Bootstrap 讲义

预习讲义如有错别字和输入错误，请及时与老师联系进行更改，谢谢！

一、Bootstrap的简介和作用

1.Bootstrap的简介

- Bootstrap是由twitter公司设计师基于html，css，js开发的简洁、开源、强大、优雅的UI框架。
- UI框架就是能够快速构建html架构的框架。
- 内置了大量的css类，供元素使用。
- Bootstrap由一个css库和一个js库组成，因为在整个ui框架中还会有许多动态需效果需要js来完成，因此，Bootstrap在动态部分是依赖jquery一个js类库完成的。 `Bootstrap = jquery + css`
- 【注意】不需要背，查询文档知道每个类的作用即可

2.准备工作

【提示】讲解的是Bootstrap v4.0 版本，到5版本会有一些改变，大家注意查询手册，比如左右变成start和end
【查看】下载Bootstrap之后会得到很多文件，有很多不需要，为大家精简了一个版本，1个css和3个js。
【拷贝】将css文件放入一个新建的css文件夹下，3个js文件放到一个新建的js文件夹下
【操作】将boot.json内的文本创建全局代码片段boot
【安装】在vscode中安装 `HTML CSS Support` 和 `IntelliSense for CSS class names in HTML` 插件，可以提示一部分css

2.安装sass【作业】

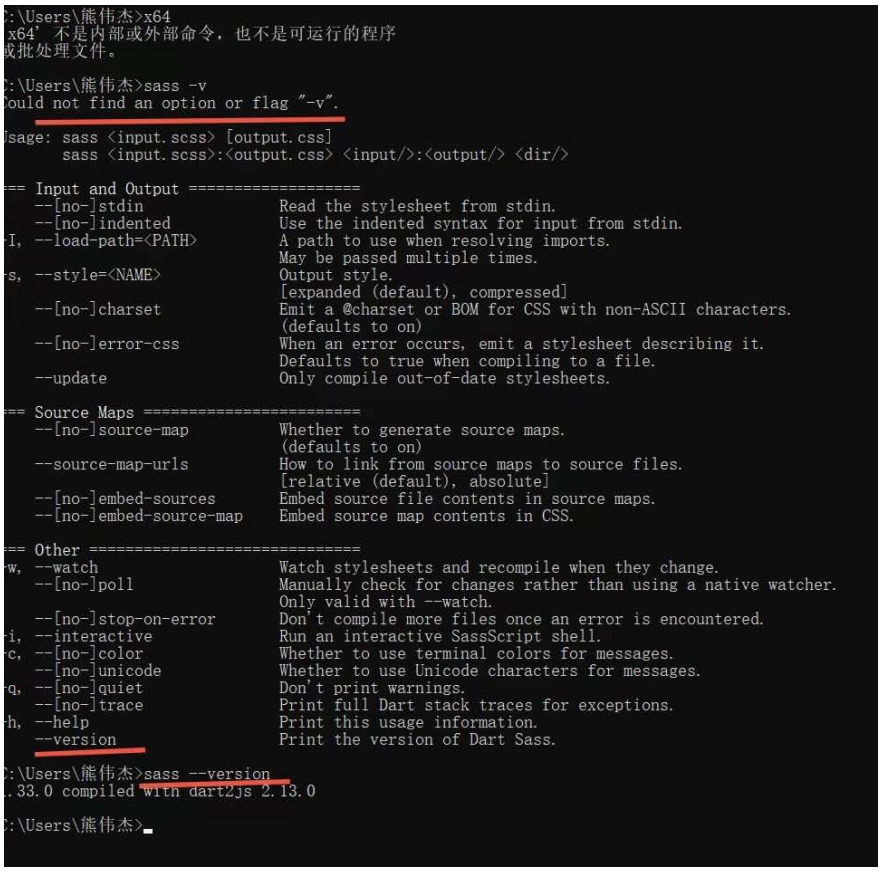
- 安装插件：Sass、Sass/Less/Stylus/Pug/Jade/Typescript/Javascript Compile Hero Pro
- 查看视频号内《sass的更多安装方法与编译方法》和《依赖ruby的安装方法》
- 先看一遍再进行安装
- 安装后编译成功，会出现css文件

（1）依赖ruby安装sass

- 依赖ruby
  - 下载rudy，网盘软件下载或网上下载
  - 查看自己电脑是32位还是64位，按照实际情况安装x86或者x64
  - 查看ruby是否安装成功，打开 `cmd` ,输入 `ruby -v`
- 安装sass
  - mac： `sudo gem install sass`
  - windows: `gem install sass`
  - 安装完之后在 `cmd` 中检查 `scss -v` 或者按照提示 `scss -version` 出现版本号就可以了
- 文件编译和监听
  - 使用终端或vscode终端在要编译的文件夹下打开，注意路径
  - 文件夹监听 `sass --watch scss:css`
  - 文件监听 `sass --watch 1.scss:1.css`
  - 可以使用简写 `sass -w scss:css`
  - 任何文件夹不能出现中文字符

（2）使用npm安装sass

- 查看是否安装npm，打开 `cmd` ,输入 `npm -v`
- 打开 `cmd` ,输入 `npm install -g sass` ， 安装成功后检查 `sass -v`
- 如果上面的方法因为网的问题没有安装成功，可以使用淘宝镜像
  - 打开 `cmd` ,输入 `$ npm install -g cnpm --registry=https://registry.npm.taobao.org`
- 再使用cnpm安装sass `$ cnpm install sass`
- 文件编译和监听
  - 使用终端或vscode终端在要编译的文件夹下打开，注意路径
  - 文件夹监听 `sass --watch scss:css`
  - 文件监听 `sass --watch 1.scss:1.css`
  - 可以使用简写 `sass -w scss:css`
  - 任何文件夹不能出现中文字符

