

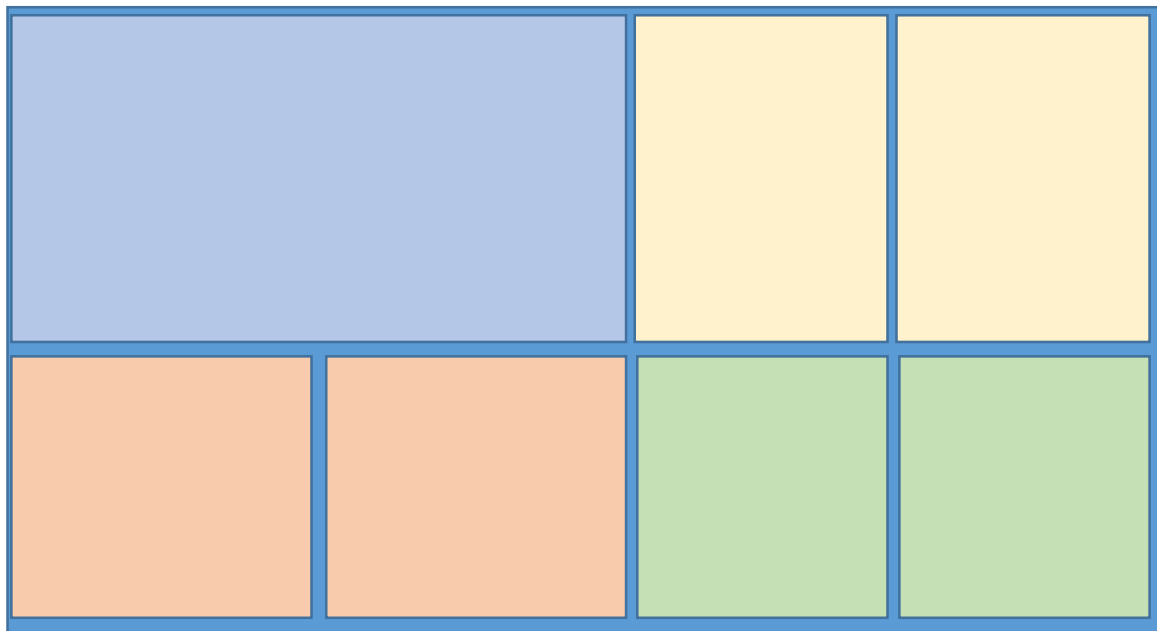


flex弹性盒子布局和grid网格布局

概述

应用浮动定位和层定位布局时有很多限制，例如很难实现盒子垂直居中、多个盒子的间隙的平均分配等。

利用弹性盒子和网格布局可以很好解决上述问题。并很容易实现类似下边所示布局。弹性盒子是一维布局模型，网格布局是二维布局模型。二者可结合使用。



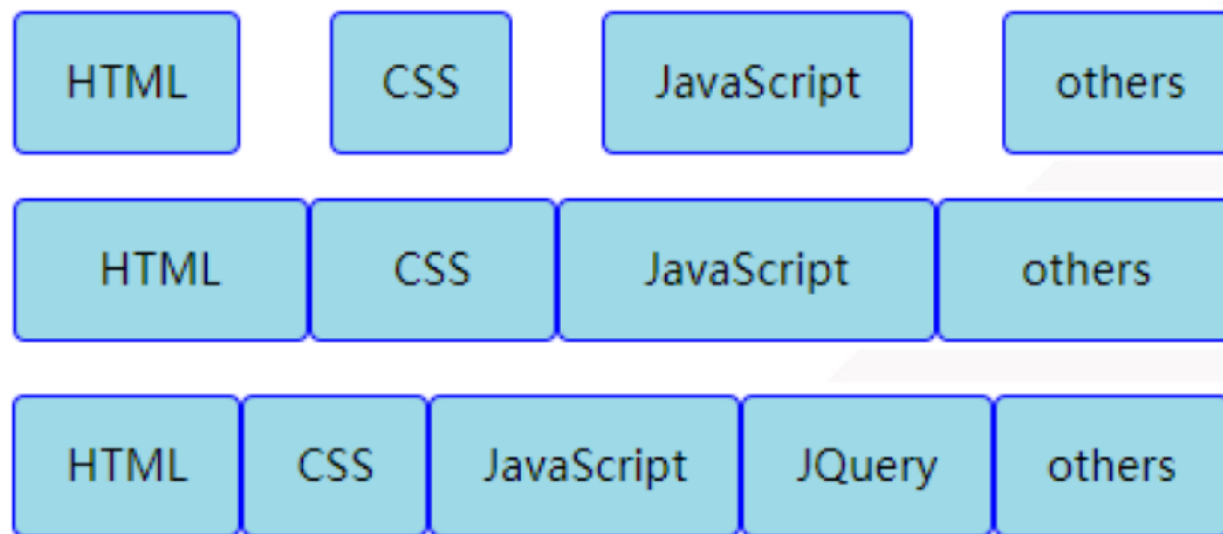
概述

元素可以

——拉伸以填充额外的空间

——收缩以适应更小的空间

flex弹性盒子布局



概述

flex弹性盒子布局可以解决如下问题

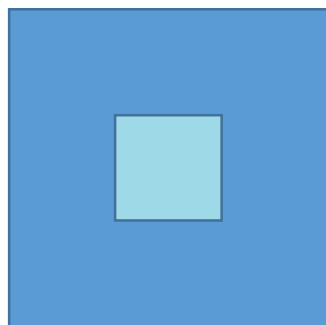
Lorem ipsum dolor		
Lorem ipsum dolor sit amet. Lorem ipsum, dolor sit amet consectetur adipisicing elit. Hic culpa iure aut?	Lorem ipsum dolor sit amet. Lorem ipsum, dolor sit amet consectetur adipisicing elit. Hic culpa iure aut? Lorem ipsum, dolor sit amet consectetur adipisicing elit. Hic culpa iure aut?	Lorem ipsum Lorem ipsum, dolor sit amet consectetur adipisicing elit. Hic culpa iure aut? Lorem ipsum, dolor sit amet consectetur adipisicing elit. Hic culpa iure aut? Lorem ipsum, dolor sit amet consectetur adipisicing elit. Hic culpa iure aut?



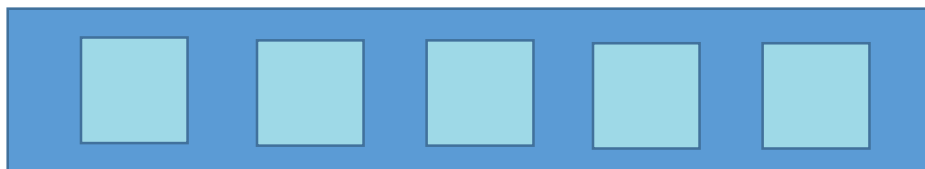
Lorem ipsum dolor		
Lorem ipsum dolor sit amet. Lorem ipsum, dolor sit amet consectetur adipisicing elit. Hic culpa iure aut?	Lorem ipsum dolor sit amet. Lorem ipsum, dolor sit amet consectetur adipisicing elit. Hic culpa iure aut? Lorem ipsum, dolor sit amet consectetur adipisicing elit. Hic culpa iure aut?	Lorem ipsum Lorem ipsum, dolor sit amet consectetur adipisicing elit. Hic culpa iure aut? Lorem ipsum, dolor sit amet consectetur adipisicing elit. Hic culpa iure aut? Lorem ipsum, dolor sit amet consectetur adipisicing elit. Hic culpa iure aut?

概述

flex弹性盒子布局可以解决如下问题



垂直居中



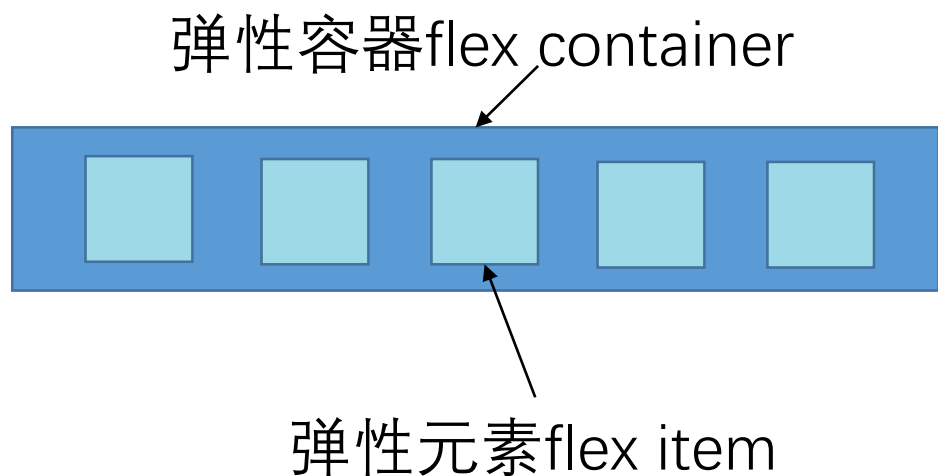
间隙的平均分配



自动占据剩余空间

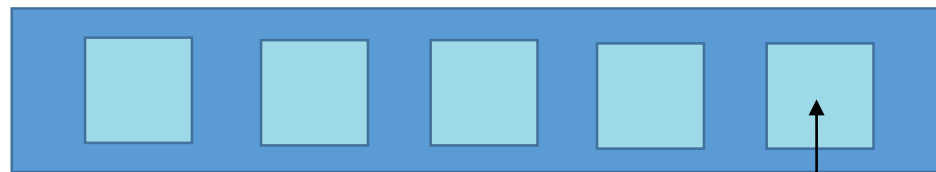
Flex弹性盒子布局

采用 Flex 布局的元素，称为 Flex 容器（flex container），简称“容器”。它的所有子元素自动成为容器成员，称为 Flex 元素（flex item），简称“元素”。



flex布局原理：就是通过给父盒子添加flex属性，来控制子盒子的位置和排列方式
弹性元素也可以作为新的弹性容器，嵌套一些新的其他的弹性元素。

Flex弹性盒子布局



display:flex

flex-direction

flex-wrap

flex-flow

justify-content

align-items

align-content

flex-grow

flex-shrink

flex-basis

order

align-self

flex

flex-direction: 设置按行还是按列进行布局

flex-wrap: 设置子元素是否换行

justify-content: 设置弹性元素在主轴上的对齐方式

align-items: 设置弹性元素在侧轴上的对齐方式
(单行)

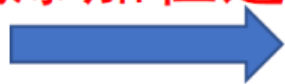
align-content: 设置弹性元素在侧轴上的对齐方式
(多行)

● 弹性容器样式

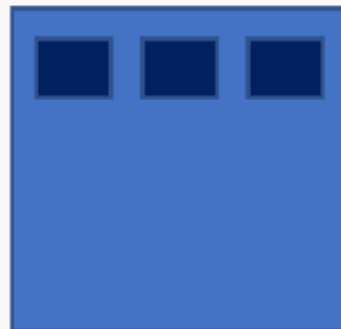
1、display属性 定义弹性容器

```
<div class="flex-container">  
  <div class="flex-item">1</div>  
  <div class="flex-item">2</div>  
  <div class="flex-item">3</div>  
</div>
```

