前端项目结构设计

北京华宇信息技术有限公司 BEIJING THUNISOFT INFORMATION TECHNOLOGY CORPORATION LIMITED

**文档变更记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 日期 | 版本号 | 修订内容 | 修订人 | 审核人 | 批准人 |
| 2016-11-23 | V1 | 创建 | 吴惟刚 |  |  |
| **2019-10-19** | V2 | Vue 技术栈 | 吴惟刚 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**目 录**

目录

[文档变更记录 i](#_Toc12070)

[目 录 I](#_Toc7073)

[1.1. 总体目的 1](#_Toc31050)

[1.2. 实用场景 2](#_Toc16305)

[1.3. 工程结构 3](#_Toc13385)

[1.3.1. 项目外层结构 3](#_Toc3399)

[1.3.2. 项目doc结构 5](#_Toc29547)

[1.3.3. 项目静态资源static结构 6](#_Toc27994)

[1.3.4. 项目源码结构 7](#_Toc15210)

[1.3.5. 项目源码components文件夹内部结构 8](#_Toc6867)

[1.3.6. 项目源码css文件夹内部结构 10](#_Toc26689)

[1.3.7. 项目源码js文件夹内部结构 14](#_Toc9128)

[1.3.8. 项目源码pages文件夹内部结构 18](#_Toc4685)

[1.3.9. 项目源码images文件夹内部结构 18](#_Toc3)

[1.3.10. 项目源码router文件夹内部结构 19](#_Toc30051)

[1.4. 其他 19](#_Toc22376)

**黄金定律**

永远遵循同一套编码规范 -- 这样可以让一个团队办事效率更高，不管有多少人共同参与同一项目，确保每一行代码都像是同一个人编写的。

## 总体目的

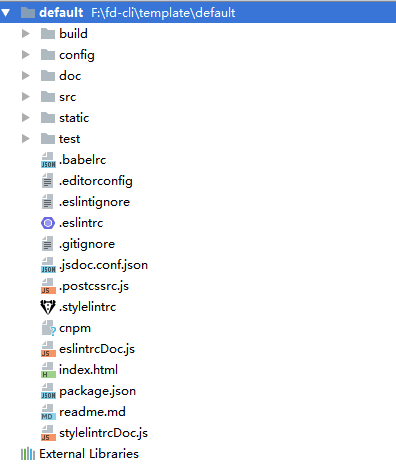
* 规范代码
* 降低维护成本
* 提升工作效率
* 提高代码的复用率

## 实用场景

* vue前后端分离项目

## 工程结构

### 项目外层结构



**default: 项目名称**

build: webpack 相关配置

config: webpack服务器相关配置

doc: 项目文档

src：前端源码文件夹

static: 前端静态资源

test: 前端单元测试目录

package.json: 项目包管理文件

readme.md： 项目说明文件

index.html： 单页面项目入口文件

**配置文件：**

.eslintrc ： eslint 配置 eslintrcDoc.js 为eslint每项说明文档

.stylelintrc ： stylelint 配置 stylelintrcDoc.js 为stylintlint每项说明文档

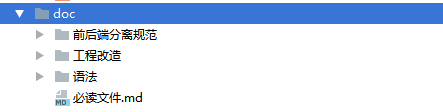
.babelrc ： babel配置

.jsdoc.conf.json ： jsdoc 文档生成配置文件

.gitignore： git 忽略文件配置

.eslintignore： eslint 验证忽略配置

### 项目doc结构



Doc 里面包含了开发文档，和工程改造说明，以及一些语法指导文档，不熟悉的在项目开始前先通读文档。使用fd命令新建项目时，务必按照必读文件做一遍

### 项目静态资源static结构

static 为项目的静态资源，内部目录结构按照【文件类型】进行划分模块，【模块内部】目录结构按照【子模块】进行划分，以此类推。



* flash：存放项目需要的flash （按照项目实际需要，自行判断是否需要这个文件夹）
* js: 存放项目外部依赖的js （按照项目实际需要，自行判断是否需要这个文件夹）
* json:存放项目所需要的所有mock json文件, 前端后项目一定不要受制于后端服务器 （必须）
* svg:存放项目所需要的所有svg文件 （按照项目实际需要，自行判断是否需要这个文件夹）
* ......