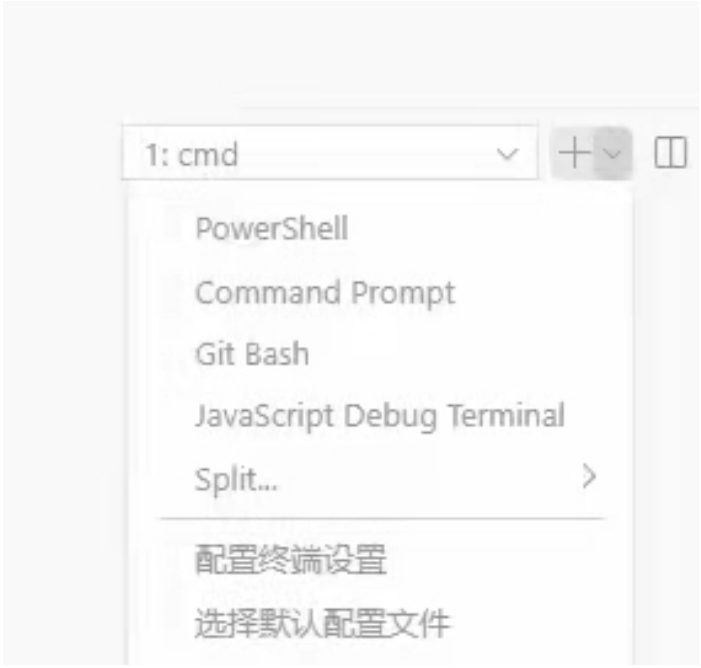


（3）编译出现的错误【重点】

编译中会出现很多错，比如编译之后没有任何效果，报错等

- sass项目的编译整个项目不允许有任何中文字符的文件夹
- 不运行可能是你的安全助手或者防火墙阻止了ruby，注意弹窗，不要阻止
- 路径写错，注意路径的关系
- 使用windows自带的cmd进行编译
- powershell编译报错，在终端切换成command prompt 也就是cmd编译，或者直接使用使用windows自带的cmd进行编译

```
PS D:\web\06CSS\DAY08> sass --watch scss:css
sass : 无法加载文件 C:\Users\web\AppData\Roaming\npm\sass.ps1，因为在此系统上禁止运行脚本。有关详细信息，请参阅 https:/go.
microsoft.com/fwlink/?LinkID=135170 中的 about_Execution_Policies。
所在位置 行:1 字符: 1
+ sass --watch scss:css
+ ~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS D:\web\06CSS\DAY08>
```



二、布局容器

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
.container	100%	540px	720px	960px	1140px
.container-sm	100%	540px	720px	960px	1140px
.container-md	100%	100%	720px	960px	1140px
.container-lg	100%	100%	100%	960px	1140px
.container-xl	100%	100%	100%	100%	1140px
.container-fluid	100%	100%	100%	100%	100%

1.定宽容器 container

container的css,设置了一些基本样式以及响应式布局，会根据屏幕宽度变更宽度，并且居中，定死最大宽度。对比一个普通的div

点

在不同的屏幕下都定义了最大宽度

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

2.变宽容器 container-fluid

宽度随着body的变化而改变

```
<div></div>
<div class="container"></div>
<div class="container-fluid"></div>
```

三、工具类

1.颜色和排版

常用颜色：

- primary - 重要
- success - 成功
- danger - 危险
- warning - 警告
- info - 信息
- light - 亮色
- dark - 深色
- secondary - 浅灰
- white - 白色
- transparent - 透明色

常用排版字号

- .h1 ~.h6 让元素的文字呈现出标题效果
- .display-1~.display-4 一种更大，更自以为是的标题样式

(1)文字颜色

- .text-primary
- .text-success

- `.text-white`

(2)背景颜色

- `.bg-primary`
- `.bg-dark`

```
<div class="box text-danger">爱生活，爱东哥</div>
<div class="box bg-primary">爱生活，爱东哥</div>
<div class="box text-white bg-success">爱生活，爱东哥</div>
```

2.文本样式

(1)文本对齐

- `.text-left` 左对齐
- `.text-center` 居中对齐
- `.text-right` 右对齐


```
<p class="text-left">爱生活，爱东哥</p>
<p class="text-center">爱生活，爱东哥</p>
<p class="text-right">爱生活，爱东哥</p>
```

(2)字体粗细和斜体

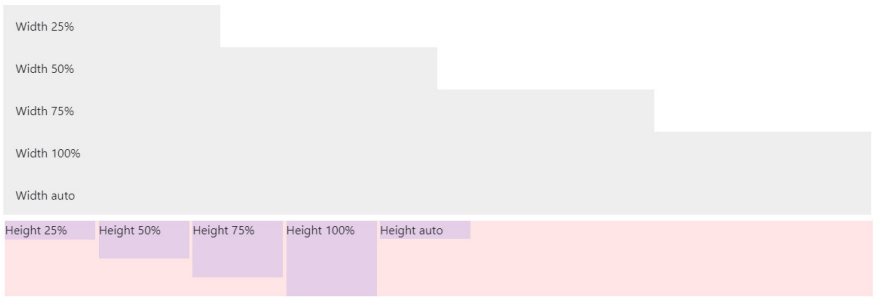
- `.font-weight-bold` 加粗
- `.font-weight-light` 细体
- `.font-italic` 斜体

```
<p class="font-weight-bold">爱生活，爱东哥</p>
<p class="font-weight-light">爱生活，爱东哥</p>
<p class="font-italic">爱生活，爱东哥</p>
```

3.调整大小

- 相对于父级的百分比，包括 25%、50%、75%、 100%
- w:宽度 `.w-50` 是父元素宽度的50%

- h:高度 `.h-25` 是父元素高度的50%
- 缺点：这能设置固定的几种百分比




```
.box {width: 400px;height: 400px;}
  <div class="container">
    <div class="bg-success box">
      <div class="w-50 h-50 bg-danger"></div>
    </div>
  </div>
```

4.边框

(1)边框属性

`.border` 基类属性

(2)边框方向

- `.border` 全边框
- `.border-left` 左边框
- `.border-right` 边框
- `.border-top` 上边框
- `.border-bottom` 下边框
- `.border-right-0` 去掉某边框的宽度
- `.border-0` 去掉所有的边框

(3)边框颜色

- `.border-primary` 多为自定义颜色

(4)圆角

- `.rounded` 圆角属性
- `.rounded-lg` 大尺寸圆角

5.间距

- m： margin值
- p： padding值
- t、r、b、l：分别代表上右下左
- `.ml-1` 默认尺寸是1，0~5
- `.pl-1` 默认尺寸是1，0~5
- 响应式 `ml-lg-*` 和 `pt-sm-*` 等
- m-auto： 代表块级元素在父级里居中

6.浮动

- `.float-left` ： 元素左浮动
- `.float-right` ： 元素右浮动
- `.clearfix` ： 子元素float后，通过给父元素添加 `.clearfix` 达到清除浮动效果

```
<div class="container">
  <div class="box clearfix bg-primary">
    <div class="float-left bg-danger"></div>
    <div class="bg-success"></div>
    <div class="bg-warning"></div>
  </div>
</div>
```

7.定位

- `.position-relative` 相对定位,定位的方向需要单独写
- `.position-absolute` 绝对定位，注意父元素需要有定位属性，定位的方向需要单独写
- `.position-fixed` 固定定位，有底部固定和顶部固定两种设置，但其他需要单独设定
- `.fixed-top` 固定定位顶部
- `.fixed-bottom` 固定定位底部
- 缺点：top、left、bottom、right 还需要单独设置

8.列表样式

`.list-unstyled` 去掉列表原有样式

9.显示

- `.d-none` 元素消失
- `.d-block` 元素按照块级显示
- `.d-inline` 元素按照内联显示
- `.d-inline-block` 元素按照行内块显示

