2.缩放
改变元素在页面中的尺寸
transform:scale(n)
取值，无单位的数字
n>1 根据转换原点放大
0< n <1 根据转换原点缩小
-1< n <0 翻转缩小 （水平和垂直方向都翻转了 180 度）
n<-1 翻转放大
transform:scale(x,y) x 和 y 缩放尺寸
transform:scaleX(x) 单独设置 x 的缩放
transform:scaleY(y) 单独设置 y 的缩放

3.旋转
改变元素在页面中的角度
transform:rotate(ndeg);
n 为正 顺时针
n 为负 逆时针
注意：
a.转换原点会影响最后的旋转效果。
b.旋转是连同坐标轴一起旋转的，会影响旋转后的位移效果。

练习

d1 先旋转 45 度，然后位移 200px
d2 先位移 200px，再旋转 45 度

4.倾斜
改变元素在页面中的形状
transform:skew(x),等同于 skewX(x)
让元素向着 x 轴发生倾斜，实际上改变的 y 轴的角度
+逆时针
- 顺时针
transform:skewY(y)
让元素向着 y 轴发生倾斜，实际上改变的时候 x 轴的角度
+ 顺时针
- 逆时针
transform:skew(x,y)

练习

06_ex 创建 200*200 的 div
设置背景颜色，鼠标悬停时，该元素向右偏移 200px;
向下偏移 200px；旋转 135deg，向着 x 轴倾斜 45 度，放大 1.5 倍

4.3D 转换

浏览器不支持 3D 位移，z 轴看不到，只能模拟
①透视距离
模拟人的眼睛到 3D 转换元素之间的距离
perspective:value;
该属性要加载转换元素的**父元素**上

②3D 旋转
transform:rotate3D(0,0,1,20deg);
取值:
a.rotateX(xdeg)
以 x 轴为中心轴，旋转元素的角度(老式爆米花机，烤羊腿)
b.rotateY(ydeg)
以 y 轴为中心轴，旋转元素的角度(旋转门)
c.rotateZ(zdeg)
以 z 轴为中心轴，旋转元素的角度(风车，摩天轮)
d. rotate3D(x,y,z,20deg);
x,y,z 取值大于 0 时，表示该轴参与旋转
取值为 0，表示不参与旋转

三.过渡

CSS 属性值，在一段时间内容平缓的变化

transition-property:background;
border-radius;
all;
能够使用过渡效果的属性
a.颜色
b.取值为数字的属性
c.转换
d.阴影
e.渐变
f.visibility

指定过渡持续时间
transition-duration:1s/1000ms

简写方式
transition:all 1s;
transition:1s;----不要写在 hover 中，写在元素本身的样式里

作业：
1.把转换的脑图自己总结
2.预习一下 boot
3.完成学子首页，（尽量使用弹性布局）

BootStrap day01

一.过渡（过度*********）

某元素的 css 属性值在一段时间内，平滑改变到另外一个值 过渡主要观察的是过程和结果

设置能够过渡的属性

支持过渡效果的样式属性 颜色的属性 /取值为数值 /transform/渐变 /visibility/阴影
指定本次过度生效的属性 transition-poperty:上面的 css 属性/all 指定过度的时长 transition-duration:过度的时长/s/ms 指定过度时间曲线函数 transition-timing-function: a.ease 默认值，慢-->快-->慢 慢速开始，快速变快，慢速结束 b.linear 匀速 c.ease-in 慢-->快 慢速开始，快速结束 d.ease-out 快速开始，慢速结束 e.ease-in-out 慢速开始，先加速再减速，慢速结束
指定延迟执行过度的时间 transition-delay: s/ms
过度属性的编写位置 a.将过度放在元素声明的样式中(元素自己的样式里)，过度效果有去有回 b.将过度放在元素的触发操作中(hover),过度效果有去无回
过渡的简写 transition：property duration timing-function delay; 最少的方式 transition: duration;

练习：翻滚吧，牛宝宝

二.动画

使元素从一种样式，改变到另外一种，再改变到其他样式..... 相当于将很多个过渡效果放到一起使用
关键帧 a.动画的执行时间点 b.该时间点上的样式

动画的实现步骤

1.声明动画及动画关键帧
@keyframes 动画名称{
//定义关键帧
0%{动画开始时的样式}
.....
100%{动画结束时的样式}
}
2.调用动画
animation-name: 动画名称;
animation-duration: 动画播放一个周期的时间
3.动画的其他属性
animation-delay
4.动画的速度时间曲线函数
animation-timing-function
取值:
ease/linear/ease-in/ease-out/ease-in-out