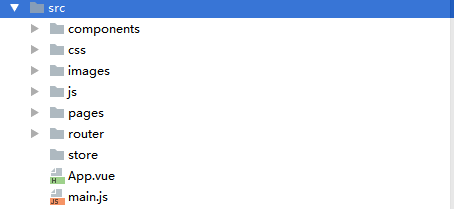
**【json 内部结构】**



json直接存放各模块的静态数据

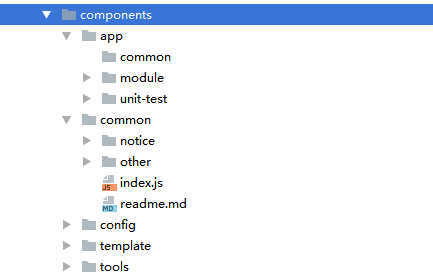
如上图：【test】为一个模块， index， details, 分别为该模块的首页和详情页

### 项目源码src结构



* components ： 存放系统所有的组件，另外包含部分工具方法 （必须）
* css ：存放项目需要的所有css(系统样式文件) （必须）
* images:存放项目需要的所有图片 （必须）
* js：存放项目需要的所有js （必须）
* pages：存放项目需要的所有页面（系统模块页面） （必须）
* router: 存放系统中的路由配置文件 （必须）
* store: 存放vuex状态管理文件 （非必须）
* App.vue ： 系统入口组件
* Main.js ： 系统入口js文件，该文件引入了 app.vue， 另外全局变量，全局组件，全局配置，路由等都是在该文件进行注册。

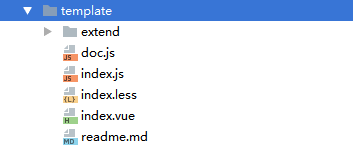
### 项目源码components文件夹内部结构



* app： 放系统业务组件
* common: 放系统通用组件
* config ： 组件的配置文件
* template： 组件编写的模块文件， 在编写公用组件和业务组件之前，大家可以直接复制template到目标区域，然后把template修改成自己的组件名即可
* tools： 通用的工具类

App, common 其内部结构根据组件的分类划分类型结构，如上图app 下有 common类，module类， unit-test类, 为提升复用性，原则上组件的css全部写到外层css目录components目录下，通用组件的css可以直接在组件内部建一个index.less，单独编写，方便其他系统复用。

组件内部结构



doc.js 为组件的文档注释，用于生成组件的API（单独提出来是为了书写文档方便）

index.vue 组件html结构

index.less 组件的css

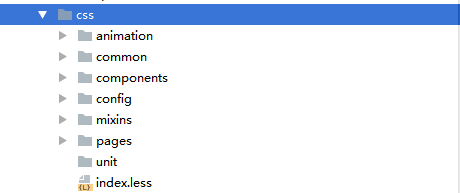
index.js 组件的js

extend： 该文件夹存放组件拆分的js

**组件内容少于200行可以将css，js 写到index.vue 文件，其他情况，js,css,html 必须**

**按照上述结构编写。**

### 项目源码css文件夹内部结构



css 目录结构按照类型进行划分，各个类型下有一个index.less，用于串联旗下的后代csss.

animation： 动画类css (非必须)

common: 公用类css **（必须）**

components： 组件类css， 内部目录结构和组件的文件夹一致 **（必须）**

config: 配置类css，因为采用less语法，所以可以配置变量 (非必须)

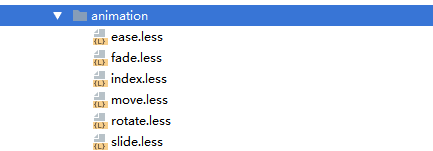
mixins： 混入类css，加强css的复用 (非必须)

pages: 页面模块类css, 各个页面的css **（必须）**

unit: 元件类css (非必须)

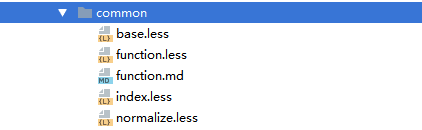
index.less 是整个项目的css 入口，里面串联各类型的css **（必须）**