- `.d-flex` 元素按照弹性布局显示
- 支持响应式 `.d-*-none` sm、md、lg、xl,需要注意的是看源码的区间

10.flex布局

(1)flex属性

- 所有需要使用flex布局的类都需要先在容器中加入以下两个类,否则不生效
- `.d-flex` 容器设置flex布局
- `.d-inline-flex` 内联元素设置flex布局
- 所有弹性都支持响应式 `.d-sm-flex` ,他规定了四个尺寸sm小尺寸,md中等尺寸,lg大尺寸,xl超大尺寸
  - `.d-flex`
  - `.d-inline-flex`
  - `.d-sm-flex`
  - `.d-sm-inline-flex`
  - `.d-md-flex`
  - `.d-md-inline-flex`
  - `.d-lg-flex`
  - `.d-lg-inline-flex`
  - `.d-xl-flex`
  - `.d-xl-inline-flex`

(2)方向

- `.flex-row` 水平方向正向排版
- `.flex-row-reverse` 水平方向反向排版
- `.flex-column` 垂直方向反向排版
- `.flex-column-reverse` 垂直方向反向排版
- 同样他规定了四个尺寸sm小尺寸,md中等尺寸,lg大尺寸,xl超大尺寸
  - `.flex-*-row` 水平方向正向排版
  - `.flex-*-row-reverse` 水平方向反向排版
  - `.flex-*-column` 垂直方向反向排版
  - `.flex-*-column-reverse` 垂直方向反向

(3)内容对齐

- `.justify-content-*` 代表水平轴的对齐方式
- 支持响应式，*号位置代表sm,md,lg,xl
- `.justify-content-*-start` (浏览器默认值)主轴起始位置对齐
- `.justify-content-*-end` 主轴结束位置对齐

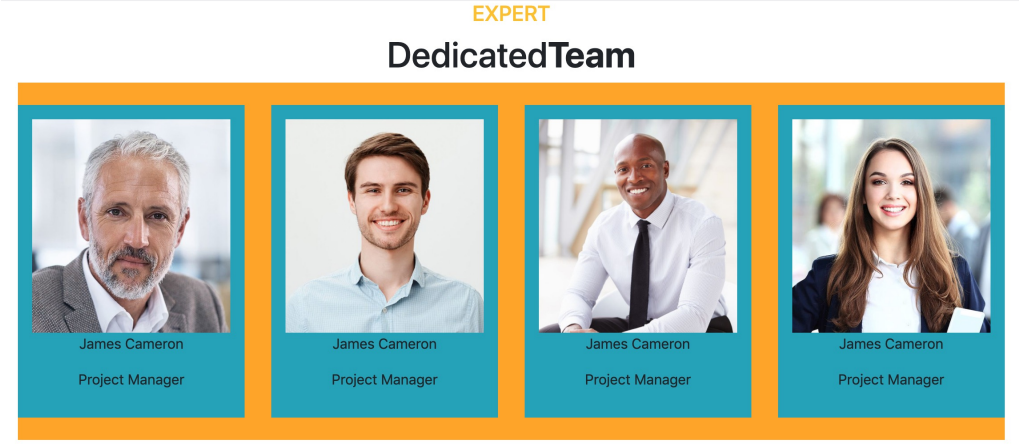
- `.justify-content-center` 居中
- `.justify-content-between` 两端对齐
- `.justify-content-around` 项目左右间距相同,两元素间距是左右两侧的一倍

(4)对齐项目

- align-items-* 代表主轴的交叉轴的对齐方式
- 支持响应式，*号位置代表sm,md,lg,xl
- `.align-items-*-start` 交叉轴起始位置对齐
- `.align-items-*-end` 交叉轴结束位置对齐
- `.align-items-*-center` 交叉轴居中对齐
- `.align-items-*-baseline` 交叉轴及基线位置对齐
- `.align-items-*-stretch` 交叉轴占满宽高

```
<div class="container">
  <!-- flex -->
  <div class="box w-100 bg-info d-flex">
    <div class="bg-danger h-100 p-5">1</div>
    <div class="bg-success h-100 p-5">2</div>
    <div class="bg-warning h-100 p-5">3</div>
  </div>
  <!-- 方向 -->
  <div class="box w-100 bg-dark d-flex flex-column-reverse">
    <div class="bg-primary w-100 p-5">1</div>
    <div class="bg-dark w-100 p-5">2</div>
    <div class="bg-danger w-100 p-5">3</div>
  </div>
  <!-- 响应式 lg的时候横向,sm的时候垂直-->
  <div class="box bg-secondary d-flex flex-lg-row flex-sm-column">
    <div class="bg-success border border-danger p-5">1</div>
    <div class="bg-success border border-danger p-5">2</div>
    <div class="bg-success border border-danger p-5">3</div>
    <div class="bg-success border border-danger p-5">4</div>
  </div>
  <!-- 内容对齐 -->
  <div class="box bg-warning d-flex justify-content-between">
    <div class="p-5 h-50 bg-info border border-dark">1</div>
    <div class="p-5 h-50 bg-info border border-dark">2</div>
    <div class="p-5 h-50  bg-info border border-dark">3</div>
  </div>
</div>
```

【练习】



四、栅格布局

- 栅格系统是用于页面布局，属于boot的核心
- 可以在不同终端显示不同效果（响应式）
- 栅格，具有行row、列col的概念

1.栅格布局的列

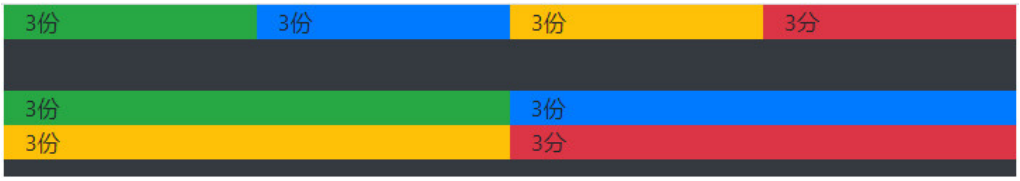
- 一行平均分为12个列
- `.col-1~12` 代表列占有行的十二分之几份

```
<div class="container pt-5 pb-5">
   <div class="row">
     <div class="col-3 text-center">
       <img src="./img/codeboy/img/footer/icon1.png" alt="">
       <p class="mt-2">品质保障</p>
     </div>
     <div class="col-3 text-center">
       <img src="./img/codeboy/img/footer/icon2.png" alt="">
       <p class="mt-2">私人定制</p>
     </div>
     <div class="col-3 text-center">
       <img src="./img/codeboy/img/footer/icon3.png" alt="">
       <p class="mt-2">学员特供</p>
     </div>
     <div class="col-3 text-center">
       <img src="./img/codeboy/img/footer/icon4.png" alt="">
       <p class="mt-2">专属特权</p>
     </div>
   </div>
 </div>
```

2.栅格布局的响应式

Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
None (auto)	540px	720px	960px	1140px
.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-

