# CSS盒子模型要点



# 1.盒子模型



Margin 外边框区域；

Border 边框区域；

Padding 内边框区域；

Content 内容区域；（含 height高，width宽）

默认情况下，设置高、宽的值，都是设置content内容区域；（显示对网页排版不适合，因为没有把几个边框区域加进来）；

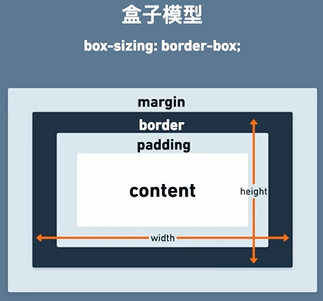
通过修改box-sizing:属性来设置大小属性;

box-sizing:content-box; 默认值；即按内容区域计算大小；

实际项目中，会把盒子大小改为box-sizing:border-box; 按边框区域设置盒子大小；

同时，要把HTML的元素margin的外边框设置为0；

这样，设置盒子高度、宽度时，就是实际盒子在页面中占用的空间大小；



# 2.文档流

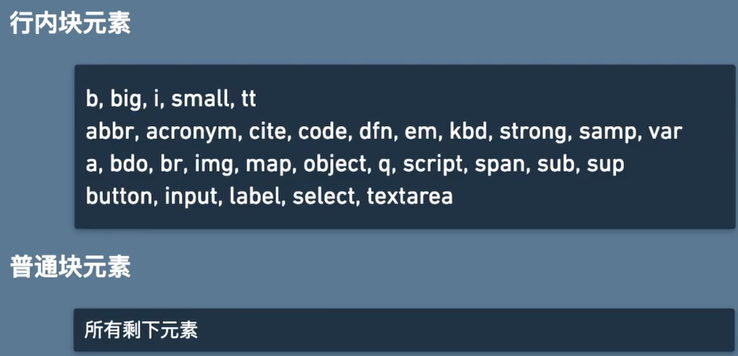
文档流：其实就是各种元素在文档中是如何排列的；



通常元素和盒子模型，在文档中是按两个方向排列“横向、纵向”；

## 2.1 方向类型：行内块、普通块

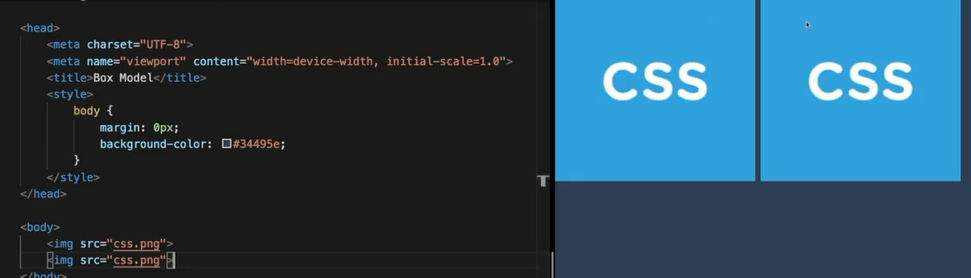
根据方向不同，分为：行内块类型、普通块类型；



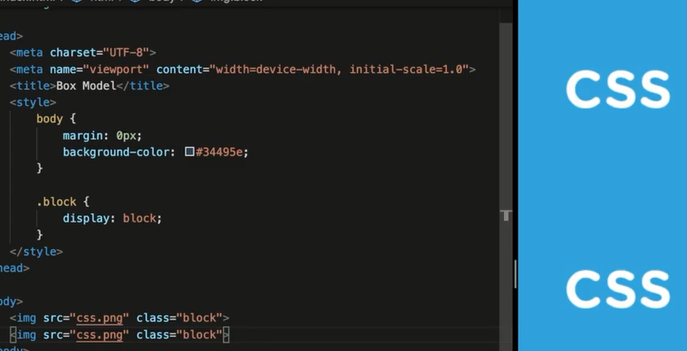
每一个HTML都有默认的块类型设置；display属性，就是指定方向模式；

### Img示例：

Img默认为行内块模型；



更改为block；变为纵向排列



# 3.定位&布局

虽然都与位置相关，但关注的点不同；

定位：关注元素本身的位置；

布局：关注容器区域的位置；

## 3.1元素定位的方式

元素本身的定位方式：1，坐标定位；2，浮动定位；

### 3.1.X 坐标定位

分为：

静态坐标 static

相对坐标 relative

绝对坐标 absolute

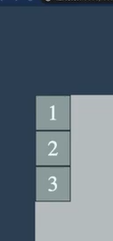
固定坐标 fixed



#### 3.1.1静态定位 static

按文档流的位置排列；

##### 示例



位置没有变化；

#### 3.1.2 相对定位 relative

相对坐标：指的是元素相对其所在文档流位置的坐标；

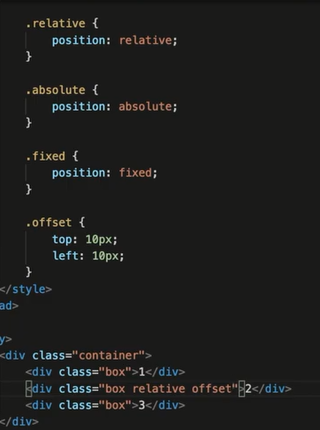
