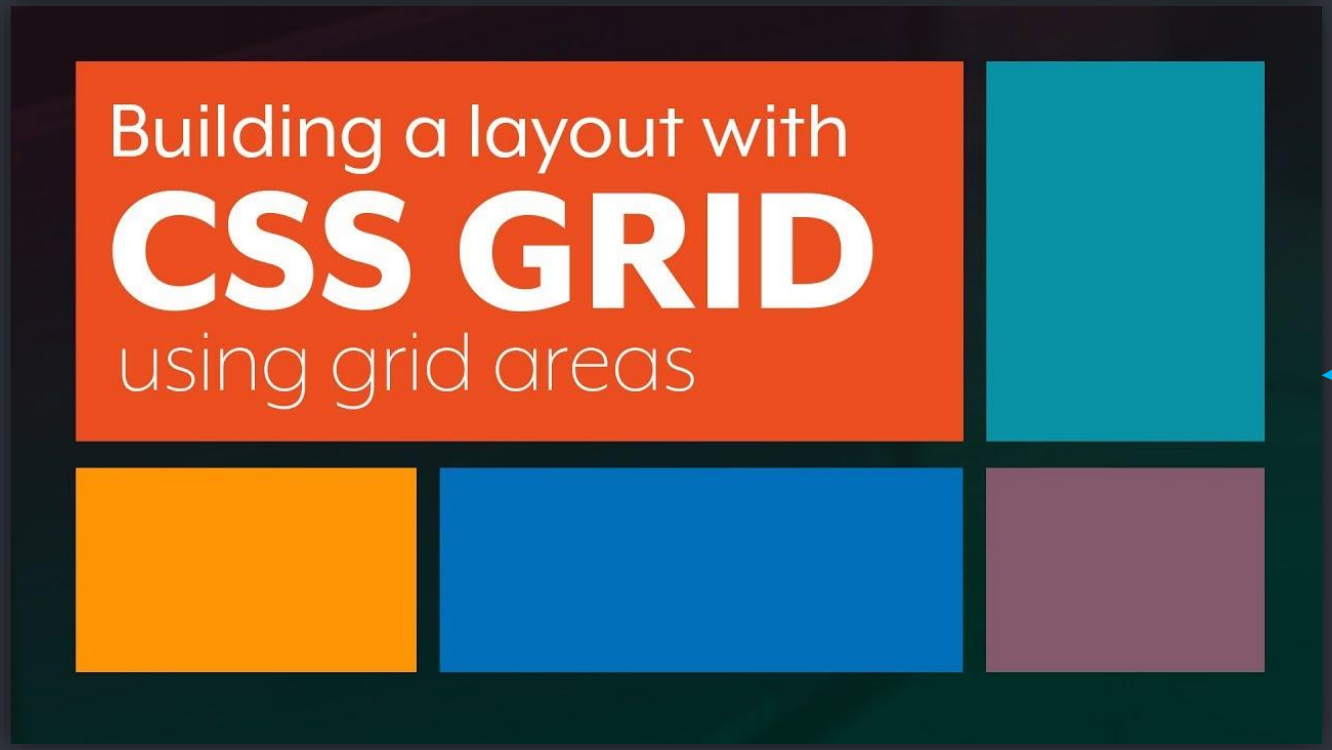
css极速入门 —— grid布局

主讲人与课程设计：耕耕

什么是grid布局?

Flex 布局是轴线布局，只能指定“项目”针对轴线的位置，可以看作是**一维布局**，**Grid** 布局则是将容器划分成“行”和“列”，产生单元格，然后指定“项目所在”的单元格，可以看作是**二维布局**，Grid 布局远比 Flex 布局强大



像一个格子一个格子的排列，更加灵活，更加强大

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，麻烦!

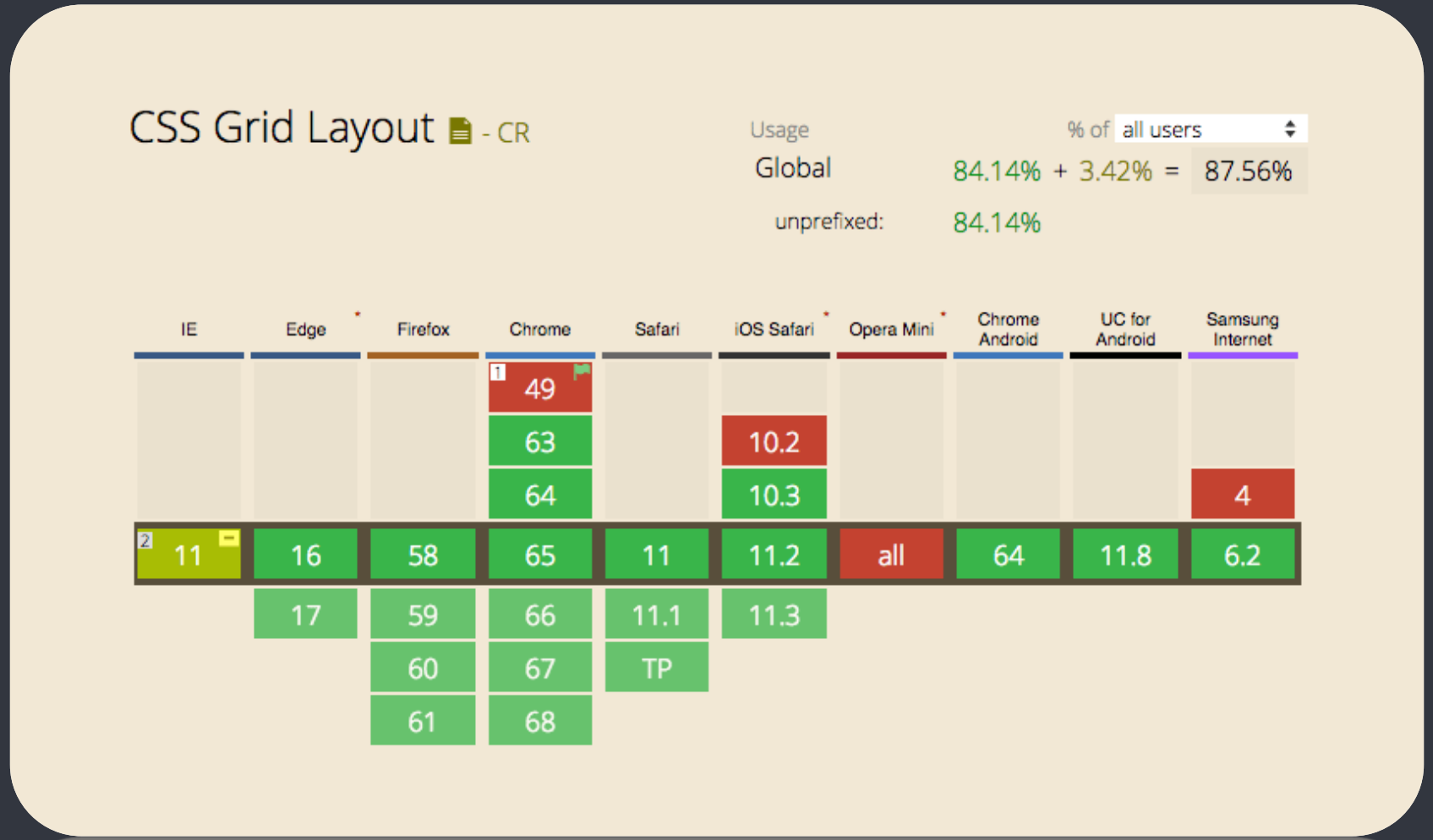
2. flex布局

有自己的一套属性，效率高，学习成本低，兼容性强!

3. grid布局

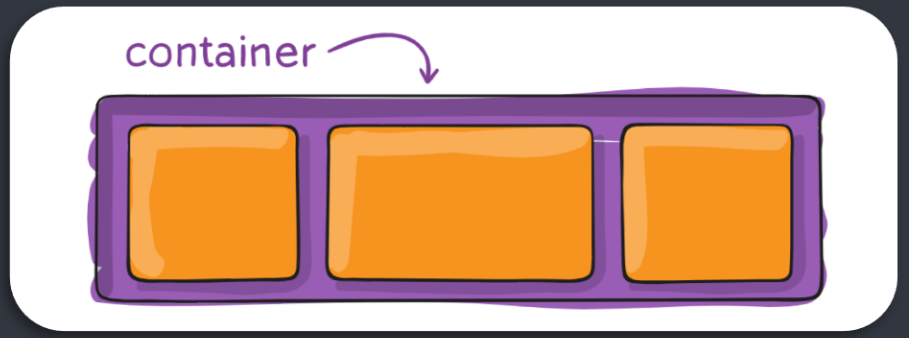
网格布局（Grid）是最强大的 CSS 布局方案。但是知识点较多，学习成本相对困难些，目前的兼容性不如flex好!

浏览器兼容性

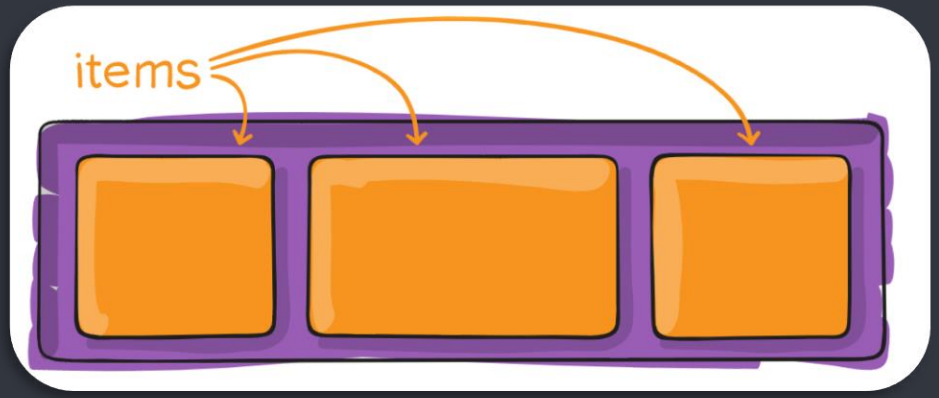

CSS Grid Layout - CR
Usage
% of all users
Global
84.14% + 3.42% = 87.56%
unprefixed:
84.14%
IE
Edge
Firefox
Chrome
Safari
iOS Safari
Opera Mini
Chrome Android
UC for Android
Samsung Internet
49
63
64
10.2
10.3
4
11
16
58
65
11
11.2
all
64
11.8
6.2
17
59
66
11.1
11.3
60
67
TP
61
68

