

拉勾教育

— 互联网人实战大学 —

《前端高手进阶》

朱德龙 前中兴软创主任工程师

— 拉勾教育出品 —

第05讲：如何管理你的 CSS 代码

如何组织样式文件

拉勾教育

— 互联网人实战大学 —

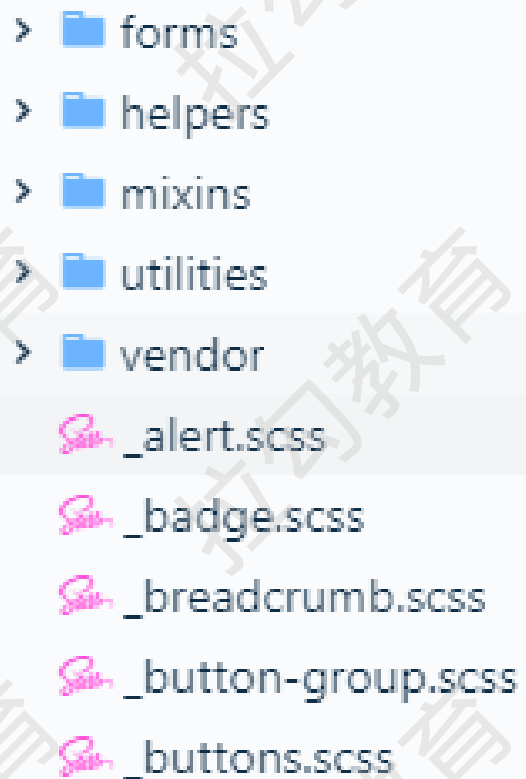
CSS 提供了 import 命令支持文件引用

但由于其存在一些**问题**（比如影响浏览器并行下载、加载顺序错乱等）导致使用率极低

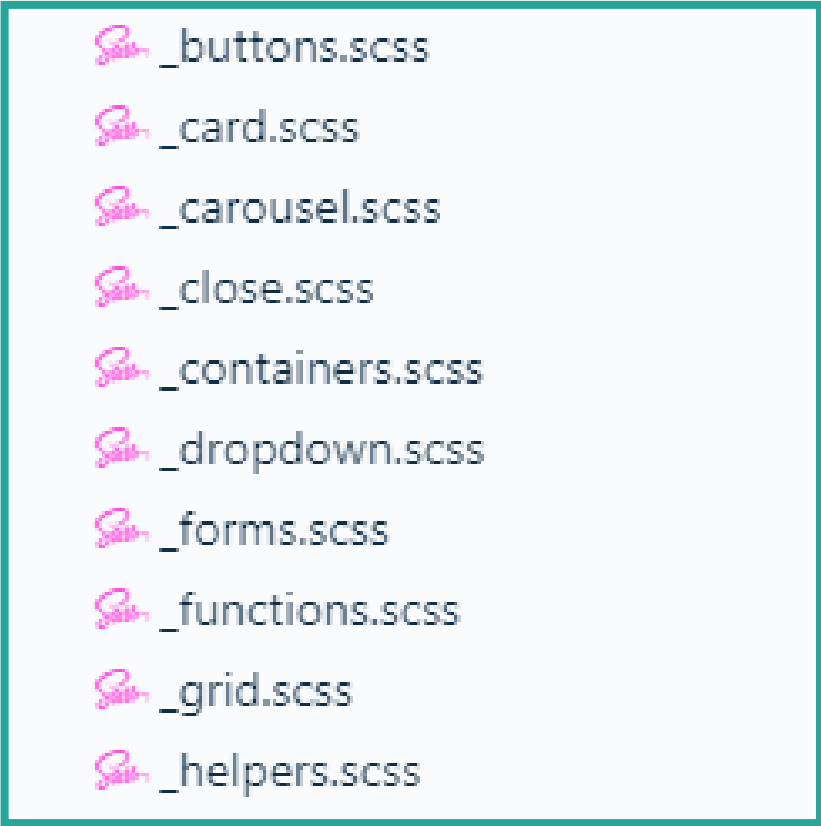
更常见的做法是通过**预处理器**或**编译工具插件**来引入样式文件











管理样式文件的目的就是为了让开发人员更方便地维护代码





- > forms
- > helpers
- > mixins
- > utilities
- > vendor
- _alert.scss
- _badge.scss
- _breadcrumb.scss
- _button-group.scss
- _buttons.scss

A rectangular box with a teal border containing a list of Sass partial files. Each file name is preceded by a small pink icon that looks like a stylized 'S' or a leaf.

-  _buttons.scss
-  _card.scss
-  _carousel.scss
-  _close.scss
-  _containers.scss
-  _dropdown.scss
-  _forms.scss
-  _functions.scss
-  _grid.scss
-  _helpers.scss

```
Salmon _helpers.scss  
Salmon _images.scss  
Salmon _list-group.scss  
Salmon _mixins.scss  
Salmon _modal.scss  
Salmon _nav.scss  
Salmon _navbar.scss  
Salmon _pagination.scss  
Salmon _popover.scss  
Salmon _progress.scss
```

