

# CSS 布局

从入门到工作：CSS 全解

# 版权声明

本内容版权属杭州饥人谷教育科技有限公司（简称饥人谷）所有。

任何媒体、网站或个人未经本网协议授权不得转载、链接、转贴，或以其他方式复制、发布和发表。

已获得饥人谷授权的媒体、网站或个人在使用时须注明「资料来源：饥人谷」。

对于违反者，饥人谷将依法追究 responsibility。

# 联系方式

如果你想要购买本课程

请微信联系 [xiedaimala02](#) 或 [xiedaimala03](#)

如果你发现有人盗用本课程

请微信联系 [xiedaimala02](#) 或 [xiedaimala03](#)

# 布局是什么

把页面分成一块一块，按左中右、上中下等排列

# 布局分类

- 两种

- ✓ 固定宽度布局，一般宽度为 960 / 1000 / 1024 px
- ✓ 不固定宽度布局，主要靠文档流的原理来布局

- 还记得吗

- ✓ 文档流本来就是自适应的，不需要加额外的样式

- 第三种布局

- ✓ 响应式布局
- ✓ 意思就是PC上固定宽度，手机上不固定宽度
- ✓ 也就是一种混合布局

# 布局的两种思路

- 从大到小

- ✓ 先定下大局
- ✓ 然后完善每个部分的小布局

- 从小到大

- ✓ 先完成小布局
- ✓ 然后组合成大布局

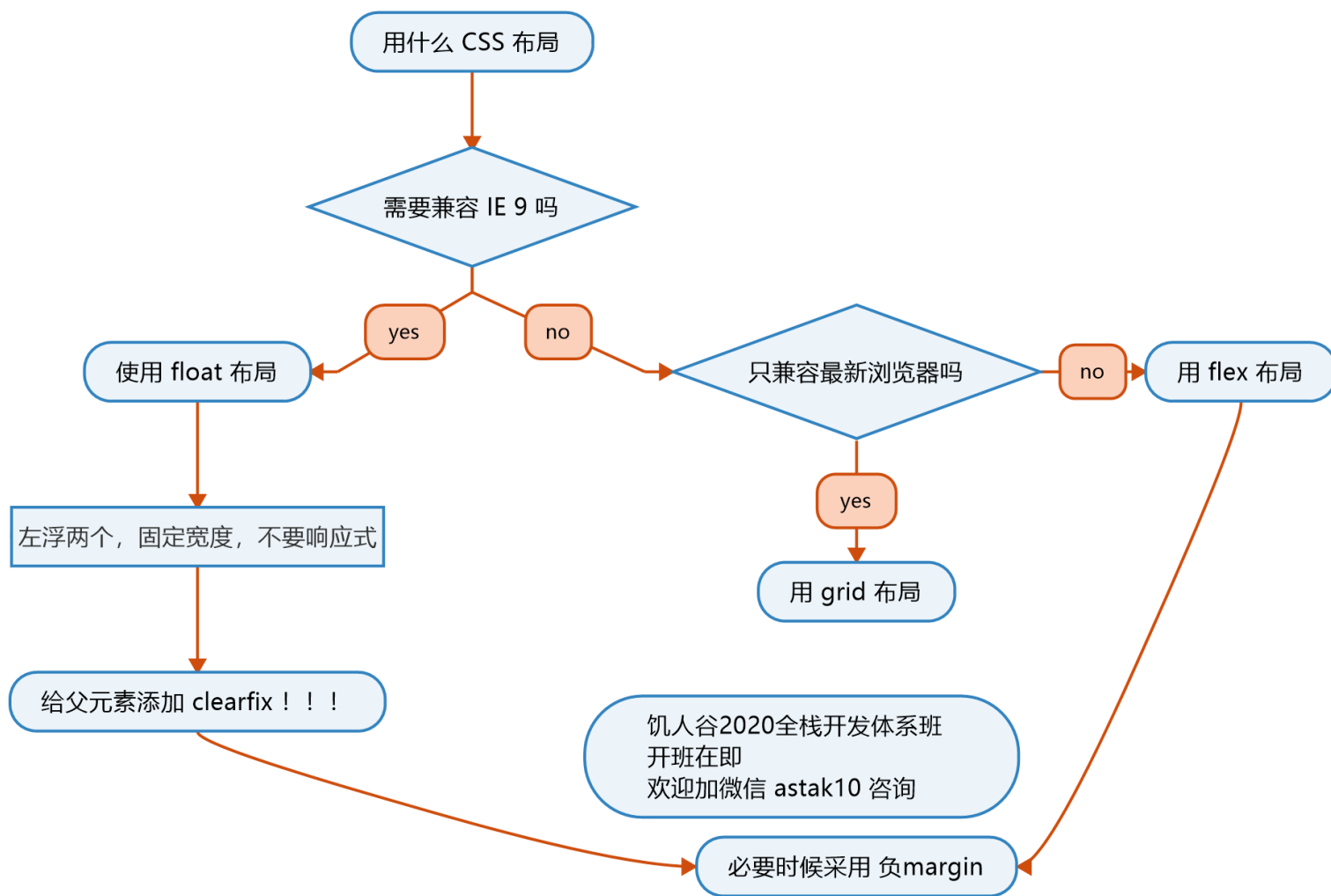
- 两种均可

- ✓ 新人推荐用第二种，因为小的简单
- ✓ 老手一般用第一种，因为熟练有大局观

# 布局需要用到哪些属性

不多哔哔，直接给你所有套路

# 一图流





# float 布局

- 步骤

- ✓ 子元素上加 float: left 和 width
- ✓ 在父元素上加 .clearfix (忘了加被我看到你就完了)

- 经验

- ✓ 有经验者会留一些空间或者最后一个不设 width
- ✓ 不需要做响应式，因为手机上没有 IE，而这个布局是专门为 IE 准备的
- ✓ IE 6/7 存在双倍 margin bug，解决办法有两个
- ✓ 一是将错就错，针对 IE 6/7 把 margin 减半
- ✓ 二是神来一笔，再加一个 display: inline-block
- ✓ 为什么可以这样？你问我，我问谁……