5.animation-iteration-count 指定动画的播放次数 取值: 具体的数字/infinite 无限次
6.animation-direction: 动画的播放方向 取值: normal 正常 0%--100% reverse 逆向播放 100%-- 0% alternate 轮流播放 奇数次正向播放 偶数次逆向播放
7.简写方式 animation: name duration timing-function delay iteration-count direction;
8. animation-fill-mode 指定动画播放前后的显示状态 a.none 默认值 b.forwards 动画完成后，保持在最后一个关键帧上 c.backwards (需要有 delay) 动画开始之前，保持在第一个关键帧上 d.both,同时设置 forwards 和 backwards
9.animation-play-state 播放状态 a.paused 暂停 b.running 播放

10.动画的兼容性

如果要兼容低版本浏览器，需要在声明动画的时候加前缀

@keyframes 动画名称{}

@-webkit-keyframes

@-moz-keyframes

@-o-keyframes

三.CSS 优化

目的：减少服务器压力

缩短服务响应时间

提升用户体验

1.优化原则

a.尽量减少 HTTP 请求的个数

b.页面顶部引入 css 文件

c.将 css 和 js 放到外部独立的文件夹中

2.CSS 代码优化

a.缩小样式文件

b.减少样式的重写（合并样式）

c.避免出现空的 src 和 href

d.选择更优的样式属性值（能使用复合，简写的写法，就不要单独定义）

e.代码压缩

f.不要在 html 里缩放图片

二.BootStrap

www.bootcss.com

1.响应式布局(重点**********)

Responsive web page 响应式/自适应的网页

可以根据浏览器的设备不同(pc,pad,phone)

自动调用对应的布局，图片，文字效果，从而不会降低用户体验

2.响应式网页必须做到的前提

a.布局：不能固定宽度，必须是**流式布局**(尽量少用定位，可以浮动)

b.文字和图片，大小随着容器大小而改变 em rem(**可伸缩的文字和图片**)

c.媒体查询技术

响应式网页存在的问题：

页面的复杂度极大的增加

只适用于内容不太多的页面(企业的官网，门户网站)

媒体查询技术属于 h5/c3 的技术，

boot 把媒体查询这件事，封装了，不需要我们自己写了

3.如何测试响应式网页

a.使用真实设备测试

好处：真实，可靠

缺点：测试任务量巨大

b.使用的三方的模拟测试软件

好处：无需添置太多真实设备，测试方便

缺点：测试效果有限，有待进一步验证

c.使用浏览器自带的模拟器测试

好处：简单方便

缺点：测试效果有限，需要进一步验证

4.如何编写响应式布局

a.在元数据标签中定义 viewport---视口

name="viewport"

content 设置能够允许网页进行操作

width=device-width 表示视口宽度就是设备宽度

initial-scale=1.0 表示视口宽度是否可以缩放 1.0 不能缩放

maximum-scale=1.0 允许缩放的最大倍率

user-scalable=0 是否允许用户手动缩放 yes/no/1/0

一般的设置

<meta name="viewport" content="width=device-width, initial-scale=1">

b.所有内容/文字/图片，使用相对尺寸，不能使用 10px 这种绝对的值

c.流式布局+弹性布局，在搭配媒体查询技术来完成响应式布局

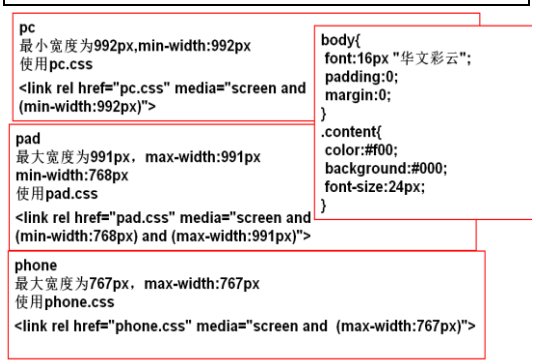
float flex

d.使用 css3 Media Query 技术做响应式网页

Media：媒体，指浏览网页的设备。

如：screen(pc/pad/phone) tv print

Media Query:媒体查询，可以自动根据当前浏览器设备的不同（尺寸，解析度，方向不同），有选择执行一部分 CSS 而忽略其他部分的 css



注意，此方案使用较少
所有的 css 文件都会被加载，每个 css 文件中，存在很多重复代码

根据媒体查询的结果，执行同一个 css 文件下的不同代码块

```
@media screen and (min-width:768px) and (max-width:991px){
    选择器{样式}}
```

