前端规范说明

市面上常用的命名规范：

camelCase（小驼峰式命名法 —— 首字母小写）

PascalCase（大驼峰式命名法 —— 首字母大写）

kebab-case（短横线连接式）

Snake（下划线连接式kebab\_case）​

1. 命名规范
   1. **项目文件命名**

1.1.1项目名

全部采用大写方式，以短横线分隔。例：MY-PROJECT-NAME。

1.1.2 目录名

参照项目命名规则，有复数结构时，要采用复数命名法。例：assets、components、directives、mixins、utils、views。

例如下面这个模板目录：

whclould-smart-port / my-project-name

|- node\_modules // 下载的依赖包

|- public // 静态页面目录

|- js // 其他静态资源文件 例如：imosSdk.min.js

|- html // 其他静态资源文件 例如：imosSdk.html

|- favicon.ico // 网站图标

|- index.html // 项目入口

|- src // 源码目录

|- api // http 请求目录

|- assets // 静态资源目录，这里的资源会被wabpack构建

|- icon // icon 存放目录

|- img // 图片存放目录

|- js // 公共 js 文件目录

|- scss // 公共样式 scss 存放目录

|- frame.scss // 入口文件

|- global.scss // 公共样式

|- reset.scss // 重置样式

|- styles // UI框架覆盖主题 文件夹

|- components // 组件

|- plugins // 插件

|- router // 路由

|- routes // 详细的路由拆分目录（可选）

|- index.js

|- store // 全局状态管理

|- utils // 工具存放目录

|- request.js // 公共请求工具

|- variable.js // 公共请求地址、参数配置文件

|- util.js // 公共方法类

|- views // 页面存放目录

|- App.vue // 根组件

|- main.js // 入口文件

|- tests // 测试用例

|- .browserslistrc// 浏览器兼容配置文件

|- .editorconfig // 编辑器配置文件

|- .eslintignore // eslint 忽略规则

|- .eslintrc.js // eslint 规则

|- .gitignore // git 忽略规则

|- babel.config.js // babel 规则

|- package-lock.json

|- package.json // 依赖

|- README.md // 项目 README

|- vue.config.js // webpack 配置

1.1.3 图像文件名

全部采用小写方式， 优先选择单个单词命名，多个单词命名以下划线分隔，banner\_first.gif

图片文件夹命名：

单个单词文件小写命名，例如：home

多个单词文件命名： camelCase 小驼峰式命名法，例如：视频组件用：monitorVideo

1.1.4 HTML 文件名

全部采用小写方式， 优先选择单个单词命名，多个单词命名以camelCase小驼峰式命名法分隔。

1.1.5 CSS 文件名和JavaScript 文件名

全部采用小写方式， 优先选择单个单词命名，多个单词命名以短横线分隔。

**1.2 vue组件命名**

1.2.1 单文件组件名

文件扩展名为 .vue 的 single-file components (单文件组件)。单文件组件名应该始终是单词小写开头 (pascalCase)，小驼峰式命名法。（myComponent.vue）

1.2.2 基础组件名

展示类的、无逻辑的或无状态、不掺杂业务逻辑的组件) 应该全部以一个特定的前缀开头 —— base。基础组件在一个页面内可使用多次，在不同页面内也可复用，是高可复用组件。

例如：日期选择组件：baseDatePicker.vue

1.2.3 紧密耦合的组件名

父子组件紧密耦合性组件以父组件名为前缀，因为编辑器通常会按字母顺序组织文件，所以这样做可以把相关联的文件排在一起。

例如：videoList.vue videoListItem.vue videoListItemPlayer.vue

1.2.4 组件名中单词顺序

组件名应该以高级别的 (通常是一般化描述的) 单词开头，以描述性的修饰词结尾。 因为编辑器通常会按字母顺序组织文件，所以现在组件之间的重要关系一目了然。如下组件主要是用于搜索和设置功能。

例如：searchContent.vue

注意：组件名尽量以完整单词命名，单词尽量语义化，方便理解。

**1.3代码参数命名**

1.3.1 name

组件名应该始终是多个单词，应该始终是 pascalCase 的。

例如：export default {

name: 'toDoList',

// ...