# 实践

## • 不同布局

- ✓ 用 float 做两栏布局（如顶部条）
- ✓ 用 float 做三栏布局（如内容区）
- ✓ 用 float 做四栏布局（如导航）
- ✓ 用 float 做平均布局（如产品展示区）
- ✓ 曾经淘宝的前端发明了双飞翼布局，不要学，已过时
- ✓ 代码

## • 经验

- ✓ 加上头尾，即可满足所有 PC 页面需求
- ✓ 手机页面傻子才用 float
- ✓ float 要程序员自己计算宽度，不灵活
- ✓ float 用来应付 IE 足以

# 问答

- 问

✓ 老师你说float适合ie，但是你刚刚是chrome啊，是我理解不对么

- 问

✓ 如果logo和nav的高度不固定。。还能用margin-top:5px吗。。

- 问

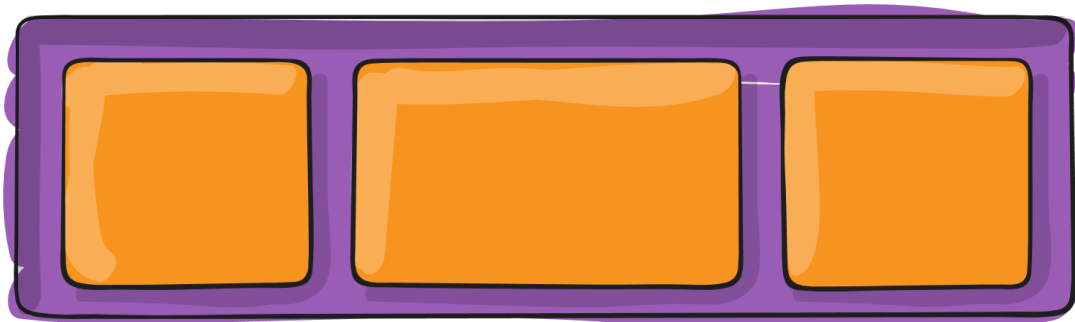
✓ 父div宽度800，四个子元素宽度分别191，不是还有border 1px吗 所以四个加起来的宽度就不止764了吧

# flex 布局

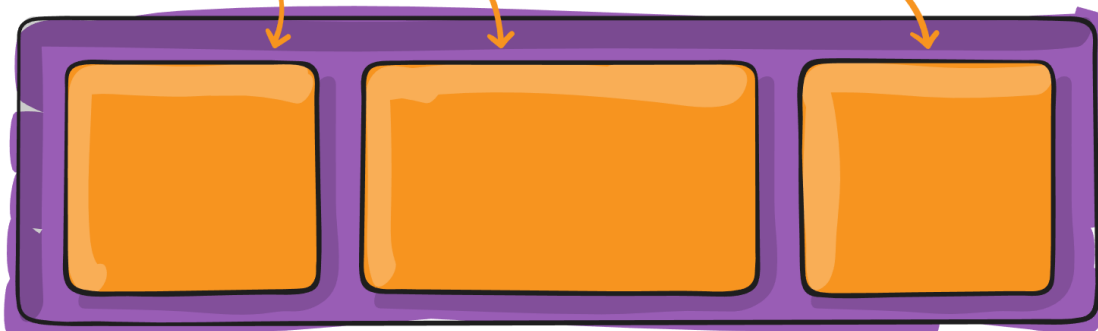
- 教程（来自 CSS Tricks）
- 把教程过一遍，然后忘掉
- 完成 [Flex青蛙游戏](#)
- 开始用 flex !

