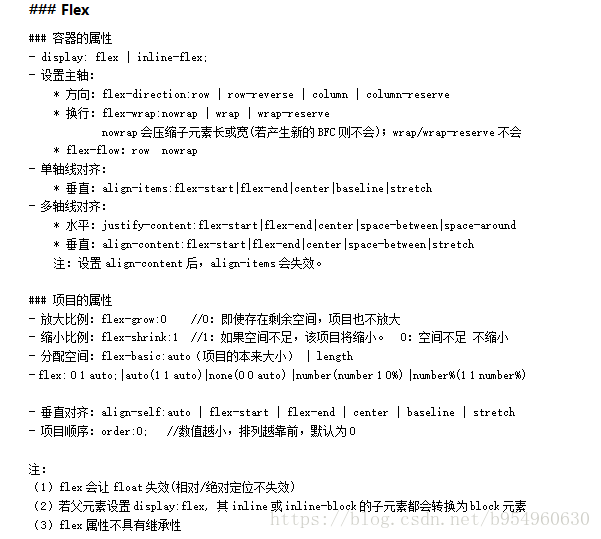
**Flex布局实战（一）：骰子**

2018-10-13 03:08:43 [我叫白天宇](https://me.csdn.net/b954960630) 阅读数 1244更多

分类专栏： [web前端](https://blog.csdn.net/b954960630/article/category/6188270) [CSS](https://blog.csdn.net/b954960630/article/category/7358928)

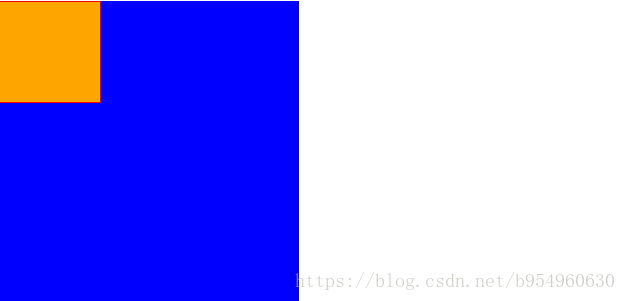
版权声明：本文为博主原创文章，遵循[CC 4.0 BY-SA](http://creativecommons.org/licenses/by-sa/4.0/)版权协议，转载请附上原文出处链接和本声明。

本文链接：<https://blog.csdn.net/b954960630/article/details/83034971>

  
参考：<http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html>

如果你对骰子布局已经了然于心，好吧，那就直接跳到  
**七、理解flex-basis与flex-grow** 看吧。

**一、单项目**

（0）初始化  
  
为父元素添加display: flex后，item自动转换为block元素

<style type="text/css">

.box{

background: blue;

display: flex; /\*为父元素添加flex后，span.item自动转换为block元素\*/

width: 300px;

height: 300px;

}

.item{

background: orange;

width: 100px;

height: 100px;

border: 1px solid red;

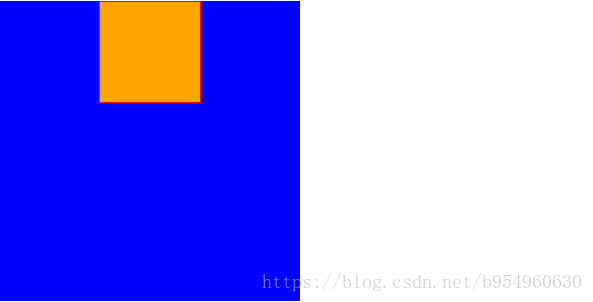
}

</style>

<div class="box">

<span class="item"></span>

</div>

（1）  


.box{

display: flex; /\*1\*/

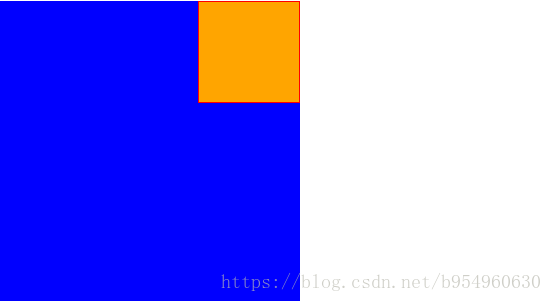
justify-content: center; /\*2\*/

background: blue;

width: 300px;

height: 300px;

}

（2）  


.box{

display: flex; /\*1\*/

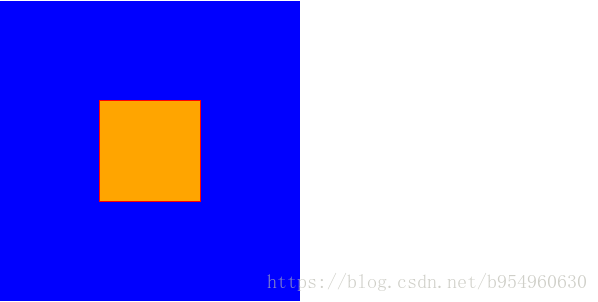
justify-content: flex-end; /\*2\*/

background: blue;

width: 300px;

height: 300px;

}

（3）  


.box{

display: flex; /\*1\*/

justify-content: center; /\*2\*/

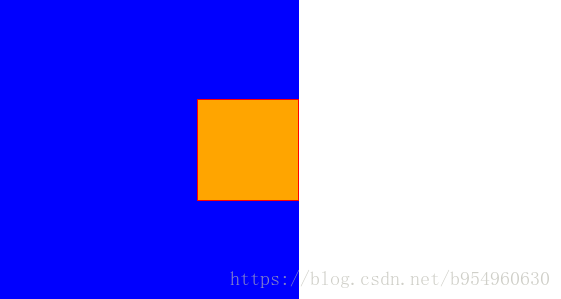
align-items: center; /\*3\*/

background: blue;

width: 300px;

height: 300px;