}

1.3.2 router

Vue Router Path 命名采用 kebab-case 格式。或者pascalCase小驼峰命名，推荐使用kebab-case 短横线。

例如： path: '/user-info', // 能解析成 user info

name: 'userInfo',

1.3.3 模板中组件

对于绝大多数项目来说，在单文件组件和字符串模板中组件名应该总是 pascalCase 的，在 DOM 模板中总是 pascalCase 的。

1.3.4 变量

命名方法：camelCase

命名规范：类型 + 对象描述或属性的方式

1.3.5 常量

命名方法：全部大写下划线分割

命名规范：使用大写字母和下划线来组合命名，下划线用以分割单词

例如：const MAX\_COUNT = 10

const URL = 'http://xxx.xxxx.com'

const COOKIE\_EXPIRES = 30

1.3.6 方法

命名方法：camelCase

命名规范：统一使用动词或者动词 + 名词形式

1.3.7 自定义事件

自定义事件应始终使用 kebab-case 的事件名。

例如：this.$emit('my-event')

1.3.8 监听事件方法

命名方法：camelCase

命名规范：handle + 名称（可选）+ 动词；动词+名词

例如：<div @current-change="handleCurrentChange"></div>

<div @click="setMatrixNum"></div>

事件与方法的区别：

1. 事件名前一般都以on开头，跟鼠标和键盘操作相关的，比如点击事件、监听事件等等。

比如：@on-focus="handleShowPop"

@click ="handleChangeType(item,index)"

② 方法可以直接调用，事件不需要调用，但是需要写一个函数且改函数赋值给相应的事件，其调用是在相应的事件触发时。

注意：通常事件和方法是连贯使用的，方法一般封装了具体要干的事，是通过事件来调用方法。

例如：我们可以通过点击事件，去调用登录的方法。

1. 代码规范

**2.1 vue2**

2.1.1 代码结构

例如下面这个模板：

<template>

<div id="my-component">

<helloComponent />

<ysVideo ></ysVideo >

</div>

</template>

<script>

import helloComponent from '../components/HelloComponent'

import ysVideo '../components/YsVideo '

export default {

name: 'myComponent',

components: {

helloComponent,

ysVideo

},

mixins: [],

props: {},

data () {

return {}

},

computed: {},

watch: {}

created () {},

mounted () {},

destroyed () {},

methods: {},

}

</script>

<style lang="scss" scoped>

#my-component {

}

</style>

2.1.2 computed

应该把复杂计算属性分割为尽可能多的更简单的属性。 小的、专注的计算属性减少了信息使用时的假设性限制，所以需求变更时也用不着那么多重构了。

2.1.3 为 v-for 设置键值

在组件上必须用 key 搭配 v-for，以便维护内部组件及其子树的状态。甚至在元素上维护可预测的行为，比如动画中的对象固化 (object constancy)

2.1.4 v-if 和 v-for 互斥(vue2)

永远不要把 v-if 和 v-for 同时用在同一个元素上，请将 v-if 移动至容器元素上。

2.1.5 指令缩写

用 : 表示 v-bind:

用 @ 表示 v-on:

用 # 表示 v-slot:

**2.2 HTML**

HTML 元素共有以下5种：

空元素：area、base、br、col、command、embed、hr、img、input、keygen、link、meta、param、source、track、wbr

原始文本元素：script、style

RCDATA 元素：textarea、title

外来元素：来自 MathML 命名空间和 SVG 命名空间的元素

常规元素：其他 HTML 允许的元素都称为常规元素

2.2.1元素及标签闭合

为了能让浏览器更好的解析代码以及能让代码具有更好的可读性，有如下约定：

① 所有具有开始标签和结束标签的元素都要写上起止标签，某些允许省略开始标签或和束标签的元素亦都要写上。

② 空元素标签都不加 “/” 字符。

2.2.2 代码嵌套

元素嵌套规范，每个块状元素独立一行，内联元素可选。

段落元素与标题元素只能嵌套内联元素。

**2.3 CSS**

2.3.1 样式文件

样式文件必须写上 @charset 规则，并且一定要在样式文件的第一行首个字符位置开始写，编码名用 “UTF-8”。

2.3.2 代码格式化

样式书写一般有两种：一种是紧凑格式 （Compact），一种是展开格式（Expanded）。

例如：展开格式（Expanded）

.jdc {

display: block;

width: 50px;