基本概念

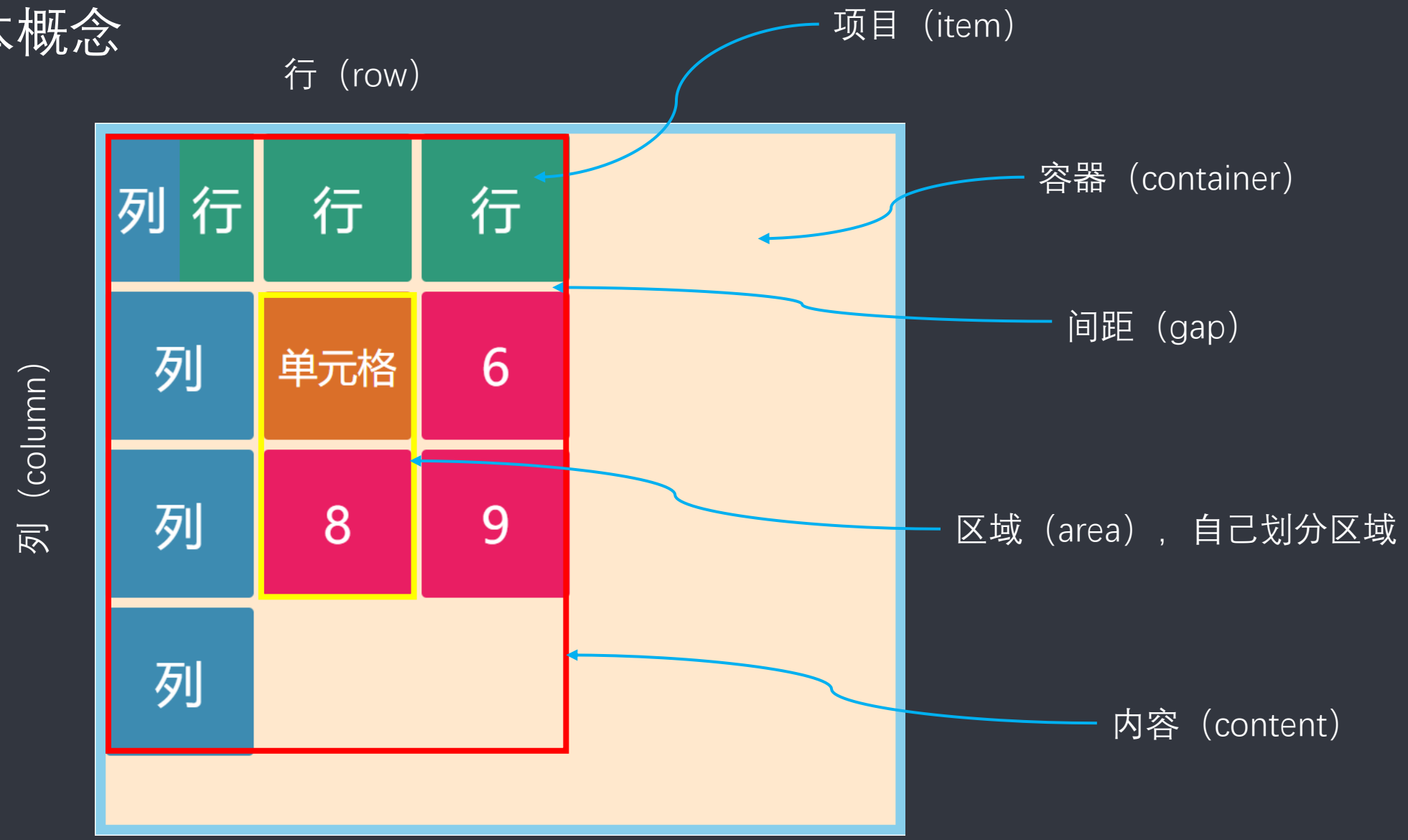
1. 容器 —— 有容器属性



2. 项目 —— 有项目属性

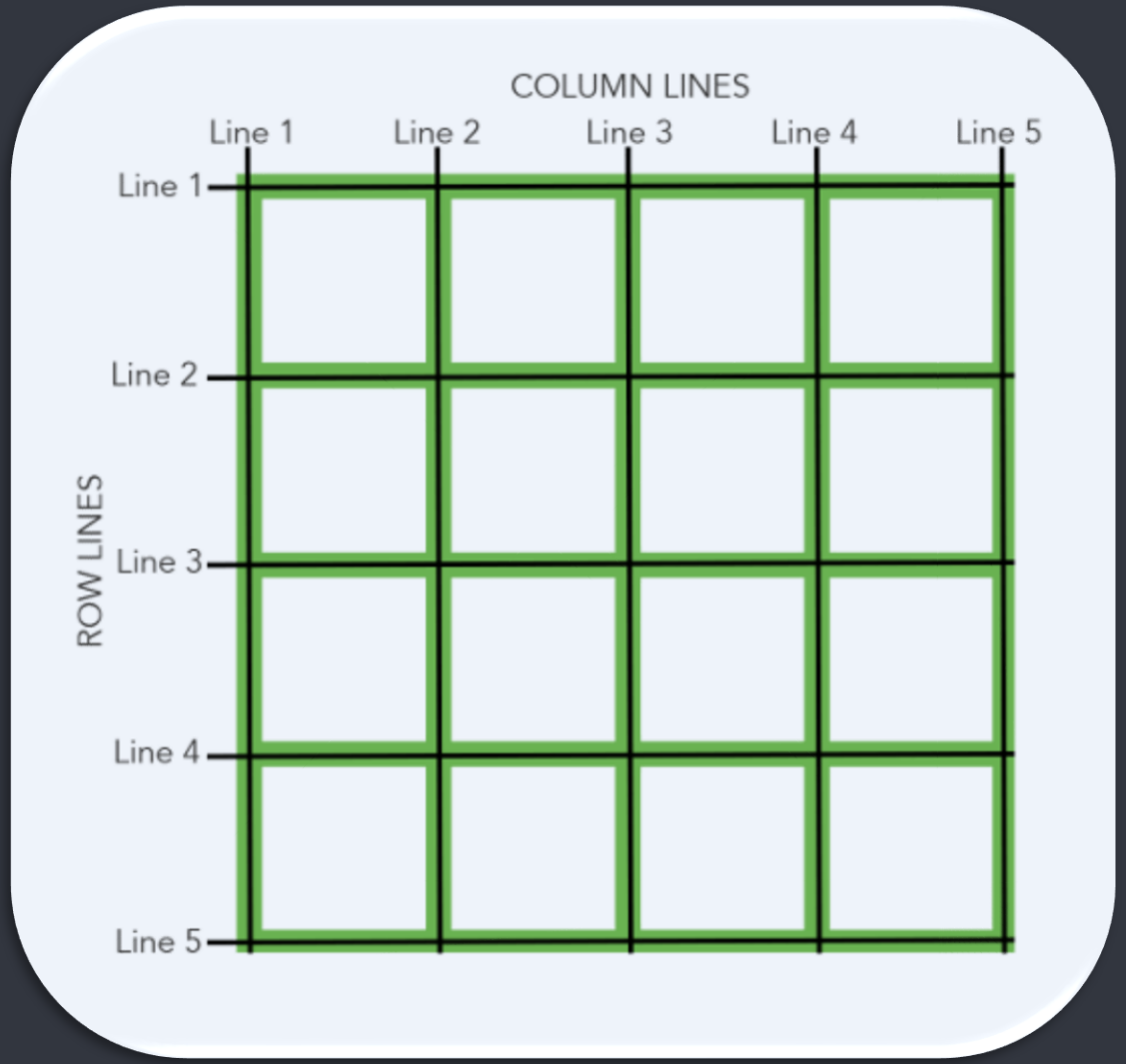


基本概念



每个grid布局，有隐藏的**网格线**，用来帮助定位的（看下一页ppt）

网格线


COLUMN LINES
Line 1
Line 2
Line 3
Line 4
Line 5
ROW LINES
Line 1
Line 2
Line 3
Line 4
Line 5

容器属性

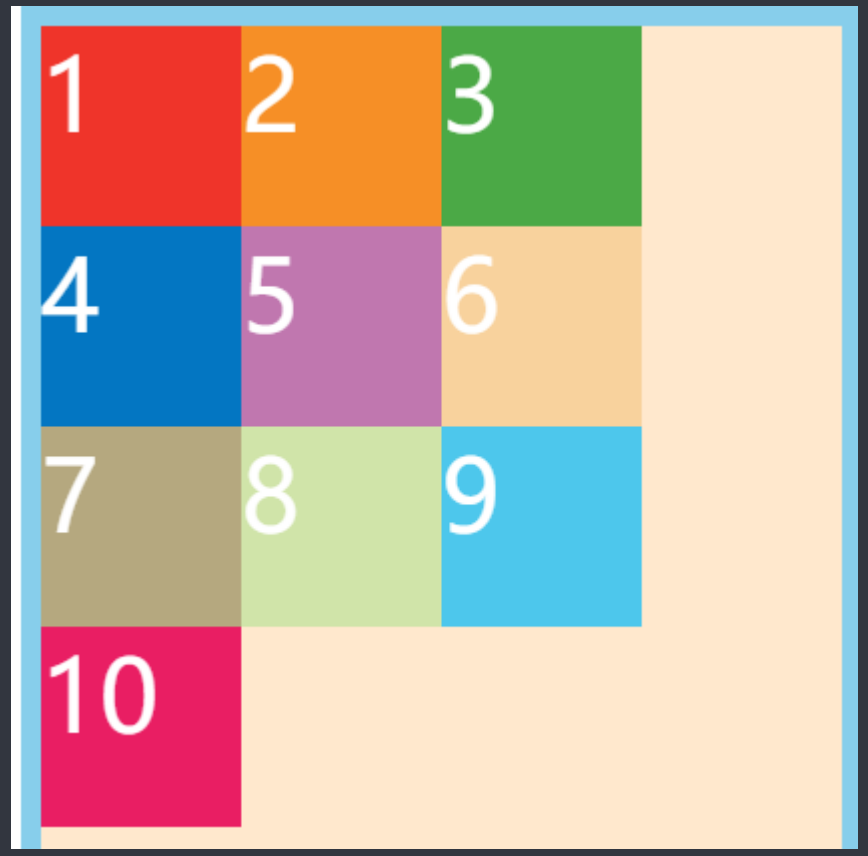
1. grid-template-columns
2. grid-template-rows
3. grid-row-gap
4. grid-column-gap
5. grid-gap (3和4的简写)
6. grid-template-areas
7. grid-auto-flow
8. justify-items
9. align-items
10. place-items(8和9的简写)
11. justify-content
12. align-content
13. place-content(11和12的简写)
14. grid-auto-columns
15. grid-auto-rows