}

（4）  


.box{

display: flex; /\*1\*/

justify-content: flex-end; /\*2\*/

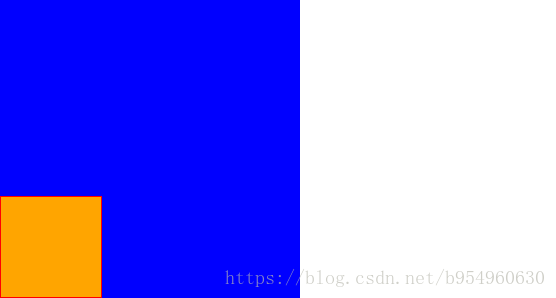
align-items: center; /\*3\*/

background: blue;

width: 300px;

height: 300px;

}

（5）  


.box{

display: flex; /\*1\*/

justify-content: flex-start; /\*2\*/

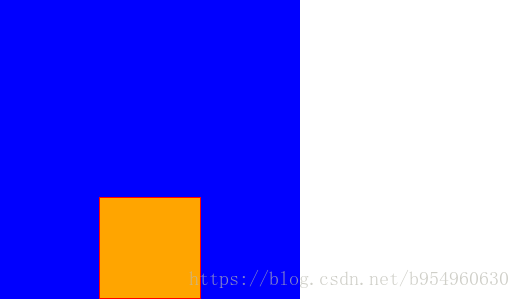
align-items: flex-end; /\*3\*/

background: blue;

width: 300px;

height: 300px;

}

（6）  


.box{

display: flex; /\*1\*/

justify-content: center; /\*2\*/

align-items: flex-end; /\*3\*/

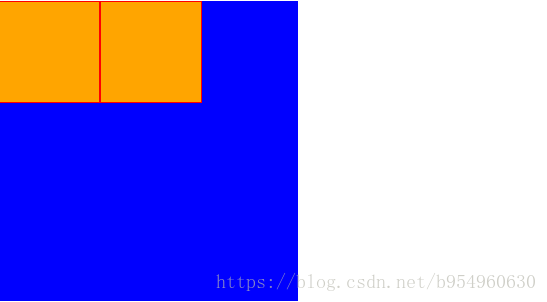
background: blue;

width: 300px;

height: 300px;

}

**二、双项目**

（0）初始化  


<style type="text/css">

.box{

display: flex; /\*1\*/

background: blue;

width: 300px;

height: 300px;

}

.item{

background: orange;

width: 100px;

height: 100px;

border: 1px solid red;

}

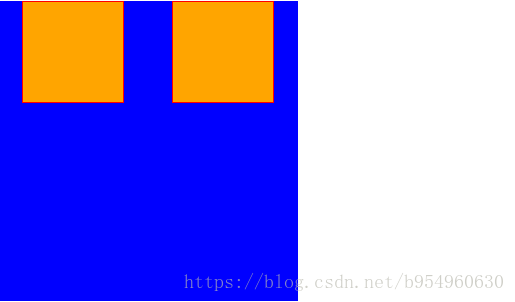
</style>

<div class="box">

<span class="item"></span>

<span class="item"></span>

</div>

（1）  


.box{

display: flex; /\*1\*/

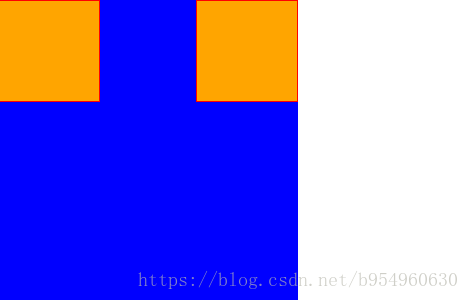
justify-content: space-around; /\*2\*/

background: blue;

width: 300px;

height: 300px;

}

（2）  


.box{

display: flex; /\*1\*/

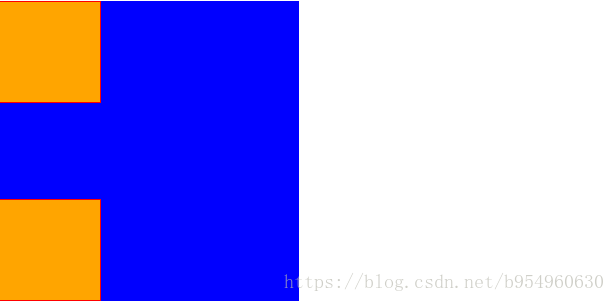
justify-content:space-between; /\*2\*/

background: blue;

width: 300px;

height: 300px;

}

（3）  
  
注：flex-direction: column；即让justify-content控制纵向，让align-items控制横向。

flex-direction: column；后相当于把(2)进行了旋转从而得到(3)

.box{

display: flex; /\*1\*/

flex-direction: column; /\*2\*/

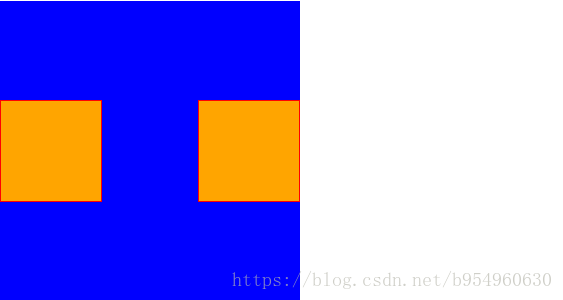
justify-content:space-between; /\*3\*/

background: blue;

width: 300px;

height: 300px;