样式添加在这里



```
.flex-container{  
  display: flex;  
}
```



弹性元素默认是按行排列

设为Flex布局以后，子元素的float、clear和vertical-align属性将失效。

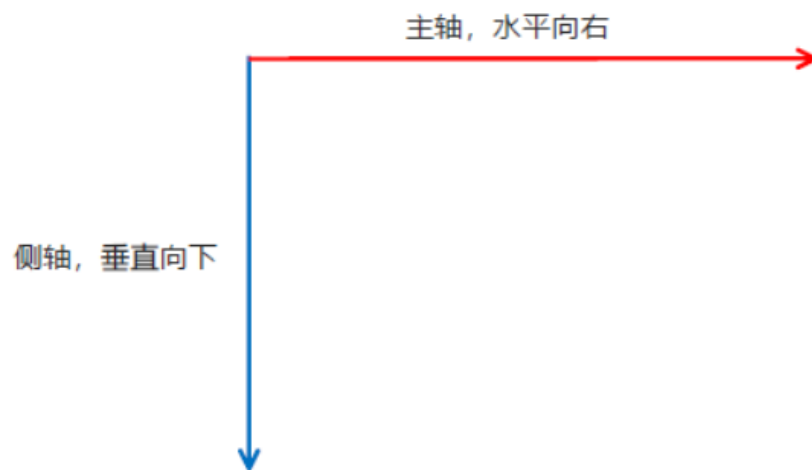
● 弹性容器样式

2、flex-direction设置主轴的方向

主轴与侧轴

在 flex 布局中，是分为主轴和侧轴两个方向，同样的叫法有：行和列、x 轴和y轴。

- 默认主轴方向就是 x 轴方向，水平向右
- 默认侧轴方向就是 y 轴方向，垂直向下


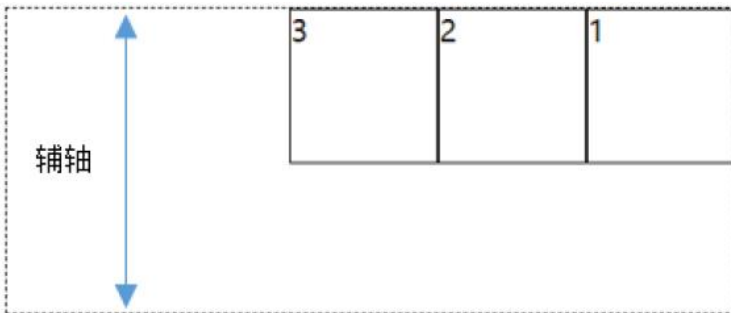


弹性容器样式

2、flex-direction设置主轴的方向

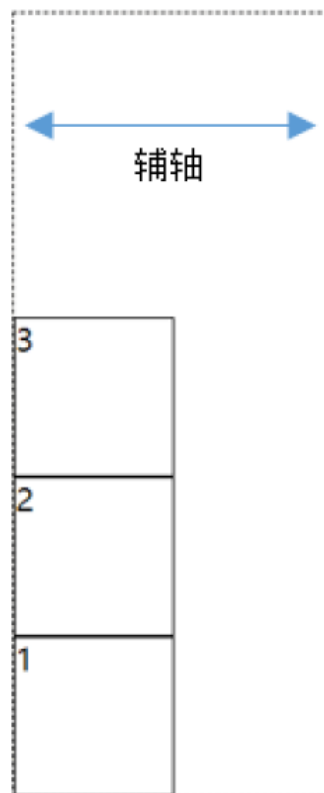
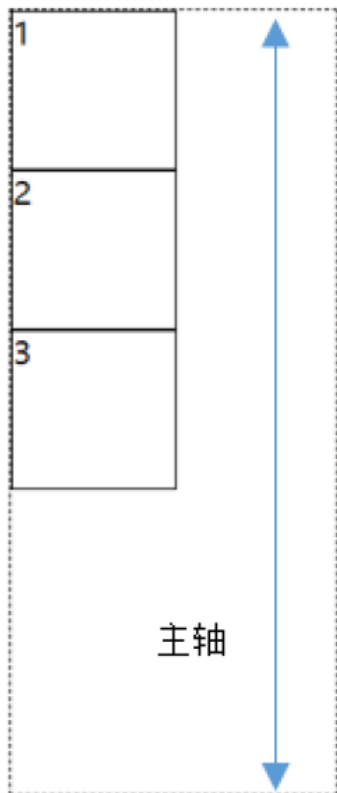
flex-direction 属性决定主轴的方向（即弹性元素的排列方向）

注意：主轴和侧轴是会变化的，就看 flex-direction 设置谁为主轴，剩下的就是侧轴，而弹性元素是跟着主轴来排列的。

属性值	说明		
row	默认值从左到右	<p>flex-direction: row</p>	<p>flex-direction: row-reverse</p>
row-reverse	从右到左		
column	从上到下		
column-reverse	从下到上		

● 弹性容器样式

2、flex-direction设置主轴的方向



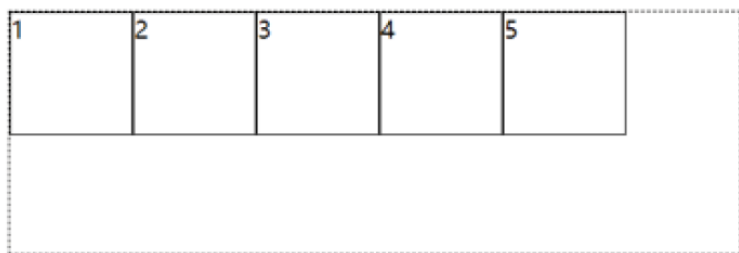
flex-direction: **column** flex-direction: **column-reverse**

弹性容器样式

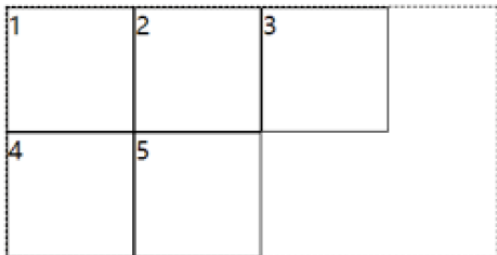
3、**flex-wrap**指定 flex 元素单行显示还是多行显示。如果允许换行，这个属性允许你控制行的堆叠方向。

默认情况下，弹性元素都排在一条线（又称“轴线”）上。

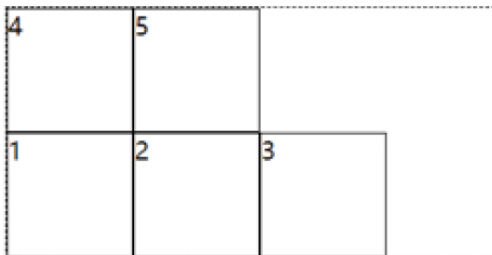
flex-wrap属性，默认是不换行的，也就是取值为**nowrap**。



flex-wrap: nowrap



flex-wrap: wrap



flex-wrap: wrap-reverse



弹性容器样式

3、flex-wrap 设置子元素是否换行

[illegible]

flex-wrap: nowrap

Lorem ipsum dolor		
Lorem ipsum dolor sit amet. Lorem ipsum, dolor sit amet consectetur adipiscing elit. Hic culpa iure aut?	Lorem ipsum dolor sit amet. Lorem ipsum, dolor sit amet consectetur adipiscing elit. Hic culpa iure aut?	Lorem ipsum Lorem ipsum, dolor sit amet consectetur adipiscing elit. Hic culpa iure aut?
Lorem ipsum dolor sit amet. Lorem ipsum, dolor sit amet consectetur adipiscing elit. Hic culpa iure aut?	Lorem ipsum dolor sit amet. Lorem ipsum, dolor sit amet consectetur adipiscing elit. Hic culpa iure aut?	Lorem ipsum Lorem ipsum, dolor sit amet consectetur adipiscing elit. Hic culpa iure aut?
Lorem ipsum dolor sit amet. Lorem ipsum, dolor sit amet consectetur adipiscing elit. Hic culpa iure aut?	Lorem ipsum dolor sit amet. Lorem ipsum, dolor sit amet consectetur adipiscing elit. Hic culpa iure aut?	Lorem ipsum Lorem ipsum, dolor sit amet consectetur adipiscing elit. Hic culpa iure aut?

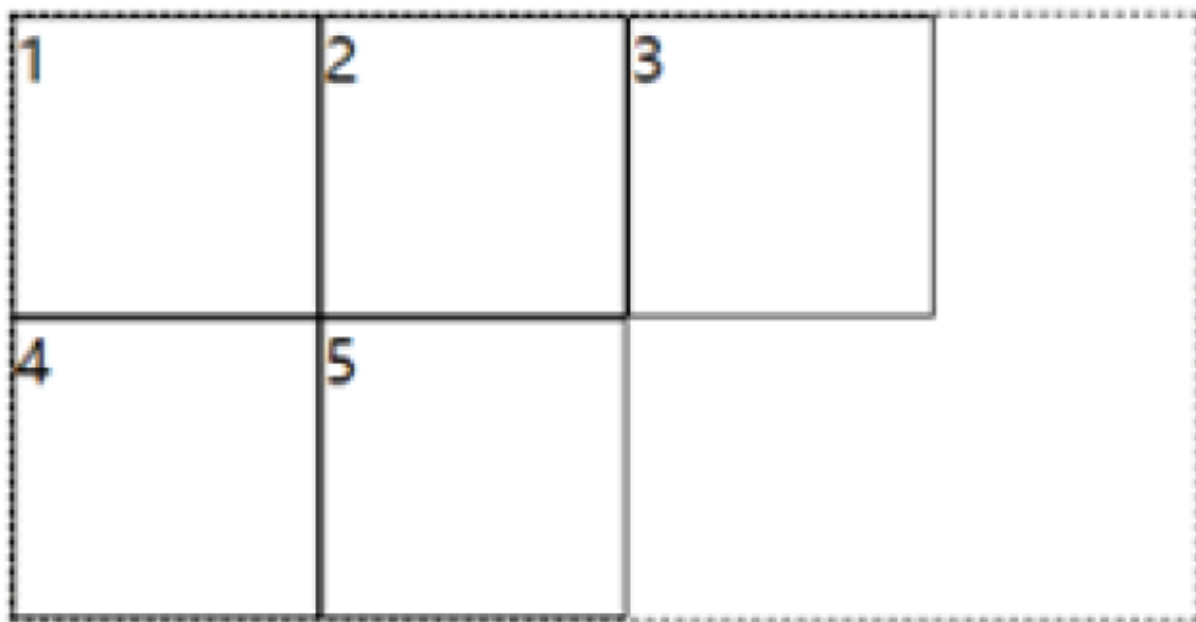
flex-wrap: wrap

● 弹性容器样式

4、flex-flow属性

flex-flow 属性是 flex-direction 和 flex-wrap 属性的复合简写属性

语法 flex-flow: row wrap;



● 弹性容器样式

5、`justify-content` 用于设置弹性元素在主轴上的对齐方式

注意：使用这个属性之前一定要确定好主轴是哪个

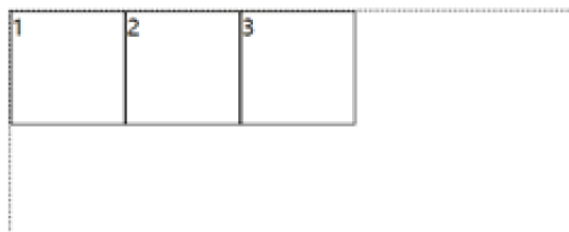
属性值	说明
<code>flex-start</code>	默认值 从头部开始 如果主轴是x轴，则从左到右
<code>flex-end</code>	从尾部开始排列
<code>center</code>	在主轴居中对齐（如果主轴是x轴则 水平居中）
<code>space-around</code>	平分剩余空间
<code>space-between</code>	先两边贴边 再平分剩余空间（重要）

