AI MARKETING PAGES COURSE — HANDOUT

AI 互動行銷頁實作課 上課講義

不寫程式,做出會動、會玩、能上線的活動網頁。

含全部模組講義、prompt 範本、常見坑與驗收清單。

Demo 對照:repo 內 demos/index.html(雙擊即開)。

2026-06 · SANWU COFFEE 為課程虛構品牌

目錄

AI 互動行銷頁實作課 — 課程總覽

> 不寫程式,做出會動、會玩、能上線的活動網頁。

這門課給誰上

你是行銷人:會企劃活動、會寫文案、天天用 ChatGPT 或 Claude 聊天,但每次要做活動網頁都得排隊等工程師或外包。這門課教你把「會描述需求」這個你已經有的能力,變成「能自己做出活動頁」的能力。

不需要會寫程式。需要的只有:一個瀏覽器、一個 AI 聊天工具(ChatGPT / Claude / Gemini 任一)、一個免費 GitHub 帳號。

完課後你能做到

1. 用 AI 對話生出一整頁有質感的活動網頁(不是套版型,是按你的活動客製)。
2. 用結構化 prompt 生出風格一致、可重現的活動視覺(主視覺、貼文圖、OG 分享圖)。
3. 幫頁面加上粒子、動態背景等互動特效,而且知道怎麼跟 AI 描述「我要哪一種」。
4. 做出拉霸抽獎、刮刮樂、心理測驗這類提升停留與分享的互動遊戲。
5. 自己把頁面部署上線(GitHub Pages),並看懂什麼情境才需要 Vercel / Render / Cloudflare。
6. (進階)讓頁面即時跟 AI 對話、讓頁面內容每天自動更新。

模組地圖

模組	標題	產出
0	行前準備:工具與心態	環境就緒
1	第一個活動頁:用 AI 對話生出整頁	你的活動 landing page
2	AI 視覺資產:從許願到可控	一套風格一致的活動視覺
3	互動特效:讓頁面活起來	頁面加上動態 hero
4	行銷遊戲:抽獎、刮刮樂、測驗	一個可玩的互動遊戲頁
5	上線:部署決策樹 + GitHub Pages	一個真實網址
6	進階:頁面即時呼叫 AI	AI 互動功能
7	進階:讓活動頁自己更新	自動化排程
8	Prompt 兵法:context、圖片與驗收清單	可重複使用的 prompt 方法
9	收 email 名單:報名頁 + 後端 + 後台	自己擁有的名單 + CSV 匯出
10	被看見:SEO / 社群分享卡 OG / AEO・GEO	搜尋、社群、AI 三管道都找得到你
11	成效追蹤:GA / Meta Pixel	知道幾人來、幾人報名、廣告有沒有效

主線是 1 到 5;**只有 6、7 是進階選修**(標準是「照著抄能動」);8 是貫穿全課的 prompt 方法,建議上完模組 1 後先翻一遍、完課後再精讀;**9、10、11(收名單 / 被找到 / 追蹤)不是進階,是完課後該補齊的行銷核心**——尤其模組 9,名單是行銷最保值的資產。

課程的三個原則

1. **零 build**:所有東西都是單一 HTML 檔,雙擊就能開,不安裝任何開發工具。
2. **免費 tier**:整條生產線(生成、特效、部署)不花一毛錢。
3. **prompt 是核心技能**:每個模組都附「給 AI 的 prompt 範本」,你帶走的不是程式碼,是跟 AI 要東西的方法。

Demo 對照

課程附三個完整 demo,都在 `demos/` 資料夾,雙擊 `demos/index.html` 即可瀏覽:

- `demos/01-landing/` — 模組 1 + 3 的成品:活動 landing page 含粒子 hero 與倒數計時
- `demos/02-lucky-draw/` — 模組 4 成品之一:拉霸抽獎 + 優惠碼發放
- `demos/03-ai-quiz/` — 模組 4 成品之二:心理測驗 + 自動產生分享圖卡

所有 demo 用虛構品牌「山霧咖啡 SANWU COFFEE」,你可以整包拿去改成自己的活動。

模組 0 — 行前準備:工具與心態

要準備的三樣東西(全免費)

1. **一個 AI 聊天工具**:ChatGPT、Claude、Gemini 任選你習慣的。免費版可以完課,但生成長網頁時付費版比較不會中途截斷。
2. **GitHub 帳號**:到 https://github.com 註冊。它在這門課扮演兩個角色:免費網頁主機(GitHub Pages)+你所有活動頁的保險箱(每次修改都有紀錄,改壞了找得回來)。
3. **一個純文字編輯器**:Windows 內建記事本就可以;想舒服一點裝免費的 VS Code(裝好就好,不用學)。

不需要:安裝程式語言、命令列工具、付費軟體。

檔案習慣(先養成,省掉一半的災難)

- 每個活動開一個資料夾,例如 `coffee-opening/`,網頁存成裡面的 `index.html`。
- 存檔編碼一律 UTF-8(記事本另存新檔時下方可選),否則中文會變亂碼。
- 檔名一律英文小寫加連字號:`hero-1920.jpg`,不要用中文與空格(上線後會變成網址的一部分)。
- AI 每給一個重要版本,先另存一份(`index-v2.html`)再繼續改,隨時有路可退。