容器属性 grid-template-*

你想要多少行或者列，就填写相应属性值的个数，不填写，自动分配

```
grid-template-columns:     100px  100px  100px;

grid-template-rows:
100px 100px 100px 100px;
```



容器属性 grid-template-* 相关

1. repeat()，第一个参数是重复的**次数**，第二个参数是所要**重复的值**

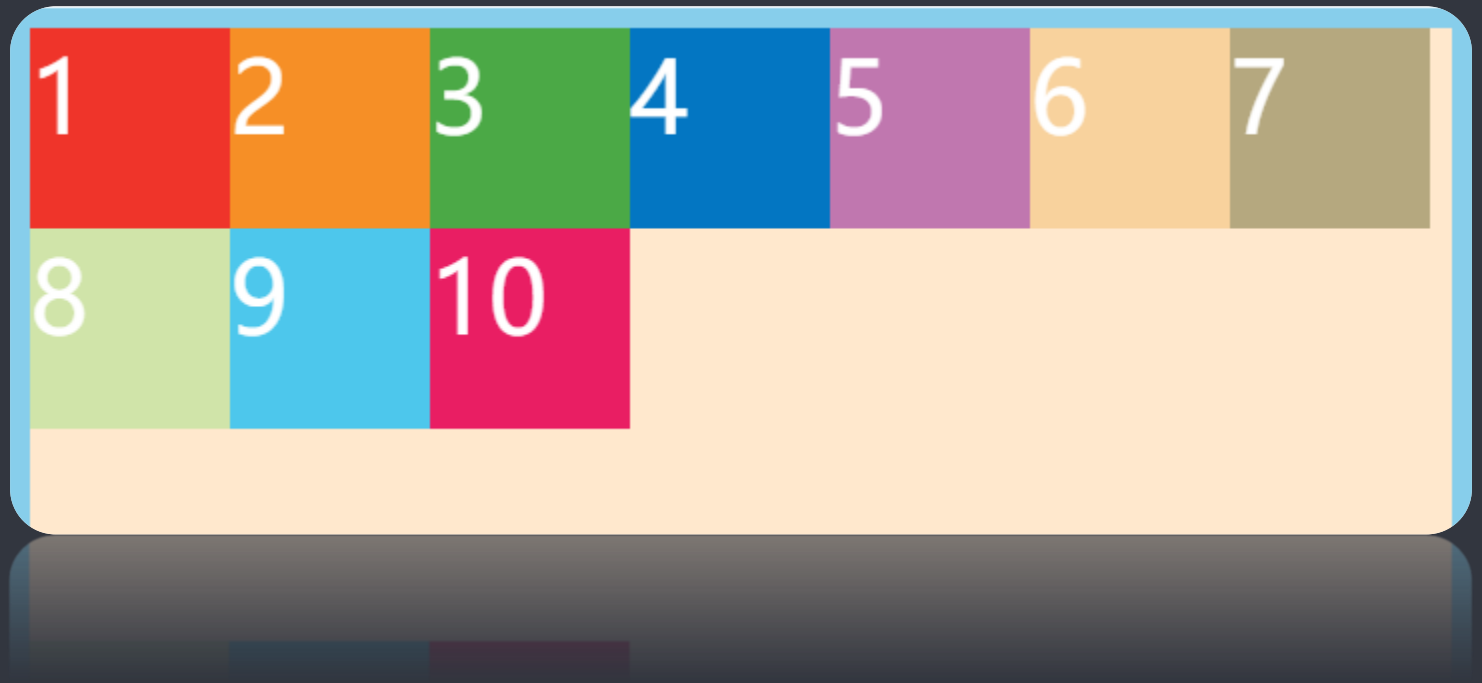
```
grid-template-columns: 100px 100px 100px;
```

```
grid-template-columns: repeat(3, 100px);
```

容器属性 grid-template-* 相关

2. auto-fill，有时，单元格的大小是固定的，但是容器的大小不确定，这个属性就会自动填充

```
grid-template-columns: repeat(auto-fill, 100px);
```



容器属性 grid-template-* 相关

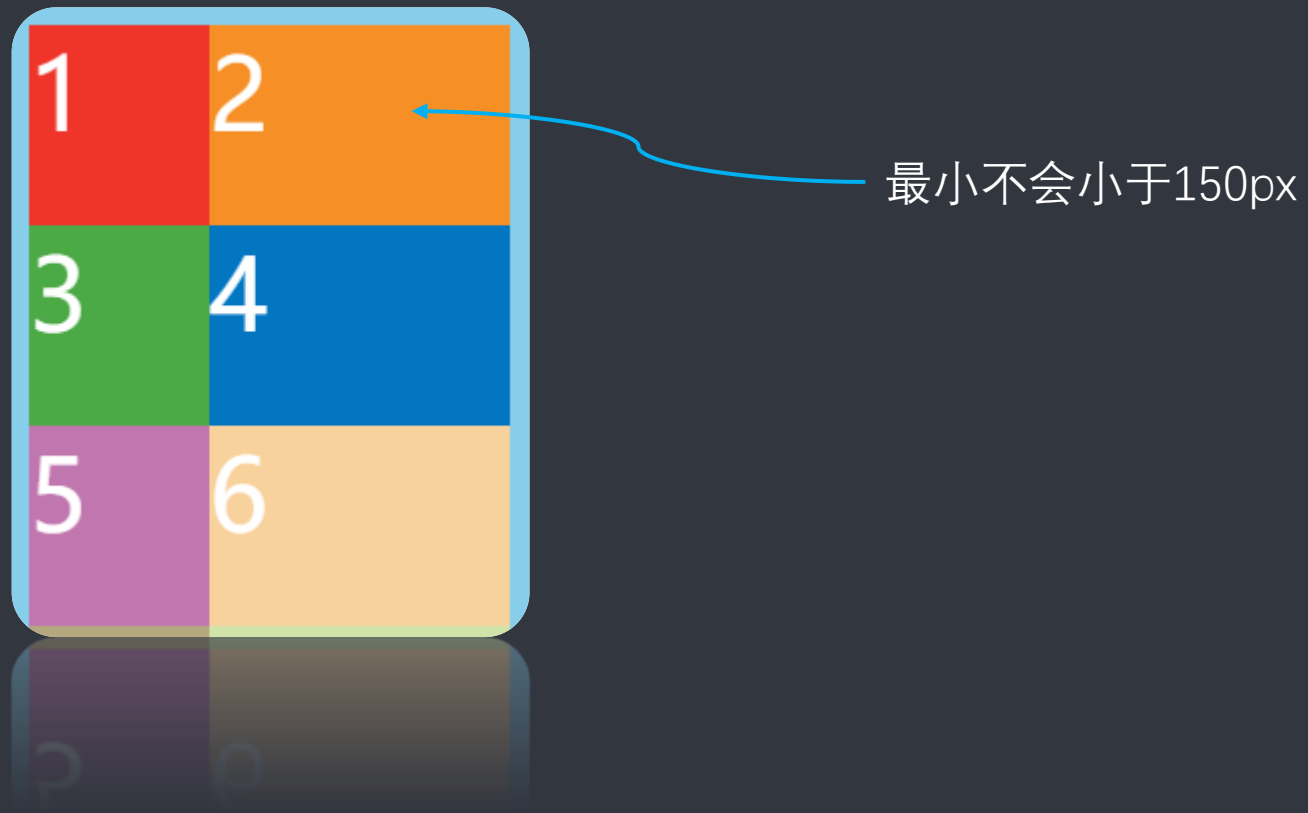
3. fr，为了方便表示比例关系，网格布局提供了fr关键字（fraction 的缩写，意为"片段"）

```
grid-template-columns: repeat(4, 1fr); //宽度平均分成4份
```

容器属性 grid-template-* 相关

4. minmax()，函数产生一个长度范围，表示长度就在这个范围之中，它接受两个参数，分别为最小值和最大值

```
grid-template-columns: 1fr minmax(150px, 1fr);
```



容器属性 grid-template-* 相关

5. auto，表示由浏览器自己决定长度

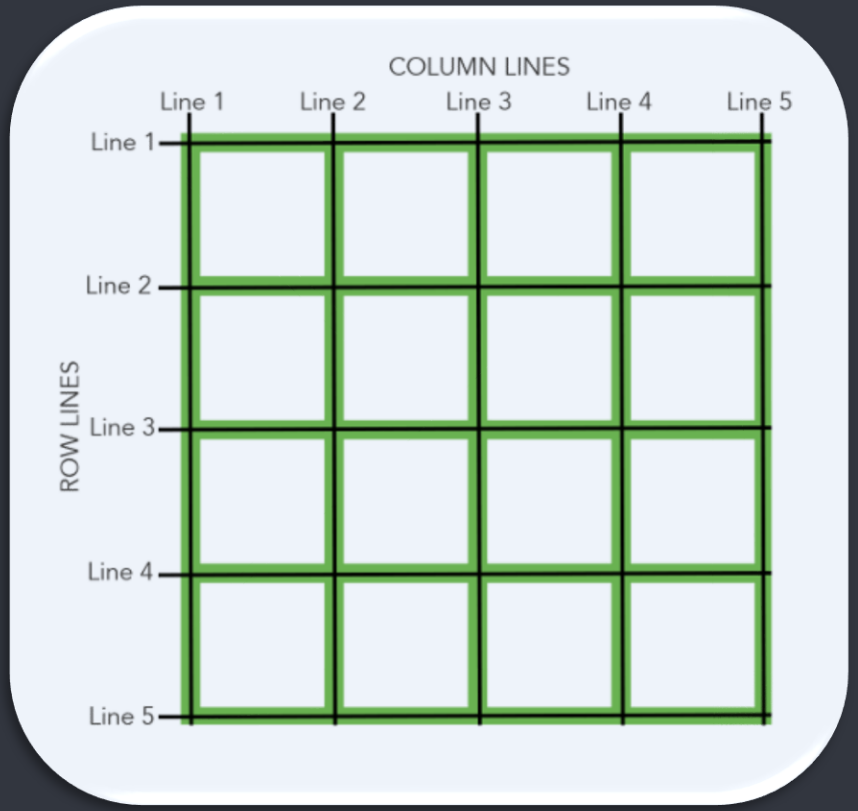
```
grid-template-columns: 100px auto 100px;
```

中间的宽度由浏览器决定，自动分配

容器属性 grid-template-* 相关

6. 网格线，可以用方括号定义网格线名称，方便**以后的引用**

```
grid-template-columns: [c1] 100px [c2] 100px [c3] 100px [c4];
```