}

（4）  


.box{

display: flex; /\*1\*/

justify-content:space-between; /\*2\*/

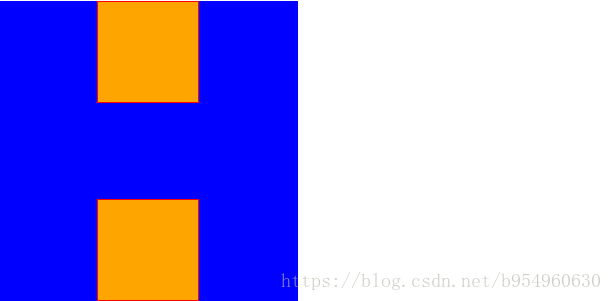
align-items: center; /\*3\*/

background: blue;

width: 300px;

height: 300px;

}

（5）  
  
注：flex-direction: column；即让justify-content控制纵向，让align-items控制横向。

flex-direction: column；后相当于把(4)进行了旋转从而得到(5)

.box{

display: flex; /\*1\*/

flex-direction: column; /\*2\*/

justify-content:space-between; /\*3\*/

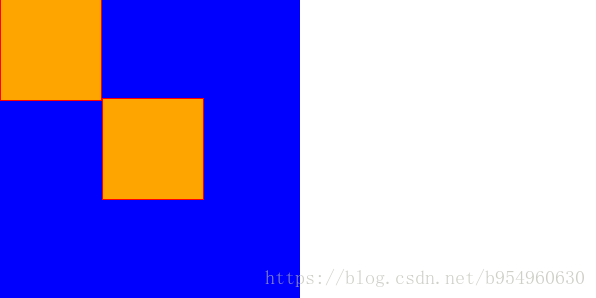
align-items: center; /\*4\*/

background: blue;

width: 300px;

height: 300px;

}

（6）  
  
如果不设置align-self: center，即为上面的(0)初始化。  
我们让第二个item元素垂直居中，即得到上图。

.box{

display: flex; /\*1\*/

background: blue;

width: 300px;

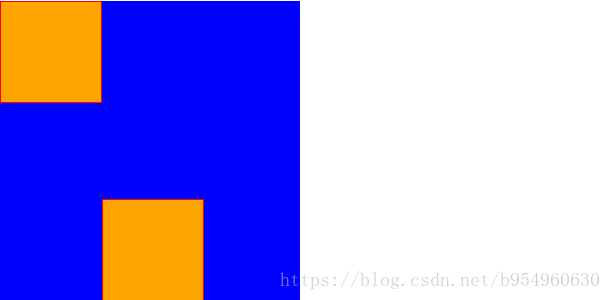
height: 300px;

}

.item:nth-child(2){

align-self: center; /\*2\*/

}

（7）  


.box{

display: flex; /\*1\*/

background: blue;

width: 300px;

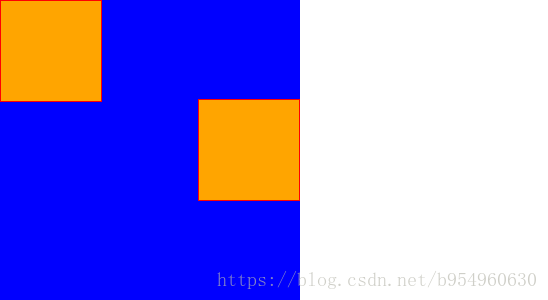
height: 300px;

}

.item:nth-child(2){

align-self: flex-end; /\*2\*/

}

（8）  


.box{

display: flex; /\*1\*/

justify-content: space-between; /\*2\*/

background: blue;

width: 300px;

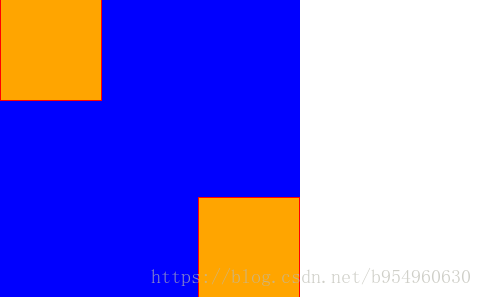
height: 300px;

}

.item:nth-child(2){

align-self: center; /\*3\*/

}

（9）  


.box{

display: flex; /\*1\*/

justify-content: space-between; /\*2\*/

background: blue;

width: 300px;

height: 300px;

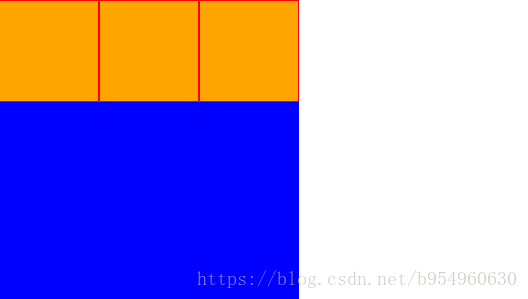
}

.item:nth-child(2){

align-self: flex-end; /\*3\*/

}

**三、三项目**

（0）初始化  


<style type="text/css">

.box{

display: flex; /\*1\*/

background: blue;

width: 300px;

height: 300px;

}

.item{

background: orange;

width: 100px;

height: 100px;

border: 1px solid red;

}

</style>

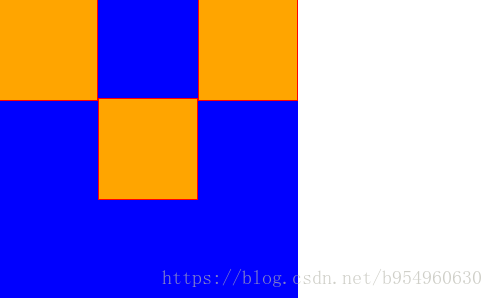
<div class="box">

<span class="item"></span>

<span class="item"></span>

<span class="item"></span>

</div>

（1）  


.box{

display: flex; /\*1\*/

background: blue;

width: 300px;

height: 300px;