弹性容器样式

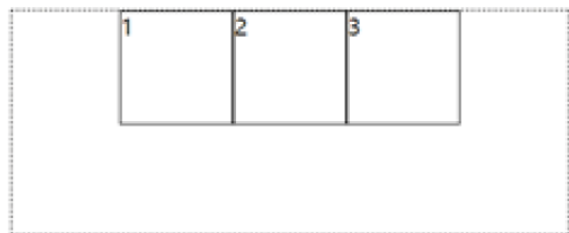
5、`justify-content` 用于设置弹性元素在主轴上的对齐方式

注意：使用这个属性之前一定要确定好主轴是哪个

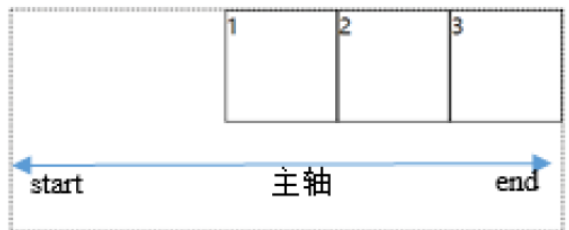
`justify-content: flex-start`



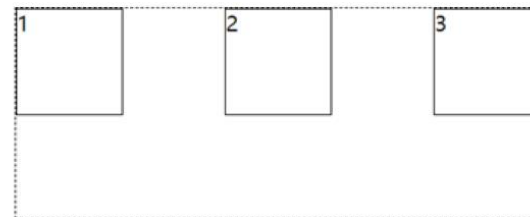
`justify-content: center`



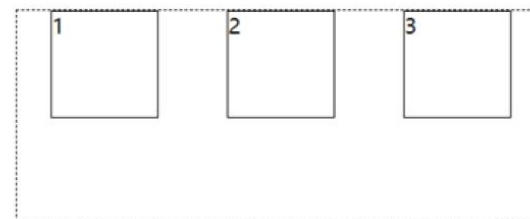
`justify-content: flex-end`



`justify-content: space-between`



`justify-content: space-around`



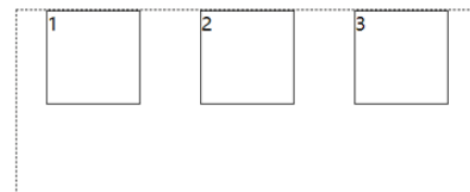
`space-between`表示子元素两端对齐，中间平均分配
`space-around`表示在子元素的两侧平均分配

弹性容器样式

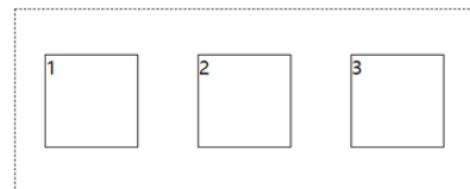
6、align-items 设置弹性子元素在辅轴上的对齐方式

属性值	说明
flex-start	从上到下
flex-end	从下到上
center	挤在一起居中（垂直居中）
stretch	拉伸（默认值）

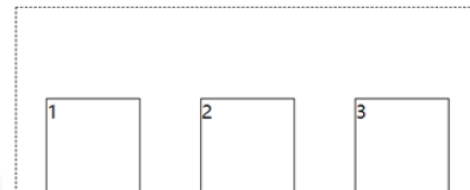
align-items: flex-start



align-items: center



align-items: flex-end

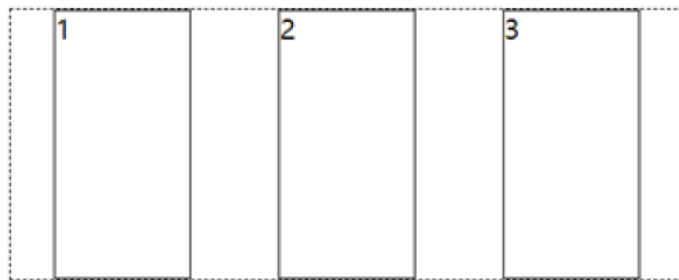


弹性容器样式

6、align-items 设置弹性元素在辅轴上的对齐方式

align-items: stretch

注意：去掉元素高度



取默认值stretch，会使所有弹性元素沿着辅轴的方向拉伸以填充父容器。

例如：生成一个弹性盒子，含三个弹性元素，按行排列，水平居中，垂直方向上拉伸占满辅轴长度。这样即使子元素内容多少不一样，也可以保证子元素高度一致。

```
<div class="flex-container">
  <div class="flex-item">1</div>
  <div class="flex-item">2</div>
  <div class="flex-item">3</div>
</div>
```

● 弹性容器样式

6、align-items 设置弹性元素在辅轴上的对齐方式

```
.flex-container{  
    display: flex;  
    flex-direction: row;  
    justify-content: space-around;  
    align-items: stretch;  
  
    width: 500px;  
    height: 200px;  
    border: 1px dashed;  
  
}
```

```
.flex-item{  
    width: 100px;  
    border: 1px solid;  
    font-size: 20px;  
  
}
```

● 弹性容器样式

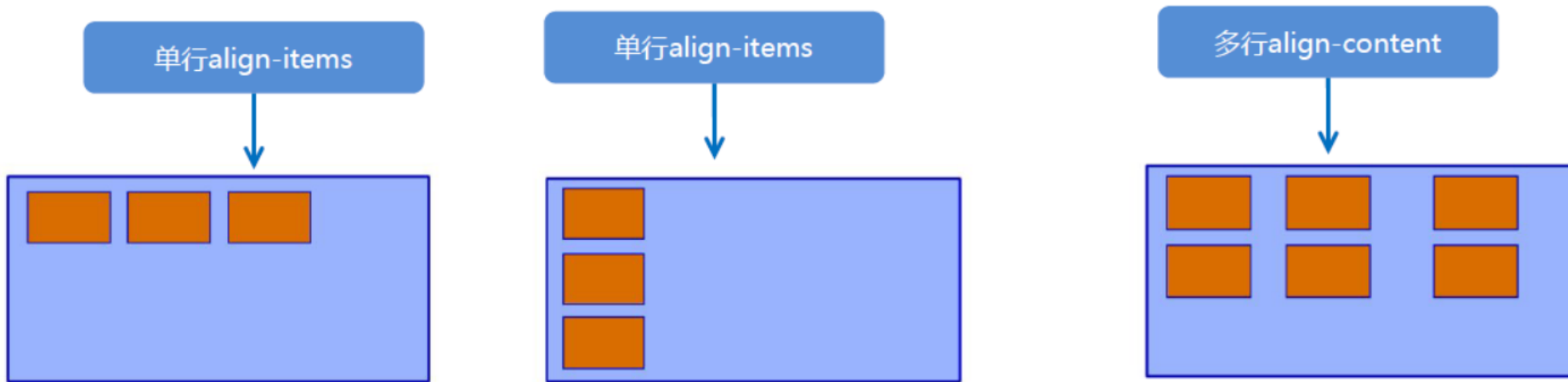
7、`align-content` 设置多行元素在容器中的整体对齐方式
只有一行或者一列，这个属性不起作用。

属性值	说明
<code>flex-start</code>	默认值在侧轴的头部开始排列
<code>flex-end</code>	在侧轴的尾部开始排列
<code>center</code>	在侧轴中间显示
<code>space-around</code>	子项在侧轴平分剩余空间
<code>space-between</code>	子项在侧轴先分布在两头，再平分剩余空间
<code>stretch</code>	设置子项元素高度平分父元素高度

● 弹性容器样式

align-content 和 align-items 区别

- align-items 适用于单行情况下，只有上对齐、下对齐、居中和拉伸
- align-content 适应于换行（多行）的情况下（单行情况下无效），可以设置 上对齐、下对齐、居中、拉伸以及平均分配剩余空间等属性值。
- 单行找 align-items，多行找 align-content



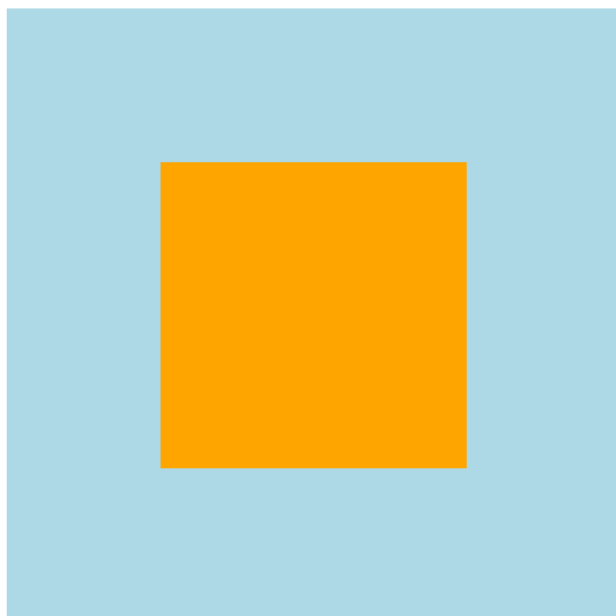
● 弹性容器样式

```
<div class="container">
```

```
  <div class="item">
```

```
  </div>
```

```
</div>
```



```
.container {  
  background-color: lightblue;  
  width: 300px;  
  height: 300px;  
}
```

```
.item {  
  background-color: orange;  
  width: 150px;  
  height: 150px;  
}
```

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

● 回顾

元素的垂直居中对齐

1、内联块元素：

`text-align: center;`

`line-height`：和父元素高度相同的高度值；

2、块级元素的垂直水平居中：

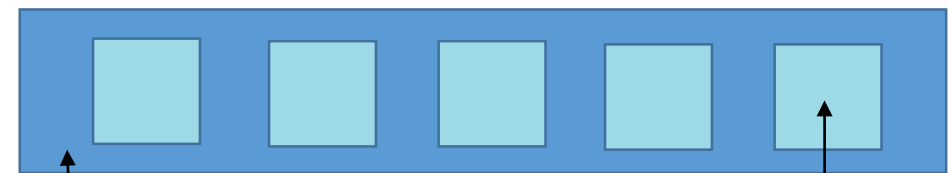
`margin: auto`（在父元素中水平居中）；

定位：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;`

弹性元素样式



弹性容器flex container

弹性元素flex item

flex-grow
flex-shrink
flex-basis

flex

align-self

order

flex-grow 用于设置弹性元素被拉大的比例

flex-shrink 用于设置弹性元素被压缩的比例

flex-basis 用于设置元素在主轴上的初始尺寸，即元素放入容器前的默认大小，可能被压缩，放入容器后，根据元素是否需要拉伸或压缩计算实际长度，其优先级高于width属性。

align-self 控制子项自己在侧轴的排列方式

order 定义子项的排列顺序（前后顺序）

● 弹性元素样式

```
<div class="flex-container">  
  <div class="flex-item">1</div>  
  <div class="flex-item">2</div>  
  <div class="flex-item">3</div>  
</div>
```

.flex-item {



样式添加在这里

}

弹性元素样式

1. flex-grow 属性

设置当父元素的宽度大于所有子元素的宽度的和时（即父元素会有剩余空间），子元素如何分配父元素的剩余空间。flex-grow 的默认值为 0，意思是该元素不索取父元素的剩余空间，如果值大于0，表示索取。值越大，索取的越厉害。

```
.flex-container{
  display: flex;
  flex-direction: row;
  align-items: stretch;
}
.flex-item{
  border: 1px solid;
}
```