# 容器 container

container



items



# flex container 有哪些样式

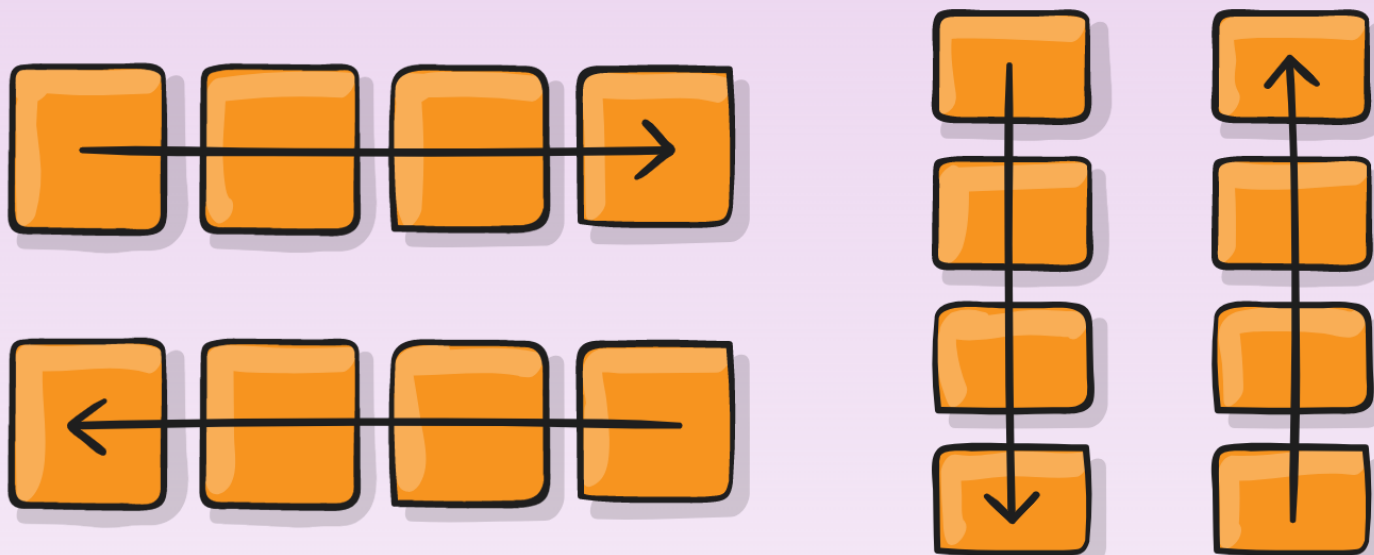
以下内容都是 container 的样式

# 让一个元素变成 flex 容器

CSS

```
.container {  
  display: flex; /* or inline-flex */  
}
```

# 改变 items 流动方向（主轴）

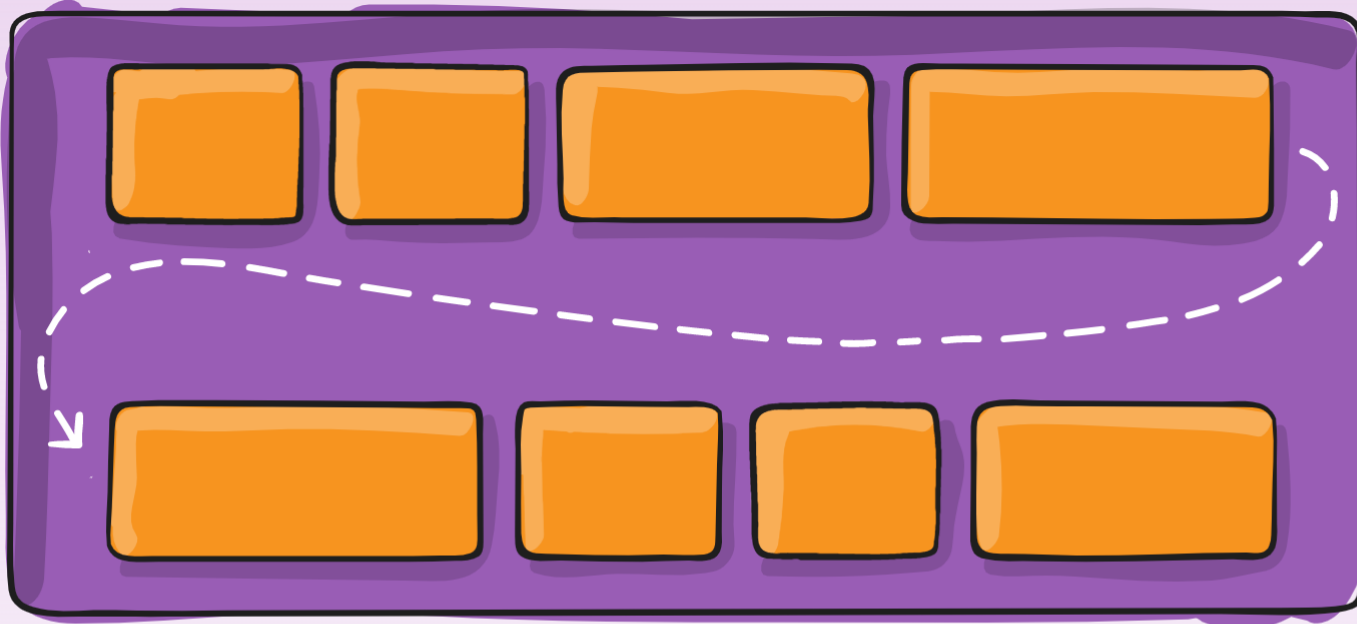


CSS

```
.container {  
  flex-direction: row | row-reverse | column | column-reverse;  
}
```



# 改变折行



CSS

```
.container{  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

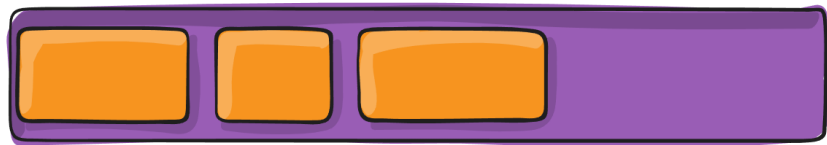
# 主轴对齐方式

默认主轴是横轴  
除非你改变了 flex-direction 方向

CSS

```
.container {  
  justify-content: flex-start | flex-end |  
}
```

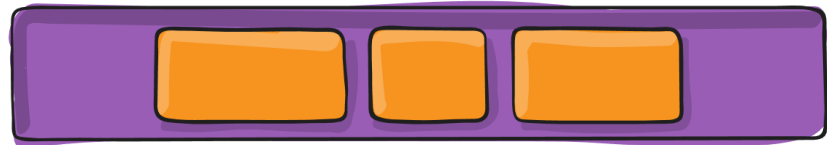
flex-start



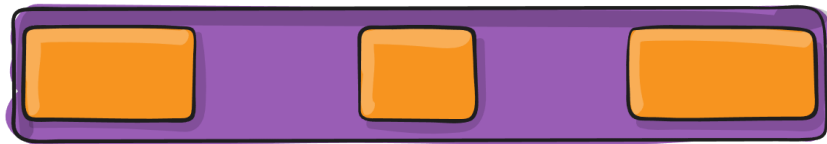
flex-end



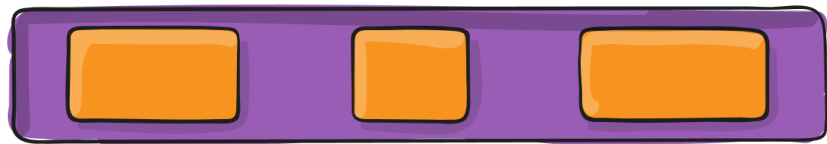
center



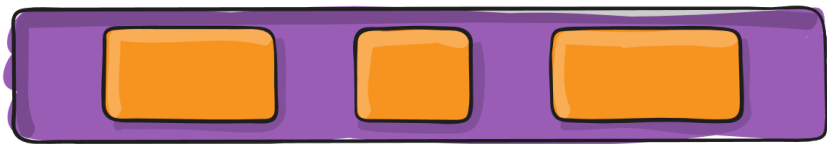
space-between



space-around



space-evenly



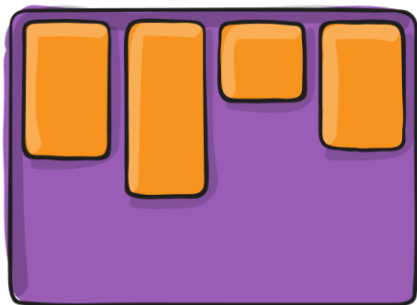
# 次轴对齐

默认次轴是纵轴

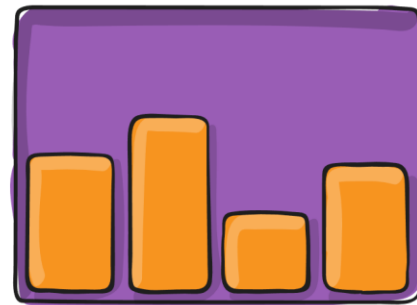
CSS