心態:你是總監,AI 是工讀生

這門課最重要的一句話:**你不是在學寫程式,你是在學驗收。**

- 工讀生(AI)動作快、不喊累,但會自作主張、會記錯你說過的話、會謊稱做完了。
- 總監(你)的工作:brief 給得清楚、一次只下一兩個修改指令、每次交付逐項檢查。
- 出問題不用自己修:把錯誤訊息或「哪裡不對」的描述原封不動丟回去,讓工讀生自己擦屁股。

每個模組的「常見坑」就是驗收清單,照著查,你會比大多數初級工程師更會驗收網頁。

30 秒測試你準備好了沒

打開 AI,貼這句:

```
給我一個完整的單檔 HTML:深綠色背景置中一行白色大字「山霧咖啡,六月見」,
字要有淡入動畫。直接給完整程式碼。
```

把回覆存成 `test.html` 雙擊開啟。看到字淡入出現,你就已經會了這門課一半的操作。

模組 1 — 第一個活動頁:用 AI 對話生出整頁

學完能做什麼

給 AI 一段「活動 brief」,拿回一個完整、可直接開啟的活動網頁檔案(單一 `index.html`),並且能用對話迭代修改,直到滿意。

核心觀念:網頁只是一個文字檔

行銷人對網頁的恐懼多半來自「以為那是工程」。事實是:一個活動頁就是一個文字檔,瀏覽器讀它、畫出來。AI 很會寫這種文字檔——你的工作不是寫,是「驗收」與「下修改指令」,這正是你天天在做的事(只是對象從設計師變成 AI)。

步驟

1. 把活動企劃壓成一段 brief

AI 生出來的頁好不好,九成取決於 brief。最少要有:

- 活動名稱、檔期、一句話賣點
- 目標客群與他們在乎的事
- 需要的區塊(hero、活動辦法、時間地點、CTA、注意事項)
- 風格參照(兩三個形容詞 + 色系)
- CTA 要使用者做什麼(填表、加 LINE、到店)

2. 用 prompt 範本開工

```
你是資深網頁設計師。請為以下活動做一個單檔活動網頁(所有 CSS 和 JS 都內嵌在同一個 HTML 檔):

【活動】山霧咖啡開幕週,6/20-6/26,買一送一
【客群】25-40 歲上班族,重視質感與儀式感
【區塊】滿版 hero(活動名 + 日期 + CTA)、三張活動亮點卡、地圖資訊、注意事項、頁尾
【風格】沉穩、霧感、留白多;主色深墨綠,輔色奶油白
【CTA】「加入 LINE 領券」按鈕(連到 https://example.com )

要求:
1. 繁體中文,手機優先(大部分人從 LINE/FB 點開)
2. 不用任何外部框架,零依賴
3. 加上 og:title、og:description、og:image 三個 meta tag(分享到 LINE/FB 會用到)
4. 文案不要 AI 樣板腔,寫得像真的品牌
```

3. 存檔、開啟、驗收

把 AI 給的完整 HTML 貼進記事本,存成 `index.html`(編碼選 UTF-8),雙擊開啟。驗收清單:

- 手機寬度看一次(瀏覽器視窗縮窄即可)
- 每個按鈕連結點一次
- 文案逐字讀:AI 很會編造「免費停車」這種你沒說過的服務

4. 對話式迭代

修改不要自己動手改檔案,回到對話下指令:「hero 字再大、卡片改成左圖右文、主色換 #1a3c34」。一次只改一兩件事,改完重存重開。

給 AI 的追加 prompt 範本

- 「整頁的留白再多 20%,現在太擠」
- 「把『活動辦法』改成三步驟圖示流程,每步一句話」
- 「加一個置底的浮動 CTA bar,捲動時固定在底部」
- 「OG 分享標題改成:山霧咖啡開幕週,整週買一送一」

CTA 設計三原則(轉換的關鍵)

活動頁的目的不是「好看」,是讓訪客**做一件事**——報名、加 LINE、購買。那顆按鈕就是 CTA(Call To Action,行動呼籲)。CTA 沒做好,頁面再美也是零轉換。三個原則:

1. **動詞要明確、具體**:不要「了解更多」「點我」這種模糊的;用「立即報名」「加入 LINE 領券」「搶開幕優惠」——一看就知道按下去會發生什麼、能拿到什麼。
2. **放在第一屏(hero)**:訪客不一定會往下捲。主要 CTA 要在打開就看得到的位置;長頁可以再用模組裡教的「置底浮動 CTA bar」讓它一直跟著。
3. **一頁一個主 CTA**:同一頁塞「報名」「加 LINE」「看影片」「下載」四顆同樣大的按鈕,等於沒有重點。選一個最想要的當主按鈕(顏色最跳),其他降級成次要連結。

給 AI 的 prompt:

```
幫我頁面設計 CTA:
- hero 放一顆最醒目的主按鈕「立即報名」,連到 [報名連結]
- 按鈕用最強調的顏色,文字夠大、手機好點
- 捲動超過一屏後,在畫面底部固定一條浮動 CTA bar(同一顆「立即報名」)
- 其他次要行動(加 LINE、看簡章)做成低調的文字連結,不要跟主按鈕搶
```

驗收:打開頁面三秒內,你能不能一眼看到「該按哪裡」?看不到就是 CTA 不夠強。(模組 9 教的 email 報名表單,就是一種把 CTA 接到真實轉換的做法。)