未规定width,height

1

2

3

`div:nth-child(1){`
`flex-grow: 1;`
`}`

`div:nth-child(2){`
`flex-grow: 1;`
`}`

`div:nth-child(3){`
`flex-grow: 2;`
`}`

1: 1: 2分配剩余空间

假如设置父元素 500px，子元素 1为 100px，子元素 2为 100px，子元素3为 200px，则剩余空间为 100px。

1:1:2的比例下

子元素1的实际宽度是 $100 + 100 \times 1/4$
子元素2的实际宽度是 $100 + 100 \times 1/4$
子元素3的实际宽度是 $200 + 100 \times 1/2$

● 弹性元素样式

2. flex-shrink属性

- 用来设置子元素的缩小比例，当父元素的宽度小于所有子元素的宽度的和时（即子元素会超出父元素），子元素如何缩小自己的宽度。
- 默认为1，当父元素的宽度小于所有子元素的宽度的和时，子元素的宽度会减小。值越大，减小的越厉害。如果值为0，表示不减小。

● 弹性元素样式

2. flex-shrink属性

假如设置父元素 400px，子元素a为 200px，子元素b为 300px。则超出空间为 100px。

如果子元素a，b都减小宽度，a设置 flex-shrink 为 3，b设置 flex-shrink 为 2。则最终 a 的大小为 自身宽度 (200px) - a减小的宽度 ($100\text{px} * (200\text{px} * 3 / (200\text{px} * 3 + 300\text{px} * 2))$) = 150px,

最终 b 的大小为 自身宽度 (300px) - B减小的宽度 ($100\text{px} * (300\text{px} * 2 / (200\text{px} * 3 + 300\text{px} * 2))$) = 250px

● 弹性元素样式

3. flex-basis 属性

用来设置元素的宽度，通常情况下大家使用 width 设置宽度。但是如果元素上同时设置了 width 和 flex-basis ，那么 width 的值就会被 flex-basis 覆盖掉。也就是说flex-basis 优先级高于width 属性。

● 弹性元素样式

4、flex属性

flex: flex-**g**row flex-**s**hrink flex-**b**asis;

- 默认值分别为0 1 auto; (即不放大, 会缩小)

flex:1 1 auto(即放大且缩小)

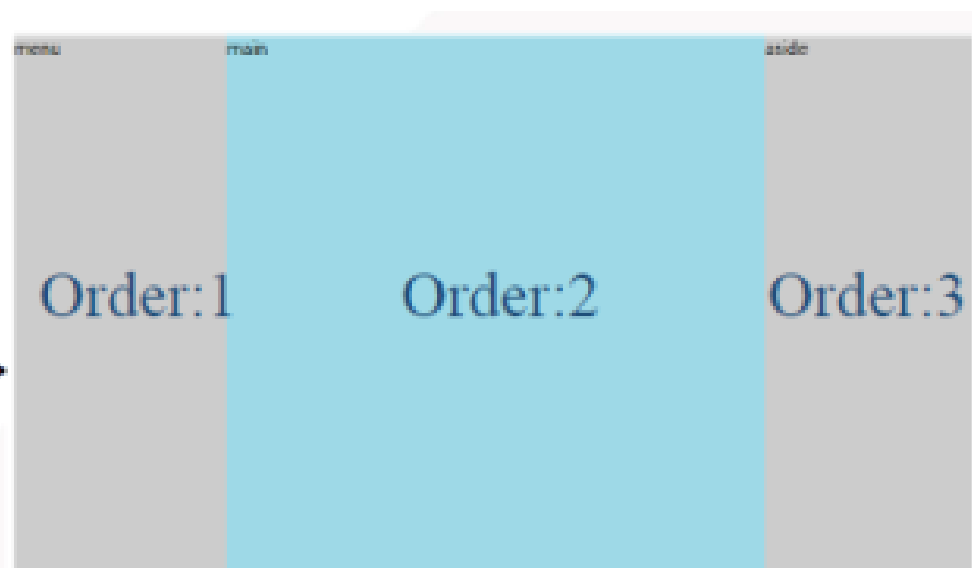
- flex-grow为1表示当弹性容器较大的时候 子元素等比例拉伸;
- flex-shrink为1表示当弹性容器较小的时候 子元素等比例压缩;
- 所以初始宽度不去进行设定, flex-basis为auto自动进行计算;

● 弹性元素样式

5. order属性

能够改变子元素在容器内的位置。不会影响到源顺序（即 DOM 树里元素的顺序），所有子元素默认order值为0，数值越小排名越靠前。相同 order 值的子元素按源顺序显示。也可以给 order 设置负值使它们比值为 0 的元素排得更前面。

```
<div class="flex-container">  
  <div class="flex-item">main</div>  
  <div class="flex-item">menu</div>  
  <div class="flex-item">aside</div>  
</div>
```



● 弹性元素样式

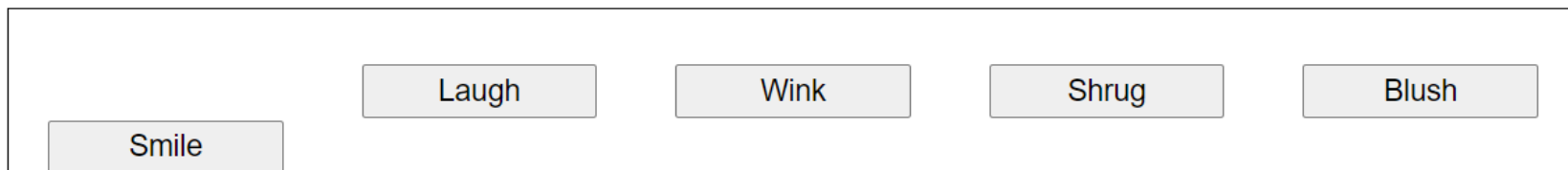
6、align-self属性 控制子项自己在侧轴上的对齐方式

align-self 属性允许单个项目有与其他项目不一样的对齐方式，可覆盖 align-items 属性，align-items 设置的是全部属性，align-self可以设置在单个子元素上。

align-self默认值为 auto，表示继承父元素的 align-items 属性，如果没有父元素，则等同于 stretch。

align-self: auto | flex-start | flex-end | center | baseline | stretch

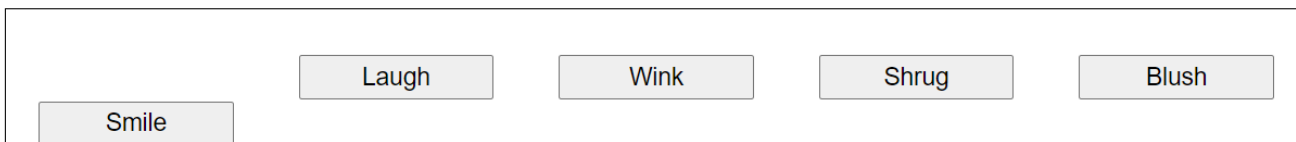
可覆盖 align-items 属性



弹性元素样式

6、align-self属性

控制子项自己在侧轴上的对齐方式



```
div {  
  display: flex;  
  align-items: center;  
  justify-content: space-around;  
}  
button:first-child {  
  align-self: flex-end;  
}
```

```
<div>  
  <button>Smile</button>  
  <button>Laugh</button>  
  <button>Wink</button>  
  <button>Shrug</button>  
  <button>Blush</button>  
</div>
```

Flex嵌套

Flex也能创建一些颇为复杂的布局。设置一个元素为 弹性元素，那么他同样也能成为一个 flex 容器，它的孩子（直接子节点）也表现为弹性元素。

Complex flexbox example

First article

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

Second article

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

Smile

Laugh

Wink

Shrug

Blush

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

Cray food truck brunch, XOXO +1 keffiyeh pickled chambray waistcoat ennui. Organic small batch paleo 8-bit. Intelligentsia umami wayfarers pickled, asymmetrical kombucha letterpress kitsch leggings cold-pressed squid chartreuse put a bird on it. Listicle pickled man bun cornhole heirloom art party.

Flex布局练习

应用弹性盒子布局，完成下边的布局效果;

定义折行，弹性元素在盒子宽度不够时自动折行，完成自适应布局。

Complex flexbox example

First article

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

Second article

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

Smile

Laugh

Wink

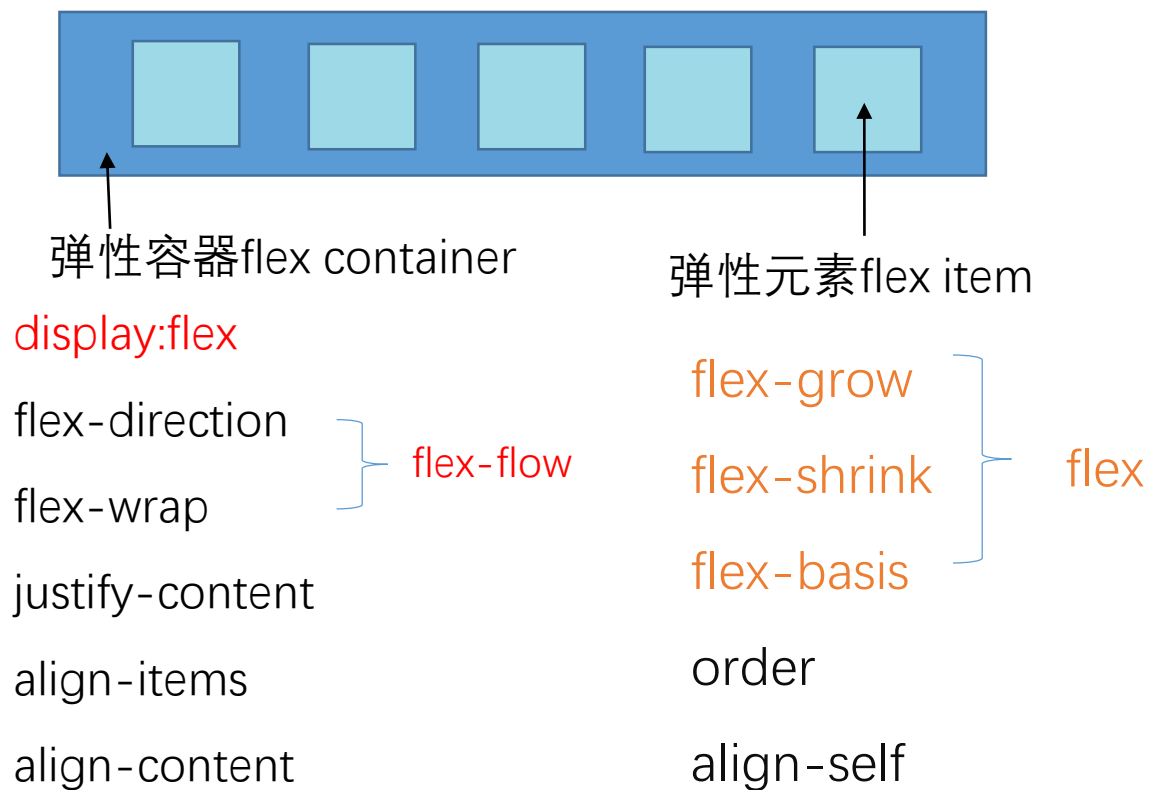
Shrug

Blush

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

Cray food truck brunch, XOXO +1 keffiyeh pickled chambray waistcoat ennui. Organic small batch paleo 8-bit. Intelligentsia umami wayfarers pickled, asymmetrical kombucha letterpress kitsch leggings cold-pressed squid chartreuse put a bird on it. Listicle pickled man bun cornhole heirloom art party.

flex布局总结



● 布局方式—常用三种

1、传统布局方式

利用position属性 + display属性 + float属性布局, 兼容性最好, 但是效率低, 较麻烦!

