JavaScript 技术管理文档

一、命名规范

1. 变量/函数命名

a 使用 ’camelCase’ 风格（如 ’userName’），禁止拼音或中文命名 。

a 命名需语义明确（如 ’getUserData’ 而非 ’getData’）。

b 布尔变量以 ’is’/’has’ 开头（如 ’isActive’）。

c 循环计数器允许单字符（如 ’i’、’j’）。

2. 常量命名

a 全大写 + 下划线分隔（如 ’MAX\_COUNT’）。

3. 类/构造函数

a 使用 ’PascalCase’（如 ’class UserModel’）。

b 文件名与类名保持一致（如 ’UserModel.js’）。

二、编码风格

4. 缩进与空格

a 缩进统一为 2 个空格，禁用 Tab 。

b 操作符两侧加空格（如 ’let sum = a + b;’）。

a 函数参数间加空格（如 ’function(a, b, c)’）。

5. 括号与换行

a 控制语句（’if’/’for’）必须加 ’{}’，即使单行 。

b 左大括号 ’{’ 不换行（如 ’if (condition) {’）。

6. 引号与符号

a 字符串统一用单引号（’'’），模板字符串用反引号（’’ ’ ’’）。

b 对象末尾禁止多余逗号（如 ’{ name: 'Alice' }’）。

三、语言特性

7. 变量声明

a 使用 ’const’ 或 ’let’，禁用 ’var’ 。

b 按作用域分组声明（如模块级 ’const’ 前置）。

8. ES6+ 语法

b 优先用箭头函数（’() => {}’）替代 ’function’ 。

b 使用解构赋值（如 ’const { id } = user;’）。

9. 错误处理

a 异步错误用 ’try/catch’ 或 ’.catch()’ 捕获 。

b 抛出 ’Error’ 对象而非字符串（’throw new Error('msg')’）。

10. 异步操作

b 优先用 ’async/await’ 替代 ’Promise.then()’ 。

四、代码结构

11. 模块化

a 使用 ’import/export’ 替代 ’require’ 。

b 一个文件仅导出一个类或功能模块 。

12. 函数设计

a 函数不超过 50 行，参数不超过 3 个 。

b 功能单一，避免副作用 。

13. 条件与循环

a 避免嵌套超过 3 层（可拆分为函数）。

b 用 ’Array.map/filter’ 替代 ’for’ 循环 。

五、注释与文档

14. 代码注释

a 复杂逻辑必须添加行注释 。

b 函数注释用 JSDoc 标注参数/返回值 。

15. 文档管理

a 公共 API 必须提供参考文档 。

b 设计文档包含目标、决策依据、替代方案 。

六、设计原则

16. 组件规范

a Vue/React 组件名用 ’PascalCase’（如 ’UserCard.vue’）。

b 组件样式加作用域（如 Vue 的 ’scoped’）。

17. 状态管理

b 全局状态集中管理（如 Vuex/Pinia）。

七、其他规范

18. 调试与日志

a 禁止提交 ’console.log()’ 。

c 开发环境允许临时调试，上线前移除 。

19. 依赖管理

a 第三方库版本需锁定（如 ’packagelock.json’）。

20. 工具配置

b 使用 ESLint + Prettier 自动格式化 。

### **规范级别说明**

| **级别** | **含义** | **示例** |
| --- | --- | --- |
| **a** | 必须遵守，否则 CI 失败 | 变量命名、缩进、const 声明 |
| **b** | 强烈推荐，需 Code Review | 解构赋值、箭头函数 |
| **c** | 允许但限制场景 | 单字符循环计数器 |