- `.col-sm-*` 小屏幕
- `.col-md-*` 中屏幕
- `.col-lg-*` 大屏幕
- `.col-xl-*` 超大屏幕



```
<!-- 正常的 -->
    <div class="box">
      <div class="row">
        <div class="col-3 bg-success">3份</div>
        <div class="col-3 bg-primary">3份</div>
        <div class="col-3 bg-warning">3份</div>
        <div class="col-3 bg-danger">3分</div>
      </div>
    </div>
    <!-- 响应式 -->
    <div class="box">
      <div class="row">
        <div class="col-lg-3 col-md-6 col-sm-12 bg-success">3份</div>
        <div class="col-lg-3 col-md-6 col-sm-12 bg-primary">3份</div>
        <div class="col-lg-3 col-md-6 col-sm-12 bg-warning">3份</div>
        <div class="col-lg-3 col-md-6 col-sm-12 bg-danger">3分</div>
      </div>
    </div>
  </div>
```

4.栅格布局的偏移值

- 元素在自己的原位置像右偏移几份
- `.offset-1~11` 偏移份数
- 偏移超出一行，列会换行


```
    <div class="container bg-dark box">
      <div class="row">
        <div class="col-2 bg-primary offset-3">3份</div>
        <div class="col-2 bg-warning">3份</div>
        <div class="col-2 bg-danger">3分</div>
      </div>
      <div class="row">
        <div class="col-1 bg-warning">1</div>
        <div class="col-1 offset-2 bg-danger">2</div>
        <div class="col-1 offset-3 bg-primary">3</div>
        <div class="col-1 offset-4 bg-light">4</div>
      </div>
    </div>
```

5.嵌套布局

嵌套布局示意图,因为栅格布局有局限,他的底层还是弹性布局,复杂的嵌套关系可以尝试混搭使用

```
.myrow > div { height: 240px; overflow: hidden;}
  <div class="container">
    <div class="row">
      <div class="col-6"><div class="col-6">
        <div class="row">
          <div class="col-6"></div>
          <div class="col-6"></div>
        </div>
        <div class="row myrow pt-2">
          <div class="col-4 pr-0"></div>
          <div class="col-4 p-0"></div>
          <div class="col-4 pl-0"> </div>
        </div>
      </div>
    </div>
  </div>
```

6.栅格布局的项目排列方式

(1) 水平轴

- `.justify-content-start` 开始位置对齐(如果横向居左)
- `.justify-content-center` 居中对齐(如果横向居中)
- `.justify-content-end` 结束位置对齐(如果横向居右)
- `.justify-content-around` 有缝隙的对齐
- `.justify-content-between` 左右两端对齐

(2) 垂直轴

- `.align-items-start` 开始位置对齐(如果横向居顶)
- `.align-items-center` 居中对齐
- `.align-items-start` 结束位置对齐(如果横向居底)

【练习】

> 使用栅格布局先制作一张卡，制作三张卡
> 使用js将数据151张全部导入
> `::before {content: "#";}` 在元素前增加内容
> `var abc += xxxxx` 拼接
> 模板字符串增强版的字符串，用反引号（`）标识，中间可以插入变量${abc}
> abc.innerHTML = 在该元素中增加内容（覆盖）

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="./css/bootstrap.css">
  <script src="./js/jquery.min.js"></script>
  <script src="./js/popper.min.js"></script>
  <script src="./js/bootstrap.min.js"></script>
  <!-- 引入我给的data.js -->
  <script src="./js/data.js"></script>
  <style>
    .z-fh::before {content: "#";}
    .z-poke .wrap {
      font-size: 13px;
      border-radius: 5px;
    }
    .z-poke>div span {border-radius: 5px;}
  </style>
</head>
<body>
  <div class="container">
    <!-- 先写行 row z-poke -->
    <div class="row z-poke" id="poke"></div>
  </div>
</body>
<script>
  // 获取行
  var poke = document.getElementById('poke');
  //看看数据
  var datajson = data;
  console.log(datajson);
  //创建一个空字符串，用于接住遍历的每一次的结果
  var pokestr = "";
  for(var i=0;i<datajson.length;i++){
    pokestr += `<div class="col-lg-4 col-md-6 col-sm-12 pt-3 pb-3">
  <div class="wrap bg-dark p-3">
    <div class="d-flex justify-content-between text-dark">
      <span class="p-2 bg-light">${datajson[i].name}</span>
      <span class="z-fh p-2 bg-light">${datajson[i].id}</span>
    </div>
    <div class="w-100 bg-light rounded mt-3 p-2">
      <img src="./img/poke/${datajson[i].id}.png" alt="" class="w-100">
    </div>
  </div>
</div>`
  }
  poke.innerHTML = pokestr;
</script>
</html>
```

五、内容

1.图片

- 在响应式或者移动端页面中,图片往往需要随着宽度的大小而变化,不能使用自己的宽度高度,但是img的最大特点是自己会保持自己的宽高比例
- `.img-fluid` 响应式图片，图片比较实用的内容比较少,其他的样式都是固定的不太适用于大多公司的要求,记一个响应式图片就可以

```
<div class="container">
  <img src="./img/响应式背景.jpeg" alt="" class="img-fluid">
</div>
```

2.表格

- `.table` 是table标签的一个基类（基本样式）
- `.thead-light` 或 `.thead-dark` 就能使 `<thead>` 区显示出浅黑或深灰
- `.table-bordered` table的边框,有设置好的的样式,但不如自己自定义
- `.table-striped` 条形纹,相当于隔行变色效果
- `.table-hover` 鼠标悬停效果

序号	图像	中文	属性
#001		妙蛙种子	草 毒
#002		妙蛙草	草 毒
#003		妙蛙花	草 毒
#004		小火龙	火

```
<style>
    .pic {
      width: 68px;
      height: 56px;
      background-image: url(./img/MSP.png);
      background-repeat: no-repeat;
    }
    table .pic1{
      background-position: -136px 0;
    }
    table .pic2{
      background-position: -204px 0;
    }
    table .pic3{
      background-position: -272px 0;
    }
    table .pic4{
      background-position: -408px 0;
    }
</style>
 <div class="container">
    <table class="table text-center table-hover table-striped">
      <thead class="thead-dark">
        <tr>
          <th>序号</th>
          <th>图像</th>
          <th>中文</th>
          <th>属性</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>#001</td>
          <td><div class="pic pic1 m-auto"></div></td>
          <td>妙蛙种子</td>
          <td>
            <div class="bg-success text-white float-left w-50">草</div>
            <div class="bg-dark text-white float-left w-50">毒</div>
          </td>
        </tr>
        <tr>
          <td>#002</td>
          <td><div class="pic pic2 m-auto"></div></td>
          <td>妙蛙草</td>
          <td >
            <div class="bg-success text-white float-left w-50">草</div>
            <div class="bg-dark text-white float-left w-50">毒</div>
          </td>
        </tr>
        <tr>
          <td>#003</td>
```