BootStrap day02

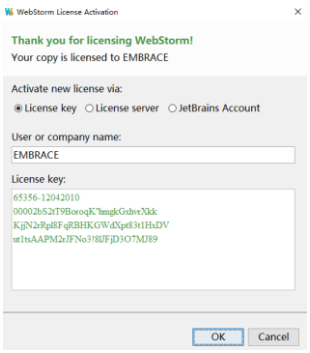
Bootstrap 用于开发响应式布局、移动设备优先的 WEB 项目

调试，使用 webstorm

a.注册码生成器

b.安装包

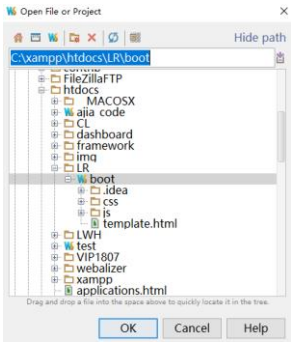
c.注册---help-->register



创建项目

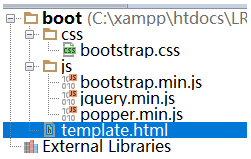
在 xampp--htdocs--创建文件夹(LR)---创建项目(boot)

使用 ws 中的 open 打开文件夹



在 boot 项目中。创建文件夹 css 和 js

把 boot 的 4 个文件复制进去



设置字体大小和颜色

在 file--->setting-->输入 font 可以改变字号

color & fonts---->html 的选项，可以更改 html 文件的字体颜色

修改 tab

general-->editor tab ---->修改成 2

修改快捷键

keymap 中选择 eclipse

所有使用 boot 的代码都必须放在 container 类的包裹中

一. 布局容器

container 定宽容器

fluid 变宽容器

二. 全局 css 样式

1.按钮相关的 class

btn 基本类

a.颜色按钮

btn-danger	危险
btn-success	成功
btn-warning	警告
btn-info	信息
btn-primary	主要
btn-secondary	次要
btn-dark	黑色
btn-light	浅色
btn-link	链接

b.不同的边框按钮

btn-outline-颜色

c.按钮大小

btn-lg	大按钮
btn-sm	小按钮
btn-block	块级按钮

2.图片相关的 class

.rounded	倒角
.rounded-circle	圆
.img-thumbnail	缩略图
.img-fluid	响应式布局，图片会缩放，但不会超过原始大小

3.文字相关

字体大小

.h1~.h6

字体颜色

text-danger/success/info/primary/warning...

文字对齐
text-*-left/right/center/justify /nowrap
*代表：lg/ sm 在大屏幕下对齐，在小屏幕下对齐
大小写
text-uppercase/lowercase/capitalize
大写 小写 首字母大写
粗体
font-weight-bold
斜体
font-italic

4.列表相关的 class

ul 中：
list-unstyled 去除标识符
list-group 创建列表组，必须写
li 中：
list-group-item 创建列表项，必须写
active 激活状态
disabled 禁用状态
li 的颜色
list-group-item-danger/success/.....

5.table 相关 class

table 的 class
table 基本类
table-bordered 带边框的表格
table-striped 隔行变色
table-hover 鼠标悬停效果
table-danger/warning.....
响应式表格
需要在 table 的父元素上，添加类 table-responsive

6.辅助类

边框
border 基础类
border-top/right/bottom/left 设置 4 个方向的边框
border-0/border-top-0/border-right-0...
取消边框
border-danger/warning... 边框颜色
rounded/rounded-0 设置/取消圆角
rounded-top/right/bottom/left

浮动
float-*-left/right *：xl/lg/md/sm
clearfix

显示
visible/invisible 显示/隐藏

背景
bg-danger/warning.....

居中对齐
mx-auto

margin
m-*(0/1/2/3/4/5)
(0/0.25/0.5/1/1.5/3rem)
mt-*/mr-*/mb-*/ml-*

padding
p-*(0/1/2/3/4/5)
(0/0.25/0.5/1/1.5/3rem)
pt-*/pr-*/pb-*/pl-*

宽度/高度
w-*(25/50/75/100)
h-*同上

三. 栅格布局(重要**********)

通过一系列的行和列的组合来创建页
container/container-fluid(容器)->row->col(12)
内容放在 col，只有列可以做 row 直接子元素

1.web 页面布局的方法

	table 布局	div+css 布局	BootStrap 中的栅格布局
优点	简单，容易控制	语义正确，渲染效率高	简单，容易控制，语义正确，渲染效率高，支持响应式
缺点	语义错误 渲染效率低(最后一个数据加载完成后，才能统一开始渲染)	控制比较麻烦 媒体查询写起来更繁琐	页面如果太复杂，不适合使用

总结：

1.最外层需要使用容器
.container 定宽容器 最大宽度是写死的
.container-fluid 变宽容器 宽度是父元素的 100%
2.容器中声明.row 弹性布局(一行内等分为 12 个单元格)
3.行中声明列 col-n(每个列都需要制定宽度占比 n/12)