2、flex布局

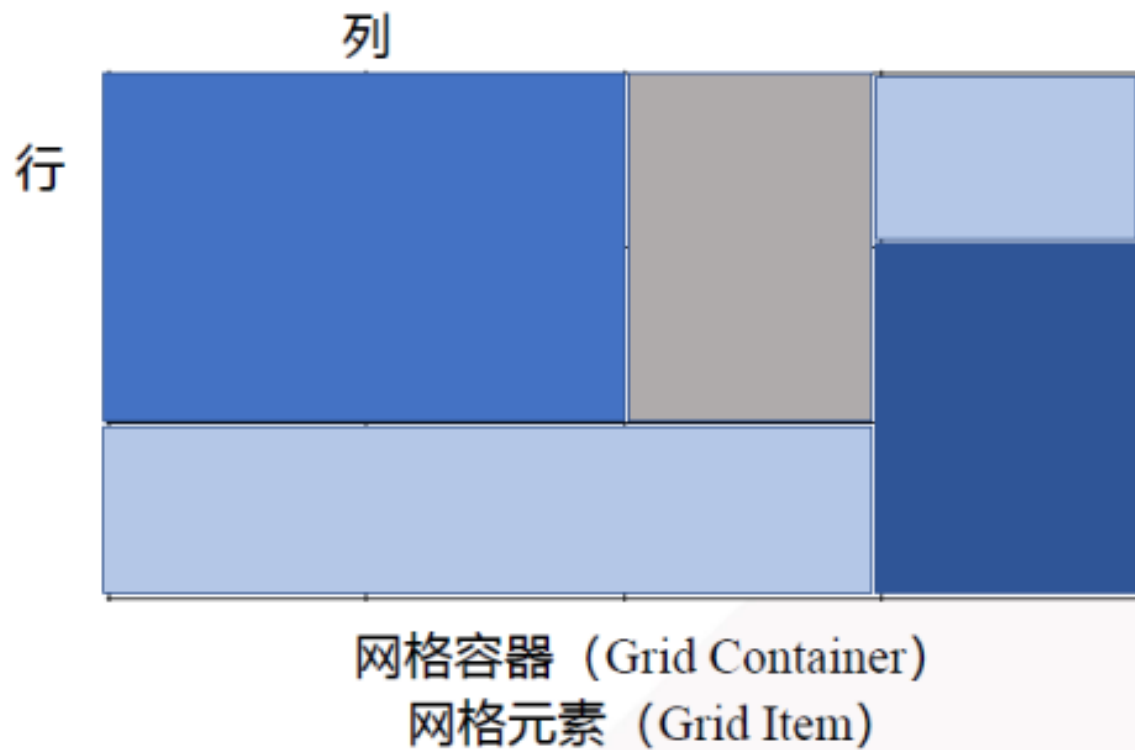
有自己的一套属性, 效率高, 兼容性强! 一维的布局模型, 往往用在页面的某一个局部位置。

3、grid布局

网格布局(grid)是最强大的布局方案, 可以用于页面中大的布局! 兼容性不如flex布局。

◆ 网格布局 (Grid Layout)

网格布局是一个用于 web 的二维布局系统，利用网格，可以把内容按照行与列的格式进行排版，能够非常轻松地实现一些复杂的布局。网格是由一系列水平及垂直的线构成的一种布局模式。



● 网格布局样式

网格容器 (grid container)

display:grid;

grid-template-columns

grid-template-rows

grid-template-areas

grid-gap

justify-items

➡ place-items

align-items

justify-content

➡ place-content

align-content

网格元素 (grid item)

grid-item-*

grid-row-*

grid-area

justify-self

align-self

➡ place-self







网格容器样式

1. display属性： 定义网格容器

```
.grid-container{  
    display: grid ;  
}
```

2. `grid-template-columns` 和 `grid-template-rows`划分网格的行和列，

取值： px 、 % 、 auto 、 fr 、 repeat() 函数

<code>grid-template-columns: 1fr 1fr;</code>	→		表示分为两列，每列占比是相等的
<code>grid-template-columns : 1fr 2fr;</code>	→		
<code>grid-template-columns : 100px 1fr 2fr;</code>	→		fr可以与一般的长度单位混合使用。
<code>grid-template-columns : repeat (3, 1fr);</code>	→		

● 网格容器样式

`grid-template-columns: repeat(3, 1fr);`

第一个传入 repeat 函数的值3，表明了后续列宽的配置要重复多少次，第二个值1fr表示需要重复的配置，这个配置可以具有多个长度设定。

例如repeat(2, 2fr 1fr)，相当于填入了 2fr 1fr 2fr 1fr。

● 网格容器样式

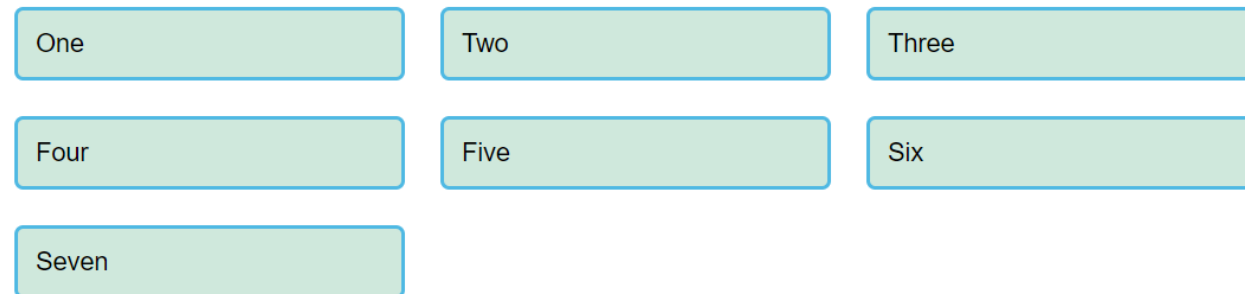
显式网格与隐式网格

- 显式网格是用 `grid-template-columns` 或 `grid-template-rows` 属性创建的。
- 而隐式网格则是当有内容被放到网格外时才会生成的。显式网格与隐式网格的关系与Flex布局的 主轴 和 侧轴的关系有些类似。
- 隐式网格中生成的行/列大小参数默认是 `auto`，大小会根据放入的内容自动调整。也可以使用`grid-auto-rows`和`grid-auto-columns`属性手动设定隐式网格的大小。

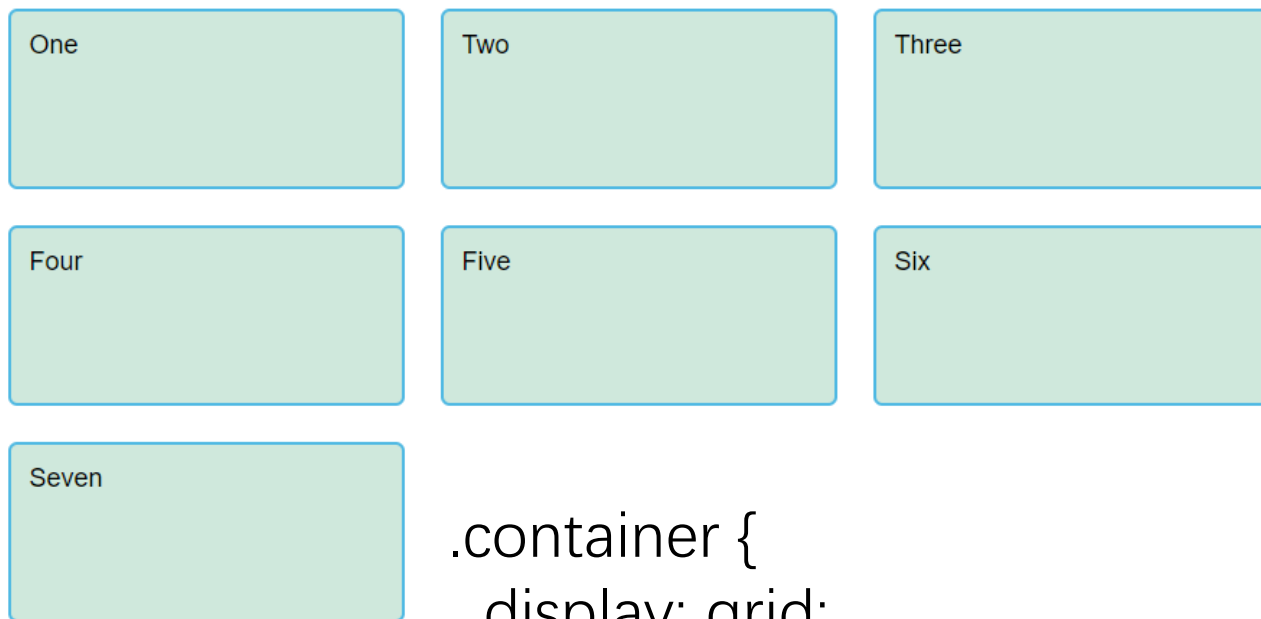


网格容器样式

Simple grid example



Simple grid example



```
.container {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  grid-auto-rows: 100px;  
  grid-gap: 20px;  
}
```

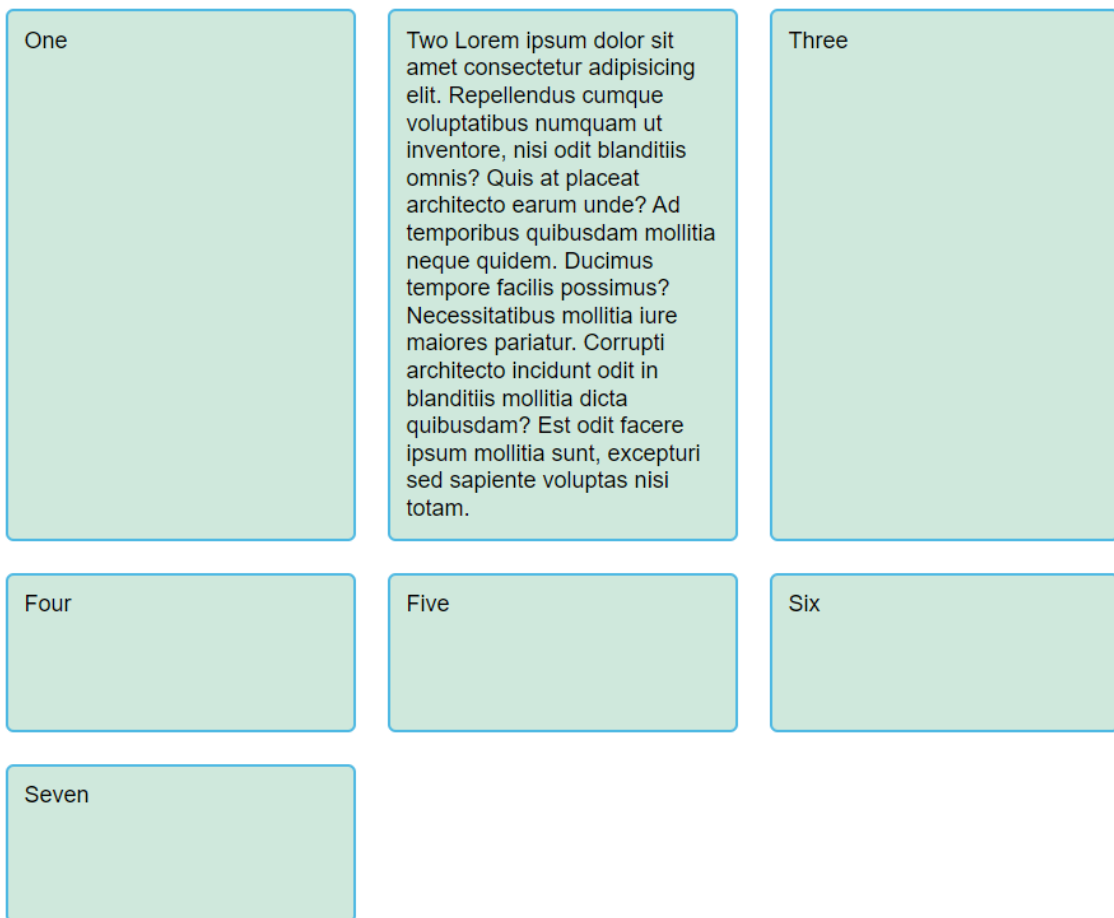
● 网格容器样式

minmax() 函数

- 100px高的网格有时可能会不够用，所以可以将其设定为至少 100px，并且能够跟随内容来自动拓展尺寸，从而保证能容纳所有内容。
- minmax()函数为一个行/列的尺寸设置了取值范围。比如设定为 minmax(100px, auto)，那么尺寸就至少为 100 像素，并且如果内容尺寸大于 100 像素则会根据内容自动调整。