 _progress.scss

 _reboot.scss

 _root.scss

 _spinners.scss

 _tables.scss

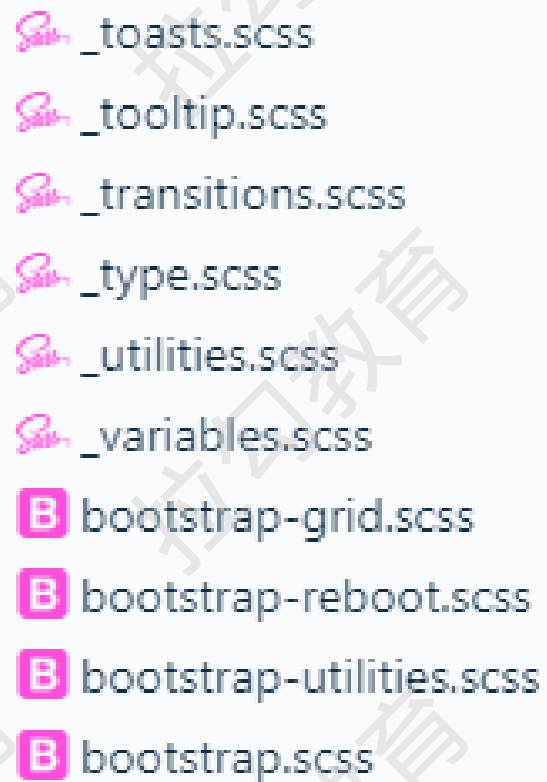
 _toasts.scss










 _tooltip.scss

 _transitions.scss

 _type.scss

 _utilities.scss



-  _toasts.scss
-  _tooltip.scss
-  _transitions.scss
-  _type.scss
-  _utilities.scss
-  _variables.scss
-  bootstrap-grid.scss
-  bootstrap-reboot.scss
-  bootstrap-utilities.scss
-  bootstrap.scss

- **forms/**: 表单组件相关样式
- **helpers/**: 公共样式, 包括定位、清除等
- **mixins/**: 可以理解为生成最终样式的函数
- **utilities/**: 媒体查询相关样式
- **vendor/**: 依赖的外部第三方样式

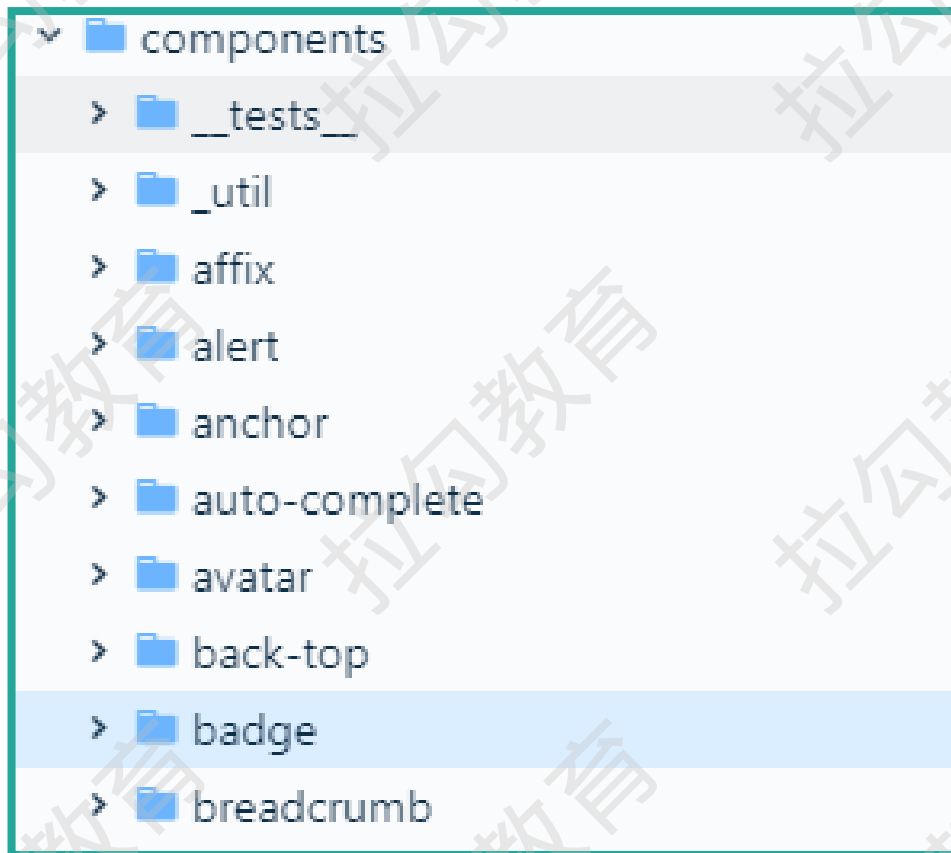













根目录存放了**组件样式文件**和**目录**

其他样式文件放在不同的目录中












目录中的文件分类清晰











但目录结构相对于大多数实际项目而言过于简单（只有样式文件）



- >  breadcrumb
- >  button
- >  calendar
- >  card
- >  carousel
- >  cascader
- >  checkbox
- >  col
- >  collapse
- >  comment
- >  config-provider

- >  config-provider
- >  date-picker
- >  descriptions
- >  divider
- >  drawer
- >  dropdown
- >  empty
- >  form
- >  grid
- >  icon
- >  input

