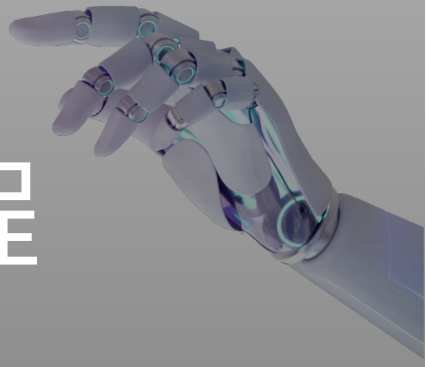


Web 前端编程

Web Front-end Programming

尹庆 Yin Qing
qyin@cuit.edu.cn

Flex 布局

为什么使用 Flex 布局

- 传统布局兼容性好, 但布局繁琐，移动端布局更难
- Flex (Flexible Box)：弹性布局，布局操作方便，布局简单，移动端应用广泛

🤔 如何快速实现如下效果?

为什么使用 Flex 布局

- Flex 布局：通过给**父盒子**设置`display: flex;`属性，控制**子盒子**的排列方式
  - Flex 容器 (flex container)：采用了 Flex 布局的元素，其子元素称为flex item
  - 父元素设置为 `flex` 布局后，子元素的 `float` 、 `clear` 和 `vertical-align` 属性将失效
  - 子元素既可以横向排列👉也可以纵向排列👇

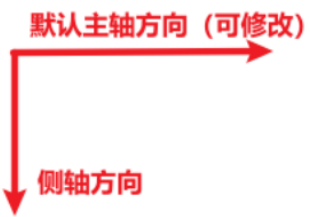
1 Flex 父项属性

常见父项属性

- `flex-direction`：设置主轴方向
- `justify-content`：设置主轴上子元素排列方式
- `flex-wrap`：设置子元素是否换行
- `align-content`：设置侧轴上子元素排列方式（wrap）
- `align-items`：设置侧轴上子元素排量方式（nowrap）
- `flex-flow`：复合属性，用于同时设置`flex-direction`和`flex-wrap`

1 Flex 父项属性

- Flex 布局分为主轴和侧轴方向
  - 默认主轴方向：x轴方向，水平向右
  - 默认侧轴方向：y轴方向，垂直向下
  - 子元素沿着**主轴**排列



- `flex-direction` 属性：设置主轴方向
  - `row` ： 水平向右 (默认值)
  - `row-reverse` ： 水平向左
  - `column` ： 垂直向下
  - `column` ： 垂直向上

💡始终有两个轴，一个轴设置成主轴，另一个轴就自动变成侧轴

1 Flex 父项属性

- `justify-content` 属性：设置**主轴**上子元素排列方式

```
- `flex-start`：从头部开始排列（默认值）
  > - 如果主轴是水平向右方向👉，则表示从左向右排列
- `flex-end`：从尾部开始排列
- `center`：在主轴居中对齐
  > - 如果主轴是水平方向👉，则表示水平居中
  > - 如果主轴是垂直方向垂直，则表示垂直居中
- `space-around`：平分剩余空间
- `space-between`：先两边贴边，再平分剩余空间
```

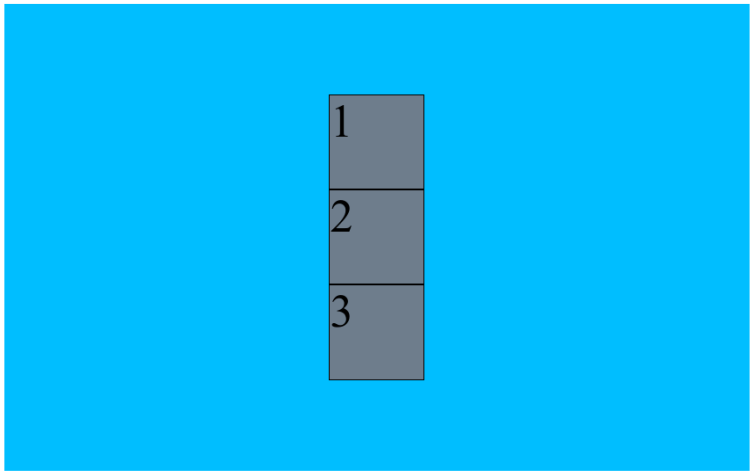
1 Flex 父项属性

- `align-items`：设置侧轴上子元素排量方式（nowrap模式）
  - `stretch` ：拉伸（默认值）
  - `flex-start` ：从头部开始排列
  - `flex-end` ：从尾部开始排列
  - `center` ：在主轴居中对齐

