# 前端开发编码规范

注意：为了提高代码质量，便于二次开发，请严格遵守以下开发规范，互相检查

## 代码目录说明

build 开发，发布构建Node脚本目录

src

components Vue文件，自定义组件目录

data 临时数据目录，菜单，页面资源

plugins 公共js目录

services Service，封装Api调用

store 全局变量

themes 主题，样式

App.vue 入口Vue

main.js 入口js

## Svn使用注意事项

* 1. 不能修改非自己创建的文件，如果需要修改，请联系文件创建者修改
  2. 编译异常，运行时chrome报异常，都不要提交
  3. 提交前检查是否有console.log相关调试信息，删掉再提交
  4. 以上三条，非常重要，请仔细检查，严格按项目管理要求执行
  5. 提交必须写注释，不能乱写
  6. 请勿删除文件，以免文件历史丢失
  7. 请勿频繁提及文件

## 命名规范

* 1. 不论是文件命名，还是变量命名，一定要语义化，一个单词或两三个单词命名，不清楚的，请查爱词霸，有道，有简写的单词就不要用全拼
  2. 严禁出现中文拼音命名
  3. 严禁出现字母加数字命名
  4. 命名尽量不要包含关键字
  5. 命名不要太短，也不要太长

### 文件命名

* + 1. 目录命名，采用横杠（英文）分隔，比如，user-manage
    2. Vue文件，Service文件命名首字母大写，其它文件首字母小写
    3. Components目录下，一个功能一个文件夹
    4. Js文件采用点号命名，比如jquery.min.js，app.config.js

### 变量命名

* + 1. 变量首字母小写，不要使用下划线，不要首字母大写
    2. Vue文件参考功能组件开发规范
    3. Css文件参考样式主题规范

### Services命名

* + 1. 一个Service对应一个RestApi请求，对应一种资源
    2. Service方法跟Api方法对应
    3. 调用HttpService的get/post方法
    4. Service还需要负责转换数据（Api有时候返回的数据属性跟需要的不一致）
    5. Service里面禁止调用其它Service（除了公共Service）
    6. Services写法参考UserService，FrameService

## 功能，组件开发规范

每一个功能，模块都是一个组件，每个组件是独立的，实现一个功能，组件可以在任何地方被重复使用，组件通过传参初始化，通过派发事件返回结果

### 组件开发

组件包括两部分，Template + Script

布局采用24列栅格布局 + 响应式布局

Dialog组件必须引入：mixins: [dialogMixin]（'plugins/dialog.mixin.js'），可以继承其方法

### 命名规范

* + 1. 模块组件命名，要能看出模块功能
    2. 属性命名，事件命名要语义化
    3. Import Service 使用解构
    4. Import 组件时，名称跟组件名称一致

### 国际化

* + 1. 页面显示资源国际化，

所有需要显示的内容都必须通过国际化函数转换：$l(‘res\_code’)

* + 1. 字典数据国际化

需要用到字典的地方，不要自己去调用字典Api获取数据，使用相关公共组件实现，比如使用select，radio，checkbox

### 注意事项

* + 1. 组件必须有name属性
    2. Template必须紧跟一个<div>，且该div必须有一个class
    3. 尽量不使用原生html标签
    4. 禁止在Vue文件里写css
    5. 禁止在标签里写style
    6. 组件属性尽量写到一行，不要超过IDE的分割线就行
    7. 所有组件属性禁止使用未定义变量，必须先在data、methods里定义
    8. 如果data字段太多，请放到对象里面，不要全部放到data下
    9. 组件属性绑定，禁止属性包含大写，只能用横杠
    10. 块级标签，如果看起来显示很紧凑，可以考虑空行
    11. 尽量不要出现滚动条，弹出框内容尽量紧凑

## 样式，主题规范

* 1. 样式和js代码Vue文件需要独立开发，独立编译，发布
  2. 样式分为两部分，ElementUI + 我们写的css，也需要分开编译，发布
  3. Npm run el-theme，Npm run qz-theme（更多请看根目录Package.json scripts部分）

### 代码目录

Themes

common 主题变量定义

el-default.css Element 默认主题

el-blue.css Element 蓝色主题

qz-default.css 强智科技默认主题

qz-blue.css 强智科技蓝色主题

dist 主题发布目录

el-default/index.css

el-blue/index.css

qz-default/index.css

qz-blue/index.css

modules Css源代码目录，我们的工作目录，每个Vue文件对应一个Css文件

index.css @import xxx.css

xxx.css

### 命名规则

* + 1. Css文件命名，采用横杠分隔
    2. Css 变量命名，也是横杠分隔

### Css开发

采用cssnext开发，如

@bn qz {

@b app-manage {

.app-list {xxx}

.app-info {xxx}

}

}

生成结果：

.qz-app-manage .app-list {xxx}

.qz-app-manage .app-info {xxx}

### 主题

为了适应多主题，大小，字体，颜色，长宽都需要定义变量，再使用

### 图标

Icomoon在线生成，除了接口返回图片链接，禁止使用img标签，用css + icon显示图片

## ES6编码规范

* 1. 能用ES6写的，就不要用原生的js写法
  2. 数组方法: 禁止使用for循环，用forEach, filter, map, some, every
  3. 对象方法：Object.keys遍历，Object.assign复制
  4. 解构，箭头函数，Import，Export

## 注释规范

* 1. Js注释主要写在Services方法上，注释跟Api文档要一致
  2. Vue文件Template部分，国际化的地方必须写中文注释
  3. Vue文件的Js部分，变量，方法都要写注释
  4. Js函数里面，逻辑比较复杂的，写清楚每个步骤的注释

## 公共方法/自定义组件使用

### Utils用法

* + 1. dictionary 字典数据转换
    2. language 页面资源多语言转换
    3. removeDuplicate 数组去重
    4. setLang, getLanguages 多语言录入

### Http 方法

* + 1. get get api
    2. postJSON application/json Post提交
    3. postForm form 格式提交表单

### 多语言录入

### 字典选择

* + 1. qz-dict-select 字典下拉选择
    2. qz-dict-radio 字典radio控件
    3. qz-dict-checkbox 字典checkbox
    4. qz-language-input 多语言录入

### 表格字典转换

QueryService.queryJSON

## Demo开发步骤