注意点：配置相对位置后，元素位置会发生偏移，但是，其原有在文档流中的位置仍然保留；



如图：左边是原有文档流的位置，右边是偏移后的位置，但原有的位置仍保留；

即：2号方块位置发生偏移，但是1号和3号位置不变；

##### 示例



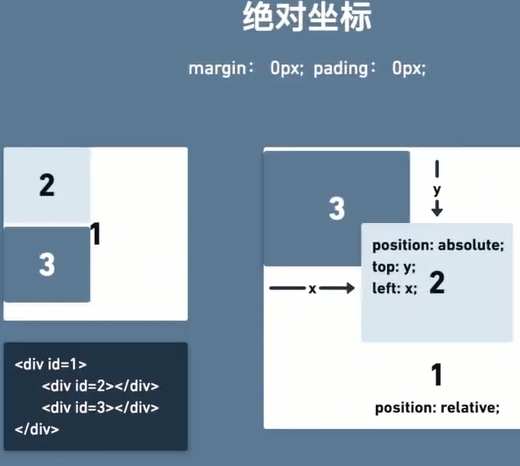
#### 3.1.3 绝对定位absolute

设定了绝对定位的元素，会脱离文档流；绝对定位的元素，会向上查找第一个**非静态定位的父级元素，作为自己的定位坐标**；

下图中：

图左边，2号方块和3号方块按常规文档流排列；

图右边，2号设置成为绝对定位后，位置发生改变；同时发现3号方块也发生了改变；这是因为2号方块脱离了文档流的位置，所以3号方快会占据2号方块的所在文档流的位置；

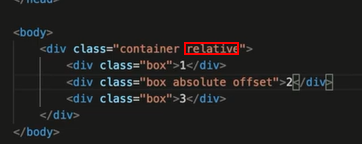


##### 示例1

2号方块脱离了容器，出现在窗口左上角；同时，3号方块占据了2号方块；

##### 示例2



绝对定位的元素，会向上查找第一个**非静态定位的父级元素，作为自己的定位坐标。**

#### 3.1.4 固定定位 fixed(窗口定位)

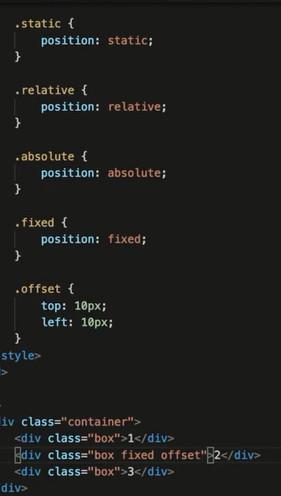
固定定位，可以叫作窗口<浏览器窗口>定位；

它是按窗口坐标定位，与文档流无关；

固定定位的元素同样会脱离文档流，而且它不会占用文档流中的位置。

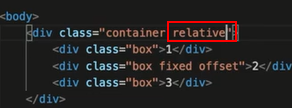


##### 示例1：



##### 示例2

因为是窗口定位，所以父元素的relative无效；



## 3.2 浮动定位

浮动元素 item

浮动容器 container

浮动定位，主要作用在于：容器与元素联动的方式进行定位；

主要应用场景是：图文排版，

### 3.2.1 浮动元素 float



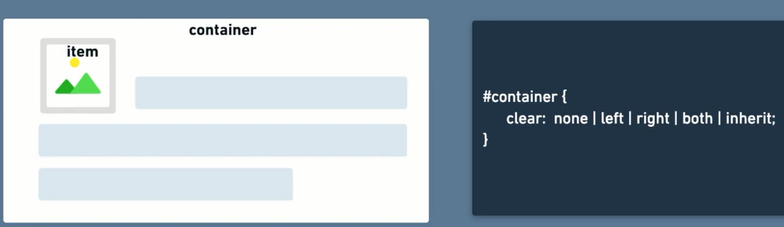
上图，其中图片所在行，文字是单行排列；

看一下浮动后的效果，此时图片被文字包围，空间利用更合理；



### 3.2.1 浮动容器 clear

用clear设置 容器内部元素的浮动位置



# 4.X 页面布局（flex与grid）

网页设计的布局分为：

Flex 弹性布局；

flex容器containter； flex元素 item;

