

# 方块笔记 (BlockNote) - 前端产品设计文档

## 1. 产品愿景

“原子化的信息方块，所见即所得的视觉看板。”

BlockNote 是一个打破传统文档流的笔记应用。它将信息单元化为“方块”，强调视觉直观性、内容自适应和极致的个性化定制。用户不仅是在记录，更是在构建属于自己的思维展示空间。

## 2. 核心功能规格

### 2.1 笔记方块 (The Block)

核心交互单元，遵循“去滚动化”原则。

- **自适应容器：**
  - **高度：**绝对自适应。内容增加，方块变高；内容减少，方块变矮。**禁止内部滚动条。**
  - **宽度：**用户可全局调节（或随模板调节）。
- **内容构成：**
  - **标题：**始终显示，用于标识笔记。
  - **正文：**支持富文本/Markdown（文字、图片、列表）。
- **交互状态：**
  - **折叠/展开：**
    - 折叠态：仅显示标题栏，高度收缩。
    - 展开态：显示完整内容。
  - **禅模式 (Zen Mode)：**
    - 点击特定按钮，当前方块在屏幕中央放大/全屏，背景压暗。
    - 提供沉浸式编辑环境，编辑完成后收缩回原列表位置。
- **媒体支持：**
  - 粘贴图片自动上传/转换并显示，宽度自适应填满方块。

### 2.2 标签与筛选系统 (Tags & Filter Templates)

核心管理逻辑，替代传统文件夹。

- **标签 (Tags)：**
  - 自由输入创建，支持多标签（如 **#灵感** **#2023** **#待办**）。
- **筛选模板 (Templates)：**
  - 本质是**标签组合 + UI配置**的集合。
  - 用户创建一个“视图”（如“工作台”），定义筛选规则（包含 **#工作**）。
  - **关键特性：**每个模板可以绑定一套独立的**UI视觉配置**。切换模板 = 切换内容 + 切换皮肤。

### 2.3 极致 UI 编辑系统 (UI Editor Mode)

核心亮点，支持“所见即所得”的界面定制。

- **编辑模式：**进入该模式后，界面元素变为可配置状态。
- **可配置项：**

### 1. 布局:

- 方块列表起始位置 (X, Y)。
- 方块统一宽度 (Width)。
- 筛选器/标签栏位置。

### 2. 视觉样式:

- 背景色 (App Background)。
- 方块样式: 背景色、字体颜色、圆角 (Border Radius)、毛玻璃/模糊效果 (Blur)、阴影。
- 标签样式: 颜色、胶囊样式。

### 3. 装饰层 (Background Decor):

- 位于最底层的独立图层。
- 支持在空白区域插入图片或文本 (“悬浮物”)。
- 装饰物支持拖拽位置、缩放。

## 2.4 辅助功能

- **导出:**
  - 单块导出: 生成卡片图片。
  - 列表导出: 生成长截图。
- **数据持久化:** 前端优先使用 LocalStorage/IndexedDB 进行本地存储。

---

## 3. 技术架构选型

### 3.1 核心栈

- **框架:** Vue 3 (Composition API + `<script setup>`)
- **语言:** TypeScript
- **构建工具:** Vite
- **样式:** Tailwind CSS (原子类) + CSS Variables (用于动态修改用户配置的颜色/尺寸)

### 3.2 关键库建议

- **状态管理:** Pinia
  - **NoteStore:** 管理笔记数据、标签 CRUD。
  - **UIStore:** 管理当前 UI 配置、模板 UI 映射。
- **动画与交互:**
  - **@vueuse/motion** 或 **Vue Transition:** 处理折叠、列表排序动画。
  - **@vueuse/core:** 提供 resize observer (监测高度变化), storage, mouse 交互等 hooks。
- **拖拽排序:** vuedraggable (Sortablejs) 或 Vue 3 DnD。
- **富文本/输入:** 简单的 **contenteditable** 封装或轻量级编辑器 (如 Tiptap 的极简配置), 优先保证图片粘贴和文本自适应。

---

## 4. 数据模型设计 (TypeScript Interfaces)

```
// 1. 笔记方块
interface NoteBlock {
  id: string;
```

```
    title: string;
    content: string; // HTML string or Markdown
    tags: string[]; // Tag IDs or names
    createdAt: number;
    updatedAt: number;
    isCollapsed: boolean; // UI state
}

// 2. 视觉配置 (主题)
interface UIThemeConfig {
    // 布局
    layout: {
        listStartX: number;
        listStartY: number;
        blockWidth: number;
        filterPosition: { x: number, y: number };
    };
    // 样式
    style: {
        appBackgroundColor: string;
        blockBackgroundColor: string;
        blockFontColor: string;
        blockBorderRadius: number;
        blockBlur: number; // px
        tagColor: string;
    };
    // 装饰物
    decorations: DecorationItem[];
}

interface DecorationItem {
    id: string;
    type: 'image' | 'text';
    content: string; // Image URL or Text content
    x: number;
    y: number;
    scale: number;
    zIndex: number;
}

// 3. 筛选模板
interface FilterTemplate {
    id: string;
    name: string;
    filterRules: {
        includeTags: string[];
        // excludeTags: string[]; // 可扩展
    };
    // 该模板绑定的 UI 主题配置
    // 如果 null, 则使用全局默认配置
    themeConfig?: UIThemeConfig;
}
```

## 5. 开发阶段规划

### 阶段一：骨架搭建 (MVP Core)

- ☐ 初始化 Vue 3 + Vite + Tailwind + Pinia 项目。
- ☐ 实现 **Block** 组件：支持输入标题/内容，**高度自适应**，折叠功能。
- ☐ 实现 **BlockList**：竖向排列，基础的数据增删。

### 阶段二：数据与标签 (Logic)

- ☐ 实现标签系统：添加标签，按标签筛选。
- ☐ 实现 **Template** 切换逻辑。

### 阶段三：UI 编辑器 (Magic)

- ☐ 实现“编辑模式”开关。
- ☐ 实现全局 CSS 变量绑定，让样式动态化。
- ☐ 开发 UI 配置面板：调整颜色、圆角、位置（实现拖拽改变布局）。

### 阶段四：完善与装饰 (Polish)

- ☐ 装饰层开发（背景图片/文本拖拽）。
- ☐ 禅模式 (Zen Mode) 实现。
- ☐ 导出功能与动效优化。