AI 教材轉換器 (AI Courseware Converter) - 開發者手冊

1. 專案概述

本專案是一個基於Web的智慧教材製作工具，旨在幫助教育人員將純文字或Word文件，透過Google Gemini AI的語意分析，自動排版精美、結構清晰的PDF教材。

1.1 核心功能

- **Word/Markdown** 匯入：支持`.docx`解析與Markdown編輯。
- **AI**兩級處理：先進行內容增強（標記重點、建議圖片），再進行格式化（轉為JSON）。
- 即時預覽：所見即所得 (WYSIWYG) 的 A4 分頁預覽。
- **PDF**生成：利用瀏覽器修復打印功能產生高分辨率PDF。

---

2. 技術架構（Tech Stack）

本專案採用現代尖端技術棧，增強客戶端加速與流暢的互動體驗。

類別	技術/函式庫	用途說明
核心框架	React 19	目前 UI 與狀態管理（使用 Hooks）。
語言	TypeScript	強型別語言，確保資料結構（Project、ContentItem）的安全性。
造型	Tailwind CSS	實用優先的CSS框架，用於快速切版。
人工智慧模型	Google Gemini API	模型版本`雙子座2.5閃光燈`，用於自然語言分析與JSON生成。
文件解析	Mammoth.js	將`.docx`轉換為HTML（透過CDN加載，掛載於`窗戶`）。
Markdown	駁回	將 HTML 轉換為 Markdown (透過 CDN 加載，掛載於`窗戶`)。
貯存	本地儲存	瀏覽器端資料持久化，支出資料庫。

3.核心業務邏輯詳解（Core Workflow）

應用程式的核心運行流程分為四個主要階段：

3.1 第一階段：輸入與解析（Input & Parsing）

- 位置：`components/Editor.tsx`->處理文件更改
- 邏輯：
    1. 用戶上傳`.docx`。
    2. **Mammoth.js**將Word轉為Raw HTML。
    3. HTML 擷取中的`<img>`標籤，轉為 Base64 並儲存於項目.圖像，將原位置替換為佔位符文字`[圖片已匯入：ID]`。
    4. 駁回將處理過的HTML轉為Markdown，存入項目.原始內容。
-

3.2 第二階段：AI智慧建議（第一步：AI建議）

- 位置：`services/geminiService.ts`->獲取AI建議
- 目的：增強內容，不改變結構。
- 及時策略：
    - 指示AI保持原始文字完整。
    - 稀缺性原則：限制重點標記數量（10%）。
    - 插入語意標籤：`[建議：重點提示]`、`[建議：定義]`、`[建議：插入圖片]`。
-
- 輸出：帶標籤的Markdown字符串。

3.3 第三階段：AI結構化生成（Step 2: Structured Generation）

- 位置：`services/geminiService.ts`->分析內容
- 目的：將標籤化文字轉換為嚴格的JSON格式供React渲染。
- 及時策略：
    - 強制模式：使用回應模式定義結構化內容項`[]`。
    - 標籤清理：指示AI移除第二層產生的標籤文字（如`[建議：定義]`），將其轉換為對應的 JSON 屬性 (type: 'definition')。
    - 目錄生成：集中力量重點生成目錄物品。
    - 表格/**JSON** 安全：嚴格禁止字符串內換行，防止JSON解析錯誤。
-

3.4階段四：渲染與PDF輸出（Rendering & Output）

- 位置：`components/PreviewPanel.tsx`&`utils/pdfGenerator.ts`
- 渲染邏輯：
    - 穿越結構化內容陣列。
    - 根據項目類型(如章節標題，桌子，關鍵點) 渲染對應的 Tailwind 元件。
    - 目錄處理：使用頁碼(如果已計算) 或↘(預設)顯示頁碼。
-

- **PDF**生成原理：
    - 不使用`jspdf`或`html2canvas`（解析度太低）。
    - 使用隱藏 **iframe/window** 技巧：
        1. 開啟一個新視窗/iframe。
        2. 寫入預覽區的HTML。
        3. 投入`@媒體印刷`CSS樣式（強制A4尺寸、刪除按鈕、優化表格分頁）。
        4. 呼叫`window.print()`觸發瀏覽器打印對話框。
    -
-

4.關鍵資料結構（Data Structures）

定義於`types.ts`，是理解專案的關鍵。

4.1 項目（專案對象）

code TypeScript

```
None
interface Project {
  id: string;
  name: string;
  rawContent: string;        // 階段一：原始 Markdown
  suggestedContent: string; // 階段二：AI 建議後的 Markdown
  structuredContent: StructuredContentItem[]; // 階段三：最終 JSON
  styles: GlobalStyles;      // 全域樣式設定
  images: Record<string, string>; // 圖片庫 (ID -> Base64)
}
```

4.2 StructuredContentItem（內容單元）

這是受歧視的聯合類型，決定了內容如何被渲染。

類型	說明	主要屬性
`章節標題`	H1標題（通常會換頁）	`內容`
`章節標題`	H2 標題	`內容`
`段落`	一般內文	`內容`

關鍵點	重點提示(藍色框)	內容
警告框	警告事項(紅色框)	內容
定義	名詞定義 (綠色框)	學期, 定義
桌子	表格	標題, 列
圖片建議	圖片佔位符	ID, 前文
目錄	目錄	items: { level, text, pageNumber }[]

5.目錄與頁碼計算機制

由於瀏覽器無法在印刷前得知PDF真實頁碼，我們採用模擬計算方式：

1. 觸發：使用者點選「計算並更新目錄頁碼」。
2. 邏輯(Editor.tsx->處理計算頁碼)：
    - 假設A4高度為**1123**像素(96 DPI下的標準像素高度)。
    - 提取每個標題元素 (h1, h2, h3) 在預訂容器中的偏移頂部。
    - 計算公式：頁碼 = Math.floor(相對高度 / 1123) + 1。
    - 更新結構化內容中的目錄物品。
3.
4. 限制：此為提示值，實際列印可能會導致邊距設定一些微誤差，但對於大多數使用者來說場景已經足夠準確。

6.環境變數與部署

- **API**金鑰：本專案使用process.env.API_KEY注入 Google Gemini API 金鑰。
- 部署：
    - 這是一個純靜態網站(Static Site)。
    - 構建後 (npm run build) 可部署於 GitHub Pages, Vercel, 或 Netlify。
    - 注意：部署時需確保建置環境包含正確的API金鑰變數。
-