```
.container {  
  align-items: stretch | flex-start |  
}
```

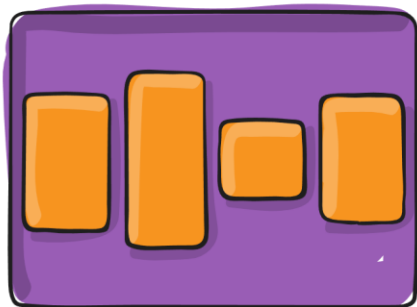
flex-start



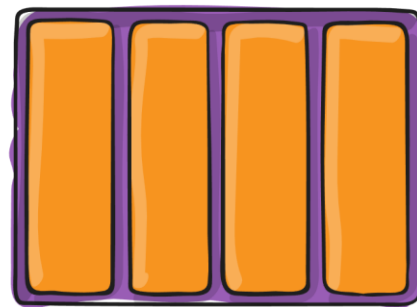
flex-end



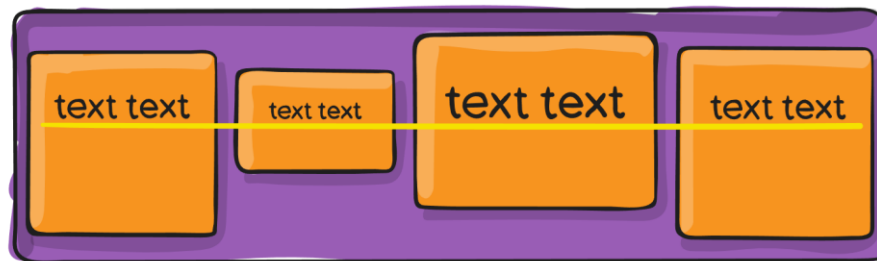
center



stretch



baseline



# 多行内容

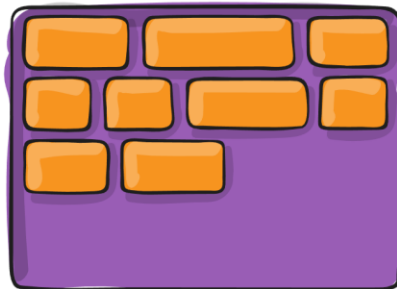
如何分布

CSS

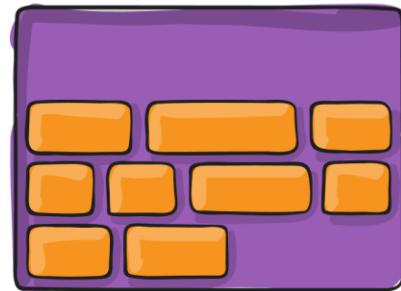
```
.container {  
  align-content: flex-start | flex-end |  
}
```

很少用到

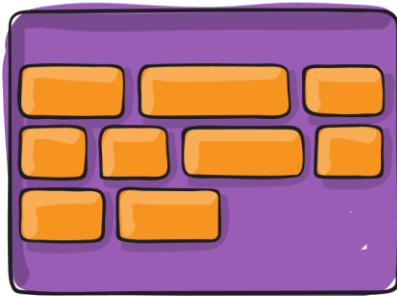
flex-start



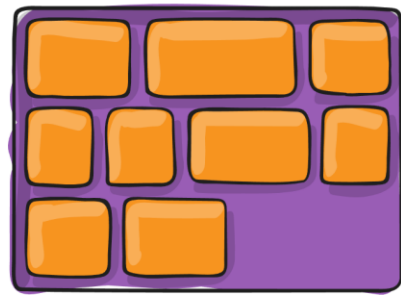
flex-end



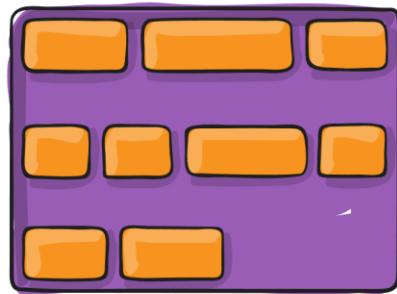
center



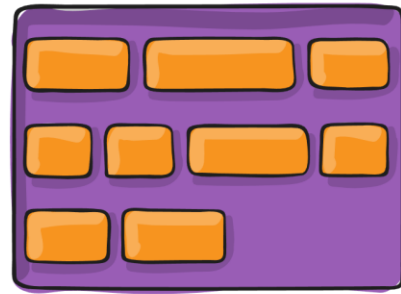
stretch



space-between



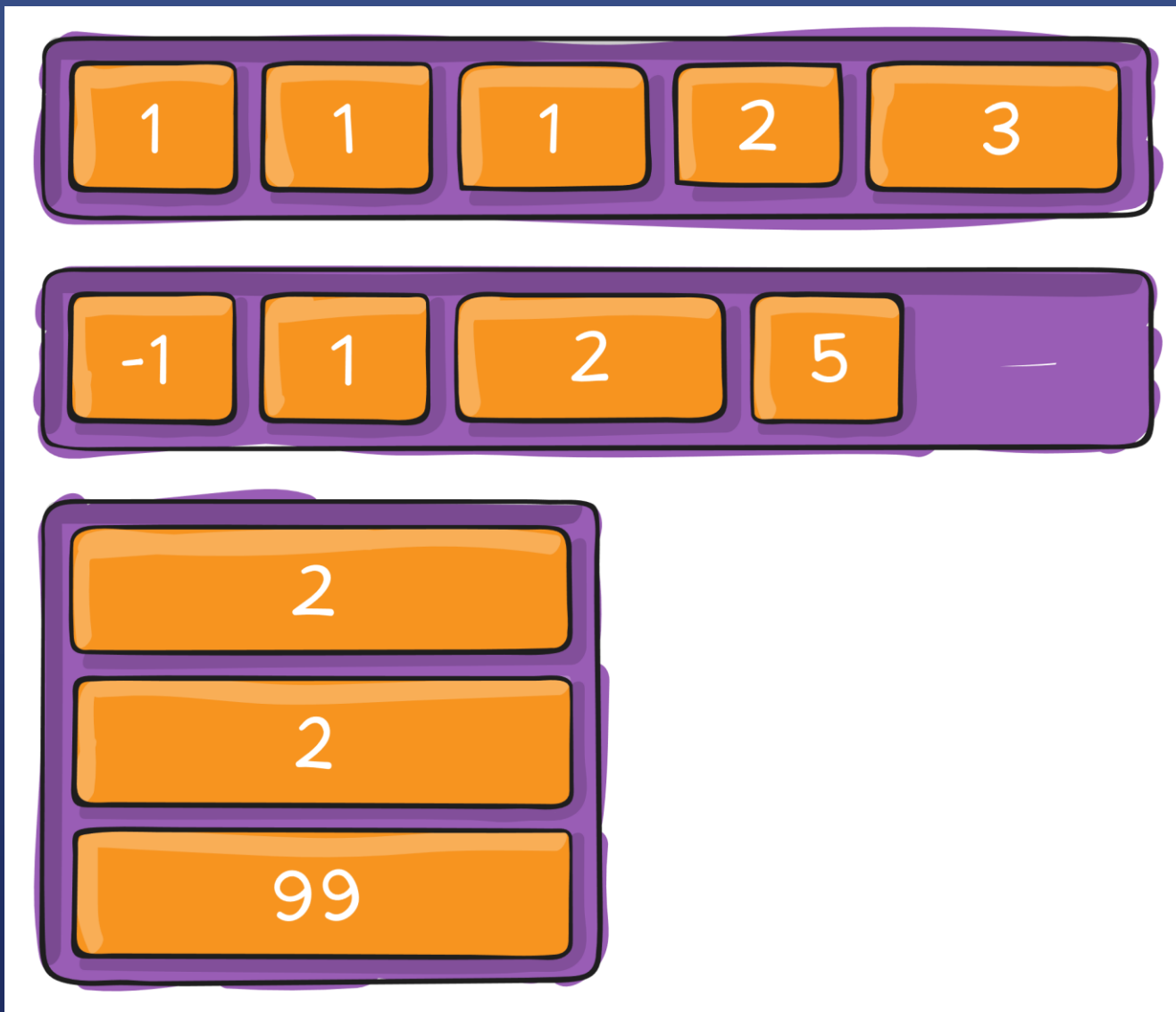
space-around



# flex item 有哪些属性

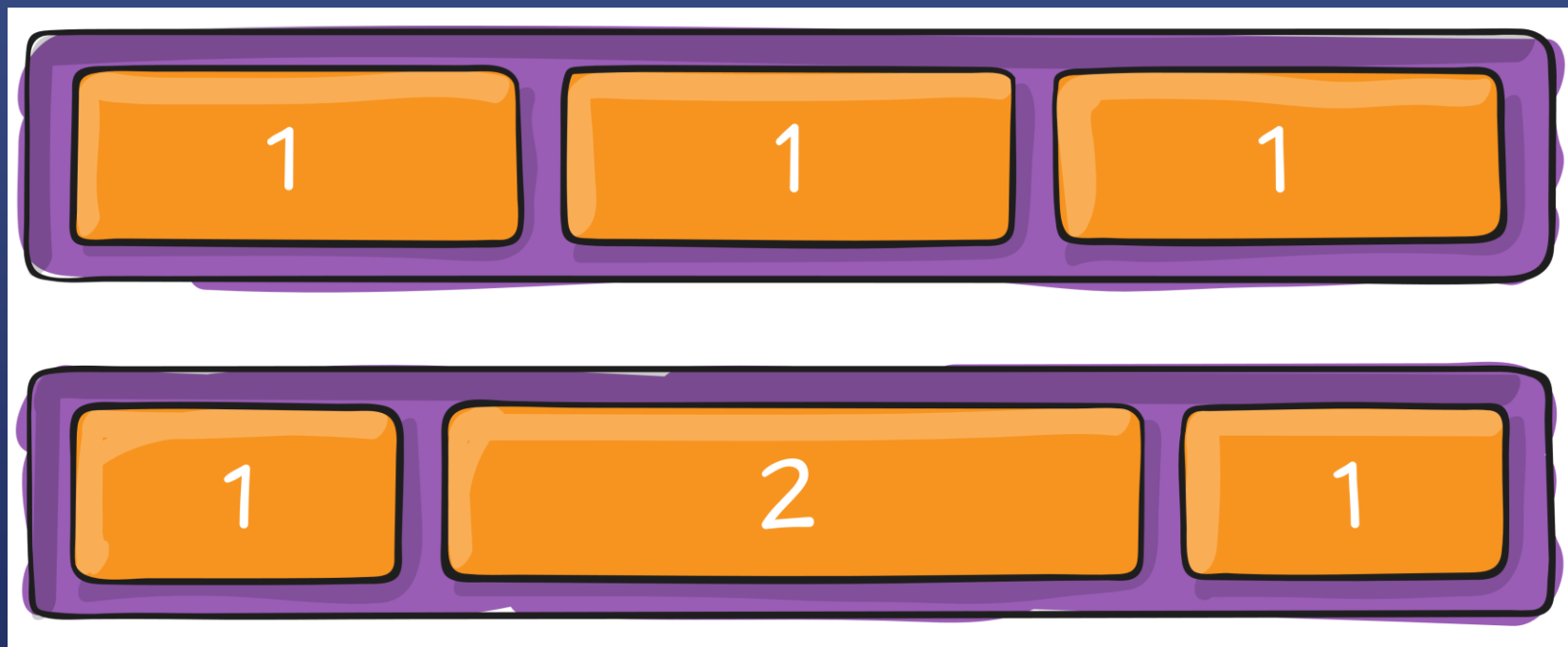
以下内容都是 item 的样式

