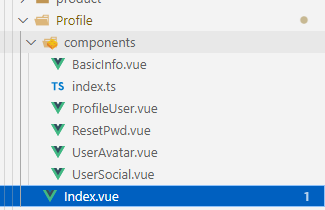
# 前端基础规范文档

不同的人有不同的开发习惯，但是为了让项目的代码可读性可维护性更好，本文档内容为一些基础的强制要求规范。前期可能会要求适应一下

## 命名规范：

统一遵循配置好的stylelint、eslint、ts配置即可，另说明：

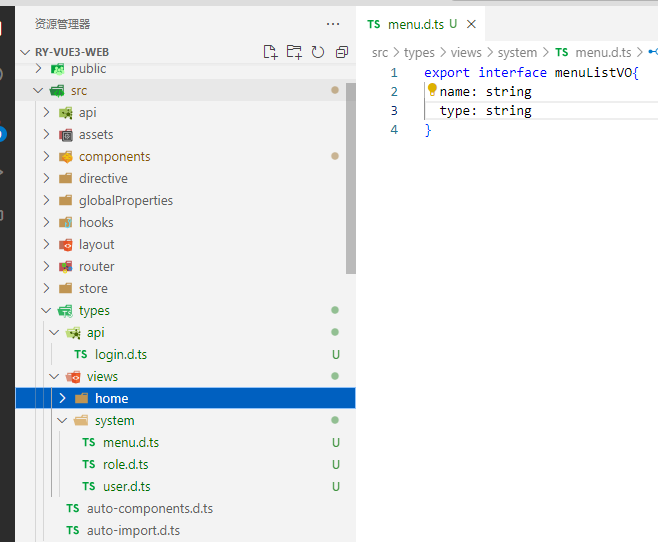
1. 页面组件(即路由对应的组件)统一（文件夹名 / index.vue 或 前缀加上page如详情页pageDetail.vue）
2. 非页面组件统一大驼峰形式，使用时候也统一大驼峰形式