**【 1. 动画类css】**



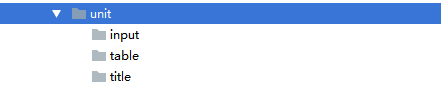
按照项目需要，大家可以在此文件夹编写一些公用的动画方法，以**【动画效果】**作为文件的名，然后在index.less 中进行引入，这样在整个系统都可以通用此动画

**【 2. 公用类css】**



* normalize.less这里是重置浏览器的样式，来源网络 （必须）
* base.less 此文件是在 normalize.less上额外增加的全局样式改造（必须）
* function.less 这里全部是功能性命名，全部严格模版执行，不可更改，华宇前端统一，有需要修改的可以提出来，然后新增进去，请各位重视。具体使用，可以查看function.md （必须）

**【3. 元件类css】**



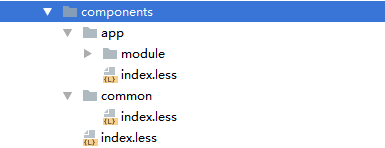
该文件夹通常用在多人协作的项目，团队成员按照元件类型，先分工写公用元件样式 ，这样可提升复用率，提升开发效率

**什么是元件？**

元件就是可以任用到任何地方，不依赖任何东西的存在

例如：一个钉子，可以运用于建筑，也可以用于船，车的建造等等。

**【4. 组件类css】**



该文件夹和components组件一一对应，app 为业务组件，common为通用组件， 各系统皆可用。

**什么是组件？**

组件可能包含多个元件，然后形成一个组件，例如：一个汽车轮子，由元件钉子，轮胎等组件

**【5. 页面模块类css】**



该文件夹存放各个页面自己的css，模块的样式基本不可通用

**什么是模块？**

简单理解，一个模块就是拥有一定功能，可能由很多元件和组件组成。

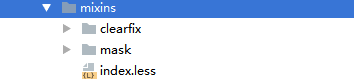
例如：一辆车，由轮胎，方向盘，车身等等组成。

**【6. 配置类css】**



该文件下的config.less 为less的配置变量， 通常用户换肤的系统，或者有类库编写会用到

**【7. 混入类css】**



该文件夹下存在的都是通用的方法，可在组件之间通用

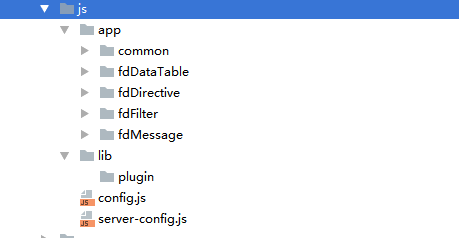
**【8. 其他类css】**

除上述分类之外，大家可以根据需要添加自己需要的类别，例如 plugin类， layout类等

* layout.css ，这里是布局的类名,集合里常用的 padding margin width height，自己按需要用还是不用

* plugin-name.css 文件命名：plugin+插件名称，引用的他人插件或者自己写的插件依赖的css.

### 项目源码js文件夹内部结构



app ：项目依赖的、所有自己写的js放这里

lib ： 项目依赖的外部库，插件js全部放这里

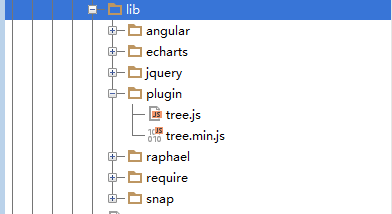
config.js 项目的配置文件，包含里请求方法，后端服务器地址，日志，调试多个配置项，大家可以在这里配置项目需要的其他项