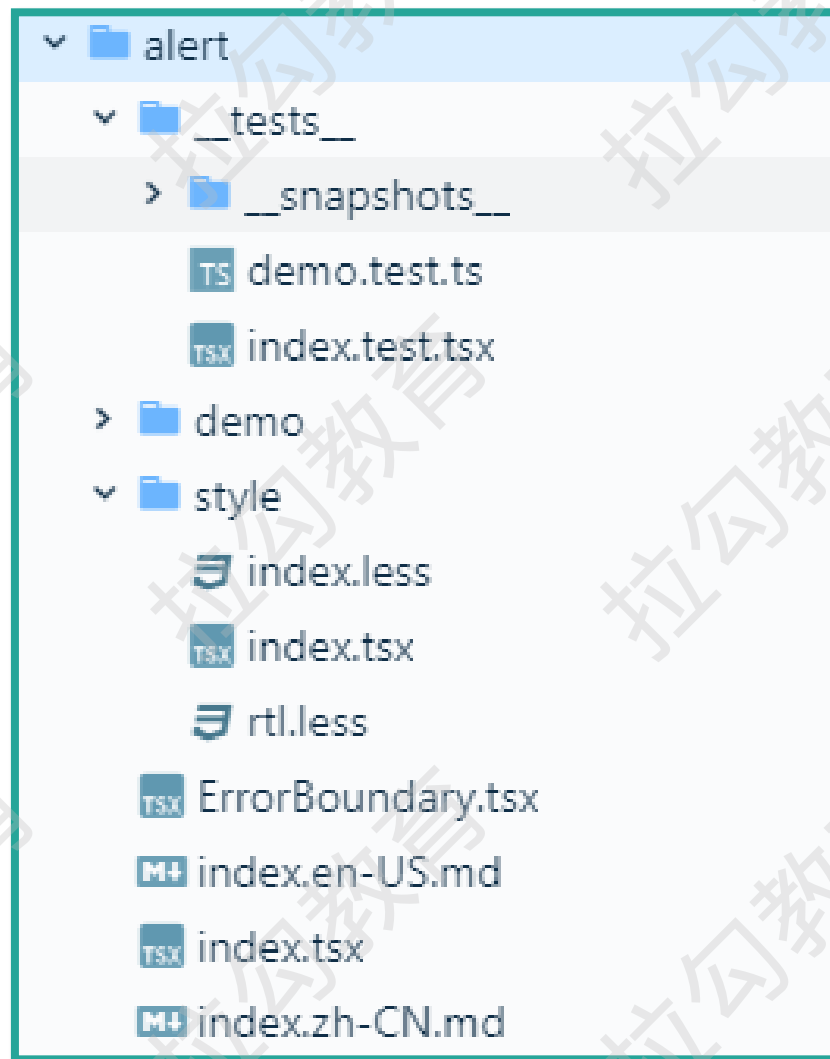
- >  input
- >  input-number
- >  layout
- >  list
- >  locale
- >  locale-provider
- >  mentions
- >  menu
- >  message
- >  modal
- >  notification

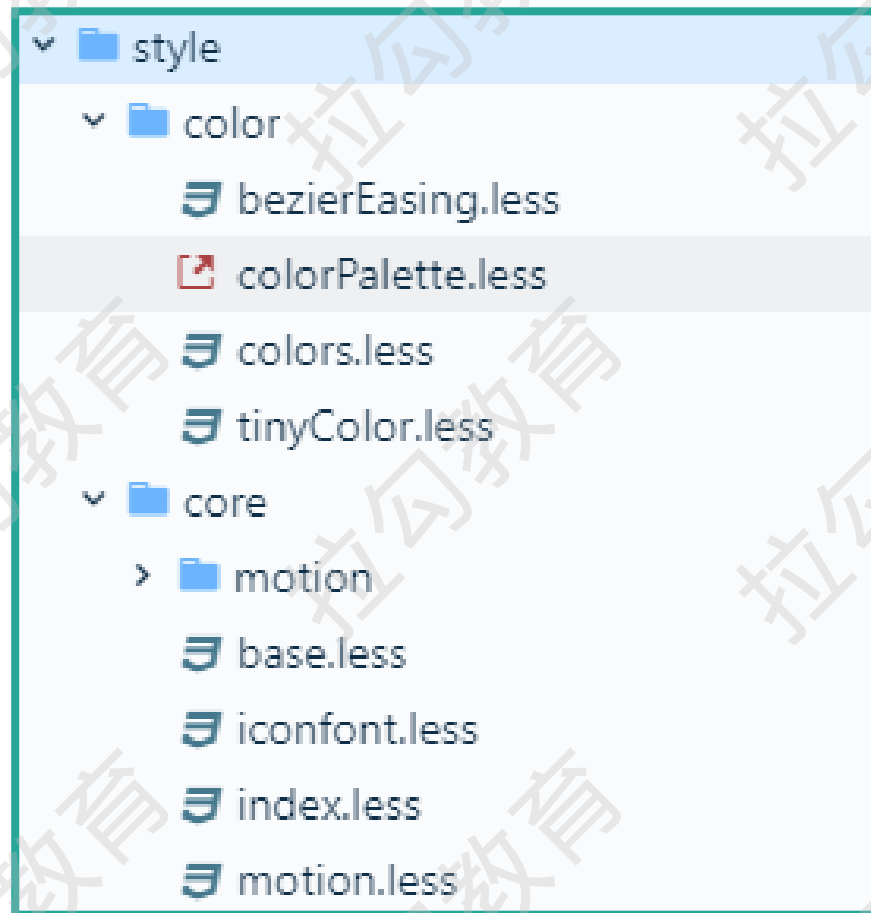
- >  mentions
- >  menu
- >  message
- >  modal
- >  notification
- >  page-header
- >  pagination
- >  popconfirm
- >  popover
- >  progress
- >  radio