容器属性 grid-row-gap / grid-column-gap

一句话解释就是，item（项目）相互之间的距离

```
grid-column-gap: 20px;
grid-row-gap: 20px;
```

```
grid-gap: 20px;
```

注意：根据最新标准，上面三个属性名的grid-前缀已经删除，grid-column-gap和grid-row-gap写成column-gap和row-gap，grid-gap写成gap

容器属性 grid-template-areas

一个区域由**单个或多个单元格**组成，由你决定 (具体使用，需要在项目属性里面设置)

```
grid-template-areas: 'a b c'
                     'd e f'
                     'g h i';
```

```
grid-template-areas: 'a a a'
                     'b b b'
                     'c c c';
```

```
grid-template-areas: 'a . c'
                     'd . f'
                     'g . i';
```

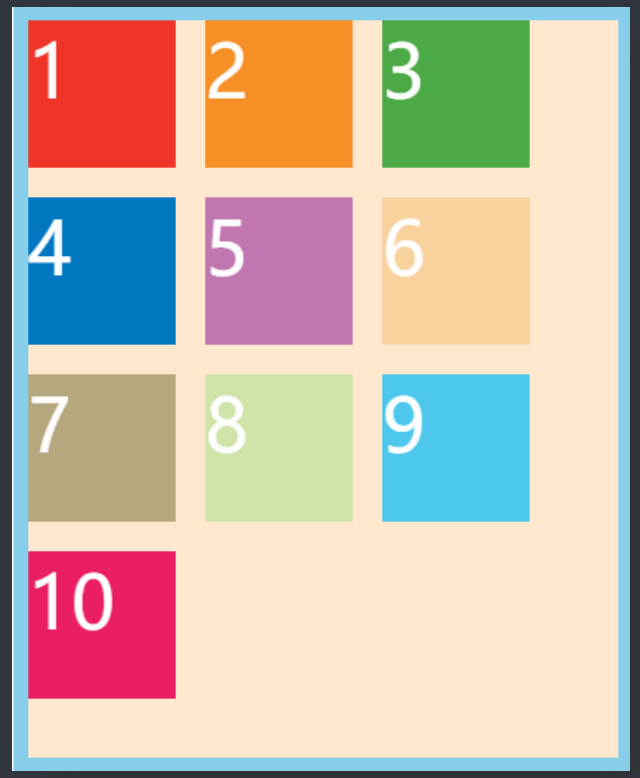
区域不需要利用，则使用"点"（.）表示

区域的命名会影响到网格线。每个区域的起始网格线，会自动命名为区域名-start，终止网格线自动命名为区域名-end

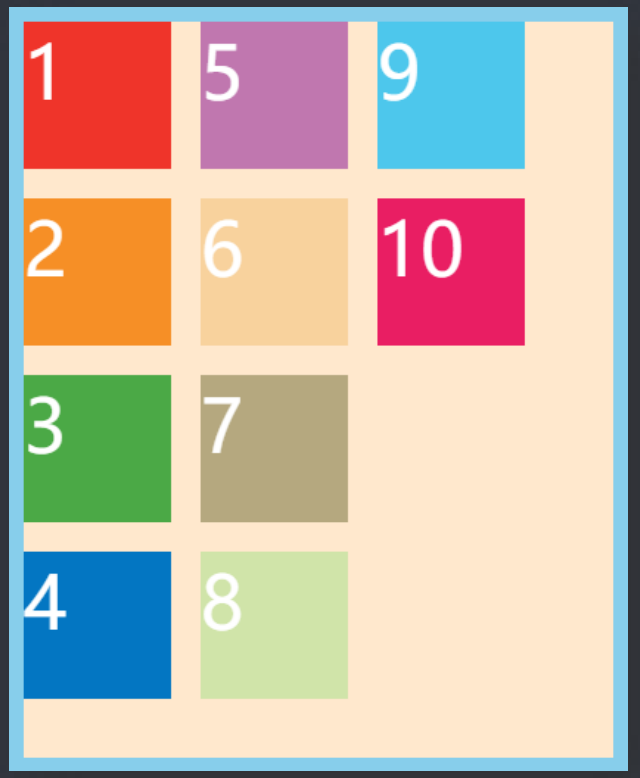
容器属性 grid-auto-flow

划分网格以后，容器的子元素会按照顺序，自动放置在每一个网格。默认的放置顺序是“先行后列”，即先填满第一行，再开始放入第二行 （**就是子元素的排放顺序**）

```
grid-auto-flow: row;
```

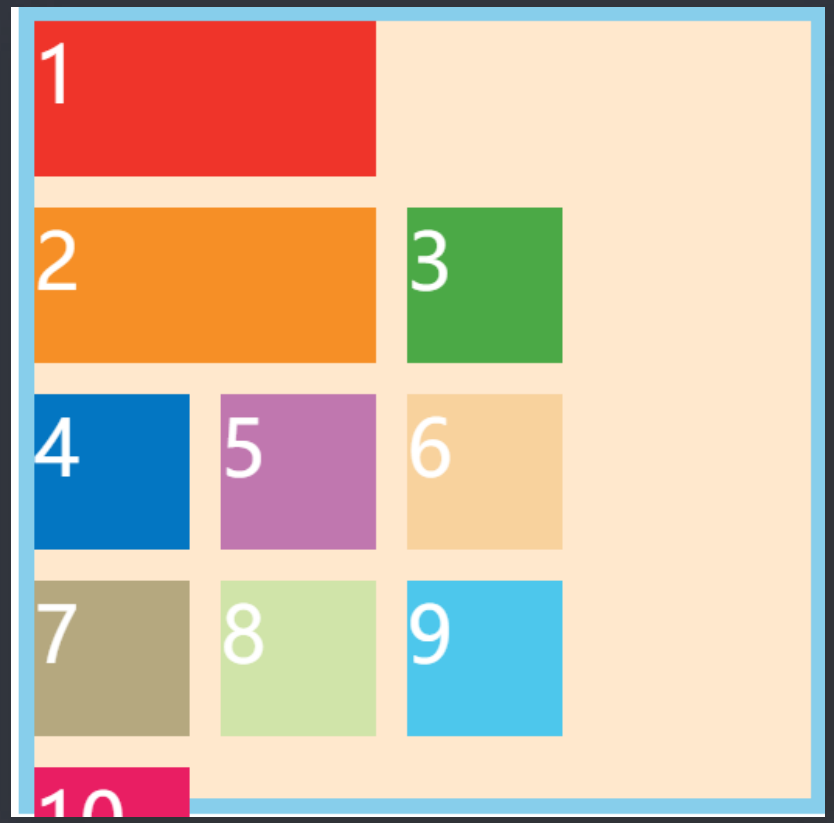



```
grid-auto-flow: column;
```

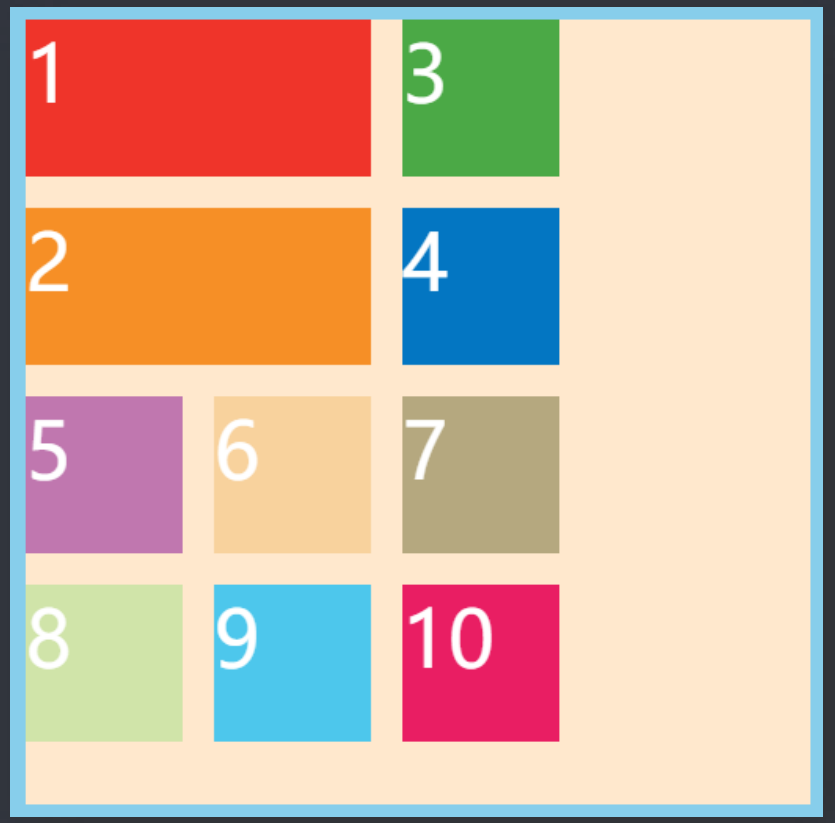


容器属性 grid-auto-flow 相关

```
grid-auto-flow: row;
```




```
grid-auto-flow: row dense;
```

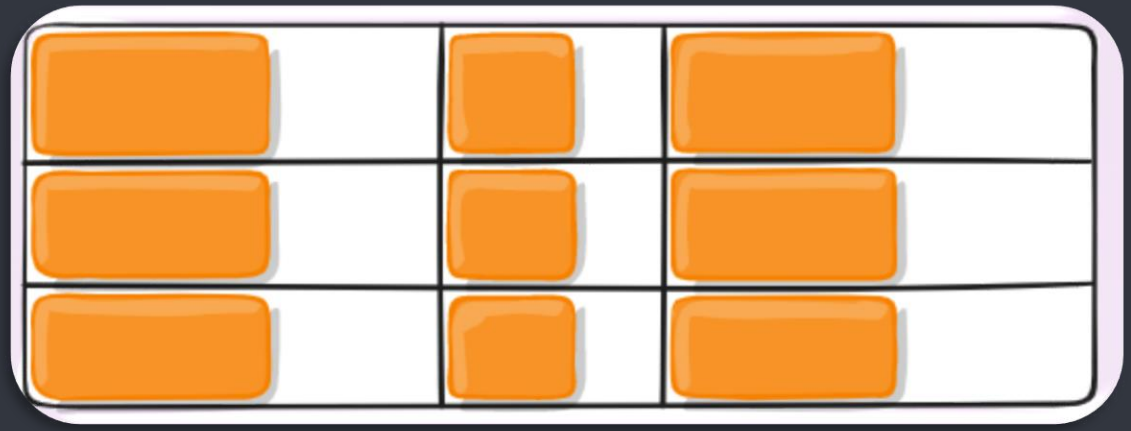


容器属性 justify-items(水平方向) / align-items (垂直方向)