```
          <td><div class="pic pic3 m-auto"></div></td>
          <td>妙蛙花</td>
          <td >
            <div class="bg-success text-white float-left w-50">草</div>
            <div class="bg-dark text-white float-left w-50">毒</div>
          </td>
        </tr>
        <tr>
          <td>#004</td>
          <td><div class="pic pic4 m-auto"></div></td>
          <td>小火龙</td>
          <td>
            <div class="col bg-danger text-white">火</div>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
```

六、组件

组件就是boot封装好的一些内容但它们有自己特有的样式组件有的是有动态交互效果的，因此有的需要js参与

1.按钮

- 可以作为按钮的标签： <a>、<button>、<input>
- .btn 按钮的基本类，渲染了基本的背景、边框、过渡等样式，必须加

(1)颜色样式

按钮的颜色都在bootstrap 中广泛应用，需要有印象。另外自己配色可以参照（自己配色容易很丑）

- .btn-default 默认样式
- .btn-success 成功
- .btn-warning 警告
- .btn-danger 危险
- .btn-primary" 首选项
- .btn-primary 重要
- .btn-info 信息
- .btn-light 浅色
- .btn-dark 深色

- `.btn-link` 连接

(2)按钮轮廓

- `.btn-outline-success` 成功
- `.btn-outline-warning` 警告
- `.btn-outline-danger` 危险
- `.btn-outline-primary` 重要
- `.btn-outline-info` 信息
- `.btn-outline-light` 浅色
- `.btn-outline-dark` 深色

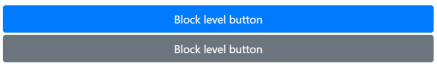
(3)按钮尺寸

按钮的尺寸是通过内边距撑开的，而不是宽高，属于开发技巧。boot是世界上公认样式写的最合理最优美的，虽然实际开发可能不用

- `.btn-lg` 大尺寸按钮
- `.btn-sm` 小尺寸按钮

(4)块级按钮

块级按钮 `.btn-block` 会随着外层容器的宽度变化，因为它的宽度是外层的100%，多用于移动端，比如：支付，下一步，登录等

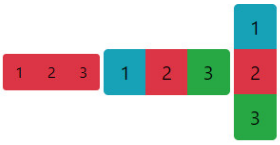


- `.btn-block` 块级按钮,父元素的同等大小

2.按钮组

按钮组就是把按钮放在以一个组里显示

- `.btn-group` 按钮组（基类）
- `.btn-group-lg` 和 `.btn-group-sm` 按钮组尺寸
- `.btn-group-vertical` 按钮组竖列显示，不需要写基类 `.btn-group`



```
<div class="container">
  <!-- 普通的按钮 -->
  <button class="btn">按钮</button>
  <button class="btn btn-danger btn-sm">按钮</button>
  <button class="btn btn-success btn-lg">按钮</button>
  <button class="btn btn-outline-danger">按钮</button>
  <button class="btn btn-block btn-primary">块级按钮</button>
  <!-- 按钮组 -->
  <!-- 一说组，必须找一个元素包起来 btn-group按钮组 -->
  <div class="btn-group">
    <button class="btn bg-danger">1</button>
    <button class="btn bg-danger">2</button>
    <button class="btn bg-danger">3</button>
  </div>
  <div class="btn-group btn-group-lg">
    <button class="btn bg-info">1</button>
    <button class="btn bg-danger">2</button>
    <button class="btn bg-success">3</button>
  </div>
  <!-- 竖排按钮btn-group-vertical -->
  <div class="btn-group-vertical btn-group-lg">
    <button class="btn bg-info">1</button>
    <button class="btn bg-danger">2</button>
    <button class="btn bg-success">3</button>
  </div>
</div>
```

3.徽章

是一种小型的用于计数和打标签的组件，比如邮件未读提示，购物车角标等等,一般涵盖在标签中用span标签来表示徽章

- `.badge` 徽章（基类）
- `.badge-pill` 胶囊徽章，可以写基类可以不些，不写会变大

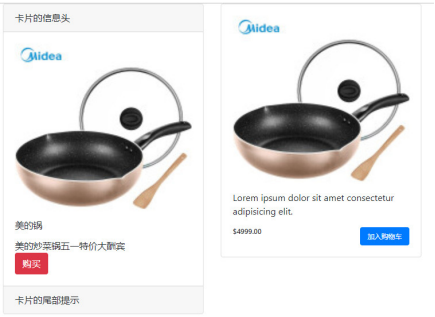
```
<div class="container">
    <p>你好,吴道华 <span class="badge bg-success text-white">新用户</span></p>
    <!-- 未读信息 -->
    <button class="btn btn-primary">收件箱 <span class="badge bg-danger">2</span> </button>
      <!-- 小练习 -->
  <div class="btn-group-vertical btn-group-lg">
    <button class="btn btn-info">收件箱 <span class="badge bg-danger">2</span></button>
    <button class="btn btn-warning">发件箱</button>
    <button class="btn btn-success">草稿箱 <span class="badge bg-danger">2</span></button>
  </div>
</div>
```

胶囊购物车:

```
<style>
.box {
margin-top: 20px;
width: 70px;
}
.box span {margin-top: -10px;}
</style>
<div class="box">
  <img src="./img/card.png" alt="" class="w-50 d-block float-left">
  <span class="badge text-light bg-danger d-block float-left">5</span>
</div>
```

4.卡片

- `.card` 卡片，最外层容器
- `.card-header` 卡片头部信息
- `.card-img-top` 卡片顶部图片 和 `.card-img-bottom` 卡片的底部图片
- `.card-body` 卡片主体部分
- `.card-text` 卡片文字介绍
- `.card-footer` 卡片的底部
- `.card-group` 卡片组，里面可以包裹多个卡片



```
<div class="card box">
  <h3 class="card-header">一个商品</h3>
  <img src="./img/guo.jpg" alt="" class="card-img-top w-75 m-auto">
  <div class="card-body">
    <h4 class="card-title">美的炒菜锅</h4>
    <p class="card-text">美的（Midea）典雅金麦饭石色涂层不粘炒锅家用炒锅电磁炉</p>
    <button class="btn btn-danger">加入购物车</button>
  </div>
  <div class="card-footer">暂时无货</div>
</div>
```

【练习】

> 学子商城列表页商品卡片
> 可以使用boot中的颜色，可以使用卡片组
> 按照布局一个卡片的宽度最小和最大的宽度都为280px