1 Flex 父项属性

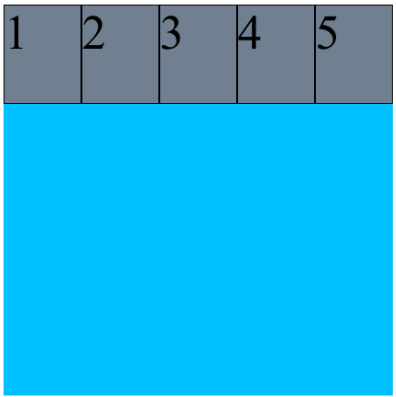
🐱课堂练习

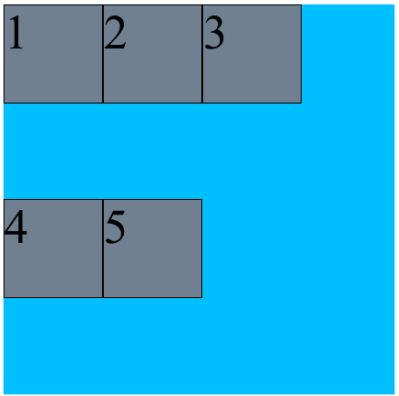
- 如何实现子盒子纵向排列+水平居中+垂直居中



1 Flex 父项属性

- `flex-wrap` 属性：设置子元素是否换行
  - `non-wrap`：Flex 项目默认排在一行上，再多也不换行（默认）
  - `wrap`：如果父盒子一行装不下子盒子，多余的子盒子自动换行





1 Flex 父项属性

- `align-content` 属性：多行模式下设置**侧轴**子元素排列方式（wrap时无效）
  - `flex-start` ：从头部开始排列
  - `flex-end` ：从尾部开始排列
  - `center` ：在主轴居中对齐
  - `space-around` ：平分剩余空间
  - `space-between` ：先两边贴边，再平分剩余空间
  - `stretch` ：拉伸（默认值）

- 💡 `wrap` 模式使用 `align-content` ， `nowrap` 模式使用 `align-items`

1 Flex 父项属性

- `flex-flow` 属性：`flex-direction` 和 `flex-wrap` 属性的复合属性

```
父盒子选择器 {
  flex-flow: row wrap;
}
```

2 Flex 子项属性

- `flex` 属性：子项目占的份数
- `align-self` 属性：控制子项自己在侧轴的排列方向
- `order` 属性：定义当前子项的排列顺序（前后顺序）

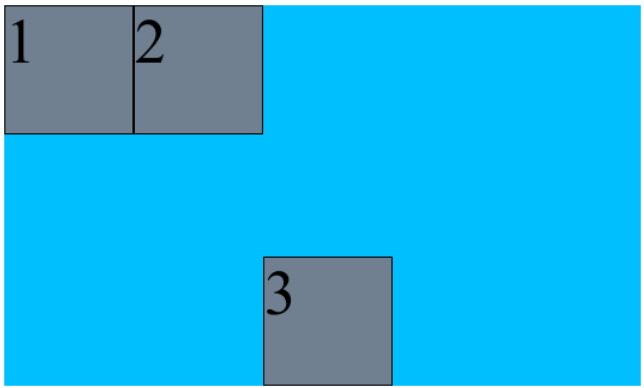
2 Flex 子项属性

- `flex`属性：“切蛋糕”，定义子项分配**剩余空间**，数值表示子项目占的份数
  - 默认值：0

```
子盒子选择器 {
  flex: 分数;
}
```

2 Flex 子项属性

- `align-self` 属性：控制子项自己在侧轴的排列方向
  - 允许单个flex项目与其他项目使用不一样的对齐方式，可覆盖父盒子的 `align-items` 属性
  - 默认值： `auto` ，表示继承父元素的 `align-items` 属性值



前端UI框架

做一名专业的 ~~CV~~ 前端工程师

Bootstrap

Bootstrap 简介

- Bootstrap：一套用于 HTML、CSS 和 JS 开发的开源工具集（前端框架）
  - 中文官方网站，官方网站

Bootstrap 简介

- 快速使用 Bootstrap，在 `index.html` 文件中添加如下代码
  - `<link>` 元素：导入 Bootstrap `CSS` 文件
  - `<scrip>` 元素：导入 Bootstrap `JavaScript` 文件

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstra
    <!-- 注意js放在head中添加defer属性启用异步加载 -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstra
  </head>
  <body>
```

1 Bootstrap 简介

1. 使用 Bootstrap 设置样式
   - 预定义样式
   - 自定义样式
2. 使用 Bootstrap 布局页面
   - 布局容器
   - 栅格系统

1 Bootstrap 设置样式

- 直接使用 Bootstrap 预定义样式（复制+粘贴）

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-danger">Warning</button>
<div class="btn btn-primary">注册</div>
<div class="btn btn-danger">启动</div>
```