Grid网格布局；

### 4.1 flex容器



设置 display:flex;

Css属性：

flex-direction排列方向

flex-wrap溢出处理

align-items对齐方式

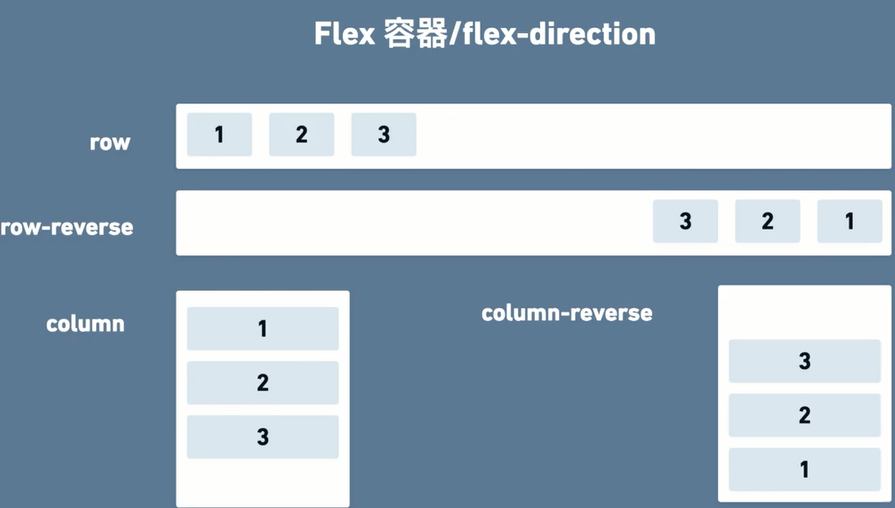
#### flex容器/flex-direction方向属性值

row横向（行）

row-reverse横向反向（行反向）

column纵向（列）

column-reverse纵向反向（列反向）

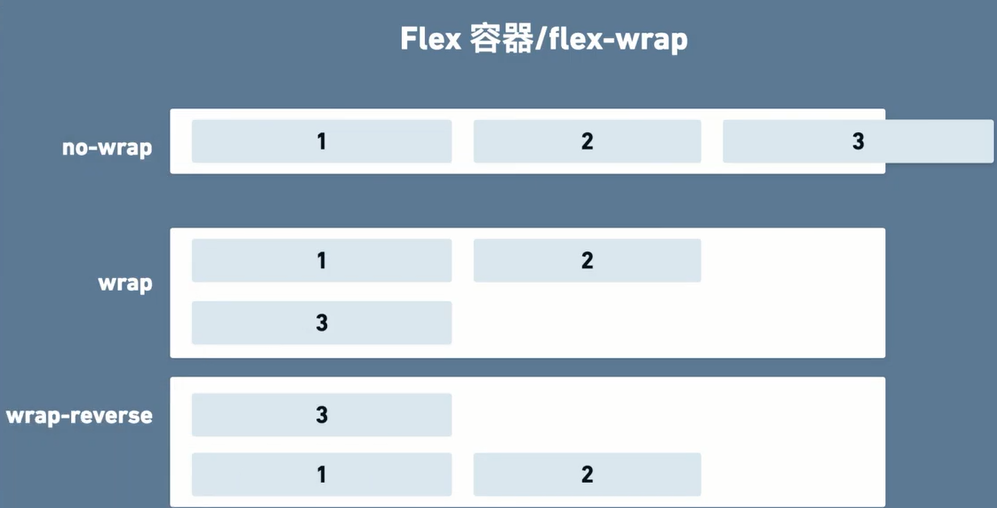


#### flex容器/flex-wrap溢出 属性值

no-wrap溢出容器宽度后不换行

wrap溢出容器宽度后，换行；

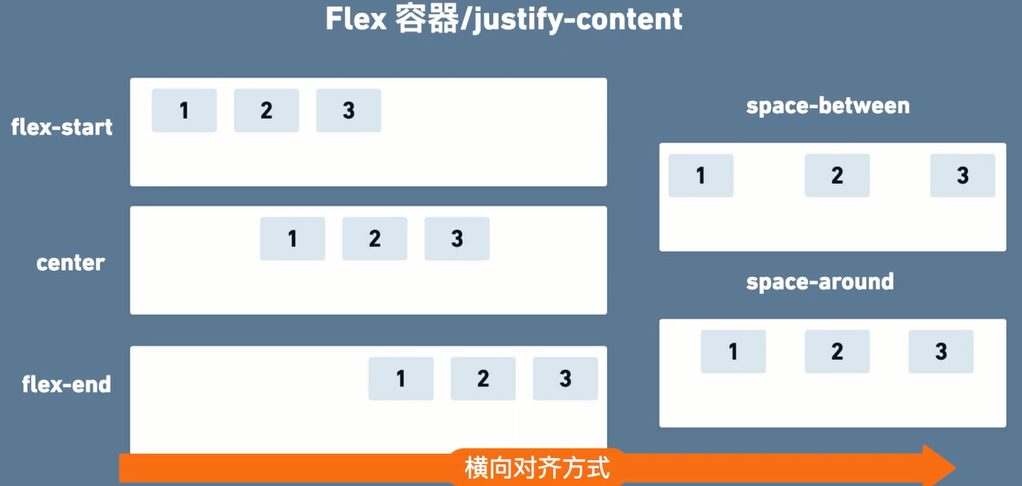
wrap-reverse溢出容器宽度，反向换行；