常見坑

1. **AI 給的檔案不完整**:頁面開起來破版,多半是回覆被截斷。指令:「請從頭重新給我完整的單一 HTML 檔,不要省略任何段落」。
2. **編造內容**:AI 會自己加「營業時間 9:00-18:00」。brief 裡沒給的資訊,驗收時逐項檢查。
3. **連結佔位符**:AI 常留 `href="#"`。上線前全頁搜尋 `#` 和 `example.com` 換成真連結。
4. **忘了 OG tags**:從 LINE 分享出去變成一片空白卡片,活動觸及直接砍半。模組 5 會教怎麼驗證;OG、SEO 與「給 AI 看的頁面」的完整做法見模組 10。

動手做

任務:挑你自己一個真實活動,用本章的四層 brief 生一個單檔 landing,存成 `index.html` 雙擊打開。 **預期成果**:打開看到 hero + 一顆主 CTA + 三個區塊;把視窗縮到手機寬度不破版、文字不溢出。 **卡關怎麼辦**:破版多半是回覆被截斷,跟 AI 說「從頭給我完整的單一 HTML 檔、手機優先,不要省略任何段落」。

對照成品

`demos/01-landing/index.html` 就是用這個流程做出來的(特效部分屬模組 3)。先開來玩,再回頭看它的 brief 長怎樣。

模組 2 — AI 視覺資產:從許願到可控

學完能做什麼

用結構化 prompt 生出「風格一致、可重現」的活動視覺:主視覺、社群貼文圖、OG 分享圖。重點不是學某個生圖工具,是學會讓任何生圖工具聽話。

核心觀念:許願 vs 下規格

新手 prompt:「幫我畫一張咖啡店開幕的圖」。每按一次生成,風格、構圖、光線全部重新抽籤——這對行銷是災難,因為活動需要的是一**套**視覺,不是一**張**圖。

解法是把 prompt 從許願改成下規格,固定六個欄位:

欄位	範例
主體	一杯拿鐵在木質吧台上,蒸氣上升
風格	日系極簡攝影,底片感
光線	清晨柔和側光,窗光
構圖	主體置右,左側留白放標題
配色	深墨綠 + 奶油白 + 木質棕
負面	不要文字、不要人臉、不要過度銳利

同樣的規格 → 高度一致的結果。要產第二張圖(貼文版)只換「主體」和「構圖」兩欄,其他欄位不動,整套視覺就自然成系列。

步驟

1. **先定規格再生圖**:把品牌色、風格形容詞寫成上面六欄,存成你的「視覺規格卡」。
2. **一次生一個用途**:主視覺(橫式 16:9,構圖預留標題位)、IG 貼文(方形 1:1)、OG 分享圖(1200x630,主體靠中,因為會被各平台裁切)。OG 分享圖怎麼掛上頁面、怎麼驗證,見模組 10。
3. **微調用追加,不要重來**:「同一張,光線改黃昏」比重寫整個 prompt 穩定。
4. **檔名與尺寸紀律**:`hero-1920.jpg`、`og-1200x630.jpg`、`post-1080.jpg`,跟模組 1 的頁面直接對接。

給 AI 的 prompt 範本

```
請用以下規格生成圖片：
主體：一杯拿鐵在深色木質吧台，熱氣緩緩上升，旁邊散落兩三顆咖啡豆
風格：日系極簡商業攝影，底片顆粒感，安靜高級
光線：清晨窗光，柔和，從左側來
構圖：16:9 橫式，主體置於右三分之一，左側大量留白（之後要疊活動標題）
配色：深墨綠、奶油白、木質棕，整體低飽和
避免：畫面中出現任何文字、人臉、logo；避免過度 HDR 銳利感
```

文字要「畫進圖裡」還是「用網頁疊上去」？

以前的建議是「絕對別叫 AI 在圖上寫中文，一定錯」。**這點現在要更新**：新一代的強繪圖模型（像 OpenAI 的 GPT image、Google Gemini 的 Nano Banana Pro 這類）寫中文已經好很多，短標題常常能寫對。所以「AI 不會寫中文」不再是鐵律。

但**推薦的做法還是「圖只負責氛圍、文字用網頁疊上去」**，理由換成兩個更實際的：

1. **之後改文字方便**：標題、日期、優惠常常要改。文字在網頁層，改一行就好；畫進圖裡就得重新生整張圖。
2. **文字能被讀到**：疊在網頁上的字，搜尋引擎與 AI 讀得到（呼應模組 10）；燒進圖裡的字，它們讀不到。

做法：生圖時構圖**留白**（規格卡的「構圖」欄寫「左側留白放標題」），回到模組 1 的對話：「hero 背景換成這張圖，活動標題疊在左側留白處，白字加細陰影」。

什麼時候直接讓 AI 把字畫進圖？**一次性、之後不會改**的視覺（例如貼文上的一句 slogan、店內海報），用新模型畫進去省事也無妨——生完檢查字有沒有錯即可。

常見坑

1. **每張圖重新發明風格**：沒有規格卡，十張圖十種風格。先卡死風格欄位。
2. **OG 圖留白不足**：LINE、FB 預覽會裁切，重要元素放中央 80% 安全區。
3. **以為「AI 一定不會寫中文」**：新模型已經好很多，別再當鐵律。但會常改的文字（標題/日期/優惠）仍建議用網頁疊——好改、又能被搜尋與 AI 讀到。
4. **圖太大拖慢頁面**：輸出後壓到 500KB 以下（任何線上壓圖工具皆可），行動網路下 3 秒打不開，使用者就走了。

