day18

四.常用组件

11.焦点轮播

|  |
| --- |
| 1.图片部分  div.carousel>div.carousel-inner>div.carousel-item\*4>img  .carousel 相对定位  .carousel-inner 溢出隐藏 w100  .carousel-item display:none  .carousel-item.active display:block  事件  给最外层div.carousel添加自定义属性  data-ride="carousel" |
| 2.左右箭头  a.carousel-control-prev>span.carousel-control-prev-icon  a.carousel-control-next>span.carousel-control-next-icon  .carousel-control-prev, .carousel-control-next定义了左右箭头的大小位置，hover等。但是这些样式不符合我们的需求，需要重写  .carousel-control-prev,.carousel-control-next{  width:4%;height:20%; top: 40%;  background-color:#aaa;border-radius: 0.25rem;  }  事件  data-slide="next/prev" 注意，事件的目标是最外层大div  所以href="#demo" demo是div.carouosel的id |
| 3.轮播指示器  ul.carousel-indicators>li  .carousel-indicators 设置ul 为弹性 定位  .carousel-indicators li 设置 li的宽高，背景，位置  我们可以按照自己的需求，修改li的样式  .carousel-indicators li{  width:0.8rem;height:0.8rem; background:#fff;  border-radius: 50%; margin-right:6px;margin-left:6px;  }  注意，选中的li需要添加样式.active。同时，可以重写active  .carousel-indicators .active{  background:#0aa1ed;  }  事件，每一个li添加data-slide-to="图片编号" 从0开始  添加事件目标 data-target="#demo" |

12.模态框

|  |
| --- |
| 模态框modal是覆盖在父窗体上的一个子窗体（类似js的alert方法）  模态框可以在不离开父窗体的情况下，与用户有一些互动  <div class="modal">  <div class="modal-dialog">  <div class="modal-content">  <div class="modal-header"></div>  <div class="modal-body"></div>  <div class="modal-footer"></div>  </div>  </div>  </div> |

五.其它组件

1.徽章(看成小按钮)

|  |
| --- |
| 基本类 badge  颜色 badge-danger/warning.....  胶囊徽章 badge-pill |

2.巨幕

|  |
| --- |
| 巨大的内边距，有背景颜色和圆角  div.jumbotron |

3.面包屑导航

|  |
| --- |
| ul.breadcrumb>li.breadcrumb-item>a  .breadcrumb 设置ul弹性，背景色，圆角  .breadcrumb-item |

4.分页条

|  |
| --- |
| ul.pagination>li.page-item>a.page-link  li有修饰类 .active .disabled |

5.进度条

|  |
| --- |
| div.progress>div.progress-bar.w-50.bg-danger  子div的宽度，就是进度条的进度  子div的背景颜色，就是进度的颜色  带条纹的滚动条 progress-bar-striped  带动画的滚动条 progress-bar-animated |

六.SCSS boot的重点，响应式+栅格+scss

|  |
| --- |
| SCSS是动态样式语言  安装scss 32位nodejs8.11以上，64位nodejs10.0以上 |

1.css有几个缺点

|  |
| --- |
| css语法不够强大，没有变量和合理的样式复用机制  导致难以维护  我们可以使用动态的样式语言，赋予css新的特性，  提高样式语言的可维护性  常用的动态样式语言  1.scss/sass（scss兼容sass，scss更接近css语法）  2.stylus  3.Less |

2.scss是什么

|  |
| --- |
| scss是一款强大的css辅助工具  它和css语法很像，  在css的基础上，添加了变量，嵌套，混合，导入，函数等高级功能  这些拓展功能和命令，让css更加的强大与优雅 |

3.安装scss

|  |
| --- |
| 1.使用发给大家的scss.rar 解压缩，把所有文件复制进nodejs文件夹  2.在线安装，保证电脑安装了npm插件  npm install -g node-sass  3.验证是否安装成功。  在cmd node-sass -v 显示版本 |