如:：

1. 文件夹命名统一小驼峰形式
2. Css样式统一烤串格式
3. Js变量和函数统一小驼峰

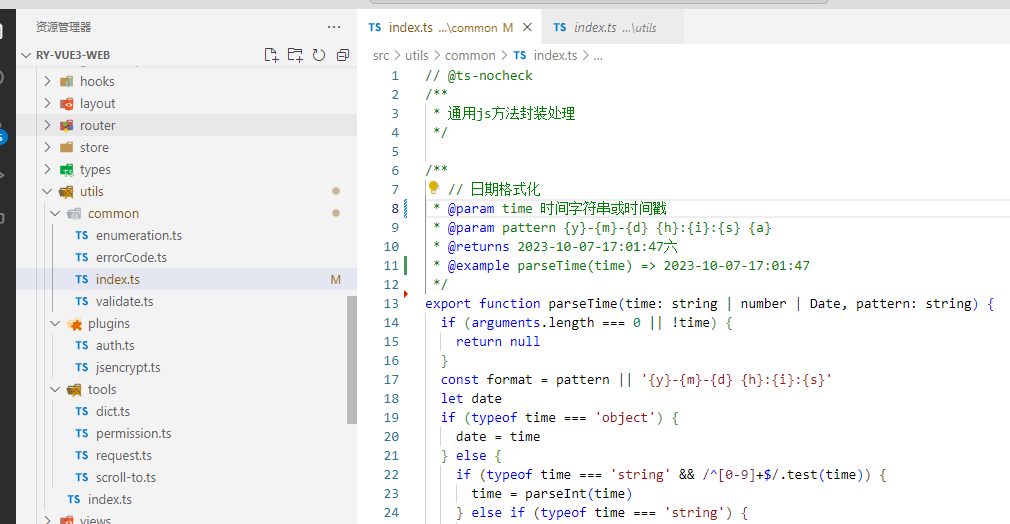
## Typescript使用规范

声明的interface或type接口对象统一放在src/types里面并且根据使用路径放置对应的文件夹中无则新创建，保证与src中的目录一一对应



## 工具库utils规范

* 每个工具库文件名要有意义
* 每个工具库工具方法要有说明和示例

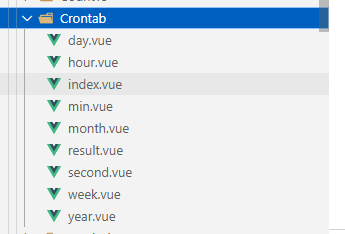


## 公共组件components 要求说明

公共组件的目的是为了让大家都能公用一些优秀的功能或者界面，提高开发效率，可如果不做好规范，写出来的公共组件实际上根本就没法大家公用，因为看起来费劲，用起来更加费劲，那还不如自己新写一个，但这样就会导致公共这个目的偏离，因此有必要强制一些规则。

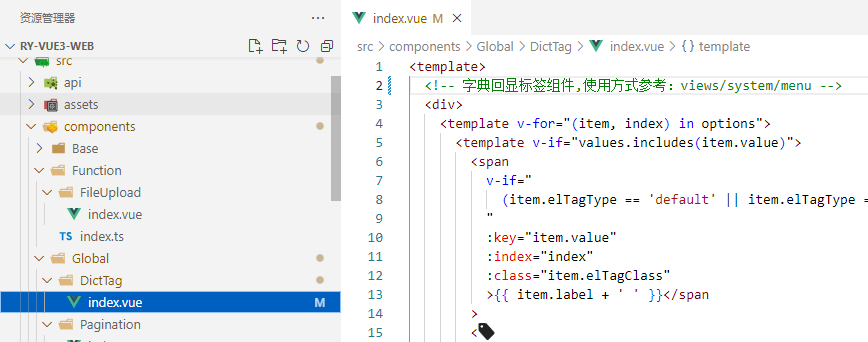
1. 谨记一个公共组件尽量只做一件事只表达一个界面功能。如果开发的功能界面比较复杂，则尽可以拆成多个子功能界面然后在组合成所需要的功能界面，多利用vue的组合特性

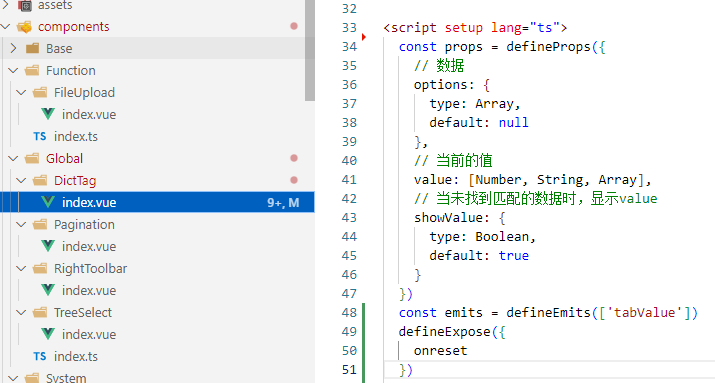
例如：这是一个cron表达式生成组件，将其根据时间功能分别拆分成了多个子组件



1. 每个公共组件的功能作用、使用示例(在哪个页面用了)要做好说明，包括props字段注释说明、emits事件、expose暴露等

例如：





1. 公共组件内容新增以满足需求可以，但是需要写好说明，而修改和删除需要谨慎，若该组件部分内容满足要求，部分不满足，则可以自己复制出来重新修改成符合自己需求的组件。

## 公共组件层级规范

公共组件层级分为全局级，视图级，模块级，组件级

* 全局级 例：src/components
* 视图级 例：src/views/components
* 模块级 例：src/views/system/components
* 组件级 例：src/views/system/user/components

组件层级划分的好处就是层次分明，共享性更好，组件跨层次的时候可以进行组件提级。后续开发过程中可以根据自己的开发需要从当前层级或上层级共享组件中看看是否有自己需要的组件拿过来使用或者改造使用，这样可以提高开发效率

Hooks的封装亦如组件层级，且每个hooks必须注明作用说明和传出内容说明

1. 无用的数据删除无需注释，如需找回直接从git过去历史中找回即可