BootStrap day03

一.栅格

col-xl-1/2/3/4... xl:Extra Large w>=1200px
col-lg-1/2/3/4... lg:large w>=992px
col-md-1/2/3/4... md:medium 768<=w<=991
col-sm-1/2/3/4... sm:small 576<=w<=767
xs:Extra small w<=575
col 只写 col，每一列平局分配

不同屏幕下，列的适应性问题
col-xl-* 只对超大屏生效
col-lg-* 对 xl/lg 生效
col-md-* 对 xl/lg/md 生效
col-sm-* 对 xl/lg/md/sm 生效

列偏移
可以通过列偏移类实现指定列和后面列的偏移。
offset-* (*:0~11)
offset-xl/lg/md/sm-*

栅格嵌套

二.弹性布局

d-flex 或者 d-inline-flex 创建一个弹性容器
flex-row 不同屏幕下主轴方向 x
flex-column 不同屏幕下主轴方向 y
lex-row-reverse
flex-column-reverse
justify-content- end/center/between /around

响应式 flex 类
d-*-flex
flex-*-row
flex-*-row-reverse
flex-*-column
flex-*-column-reverse
justify-content-*- start/end/center /between/around

三.表单

在 form 表中写的类
方向的排列
垂直表单
form-group+form-control 堆叠，垂直方向排列,宽度 100%
水平表单
form-inline 内联表单 水平排列

输入框
form-control

form-check 写在单选按钮/多选按钮的父级容器中，有相对定位
form-check-inline 写在单选按钮/多选按钮父级容器中，自带弹性布局，交叉轴垂直居中
form-check-input 自带绝对定位，需要与上面 form-check/ form-check-inline 配合使用
form-check-label 底外边距为 0
form-text 上外边距为 0.25rem

	默认的栅格布局	水平表单
行	div.row	div.form-inline
列	div.col	div.col

四.组件

boot 把一些功能性的模块，进行了封装
我们只需要按照 boot 要求的层次结构，以及类名套用，就可以完成

1.下拉菜单

①外部大包裹.dropdown
```
<div class="dropdown">
```
②内部菜单按钮
class="dropdown-toggle" 作用，画向下的小箭头
事件的激活 自定义属性 data-toggle="dropdown"
```
<button class="btn btn-primary dropdown-toggle" data-toggle="dropdown">
```
③菜单栏，div.dropdown-menu
```
<div class="dropdown-menu">
```
④菜单栏中有菜单项 a.dropdown-item
```
<a href="#" class="dropdown-item">
```
⑤菜单项有几个可选的类名
dropdown-header `<h5 class="dropdown-header">`
active `<a href="#" class="dropdown-item active">`
disabled `<a href="#" class="dropdown-item disabled">`
⑥在每个 item 之间添加分割线
```
<div class="dropdown-divider"></div>
```

2.按钮组

基本按钮组

外包裹添加 class="btn-group"

把内部的显示成一组

按钮工具栏

btn-toolbar

按钮尺寸

使用 btn-group-lg/sm 设置按钮组大小

使用 btn-group-vertical 设置垂直按钮组

3.信息提示框

1.外包裹 class="alert"

可以修改颜色 alert-danger/warning/info

如果内部的 x 想使用类.close，父级要添加 alert-dismissible

** **

2.内部有文本提示

3.内部有关闭提示框的 X.

data-dismiss="alert" class="close"

****

4.导航

(1)水平导航

①.ul 必须有类名 .nav .nav-justified

.nav-justified 本身没有任何样式，为了给子元素 item 引路

②.li class="nav-item"

③.a class="nav-link"

④.其它样式

ul 中，如果有.nav-justified，意味着导航项等宽显示

如果想使用 justify-content-*做主轴的对齐，需要把.nav-justified 删掉

flex-row/column/reverse 都可以生效

(2)选项卡导航

①.外包裹 ul class="nav nav-tabs"

`<ul class="nav nav-tabs"`

②.ul 内部需要有两套

2.1 选项卡

li class="nav-item" `<li class="nav-item">`

a class="nav-link" `<a class="nav-link"`

2.2 对应内容

div class="tab-content" tab-content 对自己没有任何修饰，为了给内部子元素 .tab-pane 引路`<div class="tab-content">`

