况，项目不放大。
- 取值越大，占据剩余的空间越多。



flex-grow属性



- flex-grow属性，专门定义项目的放大比例。
- 如果容器有足够的空间，项目可以放大。
- 其值为整数数字，无需单位。
- 默认情况，项目不放大。
- 取值越大，占据剩余的空间越多
 - 所有项目都设置flex-grow:1



flex-grow属性



- flex-grow属性，专门定义项目的放大比例。
- 如果容器有足够的空间，项目可以放大。
- 其值为整数数字，无需单位。
- 默认情况，项目不放大。
- 取值越大，占据剩余的空间越多。
 - 只给第2项设置flex-grow:2





flex-shrink属性

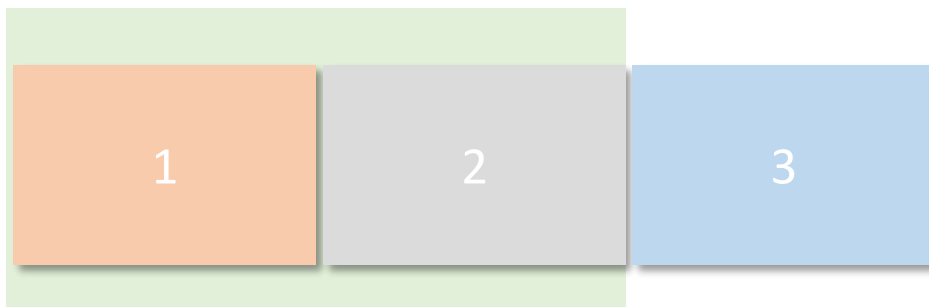
- flex-shrink属性，专门定义项目的缩小比例。
- 如果容器空间不足时，项目可以缩小。
- 缩小的比例取决于flex-shrink的值。
- 其默认值为1，表示空间不足时，则等比缩小。
- 可改为0，表示不缩小。



flex-shrink属性



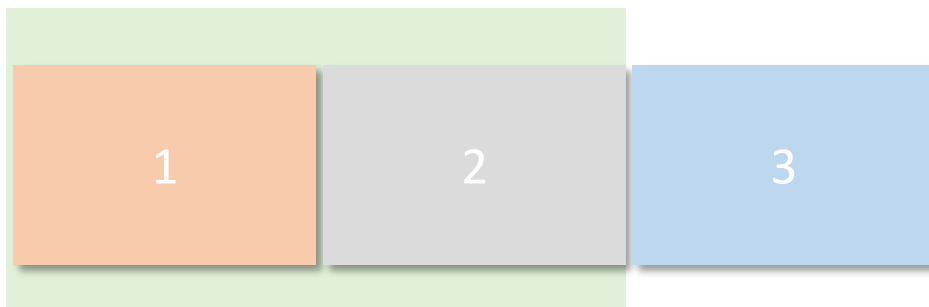
- flex-shrink属性，专门定义项目的缩小比例。
- 如果容器空间不足时，项目可以缩小。
- 缩小的比例取决于flex-shrink的值。
- 其默认值为1，表示空间不足时，则等比缩小。
- 可改为0，表示不缩小。
 - 所有项目的flex-shrink:0