```
<div class="container">
   <div class="card-group mycard">
     <div class="card m-3 border">
       <img src="./img/shop/1.jpeg" alt="">
       <div class="card-body">
         <h5 class="card-title text-info">￥5499.00</h5>
         <p class="card-text">神舟(HASEE)战神Z7M-SL7D2 15.6英寸游戏本笔记本电脑(i7-6700HQ 8G 1T+1
         <div class="row">
           <div class="col-6">
             <button class="btn btn-sm btn-dark">-</button>
             <input type="text" class="myinput" value="1">
             <button class="btn btn-sm btn-dark">+</button>
           </div>
           <div class="col-6">
             <button class="btn btn-info">加入购物车</button>
           </div>
         </div>
       </div>
     </div>
   </div>
 </div>
```

5.面包屑导航

面包屑导航（Breadcrumb）用于指示当前页面在导航层级中的位置
可以使用h5的新标签 <nav> 给它加入 aria-label属性用来给当前元素加上的标签描述，这个属性可以对不明显的标签做描述，对无障碍服务有利。它知识一个语义化属性，不加也没关系
li的 aria-current="page" 表示当前页，也是一种描述

- 面包屑导航可以使用h5的新标签 <nav> ，并且根据语义化的特点给其加入 aria-label="breadcrumb" 属性，用于描述其功能
- .breadcrumb 面包屑导航外层属性
- .breadcrumb-item 面包屑导航的项目
- .active 代表当前的项目

学子商城面包屑导航

```
<style>
/*找到分隔符的css，把content改了*/
.breadcrumb-item+.breadcrumb-item::before {
  content: ">" !important;
}
</style>
<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">首页</a></li>
    <li class="breadcrumb-item"><a href="#">学习用品</a></li>
    <li class="breadcrumb-item"><a href="#">笔记本电脑</a></li>
    <li class="breadcrumb-item active">联想</li>
  </ol>
</nav>
```

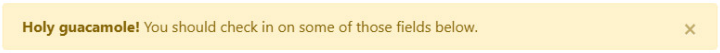
6.弹窗

组件中涉及自定义属性，通过自定义属性控制交互行为。

```
<div class="box bg-success" data-my="odiv" id="mydiv"></div>
var mydiv = document.getElementById('mydiv');
console.log(mydiv.getAttribute('data-my'));
```

因为自定义属性的存在，因此使用自定义属性做js交互

自定义属性 `data-dismiss="alert"` 通过 data 属性：通过数据API添加可取消功能，只需要向关闭按钮添加 `data-dismiss="alert"` ，就会自动为警告框添加关闭功能。



- `.alert` 设置元素为警告框组件
- `.alert-info` 颜色
- `.close` 用于警告框内单独的标签设置按钮为可关闭；
- `data-dismiss="alert"` 是给关闭按钮设置的自定义属性事件

```
<div class="container">
  <!-- 一个单独的提示窗没有任何交互的 -->
  <div class="alert alert-primary">这是一个提示窗</div>
  <!-- 有关闭标识 -->
  <div class="alert alert-danger">
    这是一个有关闭标识的提示窗
    <span class="close" data-dismiss="alert">x</span>
  </div>
</div>
```

7.折叠

折叠是一种效果，有一个类似按钮的元素，点按之后出现对应的内容。需要一个按钮类的组件来进行显示元素的控制。之前在学习css的时候有一个选择器教target，和这个类似.
使用 `data-toggle="collapse"` 自定义属性作为切换的按钮控制折叠元素，同时要用选择器锁定被折叠元素 `data-target="#show"` 。
这个选择器如果使用id，则会控制对应id元素的打开和折叠，如果是类名，则可以控制两个显示区域的切换。
在boot 4.6以上版本，有单独的手风琴效果，4.0没有

- 需要两组元素，第一组是控制元素，控制显示和隐藏的区域
- 控制元素需要使用 `data-toggle="collapse"` 设置折叠控制， `data-target="#show1"` 关联折叠元素
- `.collapse` 被折叠元素，需要有被控制的选择器，如id或者class

单独按钮的开合

```
<button class="btn btn-primary" data-toggle="collapse" data-target="#show">打开折叠</button>
<div class="collapse" id="show">点击打开按钮对应的被折叠区域</div>
```

多个按钮的开合和切换

```
<button class="btn btn-success" data-toggle="collapse" data-target="#show1">show1</button>
<button class="btn btn-danger" data-toggle="collapse" data-target="#show2">show2</button>
<button class="btn btn-warning" data-toggle="collapse" data-target=".box">切换1和2</button>
<div class="collapse box" id="show1">对应第一个显示区域</div>
<div class="collapse box" id="show2">对应第二个显示区域</div>
```

7.下拉菜单

下拉菜单与折叠类似，需要一个控制元素，关联开合的下拉组

- `.dropdown` 下拉菜单区域
- 控制按钮
  - `.dropdown-toggle` 向下三角
  - `data-toggle="dropdown"` 控制下拉的控件
- 菜单区域
  - `.dropdown-menu` 菜单列表
  - `.dropdown-item` 菜单列表项

```
<div class="container">
  <div class="dropdown">
    <button class="btn btn-success dropdown-toggle" data-toggle="dropdown">下拉菜单</button>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">家用电器</a>
      <a class="dropdown-item" href="#">食品生鲜</a>
      <a class="dropdown-item" href="#">图书影视</a>
    </div>
  </div>
</div>
```

8.表单

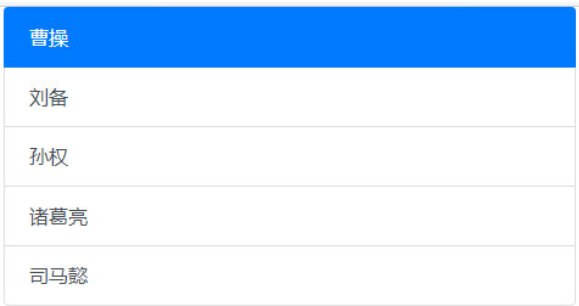
表单的涉及有自己的样式,但因为搜索或者表单区域会设计自己但样式，往往表单提供的样式不常用

`.form-control` 表单控件（基类）

```
<div class="container">
  <!-- form-control -->
  <input type="text" class="form-control">
</div>
```

11.列表组

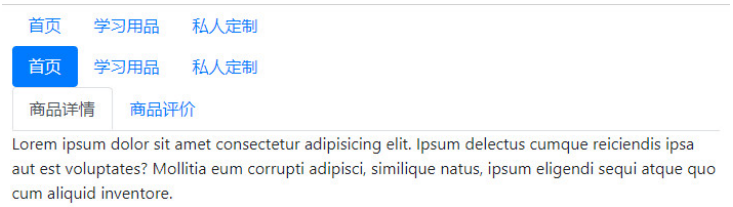
- `.list-group` 列表组（父级）
- `.list-group-item` 列表项目
- `.list-group-item-action` 列表项目的悬停效果
- `.active` 被选项
- `.list-group-item-primary` 列表项的颜色