1 Bootstrap 设置样式

- 二次开发：在Bootstrap 预定义样式基础上修改（注意优先级问题）

```
<div class="btn btn-primary login">启动</div>
```

```
.login {
  width: 100px;
  height: 50px;
}
```

2.1 布局容器

1. `container` 类：根据窗口大小设置不同的固定宽度，比如

- 大屏宽度： 1170px
- 中屏宽度： 970px
- 小屏宽度： 760px
- 超小屏（如手机）：100%

```
<div class="container">固定宽度容器</div>
```

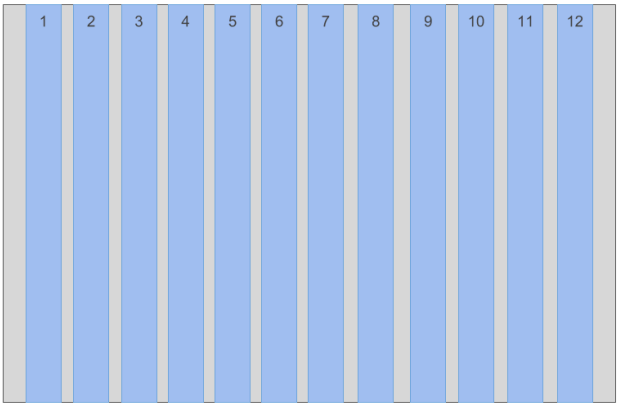
---

2. `container-fluid` 类：宽度占满窗口的容器，适用于单独做移动端开发

```
<div class="container-fluid">宽度占满容器</div>
```

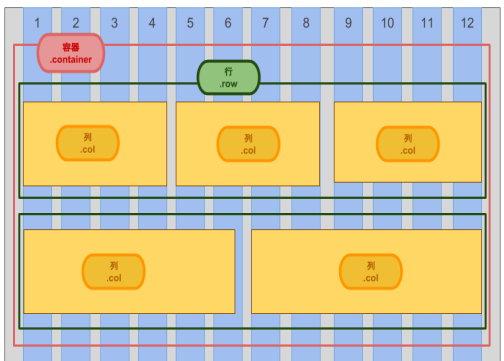
2.2 栅格系统 (grid systems)

- 12 列栅格系统：基于 CSS 实现“表格”式的布局
  - 一行默认被划分为 12 等份
  - 12 列格子可自由合并（类似表格的列合并 `colspan` 效果）



2.2 栅格系统 (grid systems)

- Bootstrap 栅格系统三大部分
  - `.container` 类：定义容器 (类似 `<table>` 元素)
  - `.row` 类：定义行（类似 `<tr>` 元素）
  - `.col` 类：定义列（类似 `<td>` 元素）



⚠ 注意：

- `.row` 必须放在 `container` 容器中
- `.col` 必须放在 `.row` 盒子中

2.2 栅格系统 (grid systems)

- `.col` 类：`class="col-类前缀-栅格数(列数)"`

```
<div class="container">
  <div class="row">
    <!-- lg 是 large（大屏幕）的缩写 -->
    <div class="col-lg-4">占4列</div>
    <div class="col-lg-3">占3列</div>
    <div class="col-lg-3">占3列</div>
    <div class="col-lg-2">占2列</div>
  </div>
</div>
```

2.2 栅格系统 (grid systems)

- `class="col-屏幕大小-栅格数(列数)"`：指定单元格在不同窗口大小下所占列数
  - 屏幕大小（`类前缀`）：指定针对的窗口大小（可组合）

	超小设备 <576px	平板 ≥576px	桌面显示器 ≥768px	大桌面显示器 ≥992px	特大桌面显示器 ≥1200px	超大桌面显示器 ≥1400px
容器最大宽度	None (auto)	540px	720px	960px	1140px	1320px
类前缀	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-	.col-xxl-

2.2 栅格系统 (grid systems)

- 列嵌套：一个列内可以继续划分若干份小列（小列数<12）


```
<div class="container">
  <div class="row">
    <div class="col-lg-3">
      <div class="row">
        <div class="col-lg-6">a</div>
        <div class="col-lg-6">b</div>
      </div>
    </div>
    <div class="col-lg-3">2</div>
  </div>
</div>
```

2.2 栅格系统 (grid systems)

- `.offset-屏幕大小-偏移量` 类：列偏移，可实现左右对齐、居中等效果

```
<div class="container">
    <!-- 1. 实现左右对齐, 偏移量 = 12-3-3 -->
    <div class="row">
      <div class="col-lg-3">左侧</div>
      <div class="col-lg-3 offset-lg-6">右侧</div>
    </div>
    <!-- 2. 实现一个盒子水平居中, 偏移量 = (12-8)/2  -->
    <div class="row">
      <div class="col-lg-8 offset-lg-2">居中</div>
    </div>
</div>
```