# item 上面加 order



# item 上面加 flex-grow

控制自己如何长胖



# flex-shrink 控制如何变瘦

一般写 `flex-shrink: 0` 防止变瘦，默认是 1



# flex-basis 控制基准宽度

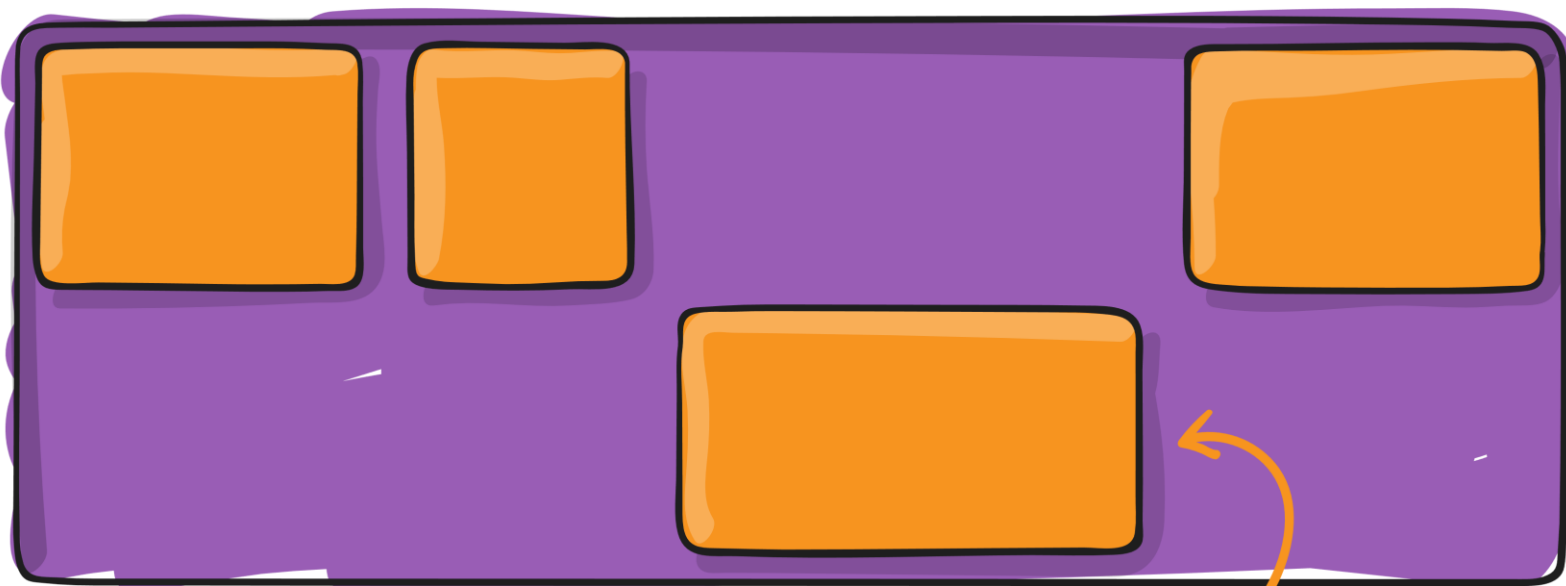
默认是 auto

flex: flex-grow flex-shrink flex-basis

缩写，空格隔开

# align-self 定制 align-items

flex-start



flex-end

# flex 所有属性讲完

忘掉它，开始干

# 重点

- 记住这些代码

- ✓ display: flex
- ✓ flex-direction: row / column
- ✓ flex-wrap: wrap
- ✓ justify-content: center / space-between
- ✓ align-items: center
- ✓ 工作中基本只用这些

# 实践

## • 不同布局

- ✓ 用 flex 做两栏布局
- ✓ 用 flex 做三栏布局
- ✓ 用 flex 做四栏布局
- ✓ 用 flex 做平均布局
- ✓ 用 flex 组合使用，做更复杂的布局
- ✓ 代码

## • 经验

- ✓ 永远不要把 width 和 height 写死，除非特殊说明
- ✓ 用 min-width / max-width / min-height / max-height
- ✓ flex 可以基本满足所有需求
- ✓ flex 和 margin-xxx: auto 配合有意外的效果

# 什么叫写死

- 写死

- ✓ width: 100px

- 不写死

- ✓ width: 50%
- ✓ max-width: 100px
- ✓ width: 30vw