}

.item:nth-child(2){

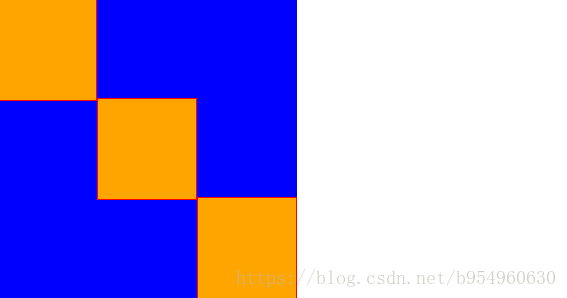
align-self: center; /\*2\*/

}

.item:nth-child(3){

align-self: flex-start; /\*3\*/

}

（2）  


.box{

display: flex; /\*1\*/

background: blue;

width: 300px;

height: 300px;

}

.item:nth-child(2){

align-self: center; /\*2\*/

}

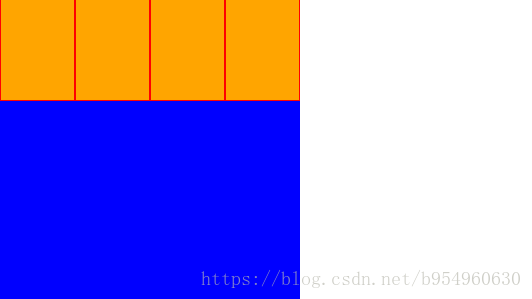
.item:nth-child(3){

align-self: flex-end; /\*3\*/

}

**四、四项目**

**1、情景一**

（0）初始化  
  
我们发现，虽然为每个item设置了98px的宽度，但是由于flex-wrap默认值为nowrap(不换行)，所以每个item的长度被压缩了。

如果我们把每个item的width设为110px, 最后得到的结果还是和上图一样，width被压缩了。

<style type="text/css">

.box{

display: flex; /\*1\*/

background: blue;

width: 300px;

height: 300px;

}

.item{

background: orange;

width: 98px;

height: 98px;

border: 1px solid red;

}

</style>

<div class="box">

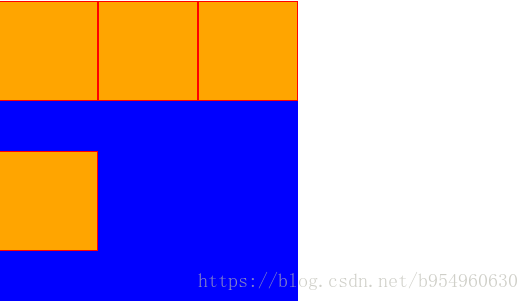
<span class="item"></span>

<span class="item"></span>

<span class="item"></span>

<span class="item"></span>

</div>

（1）  
  
将flex-wrap值设为wrap后（即允许换行），每个item又恢复为原来的宽度。

.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*2\*/

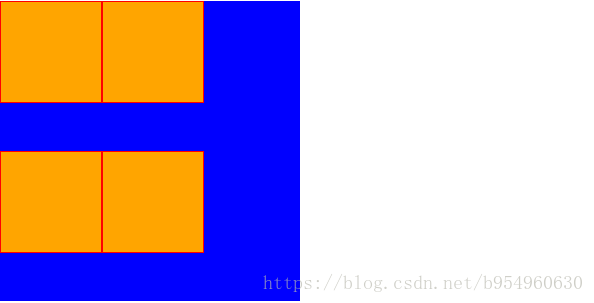
background: blue;

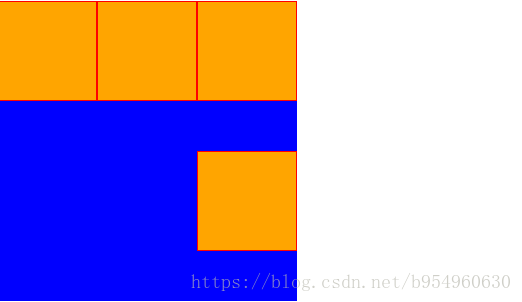
width: 300px;

height: 300px;

}

**提问：**  
在上图(1)的基础上，把item的width改为100px，且采用的是W3标准盒模型，布局会变成什么样？

**分析：**  
由于采用W3标准盒模型，所以当把item改为100px后，item的真实长度变为 (2 + 100) = 102px。  
又因为我们使用了flex-wrap: wrap;即允许换行，所以item的width不会被压缩。而box的width为300px，一行排不下3个item元素，所以排到第三个item时就会另起一行，变成下图这样。  


（2）  
  
既然我们是在父元素加 justify-content、align-items属性，所以这两个属性肯定是对其所有子元素（看成一个整体）作用的。

在这里，我们把4个item当作一个整体操作的，当设置justify-content: flex-end后，前三个item由于已经填充满了一行，所以并没有变化，而最后一个item由于没有填充满一行，所以就移到了这行中最靠后的位置。

.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*2\*/

justify-content: flex-end; /\*3\*/

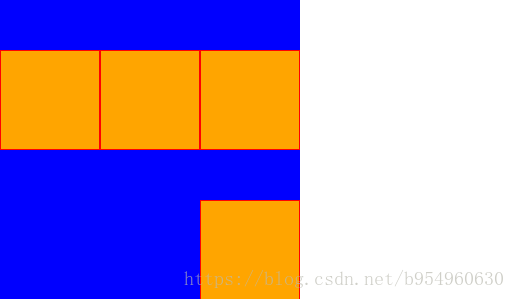
/\* 加上align-items: flex-start; 什么都不会发生，原因往下看\*/

background: blue;

width: 300px;

height: 300px;

}

（3）  
  
既然我们是在父元素加 justify-content、align-items属性，所以这两个属性肯定是对其所有子元素（看成一个整体）作用的。

通过加上align-items: flex-end;的位置变化可以看出，4个item是被看作一个整体的，所以(2)加上align-items: flex-start;什么都不会发生。