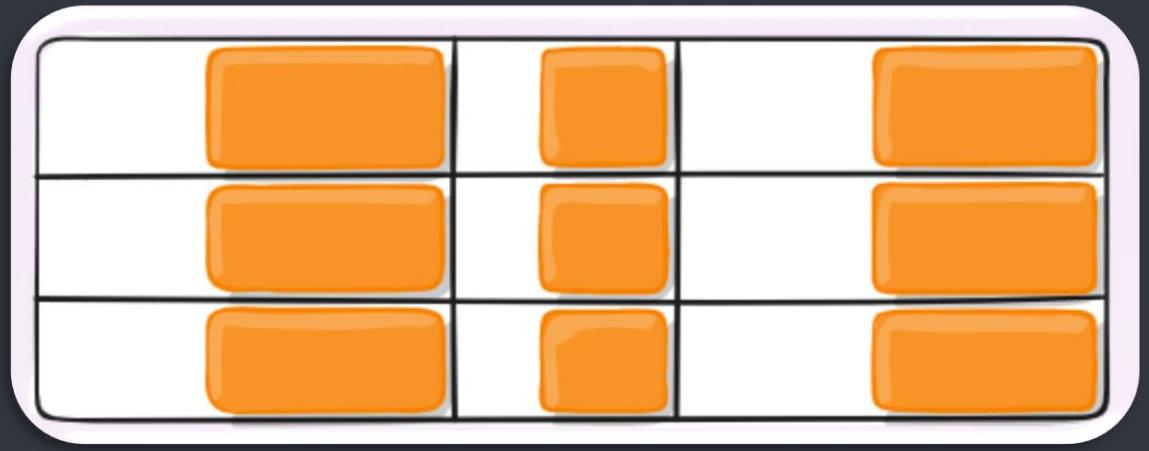
设置单元格内容的水平和垂直的对齐方式

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

justify-items: start;



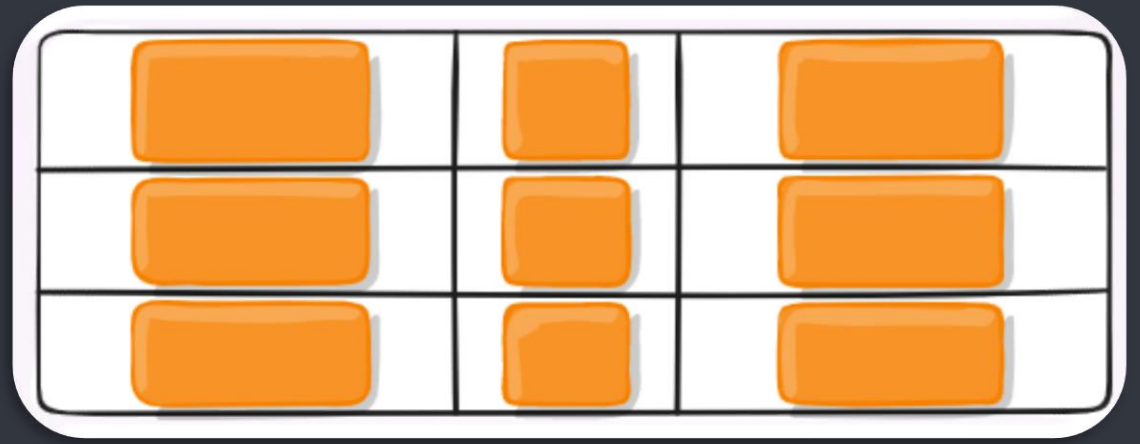
justify-items: end;



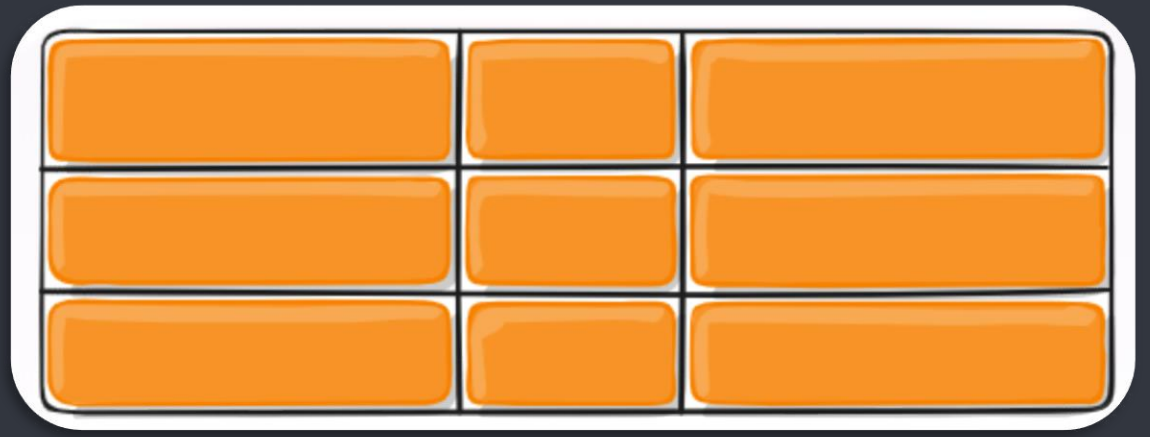
容器属性 justify-items(水平方向) / align-items (垂直方向)

设置单元格内容的水平和垂直的对齐方式

justify-items: center;



justify-items: stretch;

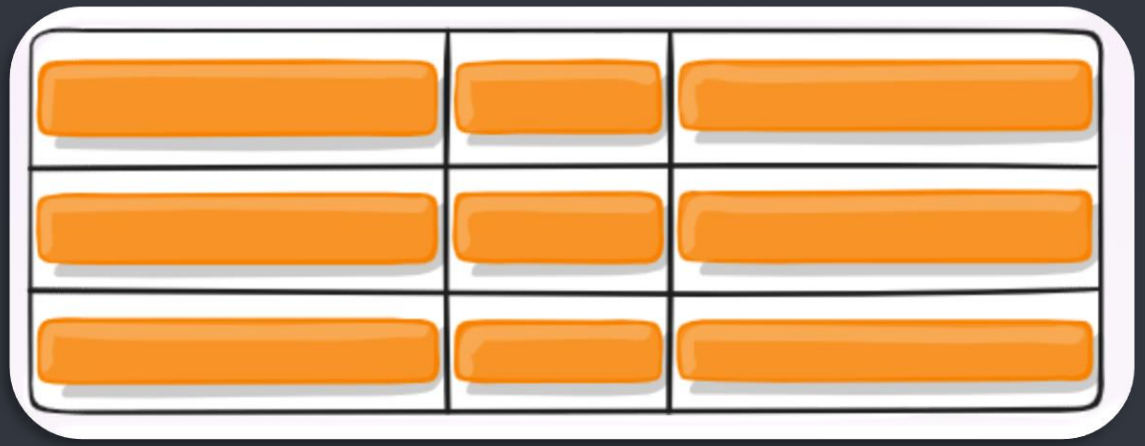


容器属性 justify-items(水平方向) / align-items (垂直方向)

设置单元格内容的水平和垂直的对齐方式

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

align-items: center;



place-items属性是align-items属性和justify-items属性的合并简写形式

place-items: <align-items> <justify-items>;

容器属性 justify-content (水平方向) / align-content (垂直方向)