}

2.3.3 代码大小写

样式选择器，属性名，属性值关键字全部使用小写字母书写，属性字符串允许使用大小写。

2.3.4 代码易读性

① 左括号与类名之间一个空格，冒号与属性值之间一个空格。

例如：

.jdc {

width: 100%;

}

② 逗号分隔的取值，逗号之后一个空格。

例如：

.jdc {

box-shadow: 1px 1px 1px #333, 2px 2px 2px #ccc;

}

③ 属性值十六进制数值能用简写的尽量用简写

例如：

.jdc {

color: #fff;

}

④ 不要为 0 指明单位

例如：

.jdc {

margin: 0 10px;

}

2.3.5 CSS 属性书写顺序

先决定定位宽高显示大小，再做局部细节修饰！推荐顺序：定位属性(或显示属性，display)->宽高属性->边距属性(margin, padding)->字体，背景，颜色等，修饰属性的定义。

**2.4 JavaScript**

2.4.1 单行代码块

在单行代码块中使用空格。

例如：

function foo () { return true }

if (foo) { bar = 0 }

2.4.2在操作符前后都需要添加空格。

例如：var sum = 1 + 2

2.4.3 注意使用规范

① 使用‘===’和‘!==’

② vue项目中尽量减少或避免进行dom操作，全部通过vue数据进行驱动。

③ 使用ES6风格编码源码,定义变量使用let,定义常量使用const,使用export,import模块化。

④ 调试信息 console.log() debugger使用完及时删除。

⑤ 全局已经引入的组件，不要重复引用注册到局部。

Ps:对于规定的CSS和JS代码规范，目前项目中使用Eslint，保存的时候会自动规范代码

1. UTMP系统

3.1 UTMP项目目录说明

NEW-UTMP-WEB // 项目名

|- node\_modules // 下载的依赖包

|- public // 静态页面目录

|- static // 其他静态资源文件

|- favicon.ico // 网站图标

|- index.html // 项目入口

|- src // 源码目录

|- api // http 请求目录

|- assets // 静态资源目录，这里的资源会被wabpack构建

|- icons // icon 存放目录

|- images // 图片存放目录

|- style // 公共样式存放目录

|- components // 公共组件

|- config // 项目相关配置

|- constants // 一些常量

|- directives // 自定义指令

|- lib // 公共js文件

|- local // 多语言设置

|- plugins // 插件

|- router // 路由

|- routes // 详细的路由拆分目录（可选）