動手做

任務：用六欄規格卡生一張活動主視覺（輸出後壓到 500KB 內），再用同一張規格卡只換「主體 / 構圖」兩欄，生一張配套貼文圖。 **預期成果**：兩張風格一致（色系、質感、光線像同一系列），主視覺左側留了放標題的空間。 **卡關怎麼辦**：風格跑掉就把規格卡的六欄固定住、只動要換的那一欄再生，不要整段 prompt 重寫。

對照成品

`demos/01-landing/` 的 hero 視覺即按本模組規格卡流程產出(規格卡內容見該資料夾 `NOTES.md`)。

模組 3 — 互動特效:讓頁面活起來

學完能做什麼

幫活動頁加上「一眼記住」的動態效果:粒子背景、動態 hero、捲動進場動畫,而且知道怎麼跟 AI 點菜、怎麼驗收特效真的在動。

核心觀念:特效是菜單題,不是技術題

你不需要會寫特效,你需要的是**叫得出名字**。「幫我加個酷酷的效果」會拿到災難;「幫我加 tsParticles 的星座連線粒子,白點細線,放在 hero 背景」會拿到精品。

點菜前先逛菜單:**web-effects-collector**(https://yazelin.github.io/web-effects-collector/)收錄了 90 個開源特效、分 12 類,每個都有現場 demo 和整合說明。挑特效的流程:逛 → 記下名字 → 貼給 AI。

行銷頁最實用的三類

類型	適用	點菜關鍵字
粒子背景	hero 區想要科技感 / 夢幻感	tsParticles、星座連線、漂浮光點
動態材質背景	質感品牌、不想太花	Vanta FOG(霧)、WAVES(波浪)
捲動進場	內容多的長頁	IntersectionObserver 淡入、上移進場

原則:**一頁最多一個搶眼特效**(通常放 hero),其他用安靜的捲動進場。特效是調味料,不是主菜。

給 AI 的 prompt 範本

```
在我這個活動頁的 hero 區塊加上 tsParticles 粒子背景:
1. 用 CDN 引入 tsParticles v3 的 @tsparticles/all bundle,並記得先呼叫 loadAll() 再 load 設定
2. fullScreen 要設為 false,粒子只出現在 hero 區塊內,不要蓋住整頁
3. 風格:細小白色光點 + 淡淡的連線,緩慢漂浮,密度低,優雅不要吵
4. 粒子層要在文字後面(z-index),不能擋到標題和按鈕
5. 手機上粒子數量減半,避免發熱耗電
```

(第 1、2 點照抄,這是 tsParticles 兩個最常見的雷,先在 prompt 裡堵住。)

驗收:特效「真的在動」的檢查法

特效壞掉時常是「整頁正常,只是那層 canvas 一片空白」,肉眼又不一定立刻發現。驗收三步:

1. 開頁面,hero 上看得到粒子在動。
2. 按 F12 開發者工具看 Console,不能有紅色錯誤。
3. 縮成手機寬度再看一次:粒子還在、不擋文字、捲動順暢。

跟 AI 回報問題時,把 Console 的紅字原文整段貼給它,比「沒效果耶」有用一百倍。

常見坑

1. **粒子蓋住整頁、按鈕點不到**:`fullScreen` 沒關,或 z-index 在文字之上。prompt 範本第 2、4 點就是在防這個。
2. **CDN 載入順序**:特效設定跑在函式庫載入前,Console 會有 `xxx is not defined`。整段貼給 AI 即可。
3. **特效太多**:hero 粒子 + 波浪背景 + 滿天 confetti = 夜市。一頁一個就好。
4. **效能**:老手機開頁面風扇狂轉,粒子數量砍半、關掉互動 hover 模式。

動手做

任務:給你 hero 區塊加一個 tsParticles 粒子背景,密度調低、`fullScreen` 設 `false`。 **預期成果**:粒子緩慢在動、不擋到標題與按鈕,F12 → Console 沒有任何紅字。 **卡關怎麼辦**:粒子蓋住整頁或按鈕點不到,就把 `fullScreen` 關掉、把粒子層的 z-index 放到文字之後。

對照成品

`demos/01-landing/index.html` 的 hero 即為「星座連線粒子 + 文字疊層」的完整實作,粒子設定有註解,可直接改密度與顏色。

模組 4 — 行銷遊戲:抽獎、刮刮樂、測驗

學完能做什麼

做出兩種最有效的活動互動:**抽獎類**(拉霸 / 刮刮樂 / 轉盤,發優惠碼用)和**測驗類**(心理測驗 + 分享圖卡,擴散用),並懂得設定中獎機率與防重抽。

