**项目代码规范文档**

1. **代码命名规范**

1. 组件名称：使用大驼峰命名，每个单词首字母大写。例如：

- `CreatePost`

- `UserProfile`

- `CommentSection`

2. 变量、函数名称：使用小驼峰命名，第一个单词首字母小写，后续单词首字母大写。例如：

- `userName`

- `fetchPosts`

- `postContent`

3. CSS 类名：使用kebab-case，单词间用连字符连接，全小写。例如：

- `post-container`

- `comment-list`

- `button-primary`

4. 常量名称：使用全大写下划线分隔单词，通常用于配置项或不变的数据。例如：

- `API\_BASE\_URL`

- `MAX\_COMMENT\_LENGTH`

5. 文件名：

- 组件文件使用大驼峰命名：如 `CreatePost.js`。

- 辅助或工具文件使用小驼峰命名：如 `formatDate.js`。

- 样式文件使用kebab-case：如 `post-container.css`。

**二、代码结构规范**

1. 文件结构：

- 组件文件放置在 `components/` 目录下，每个组件有独立文件夹。

- 公共方法放在 `utils/` 目录中，如数据处理函数、API请求等。

- 样式文件放在 `styles/` 目录中，组件对应样式文件放在组件文件夹内。

2. 导出规则：

- 单一导出组件使用 `export default`。

- 多个导出则使用 `export { ... }`，在导入时使用解构形式。

3. 代码风格规范

1. 缩进：统一使用 2个空格 进行缩进。

2. 引号：优先使用单引号 `' '`。

3. 行宽：代码行长度不超过 80字符，必要时换行。

4. 分号：每行语句末尾使用分号，确保代码清晰、减少错误。

5. 空格：

- 算术运算符两边留空格，例如 `a + b`。

- 控制结构关键词与括号之间保留一个空格，例如 `if (condition)`。

4. 注释规范

1. 单行注释：用于简单说明，在代码上方单独一行，使用 `//`。

2. 多行注释：用于复杂逻辑或函数说明，使用 `/ ... /` 格式。

5. 代码格式化

1. 格式化工具：建议使用 `Prettier` 进行代码格式化，统一团队代码风格。

2. Lint工具：使用 `ESLint` 进行代码检查，捕捉潜在错误。

6. 代码提交规范

1. 分支命名：分支名称清晰、描述本次改动。格式为 `feature/功能`、`fix/bug描述`。

2. 提交信息：每次提交需编写简洁的描述，清晰表述本次改动内容。例如：

- `feat: 添加用户创建功能`

- `fix: 修复页面加载bug`