**UniApp 编码风格文档**

**1. 项目结构**

（1）按功能模块划分目录，目录结构清晰扁平化。

（2）推荐的项目目录结构：

|  |
| --- |
| 1. ├── common/          # 公共文件（如工具函数、常量配置等） 2. ├── components/      # 通用组件 3. ├── pages/           # 页面 4. │   ├── index/       # 首页 5. │   ├── user/        # 用户模块 6. │   ├── ...          # 其他模块 7. ├── static/          # 静态资源 8. ├── store/           # 状态管理（如 Vuex） 9. ├── uni\_modules/     # UniApp 插件 10. ├── App.vue          # 入口文件 11. ├── main.js          # 主配置文件 12. ├── manifest.json    # 应用配置 13. ├── pages.json       # 路由配置 14. ├── uni.scss         # 全局样式 |

**2. 代码规范**

**2.1 文件命名**

（1）文件名使用小写字母并以下划线分隔，避免使用驼峰命名。

（2）Vue 组件文件统一以 .vue 结尾。

|  |
| --- |
| 1. index.vue            *// 页面文件* 2. user\_profile.vue     *// 用户模块文件* |

**2.2 缩进与行长**

（1）使用 2 个空格进行缩进。

（2）每行代码不超过 100 个字符，超过的部分进行换行。

**2.3 组件命名**

（1）全局组件使用 PascalCase 命名。

（2）本地组件使用 kebab-case 命名。

|  |
| --- |
| 1. *// 全局组件* 2. import HeaderBar from '@/components/HeaderBar.vue'; 3. *// 本地组件* 4. <custom-button /> |

**3. 模板规范**

**3.1 标签属性**

（1）标签属性按功能分组：指令、绑定、事件、普通属性，按顺序书写。

（2）每组属性换行，保持清晰。

|  |
| --- |
| 1. <view 2. v-if="isVisible" 3. :class="['container', customClass]" 4. @click="handleClick" 5. id="mainView"> 6. 内容 7. </view> |

**3.2 自闭合标签**

（1）单标签必须使用自闭合格式。

|  |
| --- |
| 1. *<!-- 正确 -->* 2. <image src="/static/logo.png" /> 3. *<!-- 错误 -->* 4. <image src="/static/logo.png"></image> |

**3.3 组件引用**

（1）组件引入统一放在 components 块。

|  |
| --- |
| 1. <script> 2. import CustomButton from '@/components/CustomButton.vue'; 3. export default { 4. components: { CustomButton }, 5. }; 6. </script> |

**4. 样式规范**

**4.1 样式语言**

（1）优先使用 SCSS 或 LESS，减少代码重复。

（2）避免使用内联样式，样式尽量抽取到 .scss 文件中。

**4.2 命名规范**

样式类名采用 kebab-case，使用具有语义化的名称。

|  |
| --- |
| 1. // 推荐 2. .user-card { 3. background-color: #fff; 4. } 5. .user-card\_\_header { 6. font-size: 16px; 7. } 8. // 不推荐 9. .userCard { 10. background-color: #fff; 11. } |

**4.3 全局样式**

（1）全局样式写在 uni.scss 中，避免污染组件内样式。

**5. 脚本规范**

**5.1 数据声明**

（1）页面或组件的 data 必须返回一个对象。

|  |
| --- |
| 1. export default { 2. data() { 3. return { 4. userInfo: null, 5. isLoading: false, 6. }; 7. }, 8. }; |

**5.2 方法命名**

（1）事件处理方法以 handle 开头，异步方法以 async 开头。

|  |
| --- |
| 1. methods: { 2. handleClick() { 3. console.log('Button clicked'); 4. }, 5. async fetchData() { 6. const response = await getData(); 7. this.data = response.data; 8. }, 9. }, |

**5.3 生命周期**

（1）生命周期钩子按顺序书写：onLoad → onShow → onReady → onHide → onUnload。

**6. 路由配置**

（1）页面路径在 pages.json 中按模块分组，路径命名语义化。

|  |
| --- |
| 1. { 2. "pages": [ 3. { 4. "path": "pages/index/index", 5. "style": { 6. "navigationBarTitleText": "首页" 7. } 8. }, 9. { 10. "path": "pages/user/profile", 11. "style": { 12. "navigationBarTitleText": "个人中心" 13. } 14. } 15. ] 16. } |

**7. 代码注释**

**7.1 页面注释**

（1）每个页面文件顶部需要添加页面功能说明。

|  |
| --- |
| 1. /\*\* 2. \* 用户中心页面 3. \* 提供用户信息展示与编辑功能 4. *\*/* |

**7.2 函数注释**

（1）每个函数必须有注释，说明功能和参数。

|  |
| --- |
| 1. */\*\** 2. \* 获取用户信息 3. \* @param {number} userId - 用户 ID 4. \* @returns {Promise<object>} 用户信息 5. \*/ 6. async fetchUserInfo(userId) { 7. return await api.getUserInfo(userId); 8. } |

**8. 工具推荐**

代码格式化：使用 ESLint + Prettier。

代码检查：引入 Husky 在提交代码前强制检查代码质量。