核心觀念:遊戲機制 = 行銷機制

遊戲	行銷目的	關鍵設計
拉霸 / 轉盤 / 刮刮樂	到店核銷、領券轉換	機率表、優惠碼、防重抽
心理測驗	社群擴散、品牌好感	結果要「值得曬」、一鍵存分享圖

兩種都是純前端就能做(零後端、零成本),代價是「防重抽只防君子」——對活動夠用,對高價贈品不夠(見常見坑 4)。

A. 拉霸抽獎

給 AI 的 prompt 範本

```
做一個單檔 HTML 拉霸抽獎頁:
1. 三個轉輪,符號用 emoji(咖啡、蛋糕、星星、葉子),按「抽一次」後依序停下
2. 中獎機率由一張表控制,放在程式碼最上方讓我能自己改:
   - 三個咖啡(大獎,買一送一):5%
   - 任兩個相同(小獎,折 20 元):25%
   - 其他:銘謝惠顧,顯示安慰文案
3. 中獎時跳出優惠碼(例如 SANWU-XXXX,XXXX 隨機),旁邊附「截圖到店出示」說明
4. 用 localStorage 記錄已抽過,同一台手機每天只能抽一次
5. 中大獎時撒一波 confetti 彩帶
6. 手機優先,按鈕要大
```

驗收重點

- 把機率表改成 100% 大獎,抽十次確認每次都中(測機率表真的有作用),再改回去。
- 抽完重新整理頁面,確認「今天已抽過」有擋住。
- 優惠碼每次中獎都不同。

B. 心理測驗 + 分享圖卡

先用 AI 生題庫

測驗的靈魂是題目與結果文案,這是行銷人的主場。prompt:

```
幫我設計「你是哪一型咖啡人」心理測驗:
- 5 題,每題 3 個選項,情境要生活化、有畫面感
- 4 種結果型(例如:儀式感型 / 效率型 / 社交型 / 探險型),每型給:
  名稱、兩句個性描述(要準到讓人想截圖)、推薦的一款咖啡、一句分享語
- 每個選項對應到某一型的加權,輸出成 JSON 格式
```

再讓 AI 組頁面

```
用這份 JSON 做單檔測驗頁:一題一畫面、進度條、答完依加權算出結果型。
結果頁要用 canvas 畫一張 1080x1350 的結果圖卡(型別名稱 + 描述 + 品牌 logo 文字),
提供「儲存圖片」按鈕讓使用者下載去限動分享。配色:深墨綠 + 奶油白。
```

驗收重點

- 每條路徑點一輪,四種結果都要能到達。
- 下載的圖卡打開看:字沒爆框、沒亂碼。
- 用手機模擬寬度玩一次(大多數人用手機玩)。

常見坑

1. **機率寫死在程式深處**:之後想調中獎率找不到地方。prompt 裡明確要求「機率表放最上方」。
2. **結果不值得曬**:測驗擴散全靠結果文案的「被說中感」。文案弱,技術再好也沒人分享。
3. **canvas 圖卡中文字型**:手機下載後字變預設體,要求 AI 用系統字型堆疊並在 canvas 繪製前等字型載入。
4. **把純前端抽獎當成抽獎系統**:懂技術的人可以改 localStorage 重抽、甚至直接看程式碼裡的機率。發「到店核銷」的券沒問題(核銷時人工把關),發「線上直接折抵」的高價券就需要後端(模組 5 的決策樹會講何時升級)。
5. **在 LINE / FB 內建瀏覽器存狀態**:活動頁幾乎都從 LINE / FB 點開,這些內建瀏覽器對 sessionStorage 不可靠,記狀態一律用 localStorage。

動手做

任務:把拉霸 demo 最上方的機率表(script 開頭的 `PRIZES`)改成你要的中獎率,連抽十次驗證;或請 AI 生一份你自己品牌的測驗題庫 JSON。 **預期成果**:十次抽下來的中獎率,大致符合你設定的數字。 **卡關怎麼辦**:改了沒效,確認你動到的是 script 最上方那張機率表,不是別處的判斷邏輯。

對照成品

- `demos/02-lucky-draw/` — 拉霸 + 機率表 + 優惠碼 + 防重抽 + confetti 完整實作

- `demos/03-ai-quiz/` — 測驗 + 加權計分 + canvas 分享圖卡完整實作

模組 5 — 上線:部署決策樹 + GitHub Pages 實作

學完能做什麼

把做好的活動頁變成一個真實網址,而且懂得依需求選平台,不會被帳單或休眠嚇到。

部署決策樹

從上往下走,停在第一個符合的:

```
你的頁面需要伺服器隨時運算嗎?
├─ 不用,就是靜態頁(本課 90% 情境:landing、抽獎、測驗)
│    → GitHub Pages(免費、不會產生帳單、repo 即備份)
├─ 需要收表單 / 一點點後端函式(例如名單寫進試算表)
│    → Vercel(git push 即部署,免費函式額度對活動綽綽有餘)
├─ 需要幫「頁面即時呼叫 AI」藏 API key(模組 6)
│    → Cloudflare Workers(免費額度每天十萬次,專門做輕量代理)
└─ 需要常駐後端:資料庫、即時連線、會員系統
     → Render(可跑完整伺服器;注意免費版 15 分鐘沒人用會休眠,
       第一個訪客要等它醒來,正式活動期建議升級或預熱)
```

三個免費 tier 陷阱先講明:**Pages** 只能靜態(沒有後端就是沒有);**Vercel** 免費版有流量上限,爆紅貼文等級的活動要先看額度;**Render** 免費版會休眠。

真實案例對照:本課的 demo 與 web-effects-collector 全在 GitHub Pages;ai-tarot-companion 用 Pages + Workers 組合;mori-canvas 因為要跑完整後端所以在 Render。