#### flex容器/对齐方向







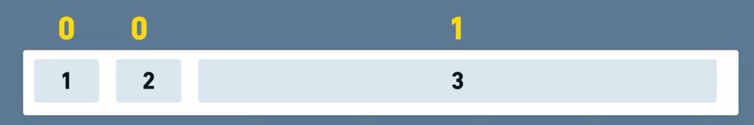
### 4.2 flex元素



#### Flex元素/flex-grow扩展 属性值

值可以设置为0或1；

值1，表示自动延长填满空间；



#### Flex元素/flex-shrink收缩 属性值

值可以设置为0或1；默认值为1；

值1，容器空间不足时，无素会等比例缩小；



#### Flex元素/flex-basis基础长度 属性值



number 像素比；

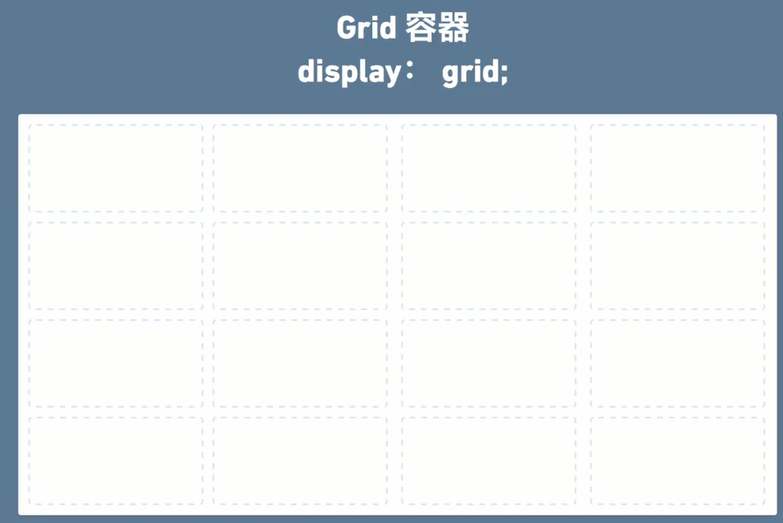


# 5.X 页面布局 Grid网格

Grid布局为分：

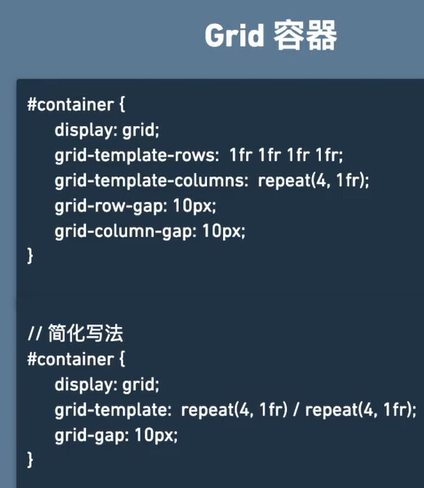
Grid容器container; Grid区域area;

