vue项目开发规范文档

一、 目录结构

```
|— build                        构建脚本目录
   |— build.js                  生产环境构建（编译打包）脚本
   |— check-versions.js         版本验证工具
   |— utils.js                  构建相关工具方法（主要用来处理css类文件的loader）
   |— vue-loader.conf.js        处理vue中的样式
   |— webpack.base.conf.js      webpack基础配置
   |— webpack.dev.conf.js       webpack开发环境配置
   |— webapck.prod.conf.js      webpack生产环境配置
|— config                       项目配置
   |— dev.env.js                开发环境变量
   |— index.js                  主配置文件
   |— prod.env.js               生产环境变量
   |— test.env.js               测试环境变量
|— node_modules                 项目依赖模块
|— mock                         mock数据目录，用于本地数据模拟
|— src                          项目源码目录
   |— assets                    资源目录，资源会被webpack构建
      |— js                     公共js文件目录
      |— css                    公共样式文件目录
      |— images                 图片存放目录
   |— components                公共组件目录
   |— common
   |— network                   存放项目的网络模块，接口
   |— tools                     自己封装的一些工具
   |— App.vue                   根组件
   |— main.js                   入口js文件
   |— routers                   前端路由目录
      |— index.js
   |— pages                     前端页面文件
   |— store                     应用级数据管理
      |— index.js               组装模块并导出，统一管理导出，也可命名为store.js
      |— state.js               单一状态树，定义应用数据结构及初始化状态
      |— getters.js             获取state中的状态，仅单向获取数据，不做任何修改
      |— actions.js             调用mutation方法对数据进行操作
      |— mutation-types.js      存放vuex常用的变量
      |— mutations.js           定义state数据的修改操作
|— static   纯静态资源，不会被webpack构建，eg:没有npm包模块
|— test                         测试
|— unit                         单元测试
|— e2e                          e2e测试
|— .babelrc                     babel的配置文件
|— .editorconfig                编辑器的配置文件
|— .gitignore                   git的忽略配置文件
|— .postcssrc.js                postcss的配置文件
|— index.html                   html模板，入口页面
|— package.json                 npm包配置文件，依赖包信息
|— README.md                        项目介绍
```

二、 UI框架选择

PC端Vue项目UI框架优先选择：Element UI、iView
移动端Vue项目UI框架：mint-ui（优先）

三、 CSS预处理器选择

less / scss / stylus
如果采用element组件，优先推荐scss

四、 文件夹、组件命名规范，组件结构规范

> 名称应统一格式
>
> 1. 大驼峰命名法 PascalCase
> 2. 短横线连接 kebab-case
> 3. 避免不常用的单词缩写，见名思意，禁用一个单词进行命名。

1)项目的文件夹创建与命名：

> 项目文件夹与其项目一级导航菜单统一，便于管理，二级导航放在一级导航文件夹下，以此类推。

2)紧密耦合的父子组件命名：

> 一般以父组件名称开头+描述性单词

一般是多个单词全拼，减少简写的情况，这样增加可读性。
例如：video-detail ...
详情：detail
编辑：editor
查看：view
删除： delete
...

3)基础组件：

> 应用特定样式和约定的基础组件,以特定单词开头，如Base开头...

当项目中需要自定义比较多的基础组件的时候，比如一些button，input，icon...
页面级组件应该放到相对应页面文件夹下，比如一些组件只有这个页面用到，其他地方没有用到的。
项目级组件一般放到公共文件夹components下base文件夹给所有的页面使用。

4) 单例组件：

> 只应该拥有单个活跃实例的组件应该以 The 前缀命名，以示其唯一性。

5)组件书写：

1. 组件结构
   遵循从上往下template，script，style的结构。
2. 采用vue推荐的通讯方式，props、emit、vuex、attrs、listeners、provide、inject、refs...
3. props 必须定义值类型

五、 组件样式

单个组件样式一般可直接写到组件下style标签下，为了防止样式污染，可添加scoped 属性，也可以通过设置作用域来防止样式污染，写样式的时候尽量少写元素选择器，为了提高代码查找速度，可以用类选择器。

六、 文件格式

UTF-8格式

七、 Template模板文件

1. 尽量使用以.vue结束的单文件组件，方便管理，结构清晰。
2. 标签语义化，避免清一色的div元素
3. 尽量减少无用的标签嵌套
4. 样式class的命名：采用BEM规范

> 1. 给组件添加命名空间，表示 模块，防止和第三方组件命名冲突
> 2. 所有单词一律小写
> 3. 单词之间用 - 分隔，命名尽量不要超过三个单词，避免命名过长
> 4. 元素名称（Element）通过 __ 与块名称（Block）分隔
> 5. 修饰符名称（Modifier）通过 -- 与块（Block）或元素（Element）名称分隔
> 6. 在组件开发中避免使用全局的OOCSS原子类，因为这会降低组件的可复用性；如：pull-left、pull-right、clearfix
> 7. 尽量避免使用子选择器，如果层次关系过长，逻辑不清晰，非常不利于维护；如: .kso-nav ul li a {}

5. 多特性，分行写，提高可读性。即一个标签内有多个属性，属性分行写。
6. .自定义标签：使用自闭标签的写法。例如：，如果自定义标签中间需要传入slot，则写开始标签和结束标签，结束标签必须加/。
7. 组件/实例选项中的空行。便于阅读和代码架构清晰。

八、 Script

在 script 标签中，你应该遵守 Js 的规范和ES6规范。

1. 组件名称：必须以大写字母开头驼峰法命名。
2. Data必须是一个函数。
3. Props定义：提供默认值，使用type属性校验类型，使用props之前先检查prop是否存在
4. 调试信息 console.log() debugger使用完及时删除。
5. 为v-for设置Key值。
6. 使用计算 规避v-if和v-for用在一起。
7. 无特殊情况不允许使用原生API操作dom,谨慎使用this.$refs直接操作dom。
8. 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。
9. 指令缩写：都用指令缩写 (用 : 表示 v-bind: 和用 @ 表示 v-on:)。
0. 使用 data 里的变量时请先在 data 里面初始化。
1. 函数中统一使用_this=this来解决全局指向问题。
2. 能用单引号不用双引号。
3. 尽量使用===。
4. 声明变量必须赋值。

...