flex-shrink属性



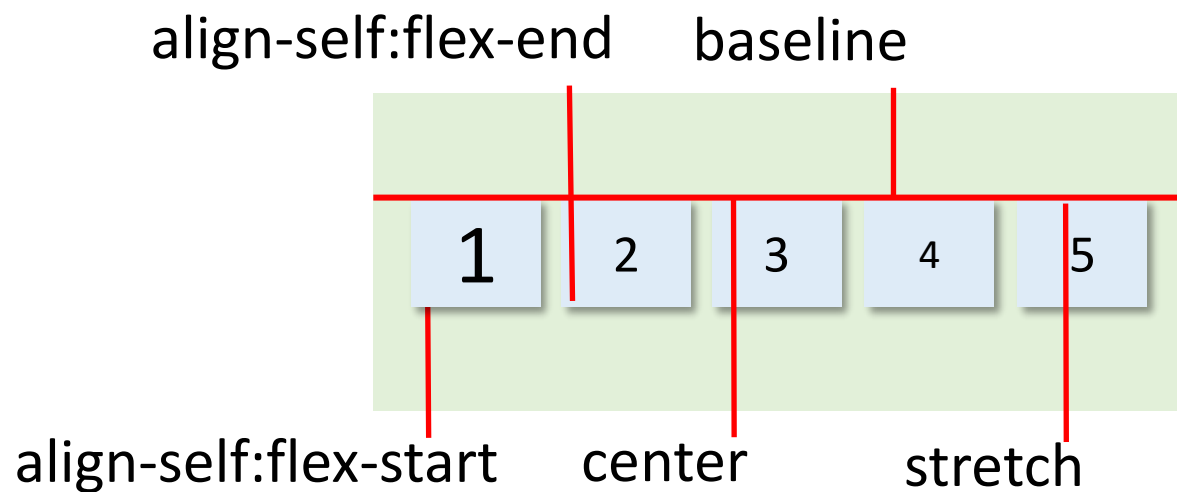
- flex-shrink属性，专门定义项目的缩小比例。
- 如果容器空间不足时，项目可以缩小。
- 缩小的比例取决于flex-shrink的值。
- 其默认值为1，表示空间不足时，则等比缩小。
- 可改为0，表示不缩小。
 - 所有项目的flex-shrink:1



align-self属性



- align-self属性，专门单独定义某一个项目在交叉轴上的对齐方式。
- 与align-items属性对比
 - align-items定义在父容器上约束所有项目的统一对齐方式
 - align-self定义在项目上，只约束当前所在的一个项目。
- 其取值和align-items完全一样。只是多了一个auto值，表示继承父元素的align-items效果。





Q3：居中的方法总结



“

3.1 水平居中总结

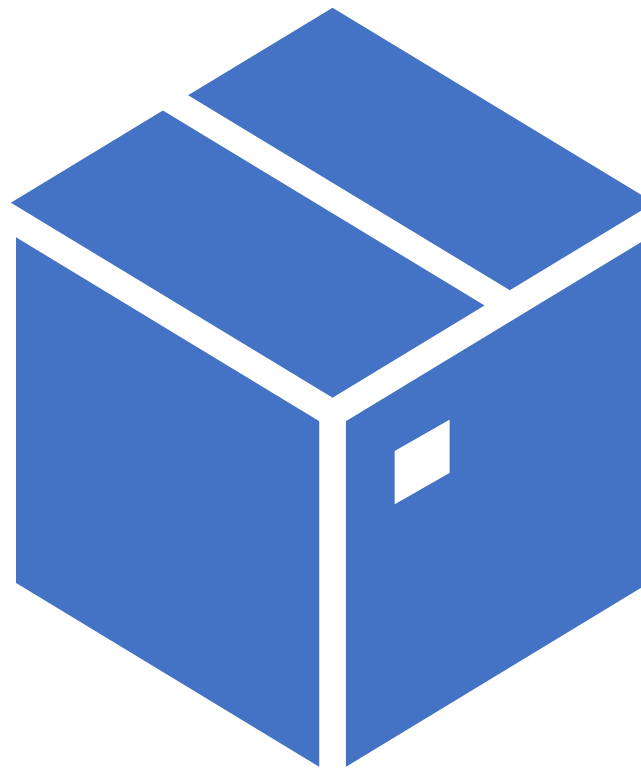
”



前提:

- 父元素必须是块级盒子容器
- 父元素宽度必须已经被设定好

不可自由嵌套的元素,
就是里面只能放内联元素的, 包括:
标题标记的<h1>、<h2>、<h3>、
<h4>、<h5>、<h6>、<caption>;
段落标记的<p>;
分隔线<hr>;
一个特别的元素<dt>(它只存在于列表元素<dl>的子一级)。



场景1:

子元素是块级元素且宽度没有设定

不存在水平居中一说

因为子元素是块级元素。没有设定宽度，那么它会充满整个父级元素的宽度，即在水平位置上宽度和父元素一致

场景2:



子元素是行内元素，子元素宽度是由其内容撑开的

这种情况下解决方案是给父元素设定

```
text-align:center;
```

场景3:



子元素是块级元素且宽度已经设定



方案一: 给子元素添加margin:0 auto;

场景3:



子元素是块级元素且宽度已经设定



方案二:通过计算指定父元素的padding-left
或padding-right或margin-left或margin-right。

给父元素和子元素都设定了box-sizing:border-box
 $(\text{父宽}-\text{子宽})/2$

场景3:

子元素是块级元素且宽度已经设定

方案三: 通过子元素相对父元素绝对定位

- 父相子绝
- 子:
 - left:50%;
 - margin-left: -子宽一半 或 transform:translateX(-50%)



场景3:

子元素是块级元素且宽度已经设定

弹性布局: 父

- `display: flex;`
- `flex-direction: row;`
- `justify-content: center;`



“

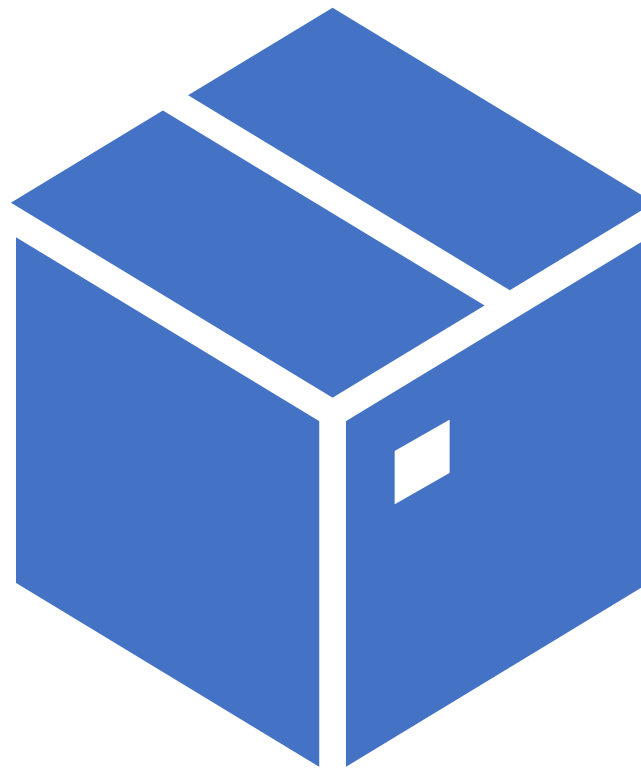
3.2 垂直居中总结

”