.tab-content> elem class="tab-pane" 组合之后，elem 就隐藏了

如果想显示 .tab-content>.active

`<div class="tab-pane active">`

③.点击选项卡的动作

a.nav-link 有事件的激活，使用自定义属性 data-toggle="tab"实现切换

`<a href="#apple" class="nav-link active" data-toggle="tab">`

④.把选项卡的 href 和内容 id 进行绑定

内容的 id `<div id="apple" class="tab-pane active">`

与选显卡 a 标签的 href 绑定

`<a href="#apple" class="nav-link " data-toggle="tab">`

⑤.页面刷新，默认显示的选项和内容没有设置

选项卡需要 active

`<a href="#apple" class="nav-link active" data-toggle="tab">`

对应的内容也需要 active

`<div id="apple" class="tab-pane active">`

作业：

1.完成选项卡导航

2.完成 boot 中 form 操作

3.完成 boot 中弹性布局

BootStrap day04

一.组件

1.导航

(1)胶囊导航
.nav-pills 替代选项卡导航的 nav-tabs
data-toggle="pill"
其它部分同选项卡导航

2.导航栏

在 ul 外层包裹一个 div，就把导航变为了导航栏
①外层包裹，div.navbar 弹性+相对定位
后面紧跟 navbar-expand-lg/md/sm (配合的类 ul.navbar-nav)
此处的响应式，是设置导航在某一个屏幕下，横向显示，
其它屏幕纵向显示
②导航可以使用 ul。添加.navbar-nav(默认弹性，主轴为 column)
配合之前的 navbar-expand-lg/md/sm，可以使主轴变为 row
③li.nav-item>a.nav-link
注意：在外包裹中，可以使 bg-color，设置导航栏颜色
整个导航栏横向显示，还是纵向显示，是靠 navbar-expand-lg/md/sm 控制的

3.折叠

控制内容的隐藏与显示
在 button 中添加 data-toggle="collapse" 和折叠目标 data-target="#内容 id" #内容 id 对应折叠内容的 id
如果使用 a 标签，直接在 href 中填写对应折叠内容的 id
在内容中添加 id 和 class="collapse"实现折叠

4.折叠导航栏

①外部包裹 div .navbar .navbar-expand-* .bg-dark .navbar-dark
.bg-dark .navbar-dark 保持一致，都是 dark/light
.navbar-dark 本身没有样式，为子元素选择器引路
影响的子元素有.navbar-brand .nav-link .navbar-toggler .navbar-toggler-icon
②外包裹内部
2.1 不折叠的选项 a. navbar-brand
2.2 折叠按钮 button.navbar-toggler .navbar-expand-* .navbar-toggler{设置什么时候显示/隐藏}
btn 需要写自定义属性 data-toggle= "collapse" data-target="#折叠内容 id"
2.3 内容 div.collapse .navbar-collapse id="折叠内容的 id"
.navbar-collapse 设置折叠导航打开的位置
2.3.1 导航的内容
ul.navbar-nav>li.nav-item>a.nav-link

5.卡片

外包裹 div class="card"
内部 3 部分
1.div class="card-header"
2.div class="card-body"
3.div class="card-footer"
在卡片中可以添加超链接，a class="card-link"

6.折叠+卡片=手风琴效果

1.卡片的 body 需要包裹在一个 div 内部，不然会卡顿
2.使用 data-parent 属性来确保所哟逃折叠，在指定的父元素下只能显示出一个

```
<h1>卡片+折叠=手风琴效果</h1>
<div id="p1">
  <div class="card">
    <div class="card-header">
      <a href="#c1" class="card-link"
        data-toggle="collapse">卡片一
</a>
    </div>
    <div id="c1" class="collapse" data-
parent="#p1">
      <div class="card-body ">
        内容一
      </div>
    </div>
  </div>
  <div class="card">
    <div class="card-header">
      <a href="#c2" class="card-link"
        data-toggle="collapse">卡片二
```

```
</a>
    </div>
    <div id="c2" class="collapse" data-parent="#p1">
      <div class="card-body ">
        内容一
      </div>
    </div>
  </div>
  <div class="card">
    <div class="card-header">
      <a href="#c3" class="card-link"
        data-toggle="collapse">卡片三
</a>
    </div>
    <div id="c3" class="collapse" data-parent="#p1">
      <div class="card-body ">
        内容一
      </div>
    </div>
  </div>
  <div class="card">
    <div class="card-header">
      <a href="#c4" class="card-link"
        data-toggle="collapse">卡片四
</a>
    </div>
    <div id="c4" class="collapse" data-parent="#p1">
      <div class="card-body ">
        内容一
      </div>
    </div>
  </div>
</div>
```

7.媒体对象

使用 div class="media"把内容包裹起来，形成对象的显示方式
.media：定义弹性布局
内部元素可以使用 align-self-center/end/.....对齐排列

练习：

8.轮播

外包裹
div class="carousel" data-ride="carousel"
创建轮播，使用 carousel 类，
让图片动起来 data-ride="carousel"