server-config.js 此文件为后端接口api的配置文件

#### config.js 基本配置如下

*/\*\*  
 \*****@file*** *config.js  
 \*****@version*** *1.0.1  
 \*****@author*** *wuwg  
 \*****@createTime*** *2019/10/17 - 20:37  
 \*****@updateTime*** *2019/10/17 - 20:37  
 \*****@see*** *[jsDoc中文文档]{****@link*** *http://www.dba.cn/book/jsdoc/JSDOCKuaiBiaoQianBLOCKTAGS/CONSTRUCTS.html}  
 \*****@description*** *config.js ,所有的配置文件 详细说明看注释 !important  
 \*/  
// 服务器配置***import** serverUrl **from './server-config.js'**;  
*// let production = process.env.NODE\_ENV !== 'production'  
// 默认开发环境***const** develop = **true**;  
*//***const** config = {  
 *// 是否是调试模式，true 为json数据，false 为服务器数据* **isDebug**: develop,  
 *// ajax查询方式 GET* **methodGet**: **'GET'**,  
 *// ajax查询方式 POST* **methodPost**: develop ? **'GET'** : **'POST'**,  
 *// ajax查询方式 PUT* **methodPut**: develop ? **'GET'** : **'PUT'**,  
 *// ajax查询方式 DELETEsocketio* **methodDelete**: develop ? **'GET'** : **'DELETE'**,  
 *// ajax查询方式 PATCH* **methodPatch**: develop ? **'GET'** : **'PATCH'**,  
 *// url链接,也就是数据的地址* **url**: {},  
 *// 是否显示日志* **showLog**: **false**,  
 *// 服务器地址* **dirServicePath**: **window**.***serverConfig***.**dirServicePath**};  
*// 本地数据***const** localUrl = {  
 *// 测试模块* **test**:{  
 *// 测试模块首页数据* **index**: **'/static/json/test/index.json'** }  
};  
config.**url** = config.isDebug ? localUrl : serverUrl;  
*// 注册全局变量，fdConfig***window**.**fdConfig** = config;

#### js文件夹lib内部结构



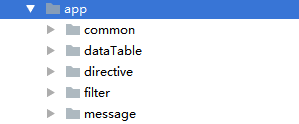
* 各种库文件

大家如果用到第三方库，可以放lib中， 以库名作为分类名称

* 插件

所有的插件js一律放到plugin中，如果涉及的插件多，种类多，可以在里面继续建文件夹分类

#### js文件夹app内部结构



**fd前缀模块代表通用模块，其他模块不可使用此类命名。 如 ： fdFilter, fdDirective**

**强制命名，目的在于统一，防止冲突，最后工具合并请求。**

* common： 公用的js文件

如global.js 等公用的方法类js

* dataTable：前台代码值对应表

fdDataTable.js 模块名字取名： fdDataTable，

具体写法看模块中的 fdDataTable.js

* directive: 通用指令

fdDirective.js模块名字取名： fdDirective, 具体写法看模块中的 fdDirective.js

* filter: 通用过滤器

fdFilter.js模块名字取名： fdFilter, 具体写法看模块中的 fdFilter.js

* message: 消息模块

fdMessage.js模块名字取名： fdMessage， 里面绑定了window.onmessage，大家根据项目需要，自行判断是否需要

**名词解释：**

app: application 应用程序

dataTable： 数据表

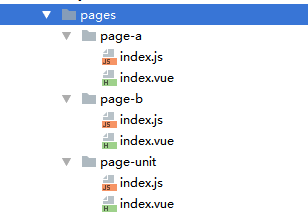
directive:指令

filter:过滤器

message： 消息

大家可以根据项目需要，在app下添加其他模块，原则：**【按模块划分类型】**

### 项目源码pages文件夹内部结构



Pages 存在路由用的各个页面， 按照模块划分内部结构，里面存在index.vue 作为页面入口

### 项目源码images文件夹内部结构



一般来说，所有图片全部放到 images文件下，如果用了插件的，用插件名为名称建立一个文件夹，存放对应的图片

### 项目源码router文件夹内部结构



此文件夹为 路由配置文件， 大家按照自己的需要配置即可。

## 其他

在大规范的前提下，自己根据需求灵活应用。

此文档规范即日起，立即生效。

2019年10月19日

吴惟刚

北京市海淀区中关村东路1号院清华科技园科技大厦C座25层 (100084)

25F, C, Science Building, Tsinghua Science Park, No.1,

East Road Zhongguancun, Haidian, Beijing 100084, China

电话 010 - 8262 2288 传真 010 - 8215 0616 /8