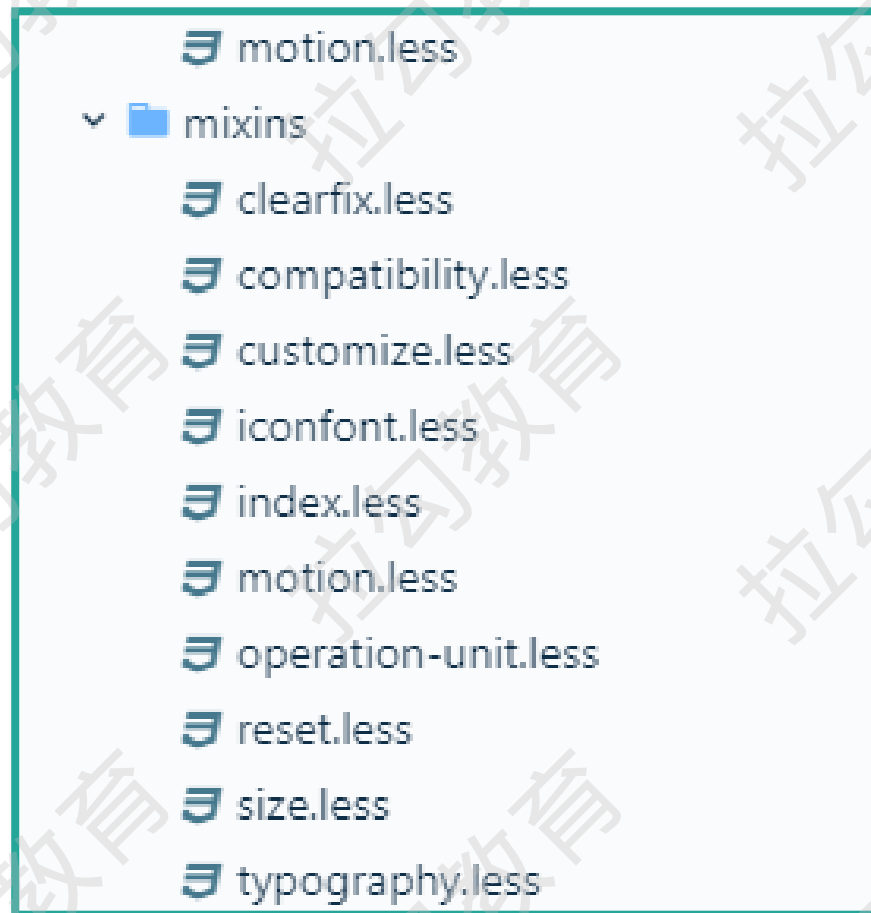
开源项目中的样式文件

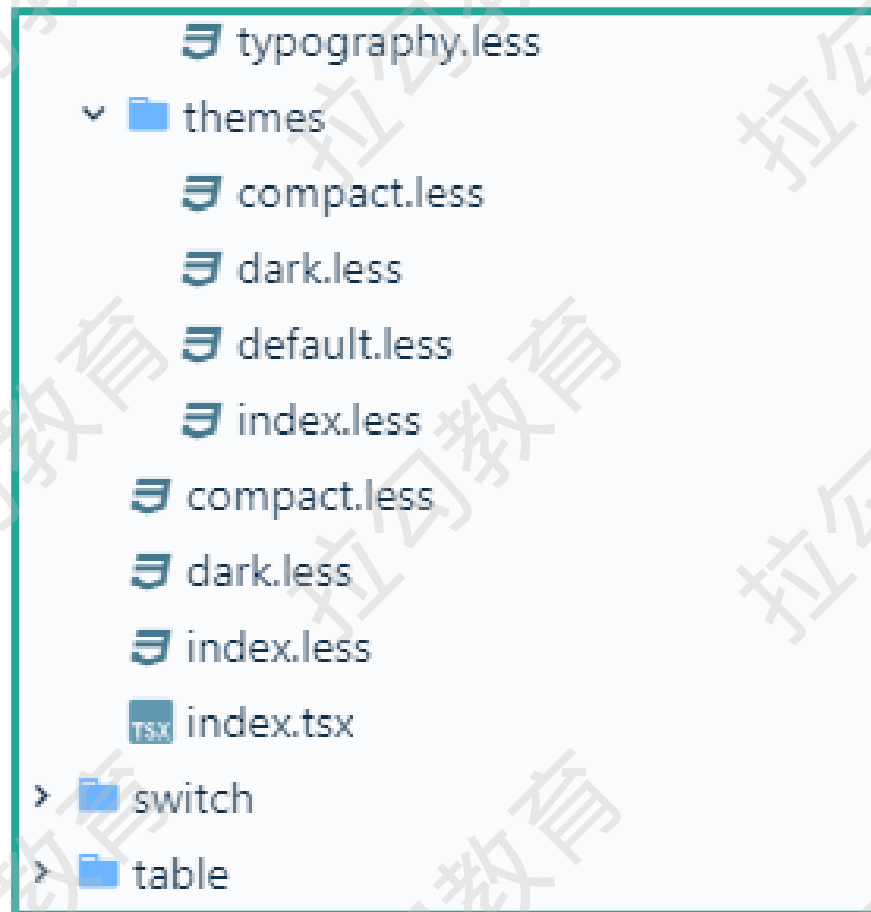
拉勾教育

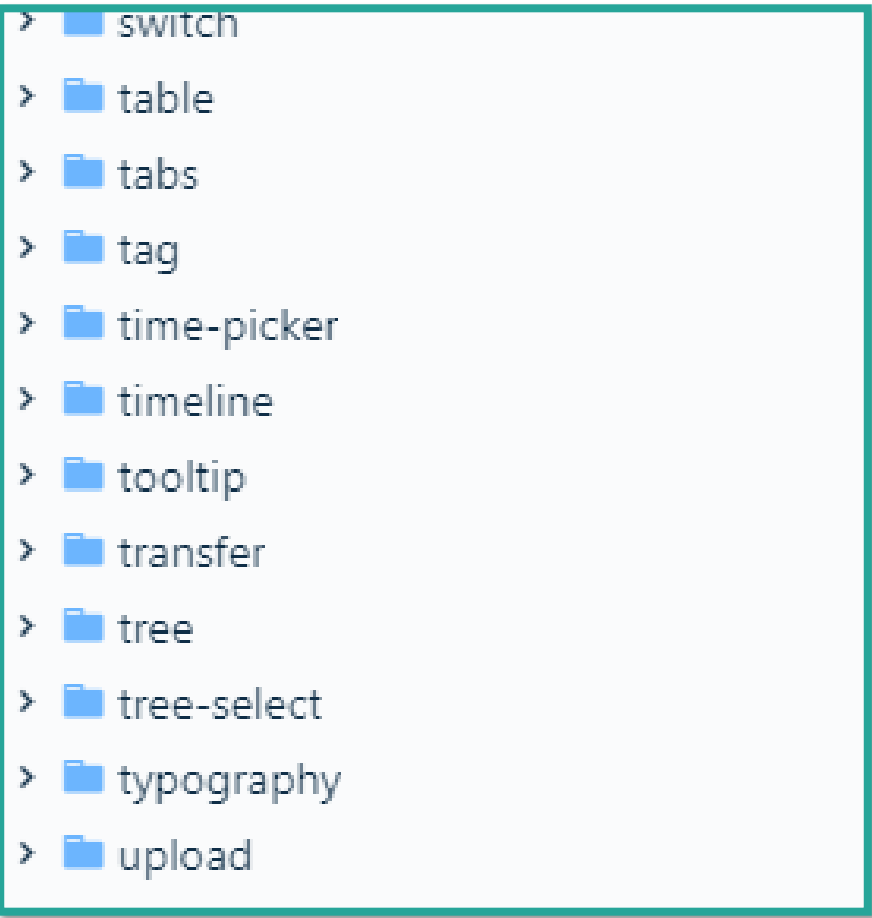
— 互联网人实战大学 —











- > switch
- > table
- > tabs
- > tag
- > time-picker
- > timeline
- > tooltip
- > transfer
- > tree
- > tree-select
- > typography
- > upload

- **color/**: 颜色相关的变量与函数
- **core/**: 全局样式, 根标签样式、字体样式等
- **mixins/**: 样式生成函数
- **themes/**: 主题相关的样式变量



开源项目中的样式文件

拉勾教育

— 互联网人实战大学 —