.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*2\*/

justify-content: flex-end; /\*3\*/

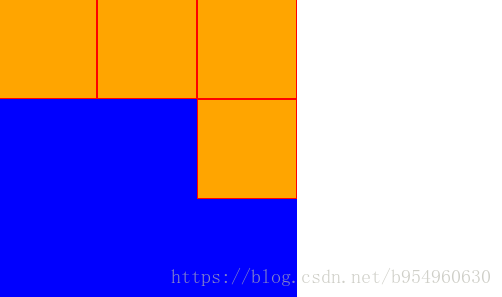
align-items: flex-end; /\*4\*/

background: blue;

width: 300px;

height: 300px;

}

（4）  
  
这里用了align-content（即：多周线对齐设置），所以再设置align-items会无效化。

.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*2\*/

justify-content: flex-end; /\*3\*/

align-content: flex-start; /\*4\*/ /\*多周线对齐设置\*/

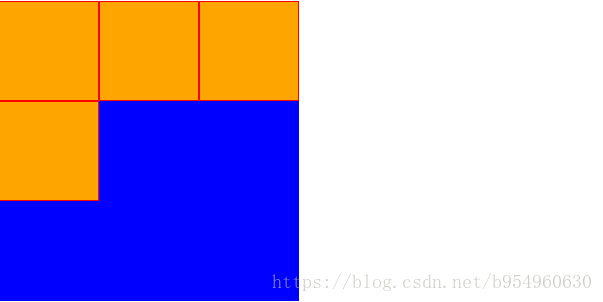
/\*因为用了align-content，所以此处设置align-items为任何值都无效\*/

background: blue;

width: 300px;

height: 300px;

}

（5）  


.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*2\*/

justify-content: flex-start; /\*3\*/

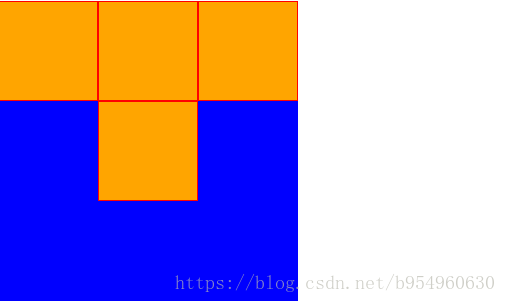
align-content: flex-start; /\*4\*/ /\*多周线对齐设置\*/

background: blue;

width: 300px;

height: 300px;

}

（6）  


.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*2\*/

justify-content: center; /\*3\*/

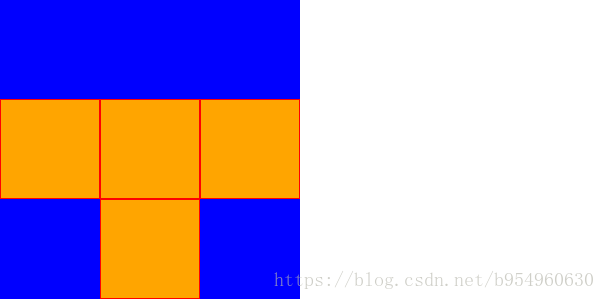
align-content: flex-start; /\*4\*/ /\*多周线对齐设置\*/

background: blue;

width: 300px;

height: 300px;

}

（7）  


.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*2\*/

justify-content: center; /\*3\*/

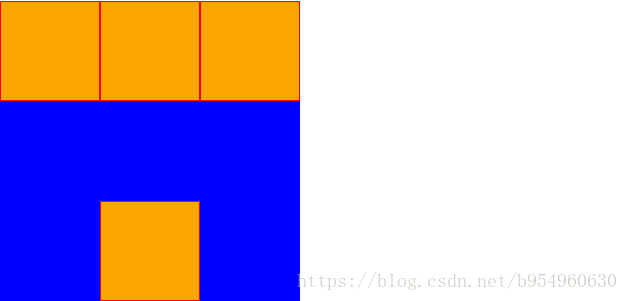
align-content: flex-end; /\*4\*/ /\*多周线对齐设置\*/

background: blue;

width: 300px;

height: 300px;

}

（8）  
  
我们可以把前三个item看成一个整体，最后一个item看成一个整体，所以设置 align-content: space-between;后变成了上图。

.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*2\*/

justify-content: center; /\*3\*/

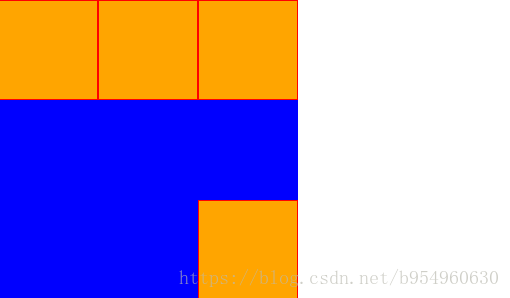
align-content: space-between; /\*4\*/ /\*多周线对齐设置\*/

background: blue;

width: 300px;

height: 300px;

}

（9）  


.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*2\*/

justify-content: flex-end; /\*3\*/

align-content: space-between; /\*4\*/ /\*多周线对齐设置\*/

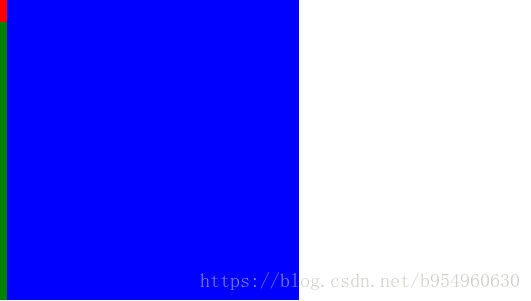
background: blue;

width: 300px;

height: 300px;

