- **命名规范**：

参照 JavaScript 和 React 的命名习惯，给变量、函数和组件取有意义的名字，使用驼峰式大小写，避免缩写或拼音。文件名要用 .js 或 .jsx 后缀 。

- **排版与格式**：

按照 ESLint 和 Prettier 的标准来格式化代码，注意缩进、空格、分号、换行等细节，使代码看起来整齐美观 。

- **注释规范**：

在合适的地方添加注释，说明代码的功能、逻辑、参数等信息，方便阅读和维护。注释要简明扼要，及时更新。

- **组件规范**：

利用 React 的思想来构建和复用 UI 组件，让组件功能单一，接口清晰。尽量用函数式组件代替类组件，用 Hooks 代替生命周期方法，用自定义 Hooks 代替高阶组件或 render props 等 。

- **样式规范**：

用 StyleSheet API 来创建和管理组件的样式，不要用内联样式或全局样式。把样式对象放在组件定义的最后，按照一定的顺序排列属性，如布局、尺寸、颜色、字体等。让样式对象的名字与组件或元素相匹配 。

- **状态规范**：

用 state 和 props 来控制组件的状态，不要直接修改 state 或 props。让 state 尽量简单和最小化，避免状态的嵌套或冗余。让 props 尽量明确和完整，避免 props 的过多或过少。用 PropTypes 或 TypeScript 来检查和文档化 props 的类型 。

- **导航规范**：

用 React Navigation 或其他库来实现应用的导航功能，如栈导航器、标签导航器、抽屉导航器等。让导航器有清晰的结构和配置，不要让导航逻辑过于复杂或混乱。把导航器和屏幕组件分开定义和管理 。

- **网络规范**：

用 Fetch API 或其他库如 Axios 来实现应用的网络请求功能，如获取数据、上传文件、发送表单等。给网络请求加上合理的错误处理和异常处理机制