🔴 网格容器样式

Simple grid example



```
.container {  
  display: grid;  
  grid-template-columns: repeat(3,1fr);  
  grid-auto-rows: minmax(100px,auto);  
  grid-gap: 20px;  
  gap: 20px;  
}
```

● 网格容器样式

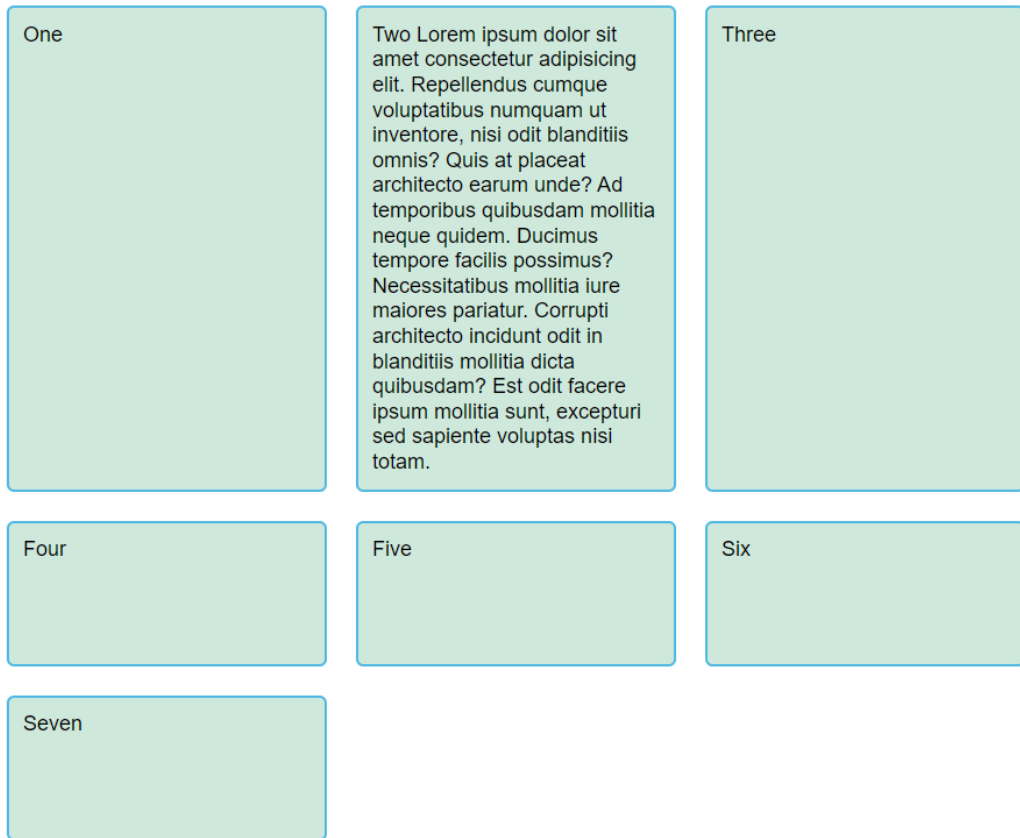
自动使用多列填充

- repeat 与 minmax 函数，组合起来，可以实现一个非常有用的功能。
- 某些情况下，我们需要让网格自动创建很多列来填满整个容器。
- 可以用到 repeat 函数中的一个关键字auto-fill来替代确定的重复次数。
而repeat函数的第二个参数，可以使用minmax函数来设定一个行/列的最小值，以及最大值 1fr。

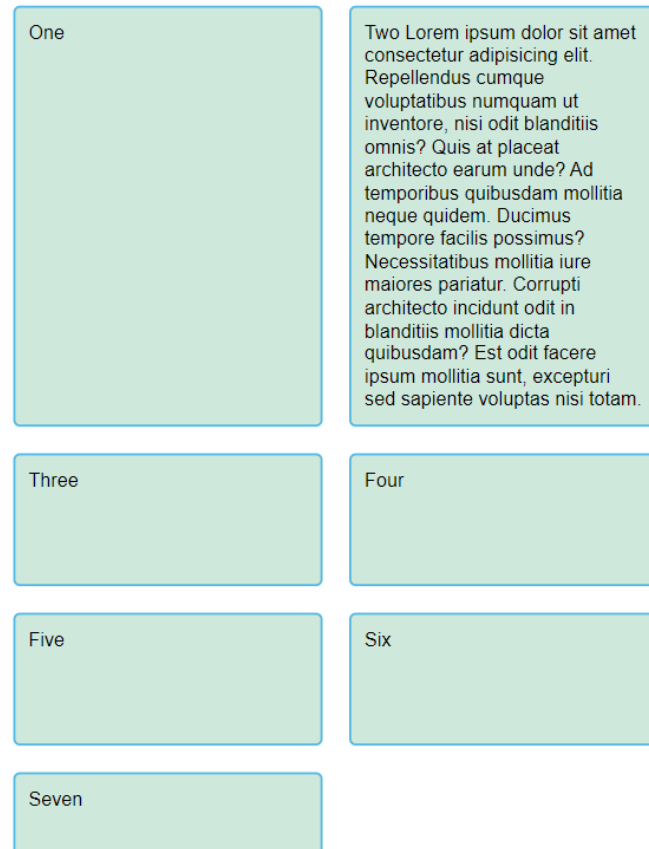


网格容器样式

Simple grid example



Simple grid example



形成了一个包含了许多至少 200 像素宽的列的网格，将容器填满。随着容器宽度的改变，网格会自动根据容器宽度进行调整，每一列的宽度总是大于 200 像素，并且容器总会被列填满。

● 网格容器样式

```
.container {  
  display: grid;  
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));  
  grid-auto-rows: minmax(100px, auto);  
  grid-gap: 20px;  
  gap: 20px;  
}
```

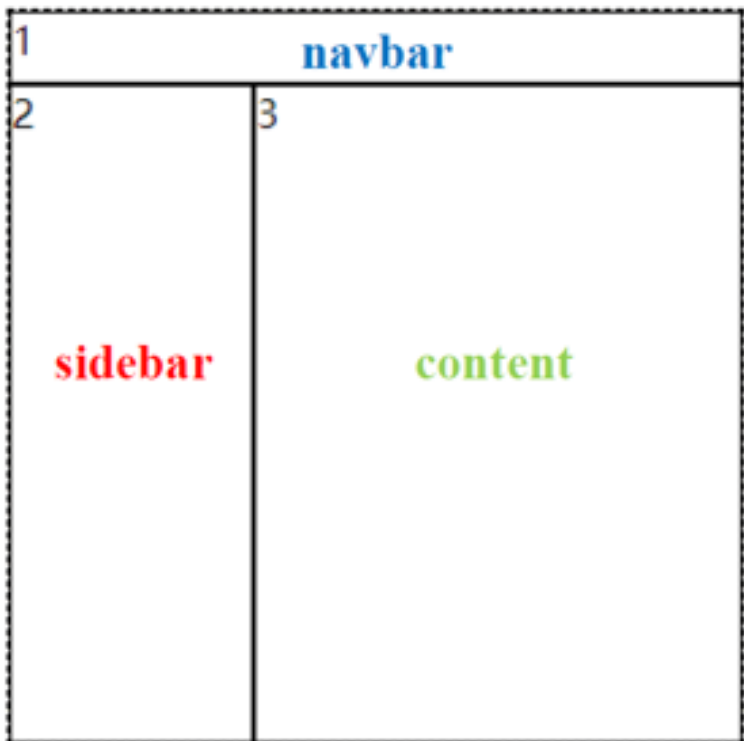



网格容器样式

3. **grid-template-areas** 属性

grid-template-areas 属性用于设置网格布局。grid-area 属性可以对网格元素进行命名。
命名的网格元素可以通过容器的 grid-template-areas 属性来引用。

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
</div>
```



```
.grid-container {
  display: grid;
  grid-template-rows: 30px 1fr 1fr;
  grid-template-columns: repeat(3, 1fr);
  width: 300px;
  height: 300px;
  border: 1px dashed;
  grid-template-areas:
    'navbar navbar navbar'
    'sidebar content content'
    'sidebar content content';
}

.grid-item {
  border: 1px solid;
}

.grid-item:nth-child(1) {
  grid-area: navbar;
}

.grid-item:nth-child(2) {
  grid-area: sidebar;
}

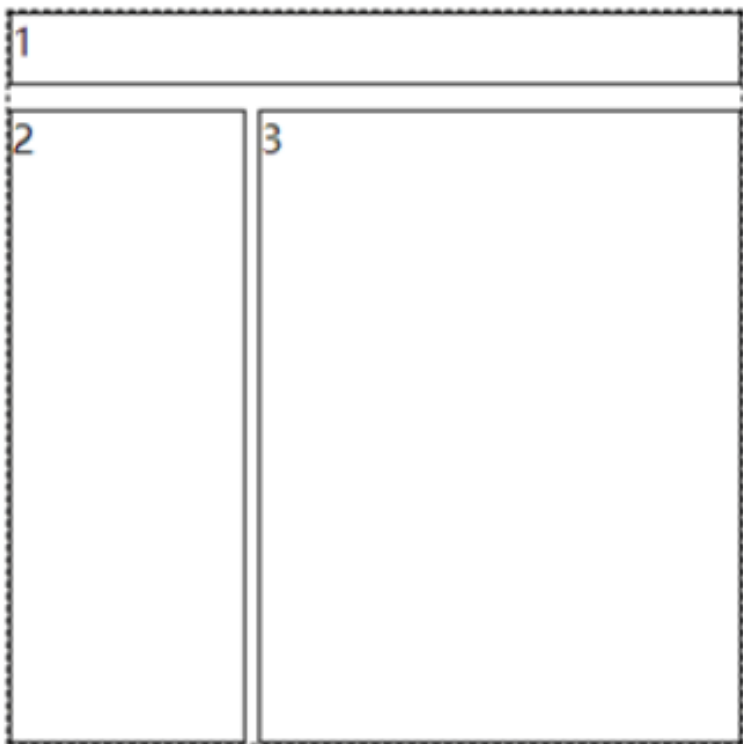
.grid-item:nth-child(3) {
  grid-area: content;
}
```

● 网格容器样式

4. **grid-gap** 属性，定义网格间隙

grid-gap: grid-row-gap grid-column-gap

使用 grid-row-gap 来定义行间隙；使用 grid-column-gap 属性来定义列间隙；