## 5.1 Grid容器display:grid;



可以定义网格的行，列；及单元格间距；

##### 示例代码

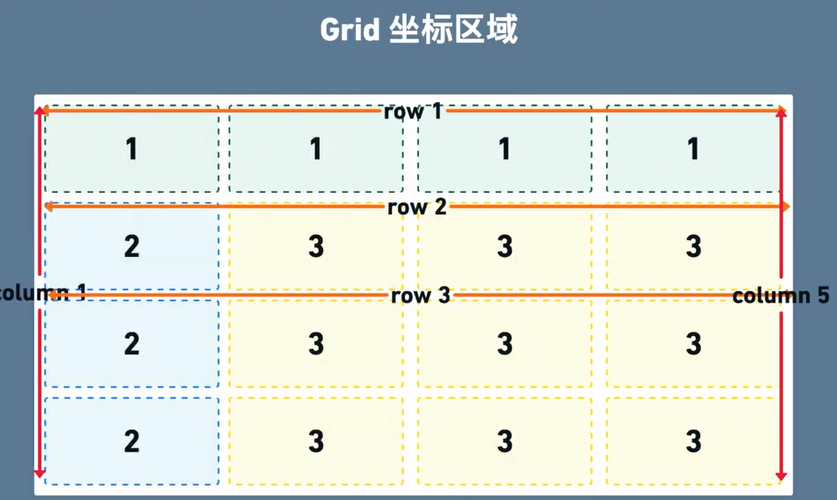


## 5.2 Grid区域

定义grid区域的方式有3种：



### 5.2.1 Grid坐标区域



开始行，开始列，结束行，结束列方式定义区域；

grid-column-start:2;

grid-column-end:5;

grid-row-start:2;

grid-row-end:5;