GitHub Pages 手把手

1. 建 repo 並上傳

1. 登入 GitHub → New repository → 名稱取 `coffee-opening`(會變成網址的一部分,用英文小寫加連字號)→ Public → Create。
2. 「uploading an existing file」→ 把 `index.html` 和圖片拖進去 → Commit changes。

2. 開啟 Pages

Settings → Pages → Source 選 `Deploy from a branch` → Branch 選 `main`、資料夾 `/ (root)` → Save。等一兩分鐘,頁面上方會出現網址:

`https://你的帳號.github.io/coffee-opening/`

3. 上線驗收清單

- 無痕視窗開網址(避免快取騙你)。
- 手機實機開一次(不是模擬,真手機)。
- 把網址貼到 LINE 給自己:**分享預覽卡片**有沒有出現你的 OG 標題和圖?沒有的話檢查 `og:image` 是不是寫成相對路徑——OG 圖必須是完整網址(`https://...` 開頭)。(OG / SEO / 給 AI 看的頁面,完整一章在模組 10。)
- 圖片全部有出現(常見錯誤:檔名大小寫不一致,`Hero.JPG` 與 `hero.jpg` 在 GitHub 上是兩個不同檔案)。

4. 更新頁面

改版就是重新上傳同名檔案 commit 一次,一兩分鐘後生效。每次 commit 都有紀錄,改壞了隨時可以回去看舊版內容。

5. 自訂網域(選配)

活動要用 `event.你的品牌.com`:Settings → Pages → Custom domain 填入網域,再到你的 DNS 商把該子網域 CNAME 指向 `你的帳號.github.io`,等生效後勾 Enforce HTTPS。

另外三個分支的最短路徑

走到決策樹其他分支時,第一次部署長這樣(都是免費起步):

Vercel(收表單 / serverless) 1. 用 GitHub 帳號登入 vercel.com → Add New → Project → 選你的 GitHub repo → Deploy,一分鐘拿到 `xxx.vercel.app` 網址。 2. 之後 git push 就自動重新部署,跟 Pages 一樣的節奏。 3. 何時選它:需要 `/api/xxx` 這種輕後端(例如表單寫進 Google Sheet)。靜態頁放 Vercel 也行,但 Pages 就夠了。

Render(常駐後端) 1. 用 GitHub 帳號登入 render.com → New → Web Service → 選 repo → 它會問啟動指令(這已經是工程師領域,通常是接手別人寫好的後端)。 2. 免費版 15 分鐘沒流量會休眠,第一個訪客要等它醒(約半分鐘)——正式活動期升級付費或接受這件事。 3. 何時選它:資料庫、會員、即時連線。行銷活動頁 99% 用不到;真用到了,這一步建議跟工程師協作。

Cloudflare Workers(AI 代理) 完整流程在模組 6;它不是拿來放網頁的,是放「幫頁面跑腿的小程式」。

進階捷徑:用 AI agent + gh CLI 一句話部署

上面的網頁點按流程是基本功;如果你已經在用 agent 式 AI(Claude Code / Codex CLI,見模組 8),部署可以變成一句話的事。原理:GitHub 官方有個命令列工具 `gh`,**你授權一次,agent 之後就能代替你操作 GitHub**。

一次性準備(自己做,兩分鐘):

1. 安裝 gh:Windows 用 `winget install GitHub.cli`,Mac 用 `brew install gh`(官網 https://cli.github.com 也有安裝檔)。

2. 授權:終端機輸入 `gh auth login`,選 GitHub.com → HTTPS → Login with a web browser,瀏覽器按確認即完成。

之後在 Claude Code / Codex 裡,部署 prompt 長這樣:

```
幫我把這個資料夾發佈成 GitHub Pages:
1. 用 gh repo create coffee-opening --public --source=. --remote=origin --push 建 repo 並推上去
2. 用 gh api 啟用 Pages(main branch 根目錄)
3. 等部署完成後,驗證網址回 200,把最終網址給我
```

agent 會自己跑 `git init`、commit、建 repo、開 Pages、輪詢到網站上線,最後丟網址給你——本課這個課程站本身就是這樣發佈的。注意兩件事:**prompt 裡永遠不需要出現任何密碼或 token**(授權存在 gh 自己的設定裡);agent 執行 `gh` 指令前通常會先問你一次,看清楚再放行,特別是 `--public` 這種會把程式碼公開的旗標。

四個平台都有官方 CLI,同一套邏輯

「授權一次,agent 代你部署」不是 GitHub 限定——決策樹上每個平台都有官方 CLI:

平台	CLI	一次性授權	agent 部署指令
GitHub Pages	`gh`	`gh auth login`	`gh repo create … --push` + 開 Pages
Cloudflare Workers	`wrangler`	`wrangler login`	`wrangler deploy`(見模組 6)
Vercel	`vercel`	`vercel login`	`vercel`(預覽)/ `vercel --prod`(正式)
Render	`render`	`render login`	建好服務後 git push 自動部署;CLI 可觸發部署、看 log

通則三條:授權都是「跑一次 login、瀏覽器按確認」;**key / token 永遠不進對話**;部署這種對外動作,agent 動手前看一眼它要跑的指令再放行。順帶一提,Vercel 甚至有 `vercel agent init` 指令,專門幫 coding agent 在專案裡生一份部署須知——平台們已經在迎接「AI 替你部署」這件事了。