```
.grid-container{  
  grid-row-gap:10px;  
  grid-column-gap:5px;  
}
```

间隙距离可以用任何长度单位包括百分比来表示，但不能使用fr单位。

● 网格容器样式

5. **justify-items** 、 **align-items** 、 **place-items** 属性

用于设置网格元素在网格中的位置

(1) justify-items: 单元格内容的水平位置

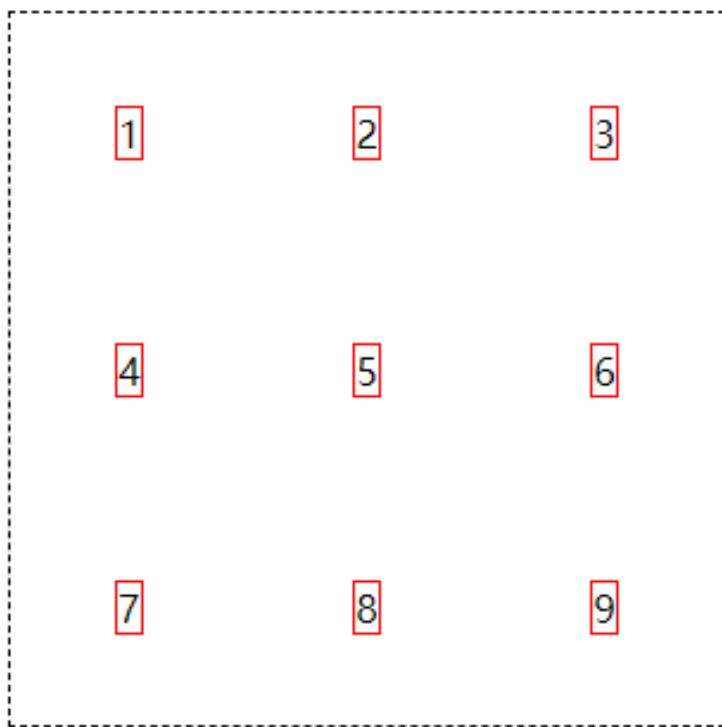
justify-items : start | end | center | 默认 stretch;

(2) align-items: 单元格内容的垂直位置

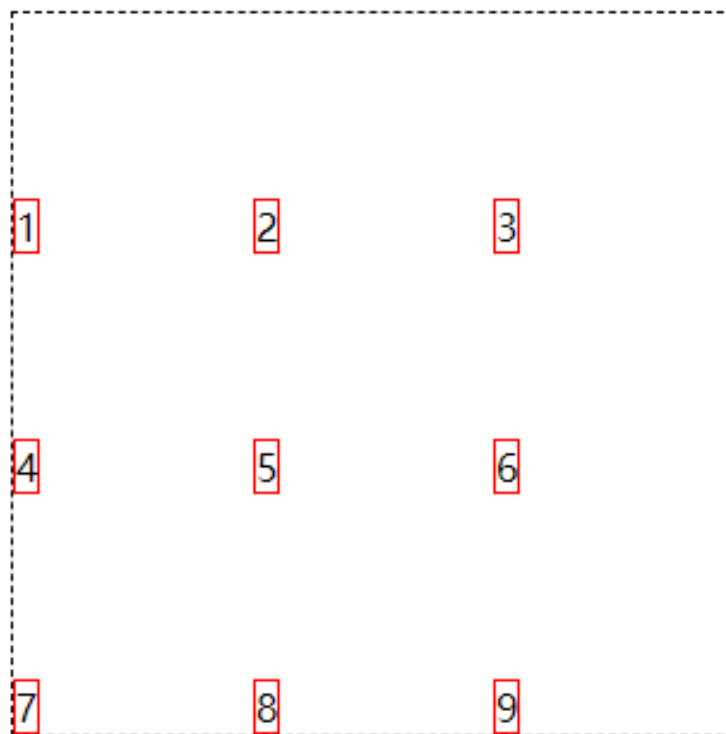
align-items : start | end | center | 默认 stretch;

place-items: align-items 属性 justify-items 属性

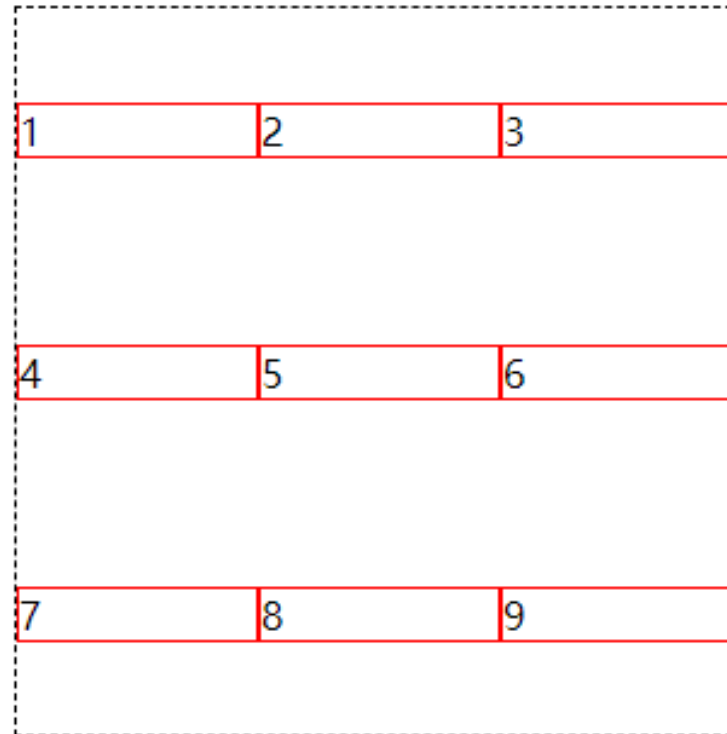
网格容器样式



justify-items: center;
align-items: center;



justify-items: start;
align-items: end;



justify-items: stretch;
align-items: center;

● 网格容器样式

6. **justify-content**、**align-content**、**place-content** 属性

用于设置整个内容区域在容器里面的位置。

(1) justify-content: 定义整个内容区域在容器里面的水平位置（左中右）。

justify-content : start | end | center | stretch | space-around | space-between | space-evenly;

(2) align-content: 定义整个内容区域在容器里面的垂直位置（上中下）

align-content : start | end | center | stretch | space-around | space-between | space-evenly;

place-content: justify-content属性 align-content属性;

● 网格容器样式

6. justify-content、align-content、place-content 属性

/ 对齐方式 */*

justify-content: center;

justify-content: start;

justify-content: end;

/ 居中排列 */*

/ 从行首开始排列 */*

/ 从行尾开始排列 */*

/ 分配弹性元素方式 */*

justify-content: space-between;

justify-content: space-around;

justify-content: space-evenly;

justify-content: stretch;

/ 均匀排列每个元素*

*首个元素放置于起点，末尾元素放置于终点 */*

/ 均匀排列每个元素*

*每个元素周围分配相同的空间 */*

/ 均匀排列每个元素*

*每个元素之间的间隔相等 */*

/ 均匀排列每个元素*

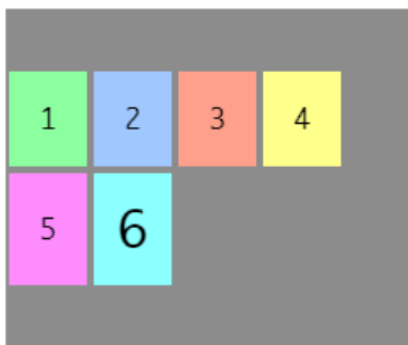
*‘auto’-sized 的元素会被拉伸来适应容器的大小 */*

网格容器样式

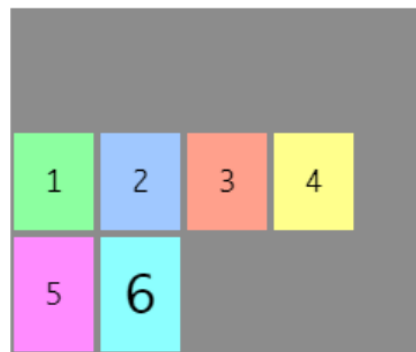
6. justify-content、align-content、place-content 属性



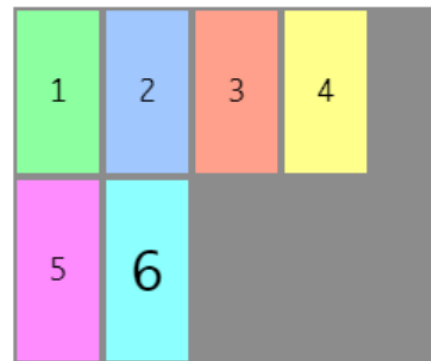
display:
align-content:



display:
align-content:



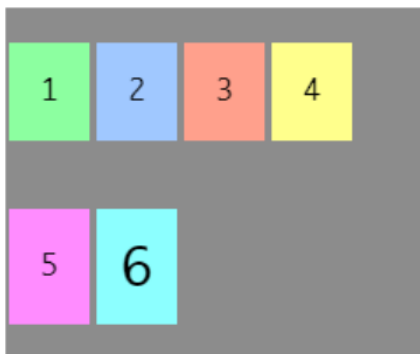
display:
align-content:



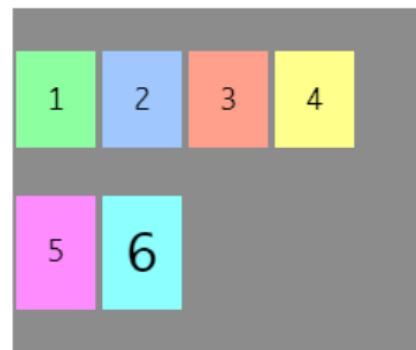
display:
align-content:



display:
align-content:



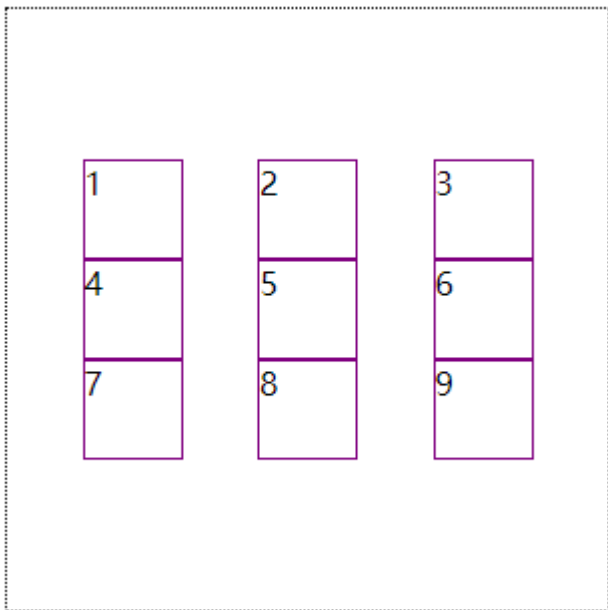
display:
align-content:



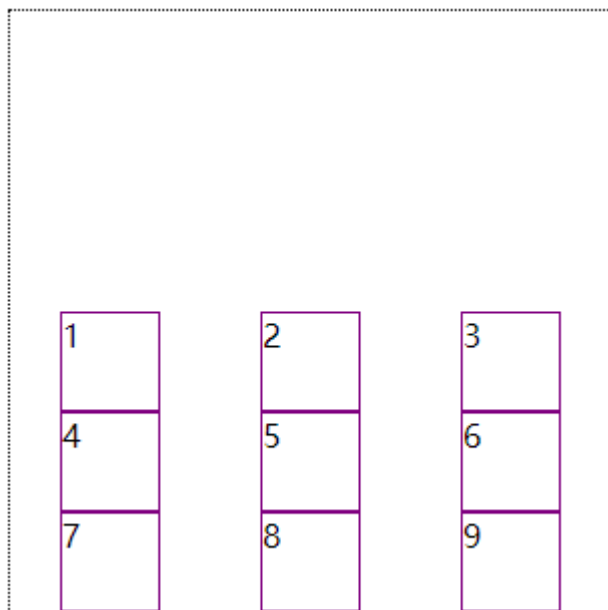
display:
align-content:



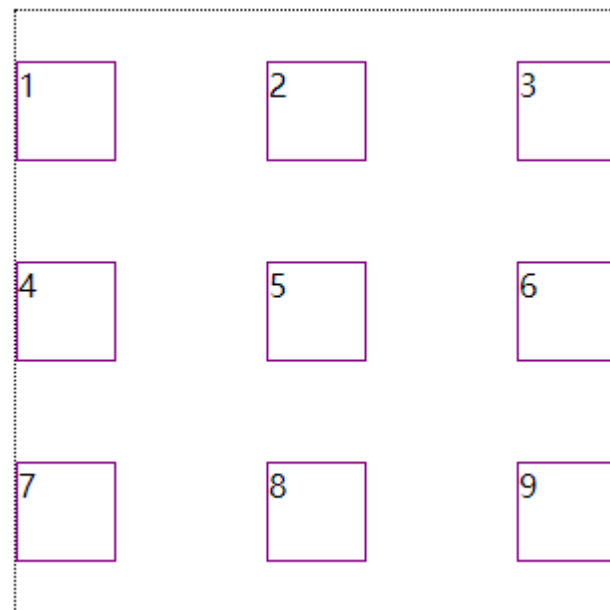
网格容器样式



`justify-content: space-evenly;`
`align-content: center;`



`justify-content: space-around;`
`align-content: end;`



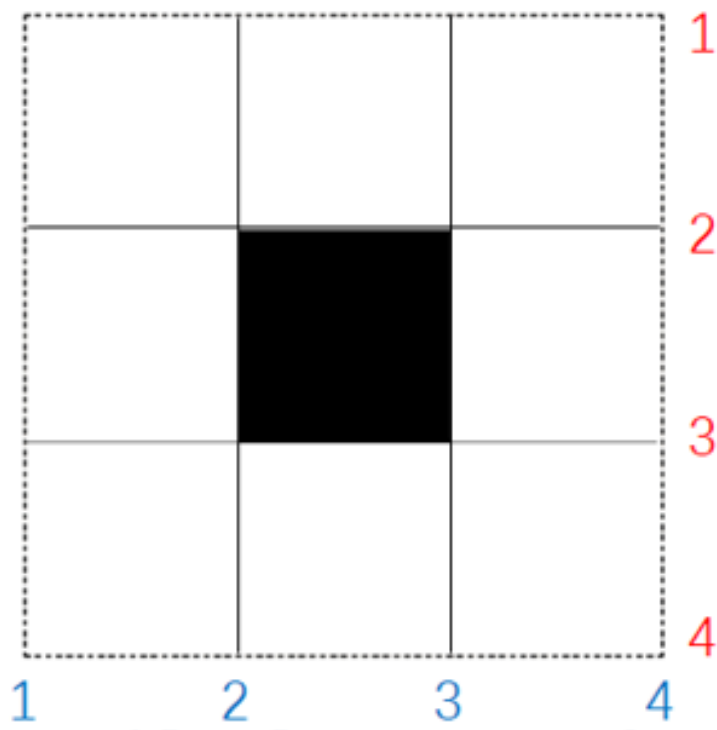
`justify-content: space-between;`
`align-content: space-around;`

● 网格元素样式

基于线的元素放置

1. **grid-column-start**、**grid-column-end**、**grid-row-start**、**grid-row-end** 属性

把网格元素按照网格线进行放置，每条网格线从1开始自动编号，元素的位置按照指定的网格线定义。



`grid-row-start:2;`
`grid-row-end:3;`

网格元素上边框从水平第2条线开始