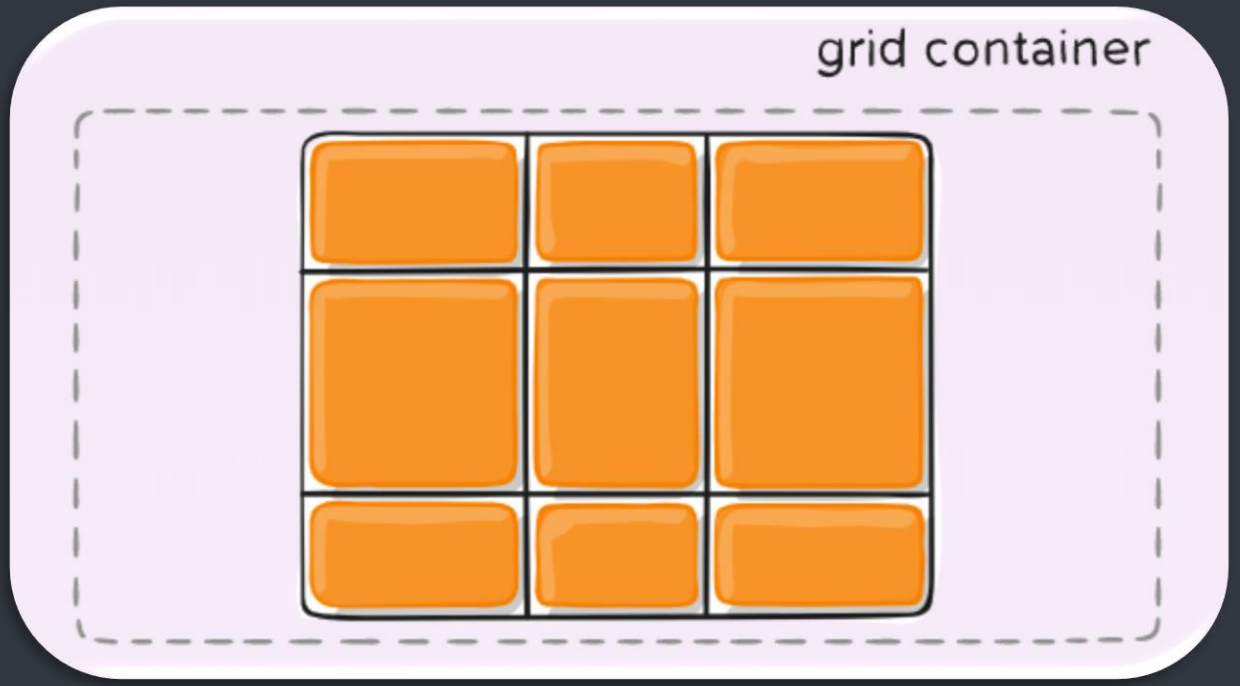
设置**整个内容**区域的水平和垂直的对齐方式

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

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

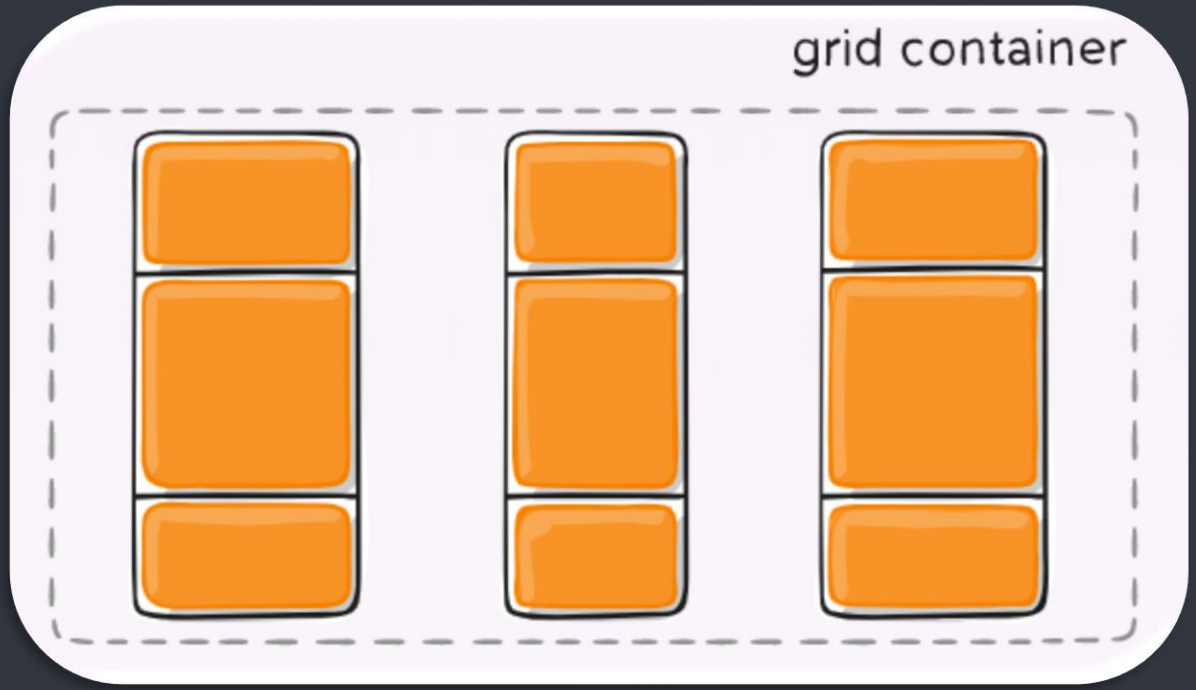
容器属性 justify-content (水平方向) / align-content (垂直方向)

```
justify-content: center;
```



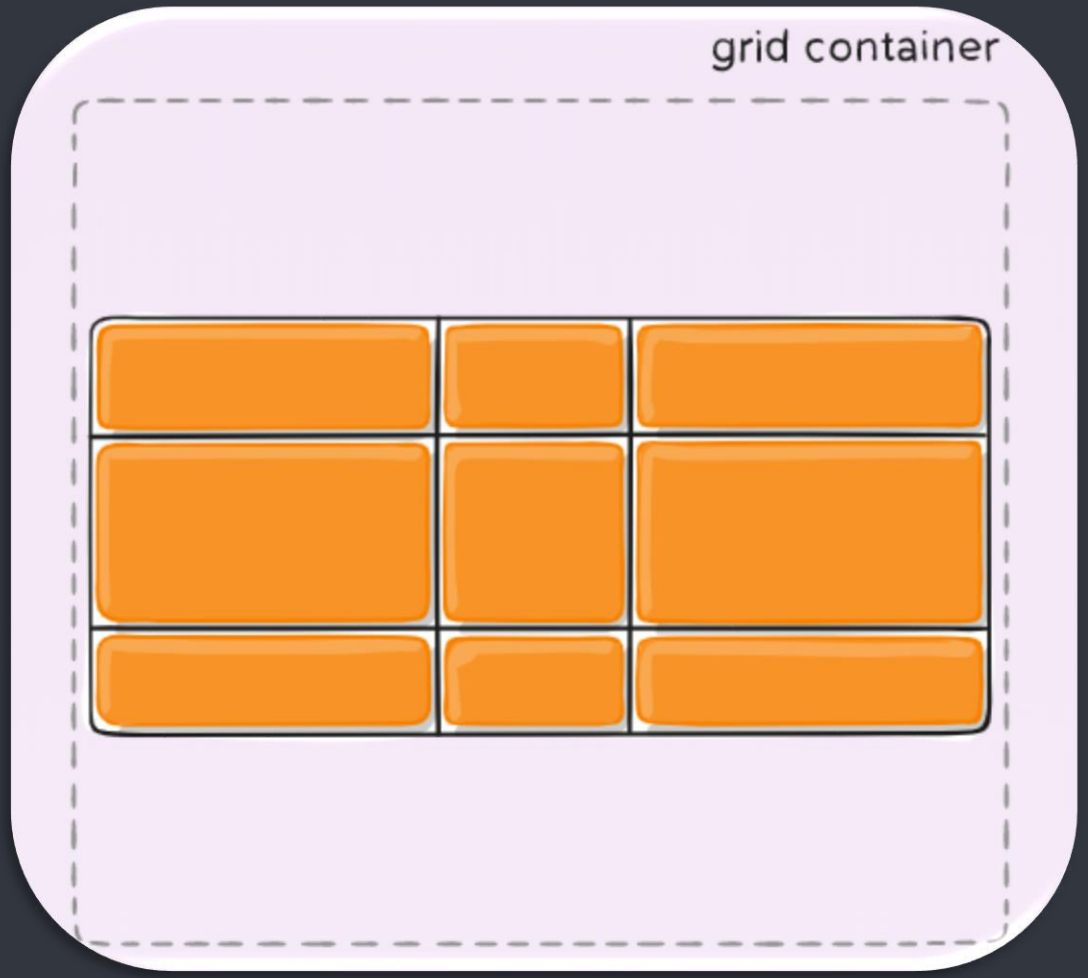
容器属性 justify-content (水平方向) / align-content (垂直方向)

justify-content: space-around;



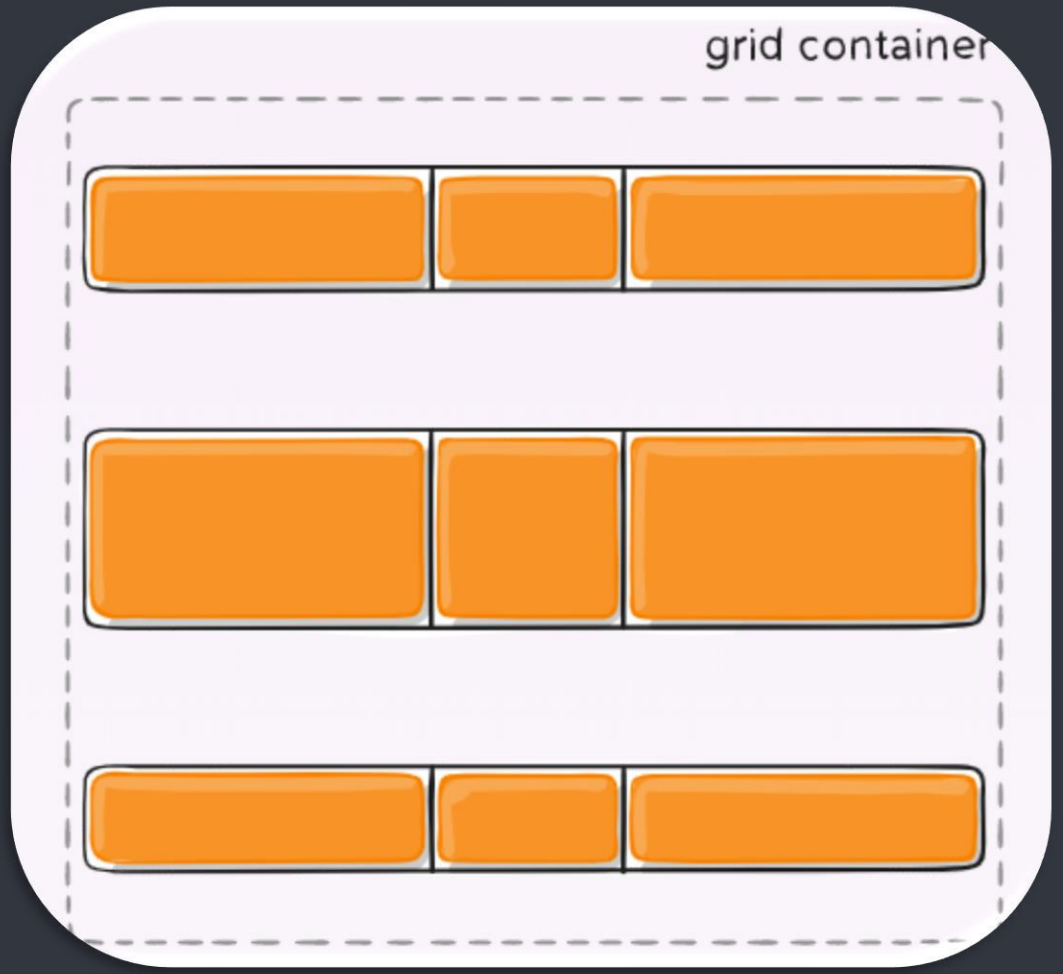
容器属性 justify-content (水平方向) / align-content (垂直方向)

```
align-content: center;
```



容器属性 justify-content (水平方向) / align-content (垂直方向)

align-content: space-around;



容器属性 grid-auto-columns / grid-auto-rows

用来设置**多出来的项目**宽和高

```
grid-auto-rows: 50px;
```

我只设置了3x3个项目，但是
实际有10个，整个属性就是用来
设置多出来的项目

项目属性

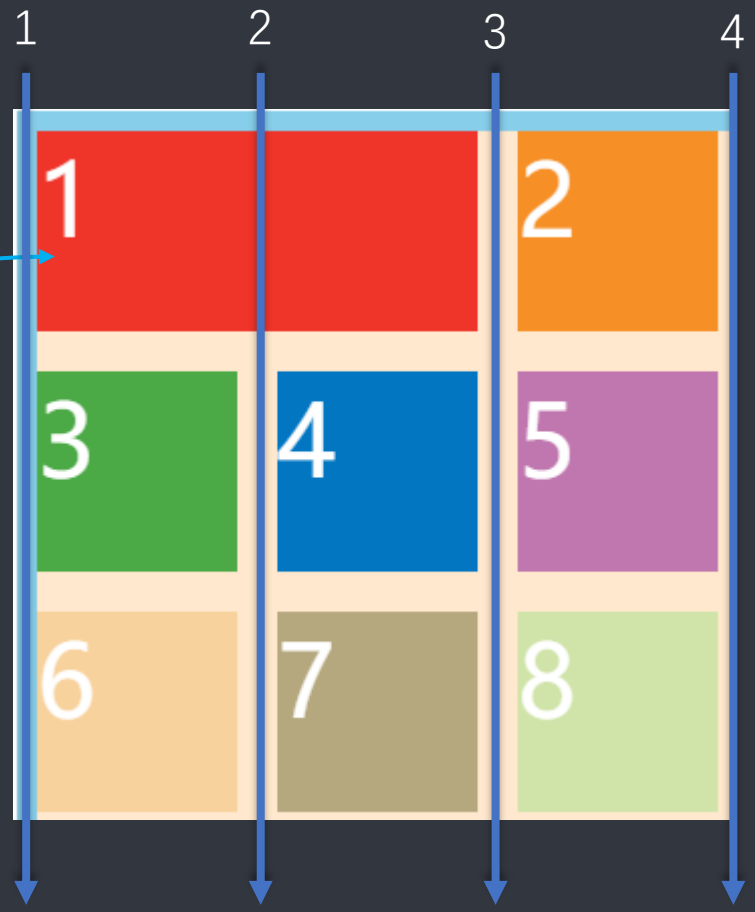
1. grid-column-start
2. grid-column-end
3. grid-row-start
4. grid-row-end
5. grid-column (1和2的简写形式)
6. grid-row (3和4的简写形式)
7. grid-area
8. justify-self
9. align-self
10. place-self (8和9的简写形式)