组件代码及相关样式放在一起，开发的时候修改会很方便

但在组件目录 **/components** 下设置 style 目录存放全局和公共样式

这些“样式”文件并不是一个单独的“组件”

再看 **style 目录** 内部结构

相对于设置单独的 color 目录来管理样式中的颜色

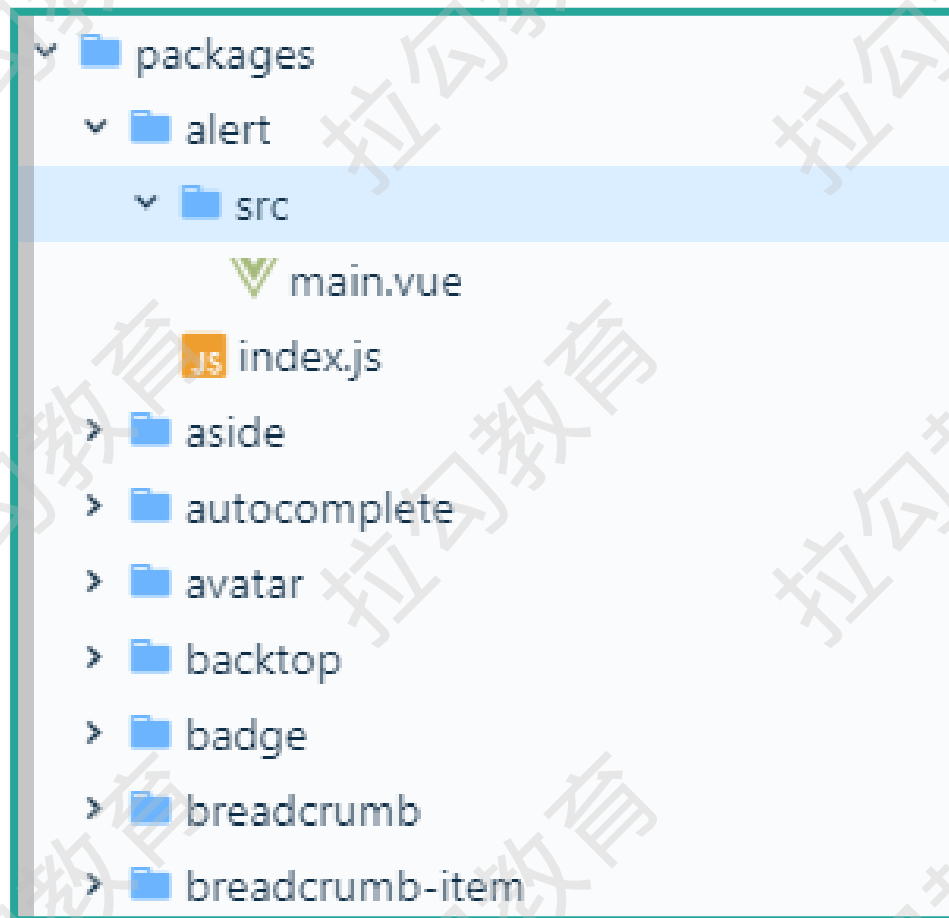
更推荐像 Bootstrap 一样设立专门的目录或文件来管理变量















开源项目中的样式文件


拉勾教育

— 互联网人实战大学 —



- >  breadcrumb-item
- >  button
- >  button-group
- >  calendar
- >  card
- >  carousel
- >  carousel-item
- >  cascader
- >  cascader-panel
- >  checkbox
- >  checkbox-button
- >  checkbox-group