```
<div class="container">
  <ul class="list-group">
    <li class="list-group-item list-group-item-action active">曹操</li>
    <li class="list-group-item list-group-item-action">刘备</li>
    <li class="list-group-item list-group-item-action">孙权</li>
    <li class="list-group-item list-group-item-action">诸葛亮</li>
    <li class="list-group-item list-group-item-action">司马懿</li>
  </ul>
</div>
```

12.导航

- 普通导航
  - .nav 导航（父级的基类）
  - .nav-item 导航项目（多用li）
  - .nav-link active 导航链接（一般用于li中的a）以及 当前的
  - .justify-content-center 导航的居中对齐
  - .justify-content-end 导航的右侧遂起
- 胶囊导航
  - .nav 导航（父级的基类）
  - .nav-pills 胶囊效果（父级）
  - .nav-pills 胶囊效果（父级）
  - .nav-item 导航项目（多用li）
  - data-toggle="pill" 使用胶囊切换（li）
  - .nav-link active 导航链接（一般用于li中的a）以及 当前的
- 选项卡
  - .nav 导航（父级的基类）
  - .nav-tabs 导航为选项卡（父级）
  - .nav-link tab链接
  - data-toggle="tab" href="#home" 具有切换的功能 以及 显示目标的id，依靠id关联
  - .tab-content 显示目标的父级
  - .tab-pane 显示部分隐藏
  - .show 显示
  - .active 默认选择
  - id="home" 显示内容对应id



```
<div class="container">
  <!-- 普通导航 -->
  <ul class="nav">
    <li class="nav-item">
      <a class="nav-link active" href="#">首页</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">学习用品</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">私人定制</a>
    </li>
  </ul>
  <!-- 胶囊导航 -->
  <ul class="nav nav-pills">
    <li class="nav-item">
      <a class="nav-link active" data-toggle="pill" href="#">首页</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" data-toggle="pill" href="#">学习用品</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" data-toggle="pill" href="#">私人定制</a>
    </li>
  </ul>
  <!-- 选项卡 -->
  <nav>
    <div class="nav nav-tabs">
      <a class="nav-link active" data-toggle="tab" href="#tab1">商品详情</a>
      <a class="nav-link" data-toggle="tab" href="#tab2" role="tab" aria-controls="nav-profile
        aria-selected="false">商品评价</a>
    </div>
  </nav>
  <div class="tab-content">
    <div class="tab-pane show active" id="tab1">Lorem ipsum dolor sit amet consectetur adipisi
      delectus cumque reiciendis ipsa aut est voluptates? Mollitia eum corrupti adipisci, simi
      eligendi sequi atque quo cum aliquid inventore.</div>
    <div class="tab-pane" id="tab2">111111111</div>
  </div>
</div>
```

13.导航条

导航条或者叫导航栏和导航的区别是:

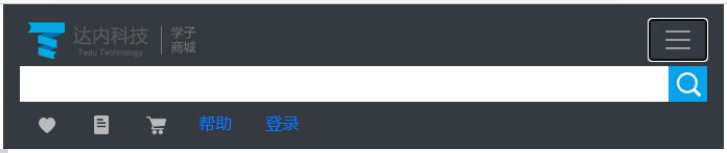
1.导航是横向，导航条可以是横向和纵向。

2.导航的结构ul>li>a,导航条的结构最外层需要有一个包裹元素如div,nav

3.boot网页的导航，缩小页面看效果，导航条缩小看效果，导航条缩小后可以响应式收缩导航内容变成三条线,叫做断点分组

- `.navbar` 导航条外层（基类）
- `navbar-dark bg-dark` 深色主题
- `navbar-light bg-light` 浅色主题
- logo和项目介绍
  - `.navbar-brand` 用于logo和项目名的外层
- 响应式
  - `.navbar` 导航条外层（基类）
  - `.navbar-expand{-sm|-md|-lg|-xl}` 菜单为竖向排列没加入该类变为横向，同样用于响应式(必要)
- 可隐藏的导航菜单
  - `.collapse .navbar-collapse` 断点分组和隐藏导航栏内容，与缩小菜单相关，在导航父级设置
  - 如果需要响应式隐藏导航，需要设置id与缩小菜单关联，如： `id=navMenu`
  - `.navbar-nav` 轻量级导航（支持下拉菜单）
- 缩小菜单
  - 一般使用 `<button>` 作为缩小菜单
  - `.navbar-toggler` 缩小菜单（缩小时展示的菜单）
  - `data-toggle="collapse" data-target="#navMenu"` 需要切换属性以及关联可切换的目标菜单,前提必须有对应id的菜单才生效
  - 一般使用 `<span>` 设置缩小菜单的图标 `.navbar-toggler-icon`
- 表单控件 `.form-inline`
- 文本 `.navbar-text` 用于添加垂直居中的文本字符串

【练习】

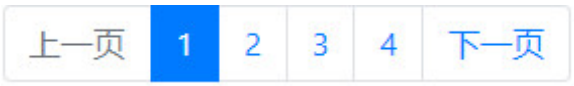


> 把navbar改成学子商城导航样式
> 注意在lg以上显示 form 可占col-lg-6 offset-lg-1 ， lg以下缩小菜单时form可占 col-sm-10

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
      <a class="navbar-brand" href="#"><img src="./img/logo.png" alt=""></a>
      <button class="navbar-toggler" data-toggle="collapse" data-target="#navMenu">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navMenu">
        <form class="form-inline">
          <input type="text" class="form-control d-block float-left">
          <div class="dropdown float-left mydown">
            <button class="btn dropdown-toggle" data-toggle="dropdown">下拉菜单</button>
            <div class="dropdown-menu">
              <a class="dropdown-item" href="#">家用电器</a>
              <a class="dropdown-item" href="#">食品生鲜</a>
              <a class="dropdown-item" href="#">图书影视</a>
            </div>
          </div>
          <a href="#" class="float-right searchimg"></a>
        </form>
        <div class="navbar-nav">
          <a class="nav-link active" href="#"><img src="./img/care.png" alt=""></a>
          <a class="nav-link active" href="#"><img src="./img/order.png" alt=""></a>
          <a class="nav-link active" href="#"><img src="./img/shop_car.png" alt=""></a>
          <a class="nav-link active" href="#"><span>注册</span></a>
          <a class="nav-link active" href="#"><span>帮助</span></span></a>
        </div>
      </div>
    </nav>
```

14.分页

- `.pagination` 分页的父级效果
- `.page-item` 分页项
- `.active` 被激活的项目
- `.disabled` 禁用的，比如第一页时的上一页，和最后页时的下一下
- `.page-link` 分页项中的链接
- 对齐方式 `.justify-content-center` 居中， `.justify-content-end` 居右



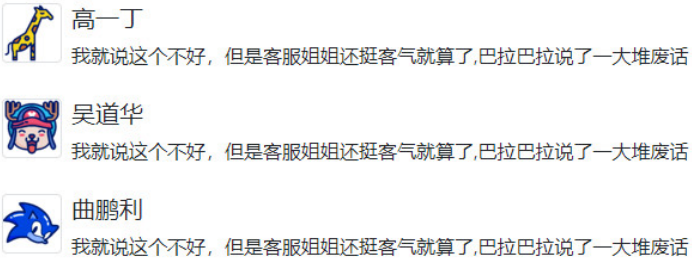
page.html，ul.pagination>li.page-item>a.page-link 之后加入 .active 和 .disabled

```
<div class="pages">
      <ul class="pagination justify-content-end">
        <li class="page-item disabled"><a class="page-link" href="#">上一页</a></li>
        <li class="page-item active"><a class="page-link" href="#">1</a></li>
        <li class="page-item"><a class="page-link" href="#">2</a></li>
        <li class="page-item"><a class="page-link" href="#">3</a></li>
        <li class="page-item"><a class="page-link" href="#">4</a></li>
        <li class="page-item"><a class="page-link" href="#">下一页</a></li>
      </ul>
    </div>
```