九、 Style

1. 使用 scoped关键字，约束样式生效的范围。

2. 避免使用标签选择器（效率低、损耗性能）。
3. 非特殊情况下，禁止使用 ID 选择器定义样式。有 JS 逻辑的情况除外。
4. 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。

十、 注释规范

> 页面注释至少占代码量的20%
> 注意在注释的前后各有一个空格。

1. HTML注释:
2. CSS注释：/* write your HTML comment! */
3. Style注释:
   a) 单行注释：// 我是less注释，和js的单行注释一样，在css中不输出
   b) 多行注释
   在这里插入代码片

```
/*
* less的多行注释，只有在compress选项未启用的时候
* 才会被输出
*/
```

c) 多行缓冲注释:

```
/*!
* less的多行缓存注释, Stylu压缩的时候这段代码无视
*/
```

4. JS注释:
   a) 行级注释（注意//后面空格）：// 正确的注释
   b) 变量声明注释：如果是在类似 Vue 项目的 data 属性中的变量，直接用行级样式跟在后面。
   例如：rightExample: ‘yes’, // 注释直接写这里
   c) 如果是在类，构造函数，或者常量定义中的变量，使用块级注释。
   例如:

```
/*
* 错误码常亮定义
* @type {number}
*/
```

d) 函数声明注释：不必要在每一个函数都写注释，但是在公共函数，还是建议补全注释，让后面的人不需要重复早轮子。
e) 复杂的业务逻辑处理说明、特殊情况的代码处理说明，对于特殊用途的变量、存在临界值、使用了某种算法思路进行注释说明

5. 方法注释
   /**
   * @author 方法创建者
   * @description 方法描述

```
* @param {string} 参数与参数类型
* @returns {Boolean} 结果及结果类型
*/
```

十一、 资源路径的配置、引入规则

路径配置

在config.js文件中配置。

```
alias: {
'@': resolve('src'), // 默认配置，设置src目录别名
'childRouter': resolve('src/pages/menuRouter'), // 子路由路径配置
'#': resolve('src/assets') // 配置assets文件夹路径
}
```

路径导入
a) Js文件中导入实例：
导入node_modules模块中的文件，直接引入即可，不需要加文件后缀名。
导入自定义文件的时候，使用相对路径或者使用路径配置别名，不许要加文件后缀名。
导入node_modules模块：import Vue from 'vue'
导入自定义文件：

```
import router from './router';
import scrollConfig from '#/js/vuescroll.config';
```

b) css样式导入需要使用 ~@ 开头

```
@import '~common/stylus/variable';
```

十二、vuex 数据中心

十三、 路由

十四、 axios

根据需要配置post、get请求，一个是取一个是贴，只需要读取文件，put（PUT 往服务器上上传文件）、delect（删除）直接对数据进行操作相对不安全 。

axios的挂载：Vue.prototype.$axios= axios。

axios使用封装后的get/post请求。

ajax的判断
首先 ajax 请求可以写在 actions也可以直接写在 .vue 页面里。
我们判断的依据是回调是否需要调用页面结构来区分，
比如在.vue页面中发送完请求后需要调用 this.$refs.element等,或者需要利用组件的独立性的效果时 后，那就写在.vue页面,否则就写在 actions 里。

十五、 api管理

新建src/ network/api.js

放置api路径，要注意 axios已经有了前缀,所以这里的 api 值需要写前缀之后的路径。当路径较多时可以再多建几个文件,分类放置。
例如：

```
// 统一管理接口
export default {
    manage: {
        fertilizerStation: '/api/AllFertSiteNameList', // 获取列表
        userLogin: '/api/Login' // 用户登录
    }
}
```

在main.js中引入：import api from ‘./request/api’ 。
使用Vue.prototype.api = api挂载到原型链上即可处处使用。

十六、 依赖规范

在package.json里增加包依赖

```
“dependencies”: {
    "axios": "^0.18.0"
}
```

十七、 Web字体规范

优先使用框架中的字体图标，比如element ui中的
使用iconfont字体图标代替图片

> 在规范中包括的字体格式有：
> woff: WOFF (Web Open Font 格式)
> ttf: TrueType
> ttf, otf: OpenType
> eot: 嵌入式 OpenType
> svg, svgz: SVG 字体

字体规则
a) 为了防止文件合并及编码转换时造成问题，建议将样式中文字体名字改成对应的英文名字，如：黑体(SimHei)、宋体(SimSun)、微软雅黑(Microsoft Yahei)。
b) 字体粗细采用具体数值，粗体bold写成700，正常normal写成400。
c) font-size必须以px为单位。
为了对font-family取值进行统一，更好的支持各个操作系统上各个浏览器的兼容性，font-family不允许在业务代码中随意设置。