常見坑

1. **網址 404**:repo 裡沒有 `index.html`(放在子資料夾裡了),或 Pages 還在建置中。檔案要在 repo 根目錄。
2. **OG 預覽不更新**:LINE / FB 會快取舊預覽。FB 有 Sharing Debugger 可以強制重抓;LINE 只能等或換網址參數。
3. **公開 repo 的心理關**:Pages 免費版 repo 是公開的——本來就會公開上線的活動頁無妨,但優惠碼邏輯等於公開(呼應模組 4 坑 4:高價券別用純前端)。
4. **改了沒生效**:瀏覽器快取。無痕視窗或強制重新整理(Ctrl+Shift+R)再判斷。

動手做

任務:把你在模組 1 做的頁部署到 GitHub Pages,拿到網址後貼到 LINE 傳給自己。 **預期成果**:用無痕視窗開那個真實網址打得開;在 LINE 裡的分享卡出現你的 OG 標題和圖。 **卡關怎麼辦**:404 多半是 `index.html` 沒放在 repo 根目錄;分享卡沒圖,檢查 `og:image` 是不是完整的 https 網址。 **自檢**:把你的頁貼進行銷頁健檢器 (https://yazelin.github.io/marketing-page-checker/)看相關項目是否變綠

模組 6(進階)— 頁面即時呼叫 AI

學完能做什麼

讓活動頁「現場」跟 AI 互動:AI 抽籤解語、AI 客製祝福文、AI 回答活動 QA。標準是「照著抄能動」,不要求理解每一行。

先講清楚:為什麼不能把 API key 放進網頁

前面所有模組的 AI 都是「製作期」用的(AI 幫你做頁面)。本模組是「使用期」AI(訪客跟 AI 互動),這需要呼叫 AI API,而 API key 等於你的信用卡——網頁的程式碼任何人按 F12 都看得到,**key 放前端 = 把信用卡貼在店門口**。

解法:中間加一個免費的「代理」(Cloudflare Worker)。網頁呼叫代理、代理拿著藏好的 key 呼叫 AI、把答案傳回來。真實案例:ai-tarot-companion(https://yazelin.github.io/ai-tarot-companion/)整站就是這個架構,前端 GitHub Pages、代理 Cloudflare Workers,全部免費 tier。

```
訪客瀏覽器 ──> Cloudflare Worker(key 藏在這)──> AI API
   GitHub Pages          免費 10 萬次/天
```

照著抄:三步驟

1. 建 Worker

註冊 Cloudflare(免費)→ Workers & Pages → Create Worker → 把下面整段貼進去取代預設程式碼:

```
export default {
  async fetch(request, env) {
    const cors = {
      "Access-Control-Allow-Origin": "*",
      "Access-Control-Allow-Methods": "POST, OPTIONS",
      "Access-Control-Allow-Headers": "Content-Type",
    };
    if (request.method === "OPTIONS") return new Response(null, { headers: cors });
    if (request.method !== "POST")
      return new Response("POST only", { status: 405, headers: cors });

    const { message } = await request.json();
    if (!message || message.length > 500)
      return new Response(JSON.stringify({ error: "訊息空白或太長" }),
        { status: 400, headers: { ...cors, "Content-Type": "application/json" } });

    const r = await fetch("https://api.groq.com/openai/v1/chat/completions", {
      method: "POST",
      headers: {
        "Authorization": `Bearer ${env.GROQ_API_KEY}`,
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        model: "llama-3.3-70b-versatile",
        max_tokens: 300,
        messages: [
          { role: "system", content: "你是山霧咖啡的活動小幫手,用繁體中文、兩三句話內溫暖回答。只回答咖啡與本次開幕活動相關問題,其他一律婉拒。" },
          { role: "user", content: message },
        ],
      }),
    });
    const data = await r.json();
    const reply = data.choices?.[0]?.message?.content ?? "我恍神了,再問一次好嗎?";
    return new Response(JSON.stringify({ reply }),
      { headers: { ...cors, "Content-Type": "application/json" } });
  },
};
```

2. 藏 key

申請一把免費的 Groq API key(https://console.groq.com ,免費額度對活動互動很夠)。回到 Worker → Settings → Variables and Secrets → 新增 Secret,名稱 `GROQ_API_KEY`,值貼上你的 key。**key 只存在這裡,永遠不出現在網頁程式碼。**

3. 網頁端接上

把 Worker 網址(形如 `https://xxx.你的帳號.workers.dev`)交給 AI:

```
在我的活動頁加一個「問 AI 小幫手」對話框:
輸入框 + 送出鈕,送出時 POST 到 https://xxx.workers.dev ,
body 是 {"message": 使用者輸入},回應的 JSON 裡 reply 欄位顯示成對話泡泡。
等待時顯示打字中動畫;錯誤時顯示「小幫手忙線中」。風格沿用本頁配色。
```

驗收

- 自己問三題活動相關問題,回答合理。
- 問一題無關問題(例如「幫我寫作業」),確認 system prompt 有婉拒。
- 按 F12 → Network 看請求:確認瀏覽器只連 workers.dev,**看不到任何 API key**。

