web前端规范

§1、概述

人多不一定力量大，还有可能一团糟。
规范的目的：统一、高效、便于维护

§2、文件规范

§2.1、文件命名规则

§2.1.1、总体原则

1、默认小写字母，专有名词可大写：
1）文件夹/文件：字母+数字+下划线
2）css：字母+数字+中划线
3）js（方法/变量）：字母+数字+驼峰式
2、杜绝拼音，统一用英文，中英文翻译参照百度翻译。
3、共性信息在前，个性信息在后。
4、大范围在前，小范围在后。
5、重要在前，次要在后。

§2.1.2、图片文件命名原则

1、主目录--images
2、子目录：每次项目迭代，新建子目录存放图片，子目录命名规则为：
“ 年月_迭代主题 ”，如 Sep18_init：2018年9月项目初始化
3、图片目录结构大体如下：

```
├─images
  ├─UI_April18
  ├─...
```

4、图片命名规则
1）命名加上属性前缀

前缀	含义
back_	背景图片
banner_	banner图
code_	码图（如二维码）
default_	默认图片

前缀	含义
figure_	配图
icon_	图标
logo_	logo
nodata_	无数据图
photo_	头像

例：icon_star　photo_doctor

§2.2、垃圾回收机制

1、在src同级目录下，建useless文件夹，专门用于存放暂时无用的垃圾文件，并定期清理。
ps：src同级文件不参与打包，不会影响包大小。
2、删改内容后注意观察全局
1）确定无用则删掉
2）将来可能有用，则移入useless文件夹。

§3、样式规范

样式统一用scss语法
详见 水果样式理论

§4、代码规范

§4.1、注释

1、方法要加注释
2、易混淆的变量和逻辑要加注释

§4.2、方法命名

驼峰式：动词+名词形式，体现动作和作用对象

§4.3、变量命名

1、驼峰式
2、特殊用途、或有某种含义的用固定前缀

前缀	含义	类型	用法示例
if	v-if的变量	Boolean	v-if="ifList"
show	v-show的变量	Boolean	v-show="showList"
is	表示"是不是"含义的	Boolean	isCheck

前缀	含义	类型	用法示例
has	表示"有没有"含义的	Boolean	hasList
able	表示"能不能"含义的	Boolean	ableSubmit // true能，false不能 :disabled="!ableSubmit"
img	表示图片	String	imgList

§4.4、代码格式（以下设置是针对vscode编辑器的）

1、语句结尾不加分号，依据**eslint**规则。
2、tab缩进，空格4
1）文件-首选项-设置
2）加入以下设置

```
    "editor.tabSize": 4,
```

3、eslint代码校验，统一代码格式
1）安装ESLint插件
2）文件-首选项-设置
3）加入以下设置，保存文件即可自动按eslint规则格式化

```
    "eslint.autoFixOnSave": true,
    "eslint.validate": [
        "javascript",{
            "language": "vue",
            "autoFix": true
        },"html",
        "vue"
    ]
```

4、常用快捷键设置
1）文件-首选项-快捷键 -> keybindings.json
2）加入以下快捷键

```
// 将键绑定放入此文件中以覆盖默认值
[
  // ctrl+d 删除当前行
  {
    "key": "ctrl+d",
    "command": "editor.action.deleteLines",
    "when": "editorTextFocus && !editorReadonly"
  },
  // ctrl+shift+d 复制当前行到下一行
  {
    "key": "ctrl+shift+d",
```

```
    "command": "editor.action.copyLinesDownAction",
    "when": "editorTextFocus && !editorReadonly"
  },
  // ctrl+shift+/ 多行注释
  {
    "key": "ctrl+shift+/",
    "command": "editor.action.blockComment",
    "when": "editorTextFocus"
  }
]
```

5、推荐安装插件
Auto Close Tag：自动闭合标签

§4.5、细节

§4.5.1、v-for循环

1、一级循环
v-for="(item, index) in mainList"
v-for="(item, index) in mainList2"
ps: 可在数据定义处注释mainList的含义
2、二级循环：j标识
1）一般：
v-for="(jitem, jindex) in item.xxx"
2）特殊：
v-for="(jitem, jindex) in jmainList"
v-for="(jitem, jindex) in mainList2"
3、三级循环：k标识

§4.5.2、无数据

1、规则定义
1）nodata = null 初始化
2）nodata === true 无数据（true可省略）
3）nodata === false 有数据（一般用不上）
2、用法
1）定义全局方法

```
switchNodata (list) {
  return !list || list.length === 0
}
```

2）数据请求得到结果

```
xxx.then(res => {
  this.list = res.data
  this.nodata = this.$util.switchNodata(this.list)
})
```