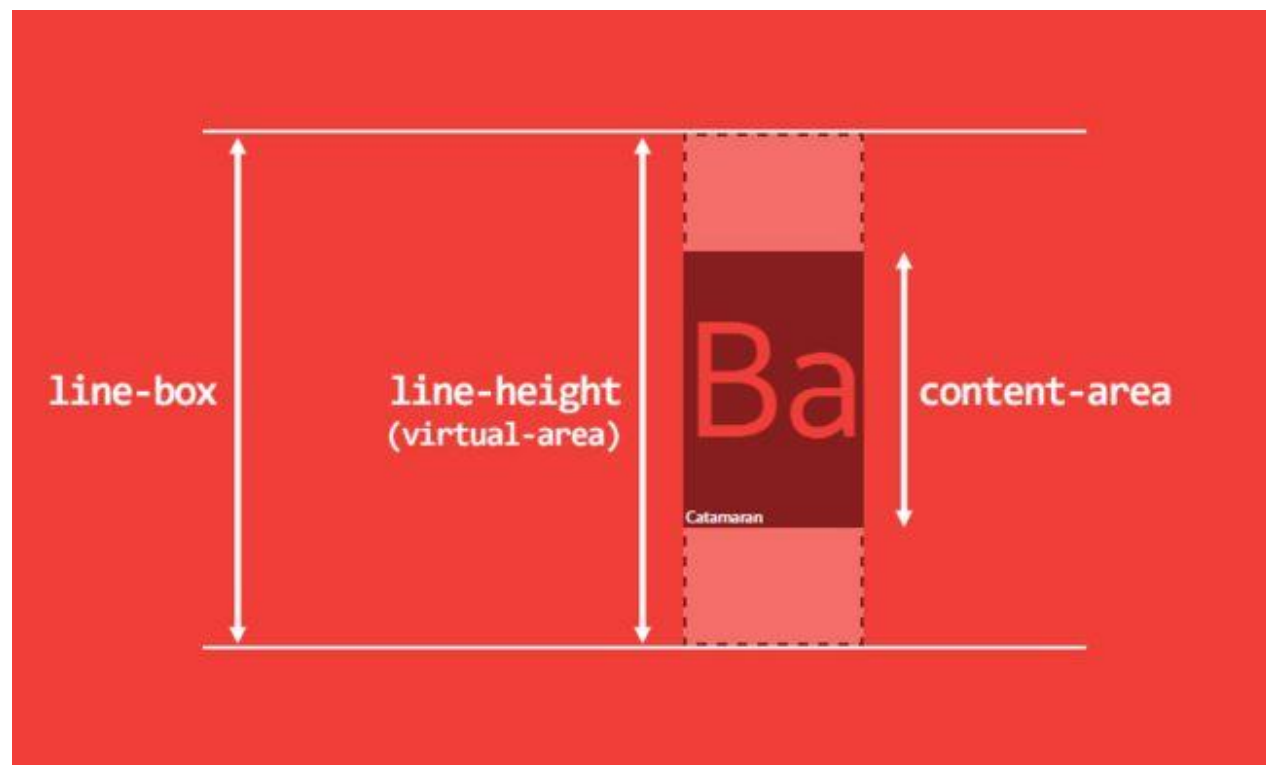
前提:

- 父元素是盒子容器



场景1:

- 子元素是行内元素，高度是由其内容撑开的
- 单行: 设定父元素的line-height为其高度来使得子元素垂直居中



场景1:

- 子元素是行内元素，高度是由其内容撑开的
- 多行: 通过给父元素设定display:inline/inline-block/table-cell; vertical-align:middle来解决

table-cell

子
子
子
子



场景2:

- 子元素是块级元素但是子元素高度没有设定
- 解法1: 通过给父元素设定
`display:inline/inline-block/table-cell; vertical-align:middle`来解决
- 解法2: flexbox
 - 父元素:
`display: flex;`
`flex-direction: column;`
`justify-content: center;`

场景3:

子元素是块级元素且高度已经设定

解法1: 计算子元素的margin-top或margin-bottom或padding-top或padding-bottom

给父元素和子元素都设定了box-sizing:border-box

$(\text{父高} - \text{子高}) / 2$

场景3:



子元素是块级元素且高度已经设定

解法2: 利用绝对定位, 让子元素相对于父元素绝对定位

父相子绝: 子元素高**已知**

子: `top:50%; margin-top: -子高一半`

场景3:



子元素是块级元素且高度已经设定


解法2: 利用绝对定位, 让子元素相对于父元素绝对定位

父相子绝: 子元素高**未知**

子: top:50%; transform:translateY(-50%)

- 父元素
 - display: flex;
 - flex-direction: column;
 - justify-content: center;

解法4：利用
flexbox





“

3.3 垂直和水平 同时居中...

”



父相子绝

已知子元素宽高:

```
top:50%; left:50%;  
margin-left:-子宽一半;  
margin-top:-子高一半
```

未知子元素宽高

```
top:50%; left:50%;  
transform:translate(-50%, -  
50%)
```



弹性布局

- .son{
- display: flex;
- justify-content: center;
- align-items: center;
- }



Q4: 移动端/响应式(思维导图)



我们的口号是：
早学早面试卷别人
晚学晚面试被人卷

paikaba
开课吧