- ✓ min-width: 80%
- ✓ 特点：不使用 px，或者加 min max 前缀

# 老板要我同时兼容 IE 和 手机

怎么办



# 前端戒律：必须先给设计稿

没有设计稿就自己画（用笔纸），老板同意再写代码

设计师只给一稿，让你做两套

怎么办

# 没设计稿，我做P

怼回去 或者 甩锅

两套界面  
必须要两套设计稿

底线不能退让

# 公司里没有设计师

怎么办

# 自己当设计师

设计稿不被老板肯定，就不要写代码

否则 996 就离你不远了

# 常用草图软件

- 跨平台

- ✓ Balsamiq
- ✓ Figma
- ✓ 墨刀
- ✓ Adobe XD

- 特点

- ✓ 直接干，不用学

# 如何在两套布局中切换

后面的课程单独讲，主要使用 @media 媒体查询



# Grid 布局

功能最强大的布局方案

# 二维布局用 Grid

一维布局用 Flex

以下内容来自 CSS Tricks

[A Complete Guide to Grid](#)

Grid 也分  
container 和 items

分别记忆

# 成为 container

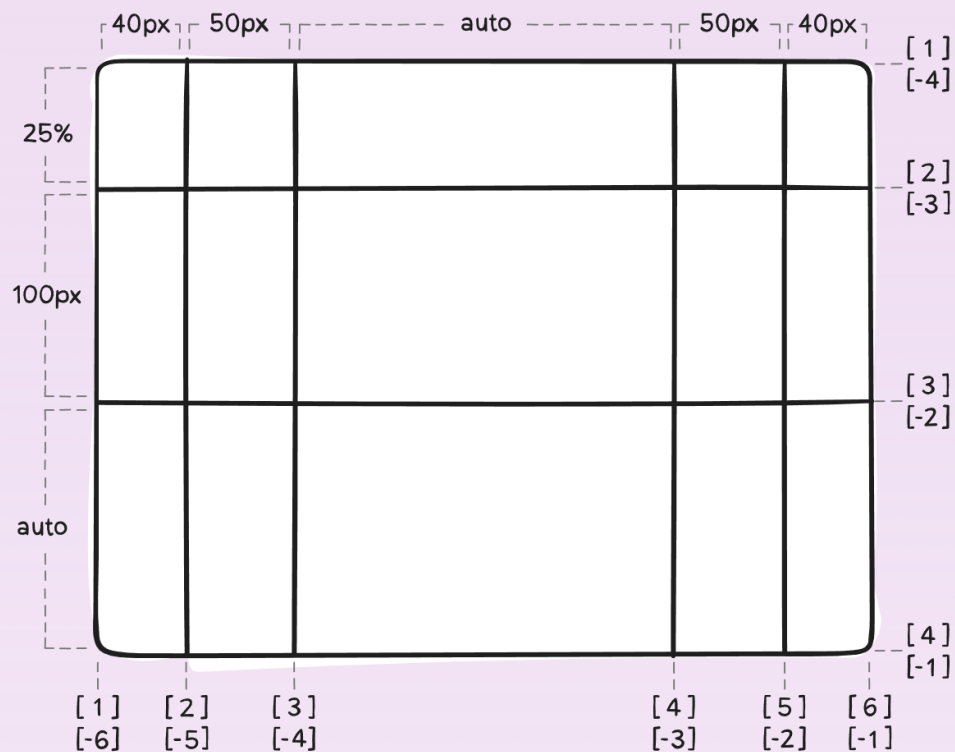
CSS

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

# 行和列

css

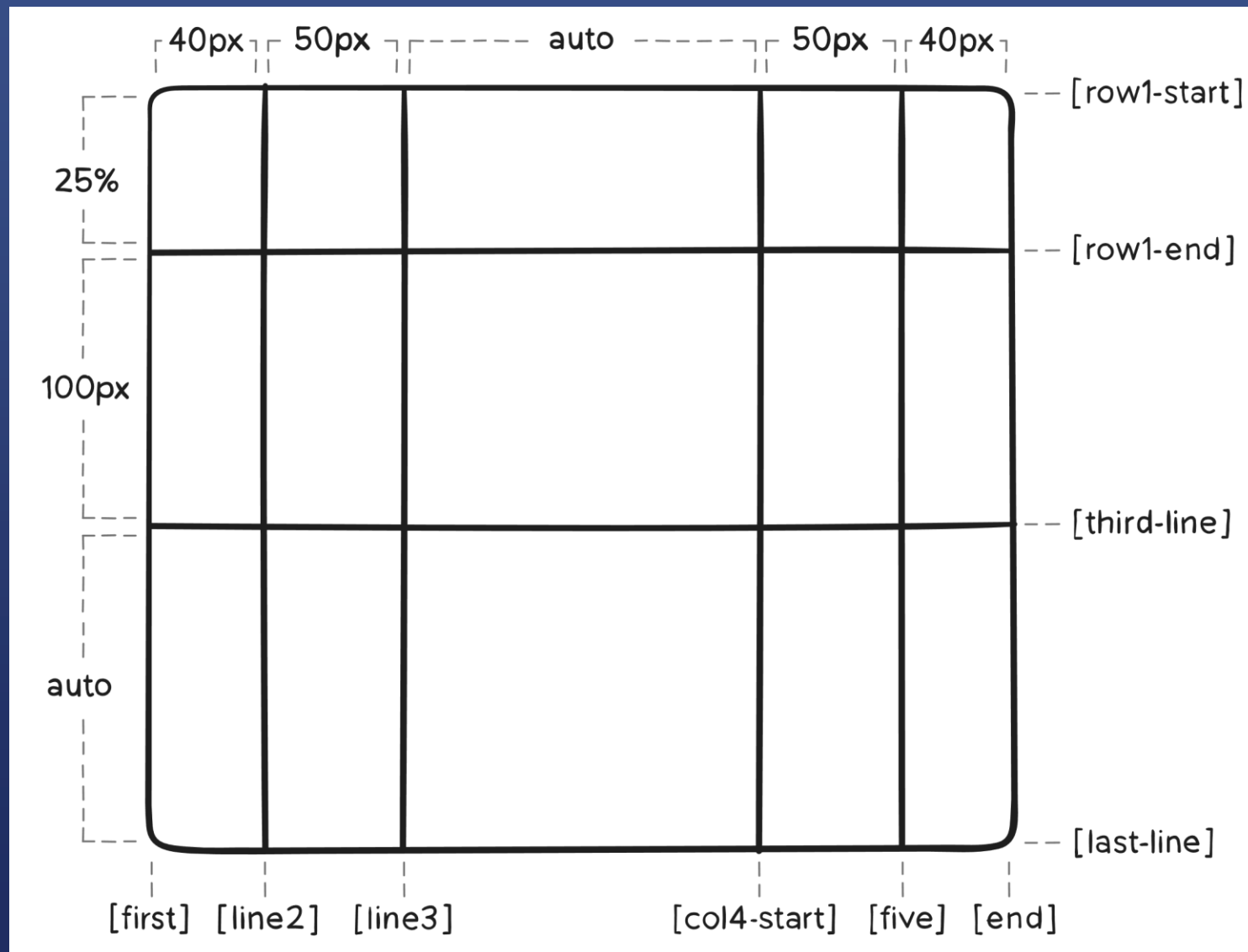
```
.container {  
  grid-template-columns: 40px 50px auto 50px 40px;  
  grid-template-rows: 25% 100px auto;  
}
```



# 你还可以给每条线取名字