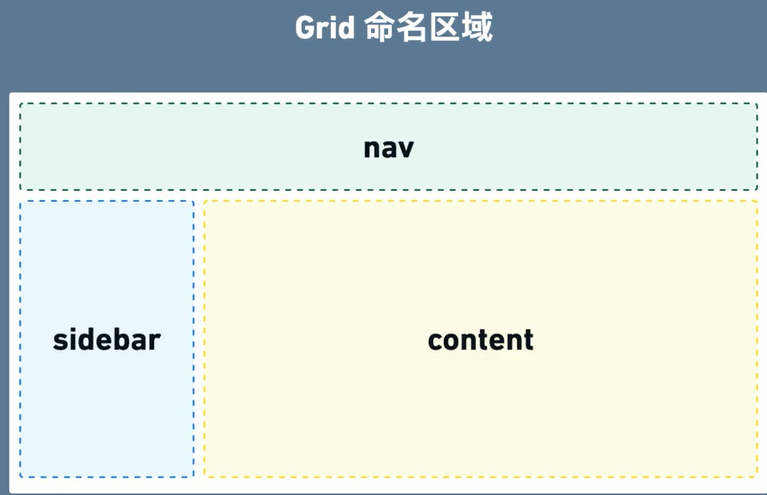
##### 示例1：

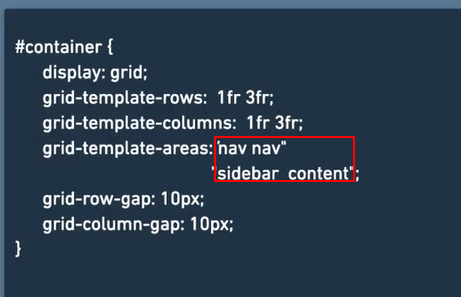


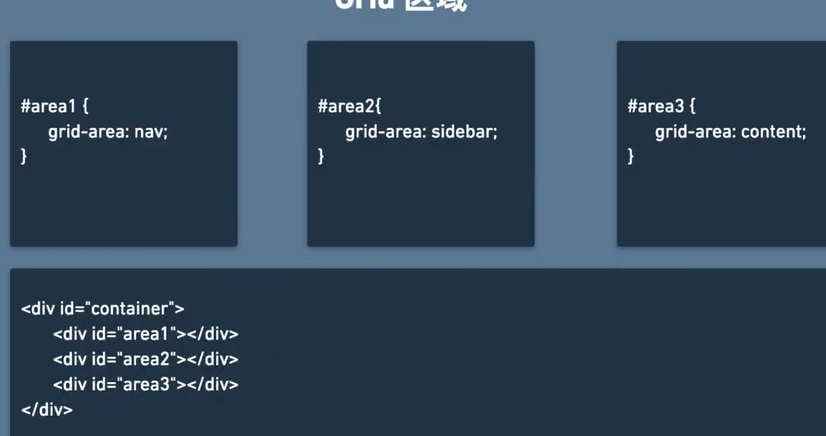
##### 示例2：



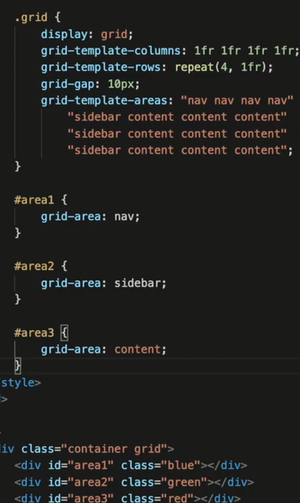
### 5.2.2 Grid命名区域



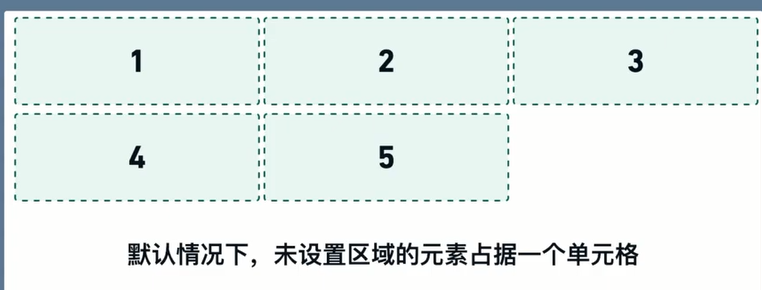




##### 示例1

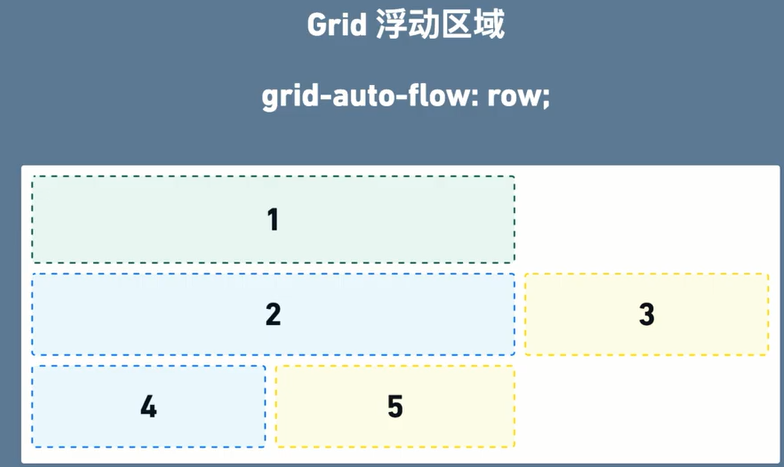
 

### 5.3 Grid浮动区载

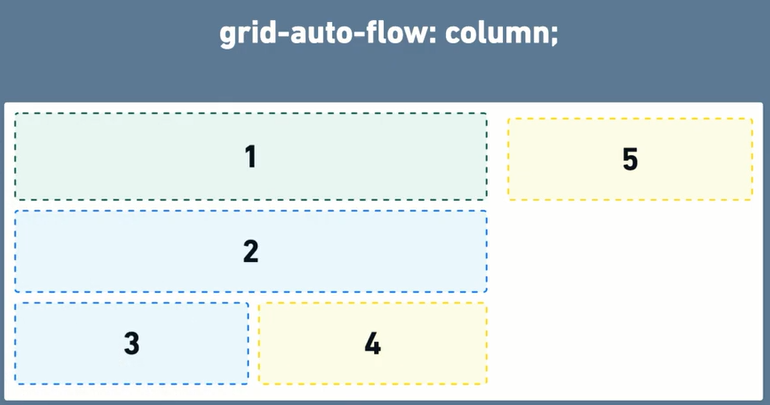


默认按横向方向排列；修改排列方向，则可以通过grid-auto-flow属性设置；

#### grid-auto-flow:row;



#### grid-auto-flow:column;



#### grid-auto-flow:dense;

更加合理利用空间；