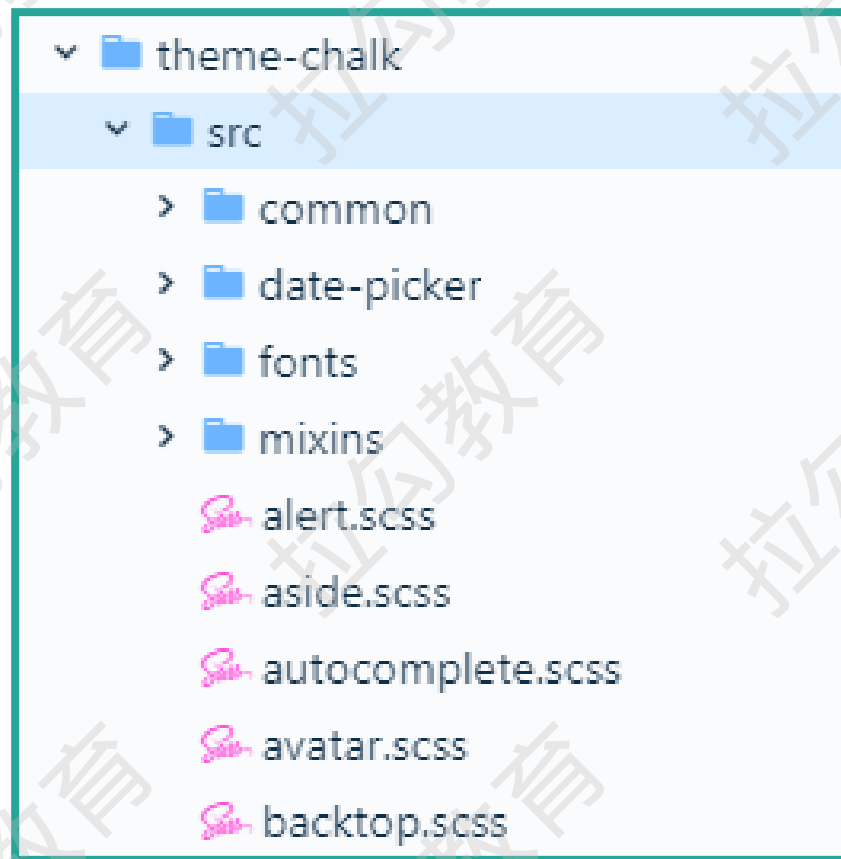
- > checkbox-group
- > col
- > collapse
- > collapse-item
- > color-picker
- > container
- > date-picker
- > dialog
- > divider
- > drawer
- > dropdown
- > dropdown-item

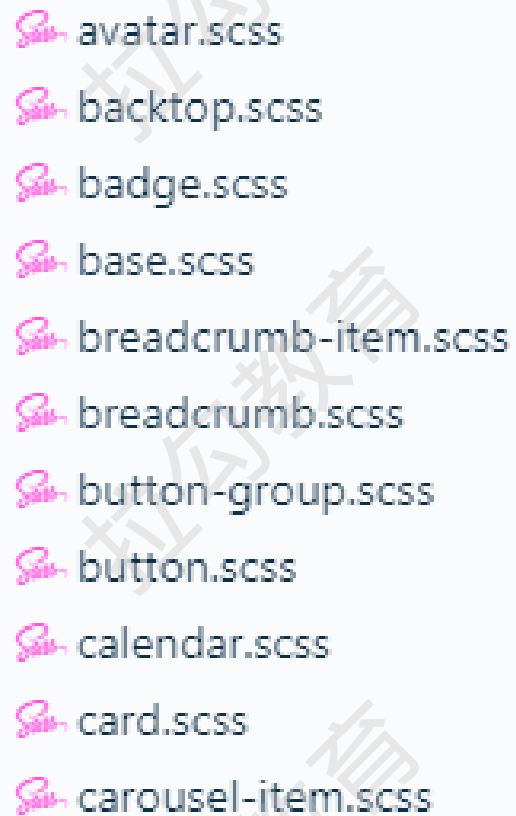
- >  dialog
- >  divider
- >  drawer
- >  dropdown
- >  dropdown-item
- >  dropdown-menu
- >  footer
- >  form
- >  form-item
- >  header
- >  icon
- >  image











开源项目中的样式文件

拉勾教育

— 互联网人实战大学 —





-  avatar.scss
-  backtop.scss
-  badge.scss
-  base.scss
-  breadcrumb-item.scss
-  breadcrumb.scss
-  button-group.scss
-  button.scss
-  calendar.scss
-  card.scss
-  carousel-item.scss

 carousel-item.scss

 carousel.scss

 cascader-panel.scss

 cascader.scss

 checkbox-button.scss

 checkbox-group.scss

 checkbox.scss

 col.scss

 collapse-item.scss

 collapse.scss

 color-picker.scss

 color-picker.scss

 container.scss

 date-picker.scss

 dialog.scss

 display.scss

 divider.scss

 drawer.scss

 dropdown-item.scss

 dropdown-menu.scss

 dropdown.scss

 footer.scss

 dropdown-menu.scss

 dropdown.scss

 footer.scss

 form-item.scss

 form.scss

 header.scss

 icon.scss

 image.scss

 index.scss

 infinite-scroll.scss

 infiniteScroll.scss

- **common/**: 一些全局样式和公共变量
- **date-picker/**: 日期组件相关样式
- **fonts/**: 字体文件
- **mixins/**: 样式生成函数及相关变量