```
.container {  
  grid-template-columns: [first] 40px [line2] 50px [line3] auto  
  [col4-start] 50px [five] 40px [end];  
  grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line]  
  auto [last-line];  
}
```

# 取名结果





# 取名有啥用

- item 可以设置范围

```
.item-a {  
  grid-column-start: 2;  
  grid-column-end: five;  
  grid-row-start: row1-start;  
  grid-row-end: 3;  
}
```

# fr - free space 巧记：份

CSS

```
.container {  
  grid-template-columns: 1fr 1fr 1fr;  
}
```

CSS

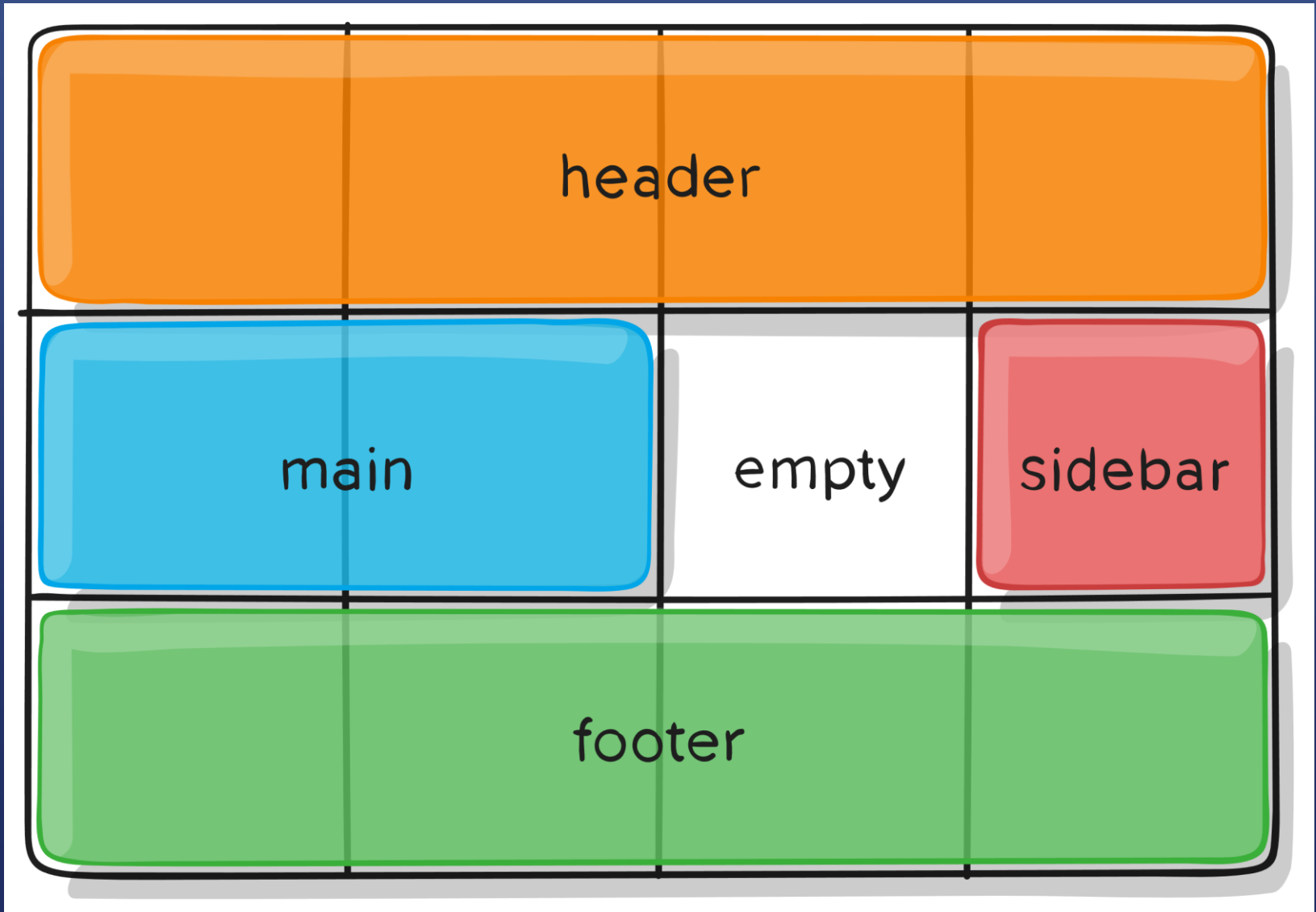
```
.container {  
  grid-template-columns: 1fr 50px 1fr 1fr;  
}
```

# 分区 grid-template-areas

CSS