3）html

```
<empty v-if="nodata">无数据</empty>
```

3、无数据组件

```
<div v-show="visible">
  <div v-if="type==='xxx'">
    <slot v-if="hasSlot">
    </slot>
    <slot v-else>
        <p>xxxxxx</p>
    </slot>
  </div>
</div>
props: {
  visible: {
    type: Boolean,
    required: true
  },
  type: {
    type: Boolean,
    required: false
  }
  hasSlot: {
    type: Boolean,
    default: false,
    required: false
  }
}
```

§5、git规范

§5.1、建分支的原则

1、git分支为成员协作服务
2、协作是以功能模块为单位

§5.2、建分支的场景

当多人进行同一个功能模块的开发，缺乏一个方便协作的分支时，才去建分支。

§5.3、分支命名

1、分支命名，不用局限于人、或功能模块，便于理解达成共识即可。
2、小写字母、中划线。

§5.4、git commit 注释规范

1、一般情况下，提交git时的注释可以分成几类，可以用几个动词开始：
1）added: 新加入的需求
2）fixed: 修复 bug
3）changed: 完成的任务
4）updated: 更新的任务
2、概述提交内容，有多条的话，分别罗列1、2、3。

§6、项目框架规范

§6.1、框架设计

详见 框架设计

§6.2、图片资源存储与引用

§6.2.1、src/assets和static的区别

详见 vuejs处理静态资产

处理静态资产

您会注意到在项目结构中我们有两个静态资产目录： `src/assets` 和 `static/` 。它们之间有什么区别？

Webpacked资产

要回答这个问题，我们首先需要了解Webpack如何处理静态资产。在 `*.vue` 组件，您的所有模板和CSS被解析 `vue-html-loader` 并 `css-loader` 寻求资产的URL。例如，在 `<img src="./logo.png">` 和中 `background: url(./logo.png)` ， `"./logo.png"` 是一个相对资产路径，将由**Webpack解析为模块依赖项。**

因为 `logo.png` 不是JavaScript，当被视为模块依赖时，我们需要使用 `url-loader` 和 `file-loader` 处理它。此模板已经为您配置了这些加载器，因此您可以免费获得文件名指纹识别和条件base64内联等功能，同时可以使用相对/模块路径而无需担心部署。

由于这些资产可能在构建期间内联/复制/重命名，因此它们本质上是源代码的一部分。这就是为什么建议将Webpack处理的静态资源放在 `/src` 其他源文件中。实际上，您甚至不必将它们全部放入 `/src/assets` ：您可以使用它们基于模块/组件来组织它们。例如，您可以将每个组件放在其自己的目录中，其静态资产就在它旁边。

资产解决规则

- **相对URL**，例如 `./assets/logo.png` 将被解释为模块依赖性。它们将替换为基于Webpack输出配置的自动生成的URL。
- **非带前缀的URL**，例如， `assets/logo.png` 将被视为与相对URL相同并被翻译成 `./assets/logo.png` 。
- **带有前缀的URL** `~` 被视为模块请求，类似于 `require('some-module/image.png')` 。如果要利用Webpack的模块解析配置，则需要使用此前缀。例如，如果您有解析别名 `assets` ，则需要使用 `<img src="~assets/logo.png">` 以确保遵守别名。
- **根相对URL**，例如 `/assets/logo.png` 根本不处理。

§6.2.2、图片资源存储

统一存放在**src/assets/images**文件夹下

§6.2.3、图片资源引用

1、给'src/assets/images'路径配置一个默认标识符：'~'，便于图片和其他资源区分，单独管理。

```
webpack.base.conf.js:
'~': resolve('src/assets/images')
```

2、image标签src引入：用于一般配图

```
html:
<image :src="imgXxx" style="width: 200rpx; height: 200rpx;"></image>
js:
data: () => ({
    imgXxx: require('~/Sep18_init/figure_xxx.png')
})
```

3、样式background引入：用于图标

```
main.scss:
// 图标
$theicons:
(truck, 60, 60),
(truck, 50, 50);
@each $icon, $width, $height in $theicons {
  .theicon-#{$icon} {
    display: inline-block;
    width: #{$width}rpx;
    height: #{$height}rpx;
    background: url(../images/Sep18_init/icon_#{$icon}.png) no-repeat;
    background-size: 100% 100%;
  }
}
```

ps: css中一定要用相对路径，才会被解释为模块依赖，编译生成base64编码。所以，将需要用background去引用图片的样式，集中放在同一个样式文件（如：main.scss）进行统一管理。

§7、CodeReview跟踪

§7.1、小程序CodeReview

详见 小程序CodeReview记录