开源项目中的样式文件

拉勾教育

— 互联网人实战大学 —

把样式当成 “**组件**” 看待

组件同级目录设立了 **theme-chalk** 目录存放样式文件

theme-chalk 目录下的全局样式 reset.scss 与组件样式同级

这也有些欠妥



这种为了将样式打包成模块

在独立项目中**直接嵌入**另一个独立项目并不推荐

更符合 Git 使用规范的做法

即是以**子模块的方式**引用进项目

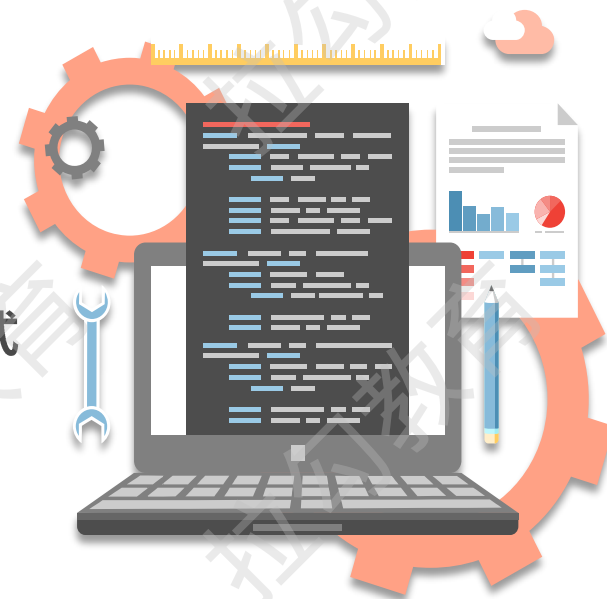
而且将组件样式和源码分离这种方式开发的时候也不方便

经常需要跨多层目录查找和修改样式



7-1 模式：这种模式建议将目录结构划分为 **7 个目录** 和 **1 个文件**

- **base/：**模板代码，比如默认标签样式重置
- **components/：**组件相关样式
- **layout/：**布局相关，包括头部、尾部、导航栏、侧边栏等
- **pages/：**页面相关样式
- **themes/：**主题样式，即使有的项目没有多个主题，也可以进行预留
- **abstracts/：**其他样式文件生成的依赖函数及 **mixin**，不能直接生成 **css** 样式
- **vendors/：**第三方样式文件



<https://github.com/HugoGiraudel/sass-boilerplate>

```
sass/  
|  
|- abstracts/  
|   |- _variables.scss  
|   |- _functions.scss  
|   |- _mixins.scss  
|   |- _placeholders.scss  
|  
|- base/  
|   |- _reset.scss  
|   |- _typography.scss  
|   ...  
|
```

<https://github.com/HugoGiraudel/sass-boilerplate>

```
| - components/  
|   | - _buttons.scss  
|   | - _carousel.scss  
|   | - _cover.scss  
|   | - _dropdown.scss  
|   | ...  
| - layout/  
|   | - _navigation.scss  
|   | - _grid.scss  
|   | - _header.scss
```

<https://github.com/HugoGiraudel/sass-boilerplate>

```
├── _header.scss
├── _footer.scss
├── _sidebar.scss
├── _forms.scss
├── ...
├── pages/
│   ├── _home.scss
│   └── _contact.scss
│   └── ...
├── themes/
│   └── _theme.scss
```

<https://github.com/HugoGiraudel/sass-boilerplate>

```
|  
|- themes/  
|   |- _theme.scss  
|   |- _admin.scss  
|   ...  
|- vendors/  
|   |- _bootstrap.scss  
|   |- _jquery-ui.scss  
|   ...  
|- main.scss
```

样式文件管理模式

拉勾教育

— 互联网人实战大学 —

main.scss 文件存在意义不大

页面样式、组件样式、布局样式都可以在页面和组件中引用

全局样式也可以在根组件中引用

每次添加、修改样式文件都需要在 main.scss 文件中同步

这种过度中心化的配置方式也不方便



layout 目录也可以去除

因为像 footer、header 这些布局相关的样式

放入对应的组件中来引用会更好

至于不能被组件化的 “_grid” 样式存在性也不大

因为对于页面布局

既可以通过下面介绍的方法来拆分成全局样式

也可以依赖第三方 UI 库来实现



样式文件管理模式

拉勾教育

— 互联网人实战大学 —

themes/ 目录也可以去除

毕竟大部分前端项目是不需要设置主题的

即使有主题也可以新建一个样式文件来管理样式变量

vendors/ 目录可以根据需求添加

因为将外部样式复制到项目中的情况比较少

更多的是通过 npm 来安装引入 UI 库

或者通过 webpack 插件来写入对应的 cdn 地址



```
src/  
|  
|- abstracts/  
|   |- _variables.scss  
|   |- _functions.scss  
|   |- _mixins.scss  
|   |- _placeholders.scss  
|  
|- base/  
|   |- _reset.scss  
|   |- _typography.scss  
|   -
```

```
|  
|- components/  
|   |- _buttons.scss  
|   |- _carousel.scss  
|   |- _cover.scss  
|   |- _dropdown.scss  
|   |- header/  
|       |- header.tsx  
|       |- header.sass  
|   |- footer/  
|       |- footer.tsx  
|       |- footer.sass
```

样式文件管理模式

拉勾教育

— 互联网人实战大学 —

```
| - header.tsx  
| - header.sass  
| - footer/  
| - footer.tsx  
| - footer.sass  
| -  
| - pages/  
| - _home.scss  
| - _contact.scss  
| -
```

如何避免样式冲突

CSS 的规则是全局的

任何一个样式规则，都对整个页面有效

如果不对选择器的命名加以管控会很容易**产生冲突**

拉勾教育

— 互联网人实战大学 —



最简单有效的命名管理方式就是制定一些命名规则

比如 **OOCSS**、**BEM**、**AMCSS**

其中推荐比较常用的 **BEM**



BEM是 Block、Element、Modifier 三个单词的缩写

- **Block** 代表独立的功能组件
- **Element** 代表功能组件的一个组成部分
 - **Modifier** 对应状态信息



HTML

```
<button class="button">
    Normal button
</button>
<button class="button button--state-success">
    Success button
</button>
<button class="button button--state-danger">
    Danger button
</button>
```

CSS

```
.button {
    display: inline-block;
    border-radius: 3px;
    padding: 7px 12px;
    border: 1px solid #D5D5D5;
    background-image: linear-gradient(#EEE, #DDD);
    font: 700 13px/18px Helvetica, arial;
}

.button--state-success {
    color: #FFF;
    background: #569E3D linear-gradient(#79D858, #569E3D) repeat-x;
    border-color: #4A993E;
}

.button--state-danger {
    color: #900;
}
```

规范约束也不能绝对保证样式名的唯一性

而且也没有有效的校验工具来保证命名正确无冲突

所以聪明的开发者想到了通过插件将原命名转化成不重复的随机命名

从根本上避免命名冲突

比较著名的解决方案就是 **CSS Modules**



```
/* style.css */  
.className {  
  color: green;  
}
```

```
import styles from './style.css';  
//import { className } from './style.css';  
element.innerHTML = '<div class="' +  
styles.className + '>';
```

```
<div class="_3zyde4l1yATCOkgn-DBWEL"></div>  
<style>  
  _3zyde4l1yATCOkgn-DBWEL {  
    color: green;  
  }  
</style>
```

