清晰且可承载足够大业务而不乱的目录结构

```
{
    common : {
        api : [
            'index.js','order.js'//'API统一管理',按业务分文件
        ],
        dicts : [
            'order.js','goods.js'//一些存在前端的状态等字典表，按业务分文件
        ],
        plugins : [
            'msbUniRequert.js' //一些自己写的插件相关，注意与工具类的区分
        ],
        rotuer:[
            'index.js','account.js','order.js' // 这里可以按照业务分模块把路由
        ],
        store: [
            'index.js','model' //vuex状态管理，按模块管理
        ],
        utils : [
              'index.js','utils.js','xx.js', // 工具类，可以按类型分文件，但所有
        ]
    },
    components:[
        'BsXx.vue','UiXx.vue' // 公共组件，注意最少两个不同页面使用到的组件才能放
    ],
    pages : {
          index : { //一个页面如果业务比较复杂可以拆分成多个模块
              components : ['Header.vue','Banner.vue'], //在文件夹内新建compon
              index.vue : ''
```

公共方法开发规范

- 公共方法统一放置utils文件夹内，可以按分类建方法文件 如：验证类verify.js 请求类request.js；
- 所有公共方法采用大驼峰命名法
- 所有的方法都从index.js输出，引入时统一引入index，不允许直接引入方法文件
- 所有方法文件如果导出的是多个方法，不允许在定义方法时导出，必须在文件底部一一导出，并附上方法简单的注释

```
//引用示例--------------------------------
// 正确
import {Req, IsPhone} from '@/common/utils';

// 错误
import {Req} from '@/common/utils/request';
import {IsPhone} from '@/common/utils/utils';


//单文件多方法公共方法编码示例--------------------------
// 正确
const IsPhone = (str) => {....}
const IsEmail = (str) => {....}
export {
    // 判断手机号
    IsPhone,
    // 判断邮箱
    IsEmail
}

// 错误
export const IsPhone = (str) => {....}
export const IsEmail = (str) => {....}

```

组件开发规范

- 请务必使用easycom模式引入第三方组件
- 根目录的components 只放置真正的组件（跨页面使用），某个页面的业务模块应该在pages的相应目录下新建components目录放置
- 所有的自定义组件文件名以大驼峰命名，且在templet中使用也用大驼峰形式使用

图片存储

- 按pages目录文件夹结构放置图片
- 有公共使用的图片请放到static/common文件夹内