}

**2、情景二**

（0）初始化  


<style type="text/css">

.box{

display: flex; /\*1\*/

background: blue;

width: 300px;

height: 300px;

}

.column{

background:green;

}

.item{

background: orange;

width: 98px;

height: 98px;

border: 1px solid red;

}

</style>

<div class="box">

<div class="column">

<span class="item"></span>

<span class="item"></span>

</div>

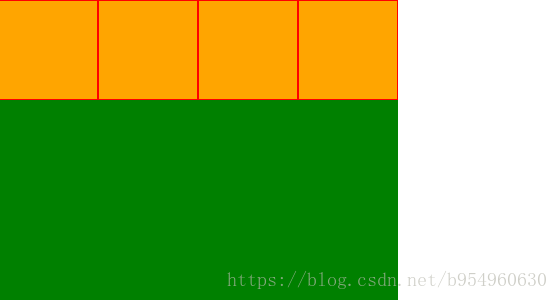
<div class="column">

<span class="item"></span>

<span class="item"></span>

</div>

</div>

（1）  


.box{

margin-top: 30px;

display: flex; /\*1\*/

background: blue;

width: 300px;

height: 300px;

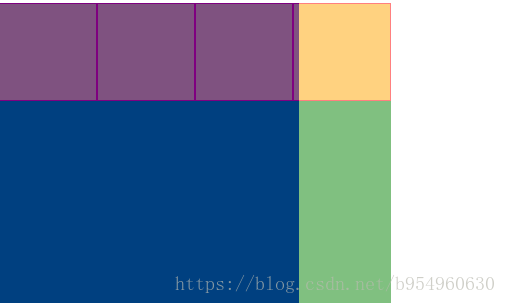
}

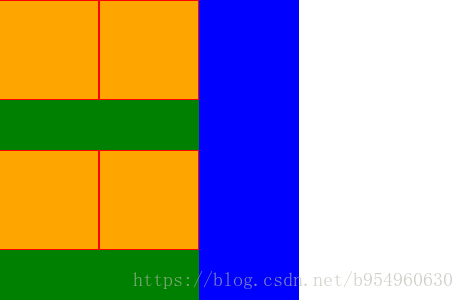
.column{

display: flex; /\*2\*/

background:green;

}

* 我们为column加上了display: flex后，item就出现了。  
  说明flex属性不具有继承性！
* 且为column加上display: flex后，column的width为200px，两个column也就是400px, 所以会覆盖box（这里column的width没有被压缩，是因为给column设置了flex后，产生了新的BFC）  
  所以为了避免这种情况发生，我们可以将column的width设为150px  
  

（2）  


.box{

margin-top: 30px;

display: flex; /\*1\*/

flex-wrap:wrap; /\*3\*/

background: blue;

width: 300px;

height: 300px;

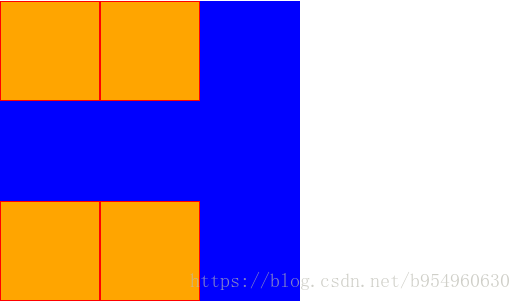
}

.column{

display: flex; /\*2\*/

background:green;

}

（3）  


.box{

margin-top: 30px;

display: flex; /\*1\*/

flex-wrap:wrap; /\*3\*/

align-content: space-between; /\*4\*/

/\*此时加justify-content: space-between;什么都不会发生\*/

background: blue;

width: 300px;

height: 300px;

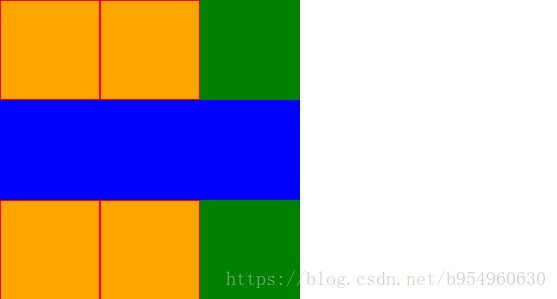
}

.column{

display: flex; /\*2\*/

background:green;

}

（4）  


.box{

margin-top: 30px;

display: flex; /\*1\*/

flex-wrap:wrap; /\*3\*/

align-content: space-between; /\*4\*/

background: blue;

width: 300px;

height: 300px;

}

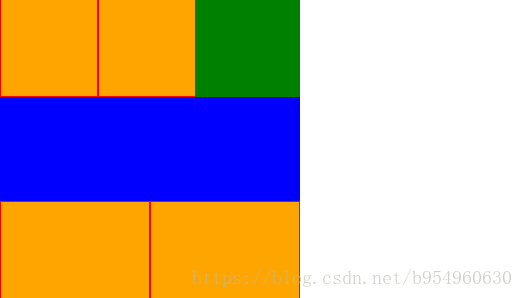
.column{

display: flex; /\*2\*/

flex-basis: 100%; /\*5\*/

background:green;

}

（5）  
  
第二个column里的两个item都设为flex-grow: 1;所以平摊(4)里的剩余空间，所以两个item长度相等。

.box{

margin-top: 30px;

display: flex; /\*1\*/

flex-wrap:wrap; /\*3\*/

align-content: space-between; /\*4\*/

background: blue;

width: 300px;

height: 300px;