```
.grid-item{  
  grid-row-start:2;  
  grid-row-end:3;  
  grid-column-start:2;  
  grid-column-end:3;  
}
```

`grid-column-start:2;`
`grid-column-end:3;`

网格元素左边框所在的垂直网格线从第2条线开始

● 网格元素样式

1. `grid-column`、`grid-row`属性

```
header {  
    grid-column: 1 / 3;  
    grid-row: 1;  
}
```

```
article {  
    grid-column: 2;  
    grid-row: 2;  
}
```

```
aside {  
    grid-column: 1;  
    grid-row: 2;  
}
```

```
footer {  
    grid-column: 1 / 3;  
    grid-row: 3;  
}
```



网格元素样式

1. grid-column、grid-row属性

This is my lovely blog

My article

Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem. Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus. Vivamus eu urna eget velit cursus viverra quis vestibulum sem. Aliquam tincidunt eget purus in interdum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Other things

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est.

Contact me@mysite.com

This is my lovely blog

Other things

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est.

My article

Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem. Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus. Vivamus eu urna eget velit cursus viverra quis vestibulum sem. Aliquam tincidunt eget purus in interdum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Contact me@mysite.com



网格元素样式

2. grid-template-areas、grid-area 属性

用grid-template-areas属性，给网格元素进行命名，并在属性中使用这些名字作为一个区域。

```
.container {  
  display: grid;  
  grid-template-areas:  
    "header header"  
    "sidebar content"  
    "footer footer";  
  grid-template-columns: 1fr 3fr;  
  gap: 20px;  
}
```

```
header {  
  grid-area: header;  
}  
article {  
  grid-area: content;  
}  
aside {  
  grid-area: sidebar;  
}  
footer {  
  grid-area: footer;  
}
```

This is my lovely blog

Other things

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est.

My article

Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem. Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus. Vivamus eu urna eget velit cursus viverra quis vestibulum sem. Aliquam tincidunt eget purus in interdum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Contact me@mysite.com



网格元素样式

2. grid-template-areas、grid-area 属性

grid-template-areas属性的使用规则如下：

- 需要填满网格的每个格子
- 对于某个横跨多个格子的元素，重复写上用grid-area属性定义的区域名字
- 所有名字只能出现在一个连续的区域，不能在不同的位置出现
- 一个连续的区域必须是一个矩形
- 使用.符号，让一个格子留空

```
.container {  
    display: grid;  
    grid-template-areas:  
        "header header"  
        "sidebar content"  
        "sidebar footer";  
    grid-template-columns: 1fr 3fr;  
    gap: 20px;  
}
```

This is my lovely blog

Other things

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est.

My article

Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem. Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus. Vivamus eu urna eget velit cursus viverra quis vestibulum sem. Aliquam tincidunt eget purus in interdum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Contact me@mysite.com



● 网格元素样式

2. grid-area 属性

还可以使用 grid-area 一次性指定所有的行/列序号： / / / ， 也就是先指定开始坐标的行/列序号，再指定结束坐标的行/列序号。

例如： 2/2/3/3： 行起始位置/列起始位置/行结束位置/列结束位置

/* 使用grid-area一次性指定所有行列序号 */

```
header{
  grid-area: 1/1/2/3;
}
article{
  grid-area:2/2/3/3;
}
aside{
  grid-area: 2/1/3/2;
}
footer{
  grid-area: 3/1/4/3;
}
```

● 网格元素样式

3. justify-self、align-self、place-self 属性

单个网格元素的对齐方式

(1) justify-self属性：单元格内容的水平位置，用法同justify-items属性。

justify-self: start | end | center | stretch (水平拉伸)

(2) align-self属性：单元格内容的垂直位置，用法同align-items属性。

align-self: start | end | center | stretch (垂直拉伸)

place-self: align-self属性 justify-self属性;

● 网格元素样式

3. justify-self、align-self、place-self 属性

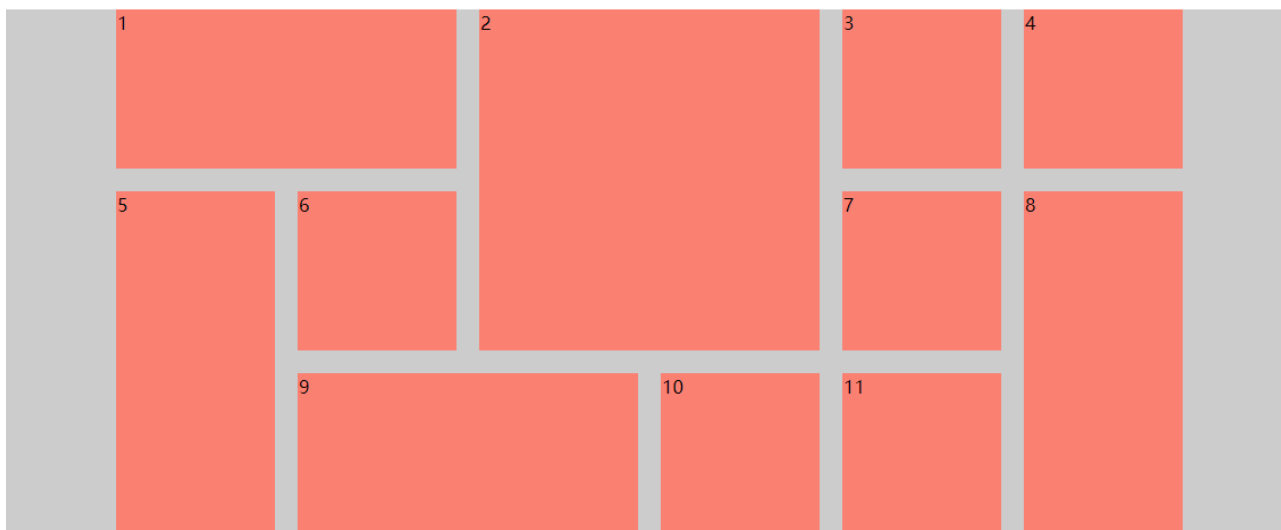


```
span:nth-child(2) {  
  justify-self: start;  
}
```

```
span:nth-child(3) {  
  justify-self: center;  
}
```

```
span:nth-child(4) {  
  justify-self: end;  
}
```

网格布局grid—案例



```
<div class="grid-container">  
  <div class="grid-item">1</div>  
  .....  
  <div class="grid-item">11</div>  
</div>
```

```
.grid-container{  
  display:grid;  
  justify-content:center;  
  align-content:center;  
  grid-template-columns:repeat(6,140px);  
  grid-template-rows:repeat(3,140px);  
  gap:20px;  
}  
.grid-item:nth-child(1){  
  grid-column: 1 / span 2;  
  grid-row: 1 / 2;  
}  
.grid-item:nth-child(2){  
  grid-column: 3 / span 2;  
  grid-row: 1 / span 2;  
}
```

grid-row: 1 / span 2;意思是网格元素从第1条水平线开始， 到跨越2个网格线的位置结束

Flex布局与Grid布局总结

- 弹性盒子布局通过设置`display:flex`实现，可以很方便的解决传统布局垂直居中、剩余空间分配的问题。
- 弹性盒子由**弹性容器**和**弹性元素**组成，有些属性用于设置弹性容器，有些属性用于设置弹性元素。
- 弹性元素可以按行或按列分布在容器内，对齐方式多样而且灵活。如果弹性容器内有剩余空白空间，弹性元素可以按照预设比例拉长；如果容器宽度不够，弹性元素可以按照预设比例压缩。利用弹性盒子布局可以很方便的实现导航栏、绝对底部等效果。
- 网格布局通过设置`display:grid`实现，并将容器划分成行和列，由此形成单元格。网格布局在二维布局方面比弹性盒子更强大。网格由**网格容器**和**网格元素**组成，可以设置网格容器和网格元素的属性。
- 网格布局可以很方便的实现行列纵横交错的布局。