内部第一层，轮播图片

img
img
img

需要使用 div 把所有图片包裹，使用类名 carousel-inner
每一个 img 还需要包裹一个 div,使用 carousle-item
carousle-item->display:none;把所有图片隐藏
.carousel-item.active 会把隐藏的 div 显示
在 carousel-item 内部,可以添加 div.carousel-caption 添加文字

内测第二部分，导航标识符
使用 ul class="carousel-indicators"
.carousel-indicators>li 样式已经写好,并且随着图片轮播，改变表示
但是不能点击便是改变图片
li class="active" 背景变为白色，被激活的 li
点击 li，改变图片
在 li 中 data-slide-to="0" 图片的下标
data-target="#当前轮播图 ID"

内部第三部分：左右箭头
a class=" carousel-control-prev"
画的左箭头
a class=" carousel-control-next"

画的右箭头

需要事件，data-slide="prev/next" data-target="#轮播图 ID"

总结：bootstrap 为我们提供了比较多的样式类，如果提供的样式不符合我们的需求，我们可以灵活的取修改样式

9.模态框

模态框(modal)是覆盖在父窗体上的子窗体

模态框可以在不离开父窗体的情况下有一些互动，提供一些交互的信息

```
<button data-toggle="modal" data-target="#myMd"> </button>
<!-- 模态框的内容-->
<div id="myMd" class="modal">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header"></div>
      <div class="modal-body"> </div>
      <div class="modal-footer">
        <button class="btn btn-warning" data-dismiss="modal">关闭</button>
      </div>
    </div>
  </div>
</div>
```

二.其它组件

1.徽章

类似于一个个小型按钮（添加了颜色和圆角）

基础样式 badge

颜色 badge-info/warning/danger/.....

样式 badge-pill

2.巨幕 jumbotron

其实就是巨大的内边距，灰色的背景

div class="jumbotron"

3.面包屑导航

按路径导航，用于有层次关系的导航

ul.breadcrumb>li.breadcrumb-item>a

修改连接的"/",需要重新编写样式

```
.breadcrumb-item + .breadcrumb-item::before {
  content: ">>>";

}
```

4.分页

最外层 ul

.pagination--->弹性，去点，圆角

.pagination-sm/lg--->/*响应式布局，让内部第一个元素和最后一个元素添加圆角（第一个元素添加左侧，最后一个元素添加右侧）*/

设置分页条大小

内部 ul>li .page-item

li>a .page-link .acitve .disabled

5.进度条

进度条的槽 div.progress

进度条：

div.progress-bar 普通进度条

通过 w-*设置进度 bg-* 进度条颜色

.progress-bar-striped 条纹进度条

.progress-bar-animated 带动画

可以在同一个槽里添加多个进度

BootStrap 重点 媒体查询技术+栅格布局+SCSS

三.SCSS，动态的样式语言

boot 的 SCSS 文件夹中，有很多_*.scss 文件。会被引入进一个不带_开头的 scss 文件中，最后会把这个文件生成 css 样式表

浏览器不认识 scss，只认识 css

1.CSS 有几个缺点

a.语法不够强大，没有变量和合理样式复用机制

b.逻辑上相关的属性值必须以字母的形式重复输出

c.难以维护

d.动态样式语言为 css 赋予了动态语言的特性极大了提高了样式语言的可维护性

常见的动态样式语言

a.SCSS/Sass (scss 兼容 sass 的；更接近 css 的写法)

b.styleus

c.Less

这些动态语言需要预处理器---浏览器不认识动态样式语言

预处理器：用一种专门的编程语言，进行 web 页面设计

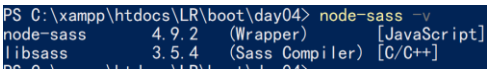
在通过编译器转化为正常 css 文件，以供项目使用

2.SCSS 的作用

有助于更好管理样式文件，以及更高效的开发项目

3.使用 scss

在服务器端使用的
(1)安装 nodejs 解释器，需要 node8.11 以上的版本
(2)安装 scss 编译程序
在线安装 npm install -g node-sass
使用 scss 压缩包安装
把 scss 文件夹中的 4 个文件，复制到 nodejs 文件夹下
如何检测是否安装成功?
在 webstrom 中 alt+f12 输入 node-sacc -v 检测版本
或者
打开项目路径，在项目文件夹中 shift+右键，打开黑窗口，检测版本



在项目中创建 scss 文件夹，然后在内部创建 01.scss，编写代码

```
$jd_red:#f10215;
$my_border:1px solid $jd_red;
$my_width:50%;
p{
  width:$my_width;
  border:$my_border;
  background: $jd_red;
}
```

保存后，进入项目文件夹，shift+右键打开黑窗口
单文件转换命令
node-sass scss文件夹名称/01.scss css文件名名/01.css
多文件转换命令
node-sass scss 文件夹名称 -o css 文件夹名称
让转换程序处于监听状态，保存 scss 文件，都会自动生成一次 css