15.媒体对象

构建高度重复的组件，例如博客评论，推文等。

- `.media` 媒体对象，创建重复模块
- `.media-body` 媒体对象的内容区域
- 直接加入 `<img>` 图片 `.media-body` 的内容会在它一侧展示，可以通过 `<img>` 和 `.media-body` 位置调换来让图片在左还是在右显示
- `.list-unstyled` 使用媒体对象列表可以包裹多个媒体对象



```
<div class="container">
   <!-- 一个媒体对象 -->
   <div class="media">
     <img src="./img/appraise/icon1.png" class="border rounded m-2">
     <div class="media-body">
       <h5 class="mt-2">高一丁</h5>
       <p>我就说这个不好，但是客服姐姐还挺客气就算了,巴拉巴拉说了一大堆废话</p>
     </div>
   </div>
   <!-- 一组媒体对象 -->
   <ul class="list-unstyled">
     <li class="media">
       <img src="./img/appraise/icon1.png" class="border rounded m-2">
       <div class="media-body">
         <h5 class="mt-2">高一丁</h5>
         <p>我就说这个不好，但是客服姐姐还挺客气就算了,巴拉巴拉说了一大堆废话</p>
       </div>
     </li>
     <li class="media">
       <img src="./img/appraise/icon2.png" class="border rounded m-2">
       <div class="media-body">
         <h5 class="mt-2">吴道华</h5>
         <p>我就说这个不好，但是客服姐姐还挺客气就算了,巴拉巴拉说了一大堆废话</p>
       </div>
     </li>
     <li class="media">
       <img src="./img/appraise/icon3.png" class="border rounded m-2">
       <div class="media-body">
         <h5 class="mt-2">曲鹏利</h5>
         <p>我就说这个不好，但是客服姐姐还挺客气就算了,巴拉巴拉说了一大堆废话</p>
       </div>
     </li>
   </ul>
   <!-- 更换顺序 -->
   <ul class="list-unstyled">
     <li class="media">
       <img src="./img/appraise/icon1.png" class="border rounded m-2">
       <div class="media-body">
         <h5 class="mt-2">高一丁</h5>
         <p>我就说这个不好，但是客服姐姐还挺客气就算了,巴拉巴拉说了一大堆废话</p>
       </div>
     </li>
     <li class="media">
       <div class="media-body">
         <h5 class="mt-2">吴道华</h5>
         <p>我就说这个不好，但是客服姐姐还挺客气就算了,巴拉巴拉说了一大堆废话</p>
       </div>
       <img src="./img/appraise/icon2.png" class="border rounded m-2">
     </li>
     <li class="media">
       <img src="./img/appraise/icon3.png" class="border rounded m-2">
```

```
        <div class="media-body">
          <h5 class="mt-2">曲鹏利</h5>
          <p>我就说这个不好，但是客服姐姐还挺客气就算了,巴拉巴拉说了一大堆废话</p>
        </div>
      </li>
    </ul>
  </div>
```

16.巨幕

巨幕就是某个图片或者某个颜色扩展至整个视口，主要用于大型活动和广告的宣传，比如王者荣耀官网

- `.jumbotron` 超大巨幕，可以不写在 `.container` 或者 `.container-fluid` 中
- `.jumbotron` 可以包裹 `.container` 或者 `.container-fluid`
- 因 `.jumbotron` 有圆角，因此，加 `.jumbotron-fluid` 可以去除巨幕的圆角

```
<!-- jumbotron-fluid 可以去除巨幕的圆角 -->
  <div class="jumbotron jumbotron-fluid myjum">
    <div class="container">
       <h1 class="display-4 text-light">欢迎来到王者荣耀</h1>
       <p class="lead">This is a simple hero unit, a simple jumbotron-style component for call
         featured content or information.</p>
       <hr class="my-4">
       <p>It uses utility classes for typography and spacing to space content out within the l
       <a class="btn btn-primary btn-lg" href="#">活动详情</a>
    </div>
  </div>
```

17.轮播

轮播图比较常见，效果也很多，而他的控制往往通过js来做。

- 父级
  - `.carousel` 定位
  - `.slide` 平滑的过渡效果，不需要可以删掉
  - `data-ride="carousel"` 负责利用定时器切换内部项目（js），不需要可以删掉
- 展示图
  - `.carousel-inner` 展示图大区域位置
  - `.carousel-item` 展示区域中的展示项目
  - `.active` 被激活的展示项目
- 左右控制按钮
  - 左右切换使用a标签，利用 `href="#banner"` 来控制轮播
  - `.carousel-control-prev` 上一个（在轮播中定位）

    - `.carousel-control-next` 下一个（在轮播中定位）
    - `.carousel-control-prev-icon` 和 `.carousel-control-next-icon` 可点击的按钮
    - `data-slide="prev"` 向左控制按钮
    - `data-slide="next"` 向右控制按钮
- 指示符
    - `.carousel-indicators` 指示符区域，在轮播中的位置
    - `data-slide-to="0"` 指示符项目控制的对应展示项目的位置，从0开始
    - `.active` 指示符激活
    - `data-target="#banner"` 指示符需要控制轮播
    - 在boot中 `.carousel-indicators li` 设置了控制符的样式
    - `.carousel-indicators .active` 控制active的样式

```
<div class="container">
  <!-- carousel轮播的控件  slide平滑过渡 data-ride="carousel"定时方法js-->
  <div id="banner" class="carousel slide" data-ride="carousel">
    <!-- 图片区域 -->
    <div class="carousel-inner">
      <div class="carousel-item active">
        <img src="./img/banner/itemCat_banner1.png" class="d-block w-100" alt="">
      </div>
      <div class="carousel-item">
        <img src="./img/banner/itemCat_banner2.png" class="d-block w-100" alt="">
      </div>
      <div class="carousel-item">
        <img src="./img/banner/itemCat_banner3.png" class="d-block w-100" alt="">
      </div>
    </div>
    <!-- 左右切换按钮 -->
    <a class="carousel-control-prev" href="#banner" data-slide="prev">
      <span class="carousel-control-prev-icon"></span>
    </a>
    <a class="carousel-control-next" href="#banner" data-slide="next">
      <span class="carousel-control-next-icon"></span>
    </a>
    <!-- 指示符 -->
    <ul class="carousel-indicators">
      <li data-target="#banner" data-slide-to="0" class="active"></li>
      <li data-target="#banner" data-slide-to="1"></li>
      <li data-target="#banner" data-slide-to="2"></li>
    </ul>
  </div>
</div>
```