项目属性 grid-column-start / grid-column-end
grid-row-start / grid-row-end

一句话解释，用来指定item的**具体位置，根据在哪根网格线**

列方向4根网格线，第一个项目占用了从第一根到第三根

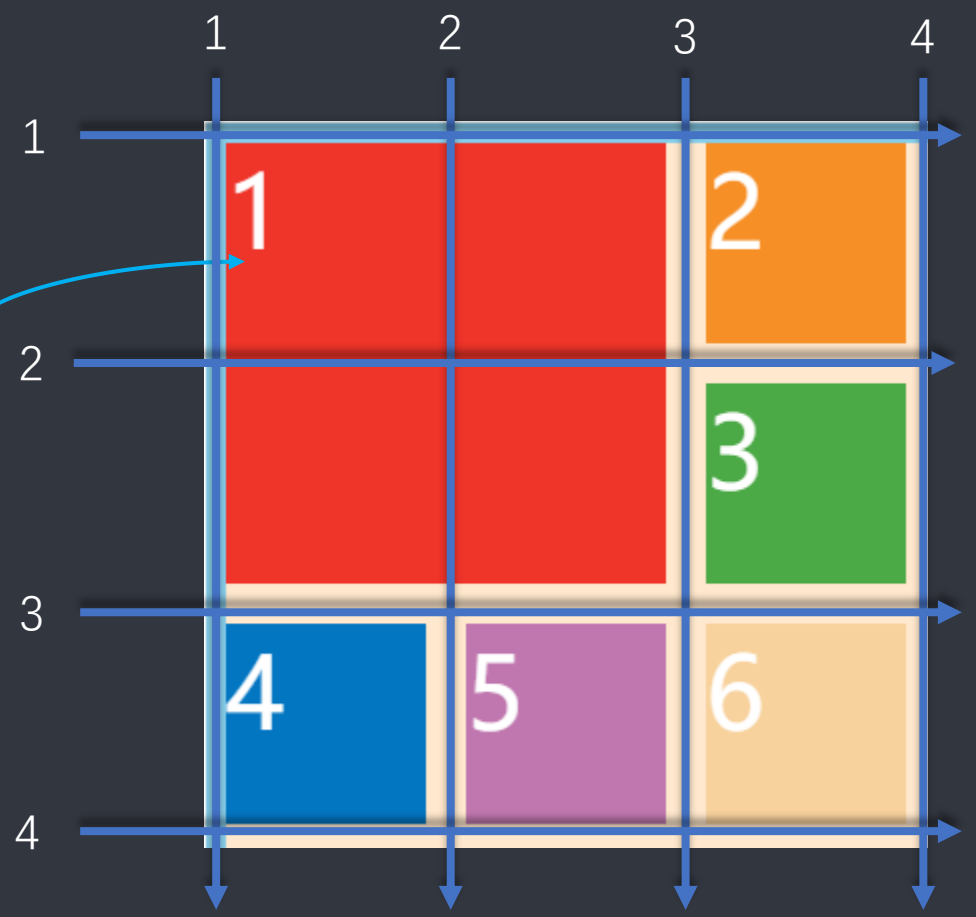
```
grid-column-start: 1;
grid-column-end: 3;
```



项目属性 grid-column-start / grid-column-end
grid-row-start / grid-row-end

一句话解释，用来指定item的**具体位置，根据在哪根网格线**

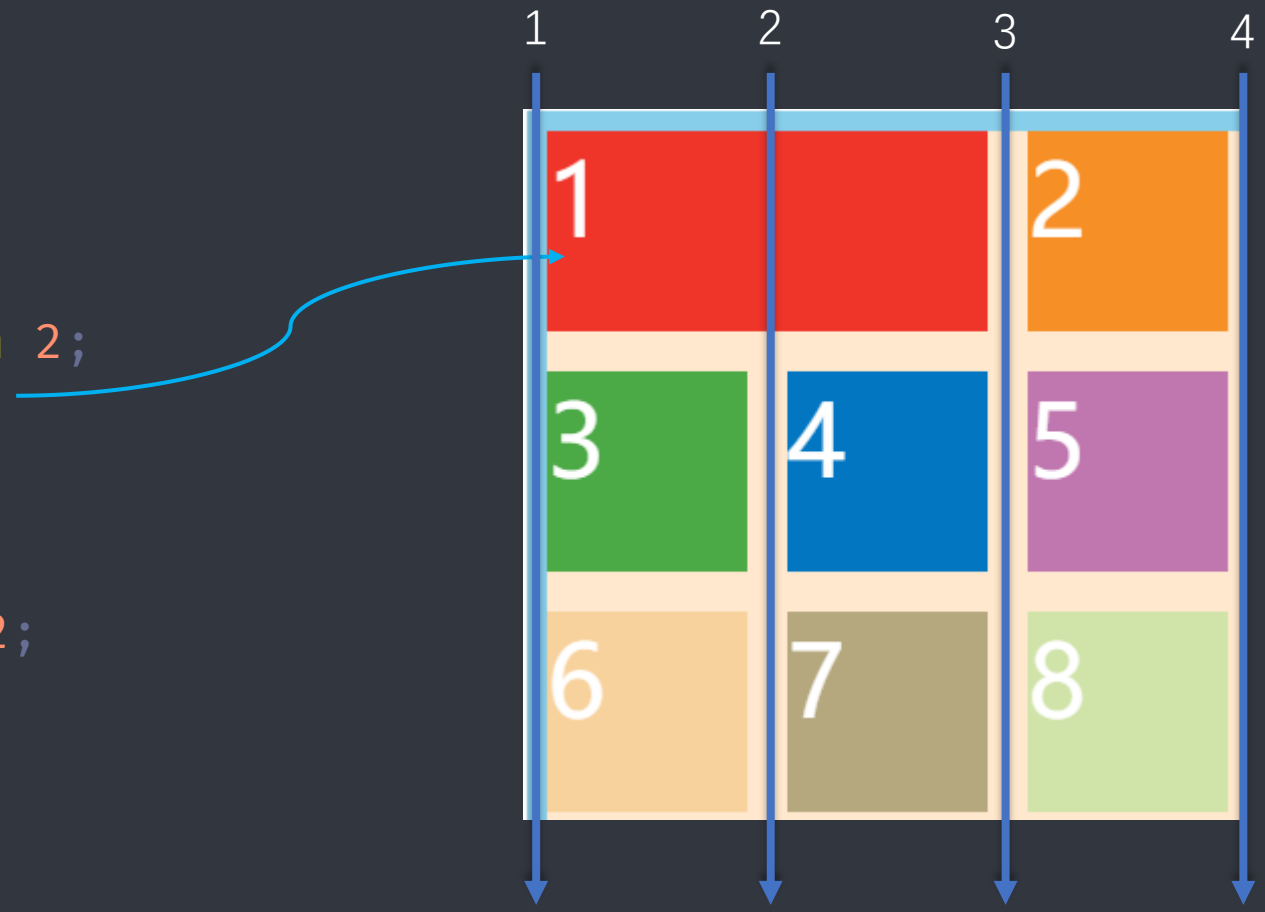
```
grid-column-start: 1;
grid-column-end: 3;
grid-row-start: 1;
grid-row-end: 3;
```



项目属性 grid-column-start / grid-column-end
grid-row-start / grid-row-end

```
grid-column-start: span 2;
```

```
grid-column-end: span 2;
```

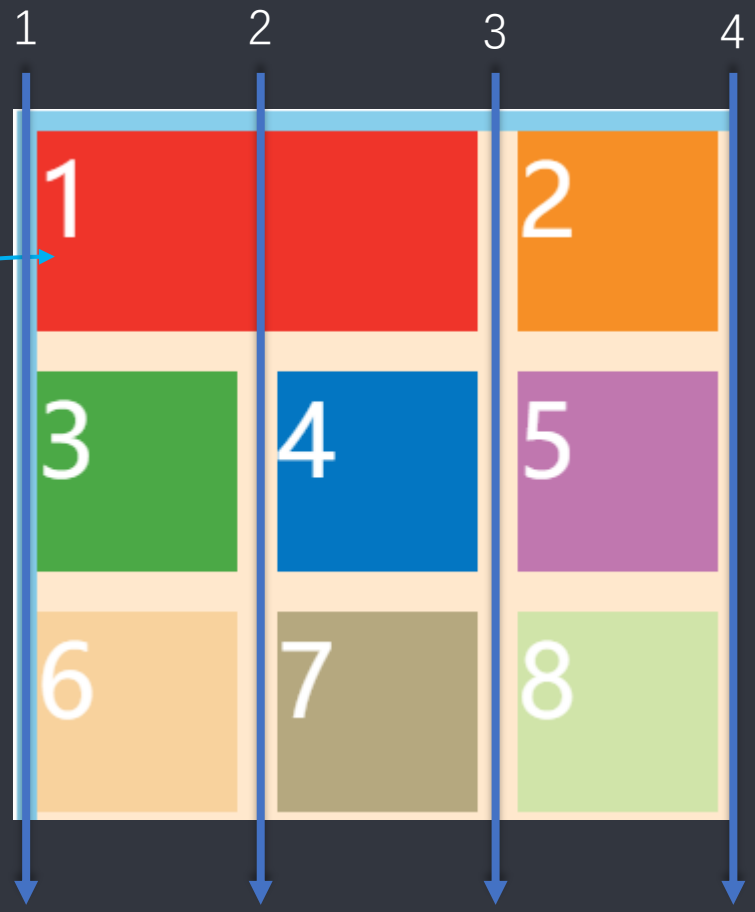


项目属性 grid-column / grid-row

grid-column属性是grid-column-start和grid-column-end的合并简写形式，grid-row属性是grid-row-start属性和grid-row-end的合并简写形式

```
grid-column: 1 / 3;
```

```
grid-column-start: 1;
grid-column-end: 3;
```