4.scss文件转换成css文件

|  |
| --- |
| scss运行服务器端，  前端页面如果想使用scss，需要先把scss文件转换成css文件 |

①单文件转换

|  |
| --- |
| node-sass scss/01.scss css/01.css  注意路径问题，如果路径错误，会报错  An output directory must be specified when compiling a directory |

②多文件转换(一次转换一个文件夹中所有文件)

|  |
| --- |
| node-sass scss(文件夹名称) -o css(文件夹名称) |

③单文件的监听（修改完.scss，一保存，自动转换成css文件）

|  |
| --- |
| node-sass -w scss/01.scss css/01.css |

④多文件的监听

|  |
| --- |
| node-sass -w scss -o css |

七.scss语法

1.变量

|  |
| --- |
| 以$开头，变量可以包含-和\_,命名规则基本与选择器命名规则一致  $jd\_red:#f10215；  注意：  变量声明在{}内，只有当前选择器的样式可以使用这个变量。  !default规则，如果此变量在之前已经声明赋值了，那么就使用之前的值  如果此变量之前没有声明赋值，就使用现在的值  $jd\_red:#f00 !default; |

2.嵌套

|  |  |
| --- | --- |
| scss文件 | css文件 |
| #content{  color:red;  h1{font-size:12px;}  p{background:#00f;  button{outline:0;}  }  }  转换成css时，会自动变成后代选择器 | #content {  color: red;  }  #content h1 {  font-size: 12px;  }  #content p {  background: #00f;  }  #content p button {  outline: 0;  } |
| 伪类也可以嵌套，但是需要占位符&。不然会多一个空格  a{  text-decoration:none;  &:hover{background:#f00;}  } | a {  text-decoration: none;  }  a:hover {  background: #f00;  } |
| 单属性嵌套  div{  border:{style: solid;width:10px;color:#ff0;}  } | div {  border-style: solid;  border-width: 10px;  border-color: #ff0;  } |
| 群组嵌套  div,h2,p{  a{color:#f00;}  } | div a, h2 a, p a {  color: #f00;  } |

练习：

|  |
| --- |
| 一个200\*200pxdiv，  红色背景，鼠标悬停时，过渡到黄色背景，并且变为圆形  使用scss进行编码 |

3导入

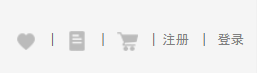
|  |
| --- |
| 在scss的语法中，如果一个scss文件以下滑线开头，那么这个文件就是一个局部scss文件  局部scss不会被转换成css文件。  一般情况下，我们会把局部scss文件导入进全局scss中.生成全局的css文件  局部scss文件\_mycolor.scss  导入语法 @import "mycolor"; 这里不要开头的\_,也不写后缀  局部文件被导入后，局部文件中样式，会在全局文件转换的css中生成，  同时局部文件声明的变量，可以在全局文件中使用 |

4.混合器

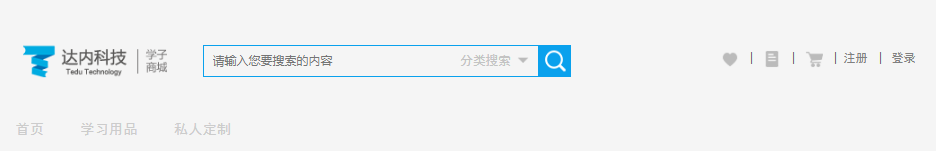
|  |
| --- |
| 把多个选择器都会使用的样式，封装进一个混合器  让需要使用的选择器去调用这个混合器。  实现代码重用，而且混合器可以带参数，更灵活  声明混合器 @mixin 混合器名称(参数1，参数2....){样式....}  调用混合器 @include 混合名称(值1，值2....);  ex:  @mixin my\_border($width,$color){  border:$width solid $color;  border-radius:50%;  box-shadow:1px 1px 1px 1px $color;  }  p{@include my\_border(1px,#ff0)};  div{@include my\_border(3px,#fff)};  #main{@include my\_border(5px,#f00)}; |

作业

1.面包屑导航



2.使用栅格完成header



3.使用栅格完成footer  


提高题：