如何高效复用样式

拉勾教育

— 互联网人实战大学 —

- display:inline-block
- clear:both
- position:relative
-

很违背 **DRY (Don't Repeat Yourself)** 原则

完全可以通过设置为全局公共样式来减少重复定义



如何高效复用样式

拉勾教育

— 互联网人实战大学 —

- 首先是具有**枚举值的属性**

除了上面提到的，还包括 `cursor:pointer`、`float:left` 等

- 其次是那些**特定数值的样式属性值**

比如 `margin: 0`、`left: 0`、`height: 100%`

- 最后是**设计规范所使用的属性**

比如设计稿中规定的几种颜色



如何高效复用样式

拉勾教育

— 互联网人实战大学 —

全局样式是**基于样式属性和值**的，是无语义的

其次对于这种复用率很高的样式应该尽量保证**命名简短**方便记忆

我们团队所使用的就是“**属性名首字母 + 横线 + 属性值首字母**”的方式进行命名



如何高效复用样式

拉勾教育

— 互联网人实战大学 —

我们团队所使用的就是“**属性名首字母 + 横线 + 属性值首字母**”的方式进行命名

对于 display:inline-block 的样式属性值

它的属性为“display”缩写为“**d**”，值为“inline-block”，缩写为“**ib**”

通过短横线连接起来就可以命名成“**d-ib**”



延伸：值得关注的 CSS in JavaScript

拉勾教育

— 互联网人实战大学 —

Web 标准

提倡**结构**、**样式**、**行为分离**（分别对应 **HTML**、**CSS**、**JavaScript** 三种语言）

但 React.js 的一出现就开始颠覆了这个原则



延伸：值得关注的 CSS in JavaScript

拉勾教育

— 互联网人实战大学 —

- ① 通过 JSX 将 HTML 代码嵌入进 JavaScript 组件
- ② 通过 CSS in JavaScript 的方式将 CSS 代码也嵌入进 JavaScript 组件

这种 “all in JavaScript” 的方式确实有悖 Web 标准

但这种编写方式和日益盛行的组件化概念非常契合，具有 “**高内聚**” 的特性



延伸：值得关注的 CSS in JavaScript

拉勾教育

— 互联网人实战大学 —

- 可以通过随机命名解决作用域问题，但命名规则和 CSS Modules 都可以解决这个问题

- 样式可以使用 JavaScript 语言特性

比如函数、循环，实现元素不同的样式效果可以通过新建不同样式类

修改元素样式类来实现



延伸：值得关注的 CSS in JavaScript

拉勾教育

— 互联网人实战大学 —

// 源代码

```
const Button = styled.button`  
  background: transparent;  
  border-radius: 3px;  
  border: 2px solid palevioletred;  
  color: palevioletred;  
  margin: 0.5em 1em;  
  padding: 0.25em 1em;  
  ${props => props.primary && css`  
    background: palevioletred;  
    color: white;  
  `}  
`
```

延伸：值得关注的 CSS in JavaScript

拉勾教育

— 互联网人实战大学 —

```
color: white;
`
;

const Container = styled.div`
  text-align: center;

  render(
    <Container>
      <Button>Normal Button</Button>
      <Button primary>Primary Button</Button>
    </Container>
  )
}
```

延伸：值得关注的 CSS in JavaScript

拉勾教育

— 互联网人实战大学 —

```
<!--HTML 代码-->
<div class="sc-fzXfNJ ciXJHl">
  <button class="sc-fzXfNl hvaMnE">Normal Button</button>
  <button class="sc-fzXfNl kiyAbM">Primary Button</button>
</div>
/*CSS 代码*/
.ciXJHl {
  text-align: center;
}
.hvaMnE {
```

延伸：值得关注的 CSS in JavaScript

拉勾教育

— 互联网人实战大学 —

```
hvaMnE {  
  color: palevioletred;  
  background: transparent;  
  border-radius: 3px;  
  border-width: 2px;  
  border-style: solid;  
  border-color: palevioletred;  
  border-image: initial;  
  margin: 0.5em 1em;  
  padding: 0.25em 1em;  
}
```


延伸：值得关注的 CSS in JavaScript

拉勾教育

— 互联网人实战大学 —

```
.kiyAbM {  
  color: white;  
  border-radius: 3px;  
  border-width: 2px;  
  border-style: solid;  
  border-color: palevioletred;  
  border-image: initial;  
  margin: 0.5em 1em;  
  padding: 0.25em 1em;
```

延伸：值得关注的 CSS in JavaScript

拉勾教育

— 互联网人实战大学 —

```
color: white;  
border-radius: 3px;  
border-width: 2px;  
border-style: solid;  
border-color: palevioletred;  
border-image: initial;  
margin: 0.5em 1em;  
padding: 0.25em 1em;  
background: palevioletred;  
}
```

延伸：值得关注的 CSS in JavaScript

拉勾教育

— 互联网人实战大学 —

styled-components 只是 CSS in JavaScript 的一种解决方案

[有兴趣的同学可以点击这里查阅 GitHub 上的资料学习其他解决方案](#)

上面收录了现有的 CSS in JavaScript 解决方案



- 推荐使用 7-1 模式简化后的目录结构

包括 pages/、components/、abastracts/、base/ 4 个目录

- 样式命名

可以采用 BEM 来命名组件

面向属性的方式来命名公共样式



说说你在项目中是如何管理样式代码的？



Next: 第05讲 《如何管理你的 CSS 代码》

拉勾教育

— 互联网人实战大学 —



下载「拉勾教育App」
获取更多内容