README

Part I. 项目本地调试和打包

1. 技术栈

`vue2 + vuex + vue-router + webpack + es6 + axios + sass + flex`

2. 目录结构

```
├── README.md                                    # 项目说明
├── build                                        # 打包
|   ├── build.js
|   ├── check-versions.js
|   ├── logo.png
|   ├── utils.js
|   ├── vue-loader.conf.js
|   ├── webpack.base.conf.js
|   ├── webpack.dev.conf.js
|   └── webpack.prod.conf.js
├── config                                       # 打包配置
|   ├── dev.env.js
|   ├── index.js
|   └── prod.env.js
├── dist                                         # 打包后目录
|   ├── index.html
|   └── static
|       ├── css
|       ├── img
|       └── js
|       └── config.sample.js                     #API环境配置样板文件（服务端需要参考此文件，
配置config.js；和config.sample.js放置在同目录）
├── index.html
├── package-lock.json
├── package.json
├── src                                          # 代码目录
|   ├── App.vue                                  # 根组件
|   ├── api                                      # api模板（yumi生成，无需修改）
|   |   ├── index.js                             # api索引（可选）
|   ├── assets                                   # 资源文件
|   |   ├── image                                # 图片资源
```

```
|   |   ├── font                          # 字体文件
|   |   ├── js                            # 第三方js插件
|   |   └── style
|   |       ├── _variable.scss            # scss变量文件
|   |       ├── mixin.scss                # scss函数文件
|   |       └── reset.scss                # 重置浏览器样式文件
|   ├── components                        # 公共组件
|   |   └── common
|   ├── const                             # 项目常量目录
|   |   ├── enum.js                       # 枚举文件
|   ├── main.js                           # 程序入口
|   ├── server                            # 网络请求（api的公共部分）
|   |   └── network.js
|   ├── page                              # 业务模块
|   |   ├── auth
|   |   ├── product                       # 商品模块
|   |   |   ├── child                     # 附属的子模块组件
|   |   |   ├── product.vue               # 模块组件
|   |   |   └── static.js                 # 组件所需要的静态数据(可选)
|   |   ├── home
|   ├── router                            # 路由组件
|   |   └── router.js
|   ├── service                           # service部分（可选）
|   ├── store                             # 状态管理
|   |   ├── actions.js                    # 根级别的 action
|   |   ├── getter.js                     # 根级别的 getter
|   |   ├── index.js
|   |   ├── modules                       # 模块部分
|   |   |   ├── auth.js
|   |   |   └── config.js
|   |   └── mutations.js                  # 根级别的 mutation
|   └── util                              # 公共的方法(原型链方式加上)
|       └── util.js
└── static
    ├── config.js                         #API环境配置
    ├── config.sample.js                  #API环境配置（sample、便于服务端配置参考）
```

3. 项目本地调试

项目使用npm管理包依赖；本地调试步骤如下：

- 运行 `npm install` 命令安装依赖库
- 如果需要修改API地址，在 `/static/config.js` 文件中修改 `API_HOST` 即可
- 运行 `npm run dev` 本地调试

4. 项目打包

项目使用webpack打包，打包配置已设置好；打包步骤如下：

- 运行 `npm install` 命令安装依赖库（如果已运行，请忽略该步骤）
- 如果需要修改API地址，在 `/static/config.js` 文件中修改 `API_HOST` 即可
- 运行 `npm run build` 完成打包；/dist即为打包生成目录

5. 服务端配置

前端抽离了接口配置的config文件；服务器端需要在/dist/static/路径下，配置config.js文件；文件内容同config.sample.js文件

Part II. 代码规范

1. 命名规范

文件夹命名

- 统一命名为小写
- 业务模块文件夹（page）代表着模块的名字
  - 由名词组成（car、order、cart）
  - 单词只能有一个 （good: car order cart） （bad: carInfo carpage）

文件命名

- 组件文件命名 PascalCase (单词首字母大写命名)
  - PascalCase 是最通用的声明约定而 kebab-case 是最通用的使用约定（备注：参阅组件命名约定）
  - 尽量是名词
  - 大写开头，开头的单词就是所属模块名字（CarDetail、CarEdit、CarList）
  - 常用结尾单词有（Detail、Edit、List、Info）
- 其它js文件命名小写

2. 代码格式

- 换行
  - 换行为2个空格

```
<div id="app">
  {{ message }}
</div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

3. vue文件结构

template

```
<template>
  <div class="container">
    <mt-header class="header" title="我的优惠券">
      <header-item slot="left" v-bind:isBack=true v-on:onclick="goBack">
      </header-item>
    </mt-header>
    <div class="tips-wrapper">
      <label class="tips">可使用优惠券{{total}}张</label>
    </div>
    <div class="list">
      <coupon-item
        class="item"
        v-for="item in items"
        :key="item.id"
        :item="item"
        :isSelected="isSelected(item)"
        v-on:onclick="onclick(item)">
      </coupon-item>
    </div>
    <div class="submit" @click="unselect">
      <label class="text">不使用优惠券</label>
    </div>
  </div>
</template>
```

1. 组件元素之间没有空行
2. 组件太长时，换行显示（换行缩进2个字符）

script

```
<script>
import CouponItem from './child/CouponItem'
import { mapState, mapMutations, mapActions } from 'vuex'
export default {
  name: 'component-name',
```

```
components: {
  CouponItem,
},
mixins: [ ],
directives: {
},
props: {
  mode: {
    type: String,
    default: null
  }
},
data () {
  return {
    items: [],
  }
},
created () {
},
mounted () {
},
computed: {
  ...mapState({
  }),
  isEmpty () {
    if (this.items && this.items.length === 0) {
      return true
    }
    return false
  }
},
watch: {
  value (val) {
  },
  currentValue (val) {
  }
},
methods: {
  ...mapMutations({
  }),
  ...mapActions({
  }),
  goBack () {
    this.$router.go(-1)
  },
  onLeftClick () {
    this.goBack()
  },
```

```
    onRightClick () {
    }
  }
}
</script>
```

1. name、components、props、data、created、mounted、computed、methods尽量按照顺序写；并且各元素之间无换行
2. computed中多个计算属性之间无换行；类似的，methods中多个方法之间无换行，可适当添加注释

style

```
<style lang="scss" scoped>
  .ui-concat-wrapper {
    display: flex;
    width: auto;
    margin-top: 8px;
    background-color: #ffffff;
    padding: 15px;
    justify-content: space-between;
    align-content: center;
    align-items: center;
    p {
      font-size: 16px;
      font-family: PingFangSC-Regular;
      color: rgba(78,84,93,1);
      line-height: 20px;
    }
    img {
      width: 12px;
      height: 13px;
      cursor: pointer;
    }
  }
</style>
```

1. 类选择器后有1个空格
2. 样式值前有1个空格
3. 使用Sass写法