BootStrap day05

一.SCSS
1.SCSS的使用

单文件转换 node-sass 文件夹/01.scss css文件夹/01.css
批量转换 node-sass scss文件夹 -o css文件夹
单文件监听 node-sass -w scss文件夹/01.scss css文件夹/01.css
多文件监听 node-sass -w scss文件 -o css文件夹

2.SCSS 的基础语法
(1)变量

使用$表示变量
变量名的命名规范，尊徐 css 选择器的命名规范
可以包含- ,_,尽量的见名知意
$jd_red:#f10215; 颜色变量
$normal-width:100px; 数值变量
$befor-content:"子曾经曰过："; 字符串变量
$border-solid:solid; 样式变量
$border1:$my_width solid $my_color; 声明变量的同时，变量的值也可以引用其他变量

```
div{
  $my_width:200px;
  width:$my_width;
}
  p{
  width: $my_width;
}
```

这样报错，**$my_width**:200**px**;使用范围是在大括号内部。

声明变量，变量值可以应用其他变量
变量定义在{}外部，整个样式文件中都可以使用
如果定义在{}内部，只能当前{}中使用

重复变量，后声明的变量会覆盖前面的变量，尽量的不要重复
$my_width:600**px !default**;
！default 规则：如果前面这个变量已经声明赋值了，就用他之前的值。
如果没有声明赋值，用现在的赋值

(2)嵌套---解决后代选择器

在 scss 可以向 css 一样，使用后代选择器

```
#content div.top {}
#content div.top h1 {}
#content div.top p {}
#content div.top p a {}
#content div.middle {}
```

这种写法导致代码重复量很大，所有 scss 推荐嵌套的写法

```
#content{
  div.top{
    h1{color:$a1;}
    p{
      a{}
    }
  }
  div.middle{}
}
```

此写法转换成 css 之后，写法与上面一样
明确表现出层次关系，无代码冗余

关于伪类的嵌套
scss 支持

```
a{}
a:hover{}
```

但是不推荐，因为这种写法，没有标明 a 标签的关系

```
a{
  color:#fff;
  &:hover{font-size: $a2;}
}
```

在 a 标签的内部嵌套写 hover，使用&符号代替 a 标签本身

(3)群组选择器的嵌套

```
nav,aside,footer{
  a{color:#444}
}
```

(4)属性嵌套

```
nav{
border:{style:solid;width:38px;color:#fff
;}
}
```

(5)scss 的导入

在 scss 中，局部文件，命名以下划线开头
_jumbotron.scss
scss 在编译时不会编译局部文件(不会编译以下划线开头的文件)
需要使用一个正常的 scss 把局部文件导入，编译这个正常的 scss 文件就可以
引用时，使用关键字@import，引用的局部文件名，省略下划线 "jumbotron"
省略后缀.scss

```
@import "jumbotron";
```

在一个局部 scss 文件中，定义一套基础样式值/颜色。
其它要生成 css 的 scss 文件去引用这个局部 scss 中的变量

_myColor.scss

```
$myRed:#f10215;
  $myBlue:#0aa1ed;
  $myYellow:#ff0;
```

在 02.scss 中引用局部文件_myColor.scss

```
@import "myColor";
  #d2{color:$myRed;}
```

这样就可以使用了

(6)混合器

使用@mixin 标识符来定义
@mixin 混合器名称{需要反复重用的样式声明}
使用混合器
@include 混合器名称;

带参数的混合器
定义：@mixin 混合器名称(形参 1，形参 2，形参 3){ 属性 1：形参 1；属性 2：形参 2；属性 3：形参 3；}
使用：@include 混合器名称(实参 1，实参 2，实参 3);

(7)继承

一个选择器可以继承另一个选择器定义的所有样式
@extend 实现继承

```
.my-rounded{
  border:1px solid red;
  border-radius: 2px;
}
.rounded_shadow{
  @extend .my-rounded;
  box-shadow: 0 0 10px;
}
```

练习

在选择器.my_border 中定义 1px solid 透明边框
设置宽度为 400px，距离左边有 20px 的外边距
在选择器.my_bottom_border 中继承.my_border。
并设置底框为红色，当鼠标悬停底边框为蓝色