}

.column{

display: flex; /\*2\*/

flex-basis: 100%; /\*5\*/

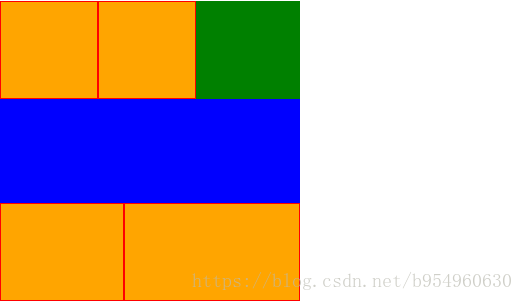
background:green;

}

.column:nth-child(2) .item{

flex-grow: 1; /\*6\*/

}

（6）  
  
第二个column里的第二个item的with是，第一个item的width的3倍

.box{

margin-top: 30px;

display: flex; /\*1\*/

flex-wrap:wrap; /\*3\*/

align-content: space-between; /\*4\*/

background: blue;

width: 300px;

height: 300px;

}

.column{

display: flex; /\*2\*/

flex-basis: 100%; /\*5\*/

background:green;

}

.column:nth-child(2) .item{

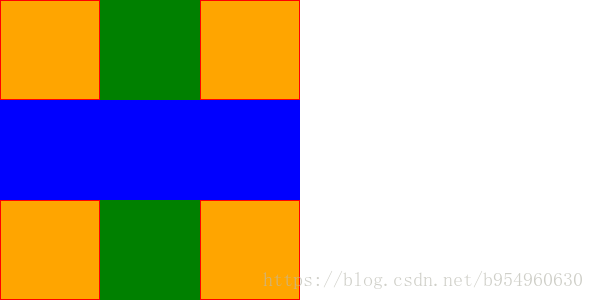
flex-grow: 1; /\*6\*/

}

.column:nth-child(2) .item:nth-child(2){

flex-grow: 3; /\*7\*/

}

（7） 这里接着图(4)扩展  


.box{

margin-top: 30px;

display: flex; /\*1\*/

flex-wrap:wrap; /\*3\*/

align-content: space-between; /\*4\*/

background: blue;

width: 300px;

height: 300px;

}

.column{

display: flex; /\*2\*/

flex-basis: 100%; /\*5\*/

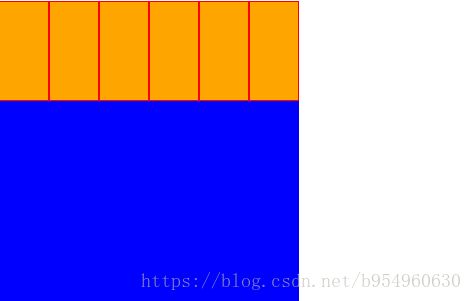
justify-content: space-between; /\*6\*/

background:green;

}

**五、六项目**

**1、情景一**

（0）初始化  


<style type="text/css">

.box{

margin-top: 30px;

display: flex; /\*1\*/

background: blue;

width: 300px;

height: 300px;

}

.item{

background: orange;

width: 98px;

height: 98px;

border: 1px solid red;

}

</style>

<div class="box">

<span class="item"></span>

<span class="item"></span>

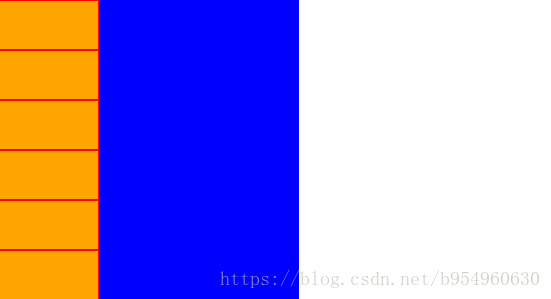
<span class="item"></span>

<span class="item"></span>

<span class="item"></span>

<span class="item"></span>

</div>

（1）  


.box{

margin-top: 30px;

display: flex; /\*1\*/

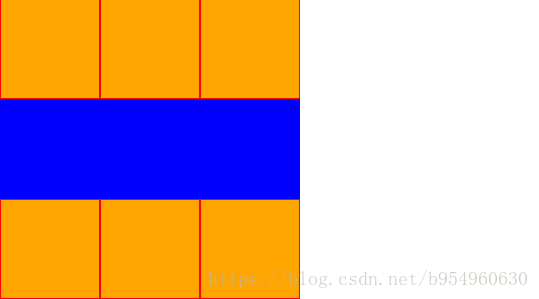
flex-direction: column; /\*2\*/

background: blue;

width: 300px;

height: 300px;

}

（2）  


.box{

margin-top: 30px;

display: flex; /\*1\*/

flex-wrap: wrap; /\*2\*/

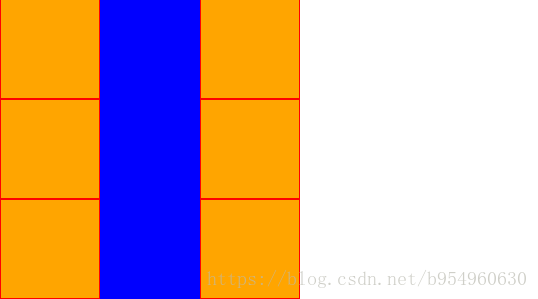
align-content: space-between; /\*3\*/

background: blue;

width: 300px;

height: 300px;

}

（3）  


.box{

margin-top: 30px;

display: flex; /\*1\*/

flex-wrap: wrap; /\*2\*/

align-content: space-between; /\*3\*/

flex-direction: column; /\*4\*/

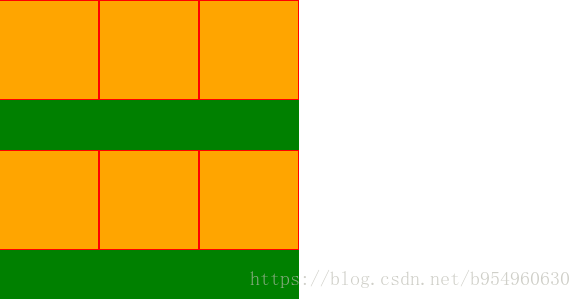
background: blue;

width: 300px;

height: 300px;