```
.item-a {  
  grid-area: header;  
}  
  
.item-b {  
  grid-area: main;  
}  
  
.item-c {  
  grid-area: sidebar;  
}  
  
.item-d {  
  grid-area: footer;  
}  
  
.container {  
  display: grid;  
  grid-template-columns: 50px 50px 50px 50px;  
  grid-template-rows: auto;  
  grid-template-areas:  
    "header header header header"  
    "main main . sidebar"  
    "footer footer footer footer";  
}
```

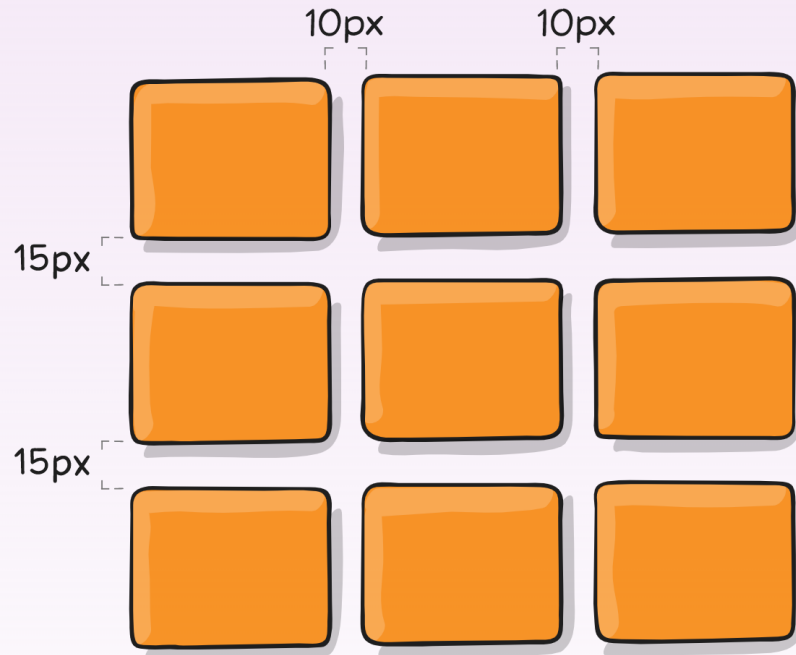
# 分区 grid-template-areas



# 空隙 gap

CSS

```
.container {  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: 80px auto 80px;  
  grid-column-gap: 10px;  
  grid-row-gap: 15px;  
}
```



# grid 属性太多

用到再说，用不到就算了 😊

# 实践

- 布局

- ✓ Grid 尤其适合不规则布局

- 经验

- ✓ 等到 Grid 普及了，前端对 CSS 的要求会进一步降低
- ✓ 目前你简单尝试一下 Grid 就可以了

# Grid Garden

<https://cssgridgarden.com/#zh-cn>



# 再见

下节课讲 CSS 定位和层叠