|- index.js

|- store // 全局状态管理

|- theme // 配置主题样式

|- views // 页面存放目录

|- App.vue // 根组件

|- main.js // 入口文件

|-main.less // 公共样式

|- tests // 测试用例

|- babelrc //

|- .editorconfig // 编辑器配置文件

|- .env.development // 开发环境配置

|- .env.local // 本地开发环境配置

|- .env.prod // 生产环境设置

|- .env.sit // 自定义环境配置

|- .eslintignore // eslint 忽略规则

|- .eslintrc.js // eslint 规则

|- .gitignore // git 忽略规则

|- .postcssrc.js // 转换CSS的工具配置

|- travis.yml // github 用于说明持续集成步骤配置文件

|- cypress.json // 基于JS的前端自动化测试工具配置

|- jsconfig.json // 指定JS 语言服务提供的功能的根文件和选项

|- package-lock.json // node\_modules中所有包的信息（版本、下载地址）

|- package.json // 依赖

|- README.md // 项目 README

|- vue.config.js // webpack 配置

目前做的UTMP系统，里面包含报销系统等小系统，目录结构须按照模板目录来。

比如，在components里建一个文件夹reimbursement，里面放报销系统需要的公共组件。（如果有新增的文件类型参照模板目录创建）

例如：| - components

| - reimbursement

| - business

| - common

Api 同理

| - api

| - reimbursement

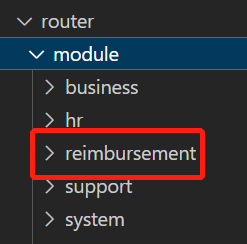
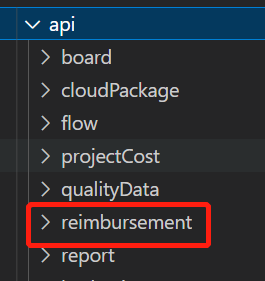
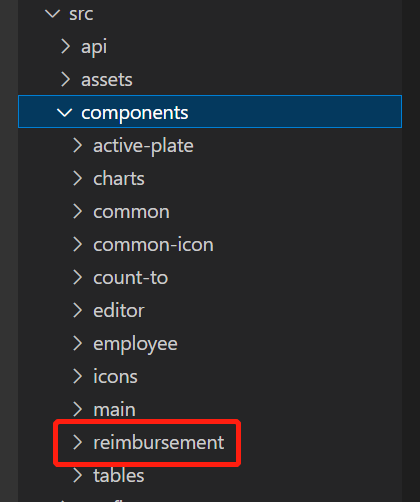
| - business

router同理

| - router

| - reimbursement

| - business



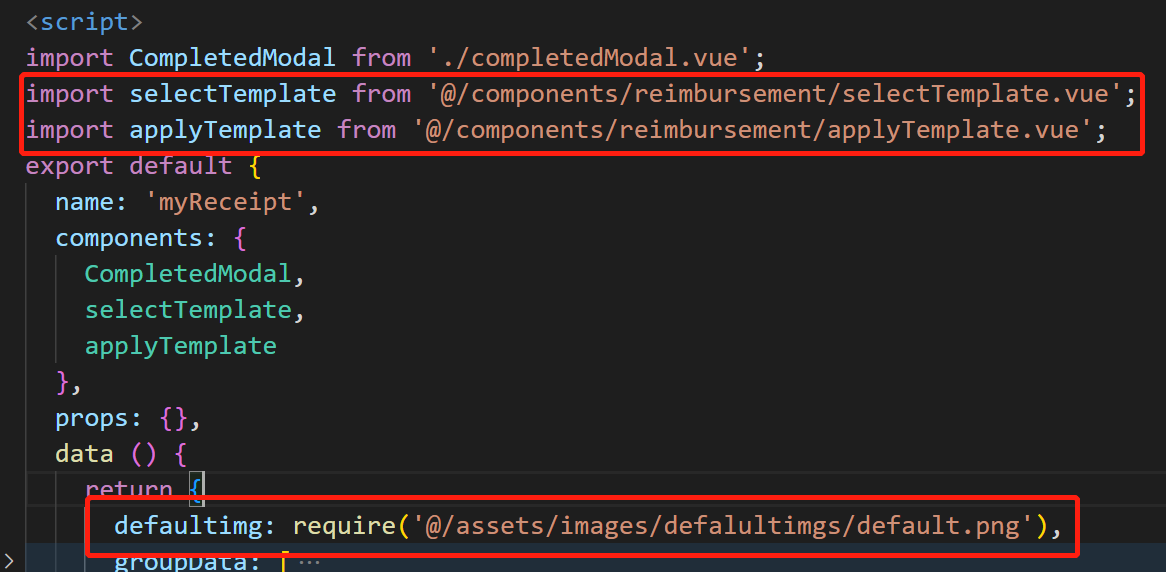
3.2 配置jsconfig.json

在文件中设置paths，方便在页面中引入子组件或图片。

Jsconfig.json:



在我的单据（myReceipt.vue）中使用：



3.3 使用第三方库view UI

目前项目里引入的是view UI第三方ui库，使用专有功能组件时优先考虑view UI，如果没有，再考虑使用其他的，不要引入多个重复功能的js文件。