}

**2、情景二**

（0）初始化  


<style type="text/css">

.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*3\*/

background: blue;

width: 300px;

height: 300px;

}

.row{

display: flex; /\*2\*/

background: green;

}

.item{

background: orange;

width: 98px;

height: 98px;

border: 1px solid red;

}

</style>

<div class="box">

<div class="row">

<span class="item"></span>

<span class="item"></span>

<span class="item"></span>

</div>

<div class="row">

<span class="item"></span>

</div>

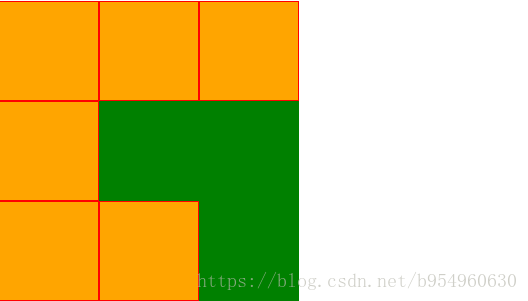
<div class="row">

<span class="item"></span>

<span class="item"></span>

</div>

</div>

（1）  


.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*3\*/

background: blue;

width: 300px;

height: 300px;

}

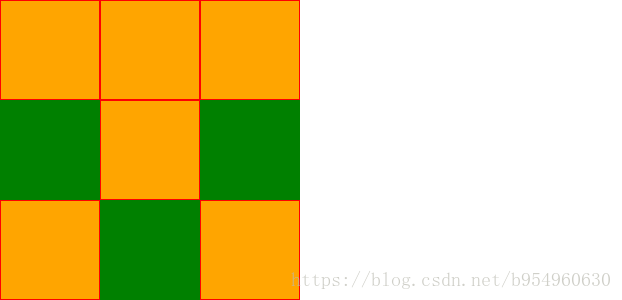
.row{

display: flex; /\*2\*/

flex-basis: 100%; /\*4\*/

background: green;

}

（2）  


.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*3\*/

background: blue;

width: 300px;

height: 300px;

}

.row{

display: flex; /\*2\*/

flex-basis: 100%; /\*4\*/

background: green;

}

.row:nth-child(2){

justify-content: center; /\*5\*/

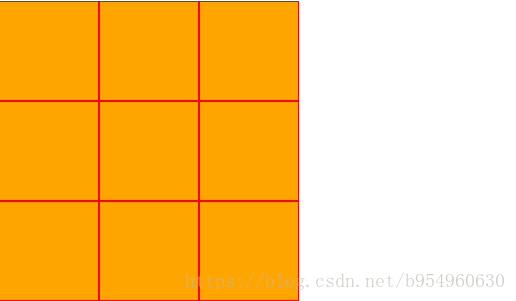
}

.row:nth-child(3){

justify-content: space-between; /\*6\*/

}

**六、九项目**



<style type="text/css">

.box{

display: flex; /\*1\*/

flex-wrap: wrap; /\*2\*/

background: blue;

width: 300px;

height: 300px;

}

.item{

background: orange;

width: 98px;

height: 98px;

border: 1px solid red;

}

</style>

<div class="box">

<span class="item"></span>

<span class="item"></span>

<span class="item"></span>

<span class="item"></span>

<span class="item"></span>

<span class="item"></span>

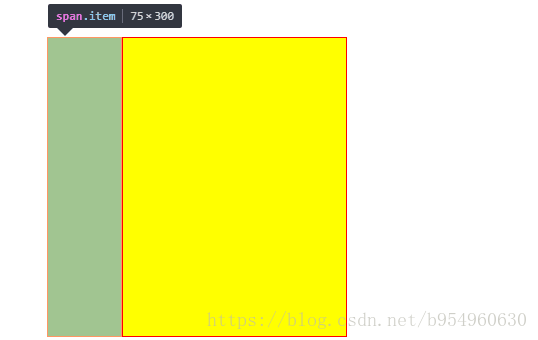
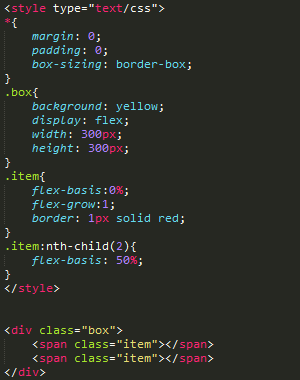
<span class="item"></span>

<span class="item"></span>

<span class="item"></span>

</div>

**七、理解flex-basis与flex-grow**

**问题:** 为什么靠左边.item的width值为75px呢？怎么计算得出的？  
  


**答：**  
div里有两个span，根据css，第一个占比0%（来自于类选择器的flex-basis属性），第二个占50%（这是由于css子选择器nth-child覆盖率了类选择器的flex-basis属性），到目前为止左边span占0px，右边占300/2=150px。但两个span同时拥有flex-grow：1 的属性，此属性在未手动设置的情况下默认为0，是指即使所有的item没有占满整个容器也仍然会保持原来大小，而当所有item此属性设为1时会等比例增长来填充容器内剩余空间。而此时容器内剩余300 -150 =150px 的空间，当两个item等比例增长填充容器剩余空间后，第一个会占150/2=75px，第二个会占150+150/2=225px。 从而解释了为什左边item占75px。‘

**问题2:**  
接着上面，如果去掉.item的flex-grow:1; 布局会变成什么样？即：  
在这里插入图片描述

**答：**  
结果很明显，就是按上面逻辑分析：  
左边的.item宽度为0，右边的.item宽度为150px  