二.运算

1.数字

加减乘除模，取整等运算 + - * / %

如果必要的情况下，会在不同的单位间转换(scss 支持)

rem 和 em 转换不了

(1)加法

+可以做字符串连接使用

如果第一个数值没有双引号，结果没有双引号

如果第一个数值有双引号，结果是有双引号的

(2)除法

在 scss 中通常"/"起到分割数字的作用

同时也具备除法运算功能

在以下情况视为除法运算

a.如果值，或者值的一部分，是变量或者函数返回值

```
width:$width/2;
```

b.如果值被圆括号包裹

```
height:(500px/2);
```

c.如果值是算术表达式的一部分

```
margin-left:5px+8px/2px;
```

其它时候，都认为/是分隔符

(3)运算表达式与其它值连用时，用空格做连接

```
margin:20px+12px auto;
```

(4)在有引号的字符串中，使用#{}插值语句可以添加动态的值

```
content:"i ate #{3+3} pies"
```

2.颜色

颜色值的元素是分段计算的，也就是分别计算红色，绿色，蓝色的值

#112233+#112233=#224466

11+11 22+22 33+33

```
background:
rgba(255,0,0,0.5)+rgba(0,255,0,0.5);
```

颜色值包含 alpha，必须有相等的 alpha 值才能进行计算

算术运算符不会作用于 alpha

练习

定义两个变量分别是 10px,和 20px;

定义两个颜色分别是#333 和#666

两个变量相加后乘以 2，赋值给 p 选择器 width

两个变量相加后除以 2,赋值给 p 选择器 height

两个颜色相减，赋值给字体颜色

三.函数

1.SCSS 定义了多种函数，有些函数可以直接被 css 语句调用

（1）颜色函数

rgba(red,green,blue,alpha)

hsl(hue,saturation,lightness);

hue:色调 取值 0-360 3 个色段,每 120 一个

saturation：饱和度 取值 0.0%--100.0%

lightness：亮度 取值 0.0%--100.0%

（2）数字函数

round($value) 四舍五入

ceil($value) 向上取整

floor($value) 向下取整

min($value1, $value2, $value3,.....)找出几个数的最小值

max($value1, $value2, $value3,.....)找出几个数的最大值

random() 获取随机数

（3）字符串函数

To-upper-case

To-lower-case

unquote($string) 去掉字符串双引号

quote($string)

2.自定义函数

scss 支持自定义函数

使用关键字 @function 函数名(参数){@return 结果}

函数名($value);

练习

定义一个带参数函数 get_mywidth($n)

函数体定义两个变量，60px 和 10px

函数最后返回结果 $n 乘以最大值，加上$n-1 乘以最小值

并将结果赋值给 div 的高度

四.控制指令

当@if 的表达式返回值不是 flase 和 null 的时候执行{}内代码

```
@if(1+1==2){border-radius:5px;}
@if(5<3){border-radius:10px;}
@if(null){border-radius:15px;}
```

多重 if

@if 块后面可以跟多个@else if 块，或者跟一个@else 块

```
@if($type==sun){
  background:#f00;
}@else if($type==m){
  background: #ff0;
}@else if($type==mark){
  background: #f0f;
}@else {
  background:#0ff;
}
```

练习

声明一个函数，在函数中判断 如果参数大于 1，返回 200px 如果参数==1 返回 100px 否则返回 0。在 div 中，给 width 赋值为该函数的返回值

项目

使用 bootstap 完成学子商城 本来的学子商城没有使用响应式，也没有使用 boot 我们做的项目效果会与原网站有差别 通过服务器打开项目

BootStrap day06

学子商城项目

首页 商品列表 商品详情页

思路

原网页没有使用响应式
我们使用 boot 和响应式
1.确定大体位置和结构布局（宽高） 2.边框，背景，圆角 3.文字相关(字体，对齐) 4.hover，过渡，动画 5.微调 从上往下，从外往内，从左往右

作业：

1.使用 boot 响应式完成学子首页

2.尽量完成学子商城，产品列表页

BootStrap day07

原网页没有使用响应式
我们使用 boot 和响应式
1.确定大体位置和结构布局（宽高） 2.边框，背景，圆角 3.文字相关(字体，对齐) 4.hover，过渡，动画 5.微调 从上往下，从外往内，从左往右

目标

学子商城主页

产品列表页

产品详情

product_details.html

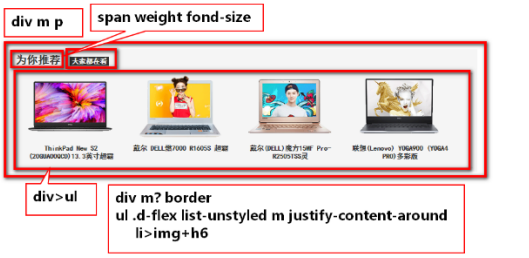
15 分钟，14:51 我写

1.导入头和脚

2.完成中间部分的面包屑导航

作业

1.完成商品详情页，无缝滚动（只写样式不写滚动）



2.完成产品详情页，下面参数规格部分，提高题