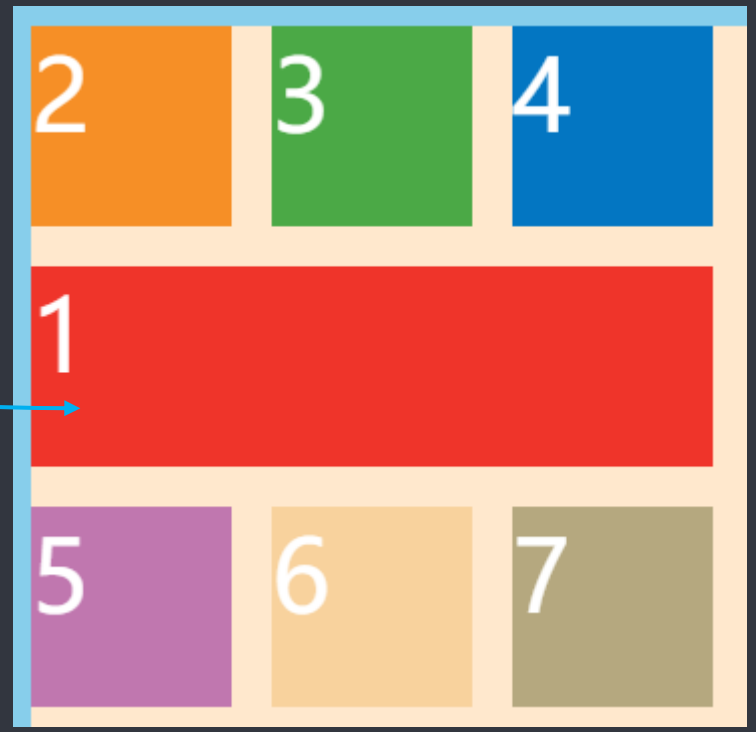


项目属性 grid-area

指定项目放在哪一个区域

```
grid-template-areas:'a a a'
                    'b b b'
                    'c c c';

grid-area: b;
```



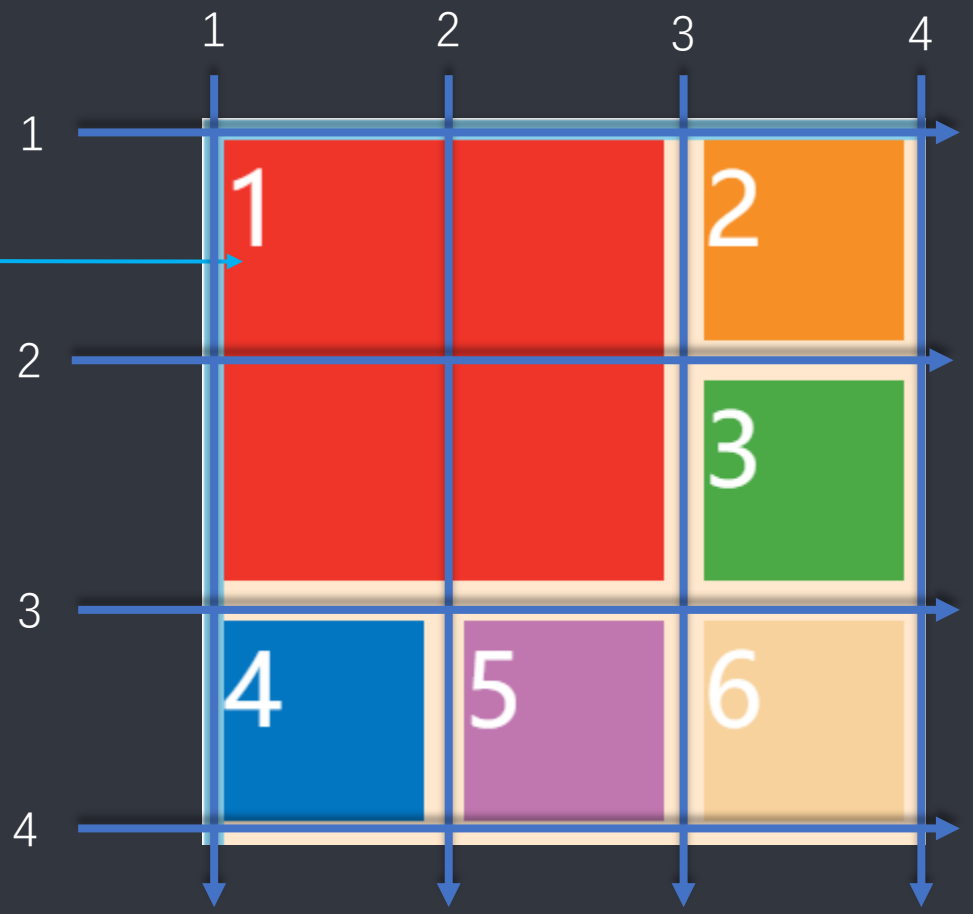
项目属性 grid-area

grid-area属性还可用作grid-row-start、grid-column-start、grid-row-end、grid-column-end的合并简写形式，直接指定项目的位置

grid-area: <row-start> / <column-start> / <row-end> / <column-end>;

```
grid-column-start: 1;
grid-column-end: 3;
grid-row-start: 1;
grid-row-end: 3;
```

```
grid-area: 1 / 1 / 3 / 3;
```



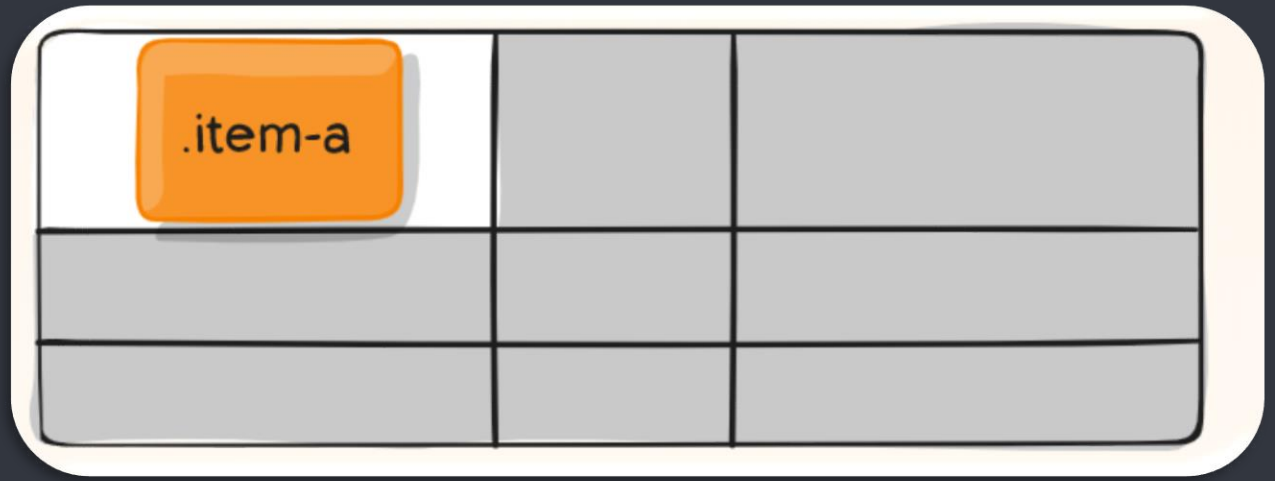
项目属性 justify-self / align-self / place-self

justify-self属性设置单元格内容的水平位置（左中右），跟justify-items属性的用法完全一致，
但只作用于单个项目 (水平方向)

align-self属性设置单元格内容的垂直位置（上中下），跟align-items属性的用法完全一致，
也是只作用于单个项目 (垂直方向)

```
justify-self: start | end | center | stretch;
```

```
justify-self: center;
```



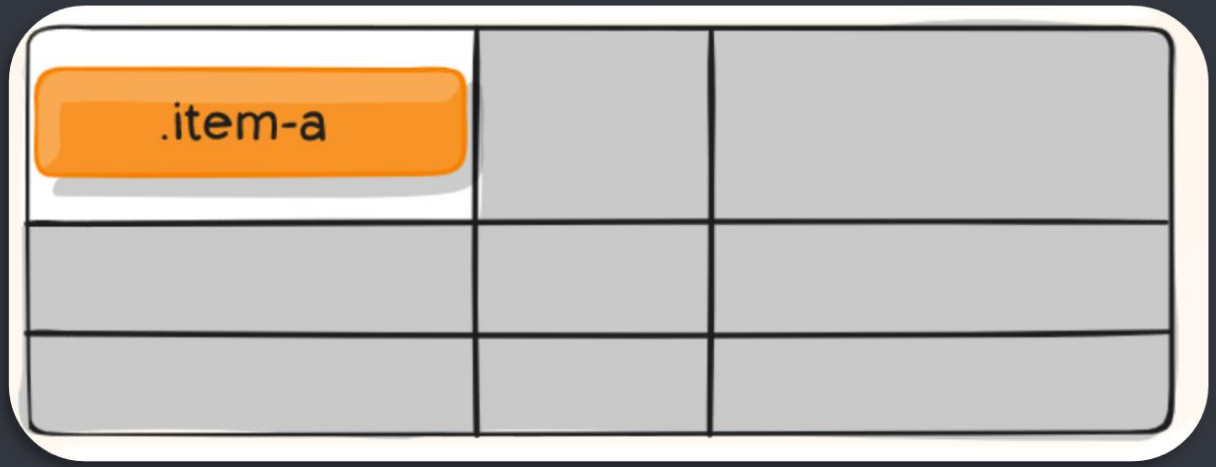
项目属性 justify-self / align-self / place-self

justify-self属性设置单元格内容的水平位置（左中右），跟justify-items属性的用法完全一致，
但只作用于单个项目 (水平方向)

align-self属性设置单元格内容的垂直位置（上中下），跟align-items属性的用法完全一致，
也是只作用于单个项目 (垂直方向)

```
align-self: start | end | center | stretch;
```

```
align-self: center;
```



place-self属性是align-self属性和justify-self属性的合并简写形式

```
place-self: center center;
```

Finally! End! Thank You!