常見坑

1. **CORS 錯誤**(Console 紅字含 "CORS"):上面範本已含 CORS 處理,通常是你改壞了 headers 區塊,整段還原即可。
2. **沒設長度限制與題目範圍**:會被當免費 ChatGPT 玩,額度被吸乾。範本裡 500 字限制與 system prompt 範圍限制就是在防這個。
3. **把 key 貼進前端 prompt**:給 AI 組頁面時,prompt 裡只能出現 Worker 網址,不能出現 Groq key。
4. **想要更高品質的回答**:把 Worker 裡的 Groq 換成其他供應商(Claude、OpenAI)只需改 fetch 那段的網址、模型名與 key,架構完全相同。

進階捷徑:wrangler CLI(配 AI agent 用)

上面三步驟是網頁點按路線,人人能走。如果你已經在用 Claude Code / Codex(模組 8 的 agent 式),Cloudflare 也有官方 CLI 叫 **wrangler**,跟模組 5 的 gh 同一個邏輯:**授權一次,agent 之後代你部署**。

1. 一次性準備:`npm install -g wrangler` → `wrangler login`(瀏覽器按確認)。
2. 之後給 agent 的 prompt:

```
幫我把這份 worker.js 部署成 Cloudflare Worker:
1. 建 wrangler.toml(名稱 my-event-ai,main 指向 worker.js)
2. wrangler deploy,把網址給我
3. 我會自己跑 wrangler secret put GROQ_API_KEY 貼 key
   (key 不進對話,跟模組 8 的紀律一致)
4. 部署完用 curl 測一題活動問題、一題無關問題,確認範圍限制有效
```

本課的示範 Worker 就是這樣部署的。秘訣同模組 8:**驗收清單寫進指令**(第 4 點),agent 測完才交。

動手做

任務:照本章三步驟建一個 Cloudflare Worker AI 代理,在你的頁面加一個「問 AI」對話框。 **預期成果**:問活動相關問題會得到合理回答;F12 → Network 翻過每個請求,看不到任何 API key。 **卡關怎麼辦**:Console 出現含 CORS 的紅字,把 Worker 的 headers 區塊整段還原成範本。

對照成品

`demos/06-ai-helper/` — 完整的「問山霧」聊天頁,**開箱即聊**:A 模式已預填課程示範 Worker(真的在 Cloudflare 上跑,key 在它的 Secret 裡,含每分鐘 8 次限流),也可換成你自己的;B 模式貼自己的 Groq key 直連體驗(key 只存你的瀏覽器)。示範 Worker 的部署版原始碼(多了限流)在 dev repo 的 `worker-deploy/`。

模組 7(進階)— 讓活動頁自己更新

學完能做什麼

讓已上線的活動頁定時自己變:每天換一句活動標語、倒數天數自動更新、甚至每天自動換一張 AI 生成的圖。標準同模組 6:照著抄能動。

核心觀念:GitHub 不只存檔,還能定時幫你做事

GitHub 有個免費功能叫 Actions:你放一個「排程指令檔」進 repo,GitHub 就會按表操課(每小時、每天、每週)自動執行,執行結果 commit 回 repo → Pages 自動更新 → 訪客看到新內容。全程不需要你的電腦開機。

真實案例:catime(https://github.com/yazelin/catime)每小時自動生成一張新的 AI 貓圖並發布,整個系統就是一個排程檔 + 一支腳本,跑在免費額度內。

照著抄:每天自動換標語

1. 把頁面裡要變的部分抽成資料檔

讓 AI 改造你的頁面:「把 hero 的標語改成從 `daily.json` 讀取 `slogan` 欄位顯示」。repo 裡新增 `daily.json`:

```
{ "slogan": "開幕週第一天,霧還沒散,咖啡先香了。" }
```

2. 放排程檔

在 repo 建立檔案 `.github/workflows/daily.yml`(路徑要一字不差):

```yaml
name: Daily Update
on:
  schedule:
    - cron: "0 0 * * *"   # 每天 UTC 00:00 = 台北 08:00
  workflow_dispatch:        # 留一顆手動按鈕方便測試
permissions:
  contents: write
jobs:
  update:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Pick today slogan
        run: |
          python3 - <<'EOF'
          import json, datetime
          slogans = [
            "開幕週第一天,霧還沒散,咖啡先香了。",
            "今天的山霧,配今天的拿鐵。",
            "第三天了,熟客都開始有自己的位子。",
            "週四,適合一個人的吧台時光。",
            "週五晚上,咖啡因換成故事。",
            "週末第一杯,留給早起的你。",
            "最後一天,買一送一,送給還沒來的那個人。",
          ]
          i = datetime.date.today().toordinal() % len(slogans)
          json.dump({"slogan": slogans[i]}, open("daily.json", "w"), ensure_ascii=False)
          EOF
      - name: Commit
        run: |
          git config user.name "github-actions[bot]"
          git config user.email "github-actions[bot]@users.noreply.github.com"
          git add daily.json
          git diff --cached --quiet || git commit -m "daily: update slogan"
          git push
```

3. 測試

repo → Actions 頁籤 → Daily Update → Run workflow(這就是 `workflow_dispatch` 留的手動按鈕)→ 跑完後看 `daily.json` 內容變了、過一兩分鐘頁面上的標語跟著變。

重要:GitHub 排程不保證準時,也可能整次跳過

這是 GitHub Actions 排程**官方文件就寫明**的限制:`schedule` 觸發在尖峰時段會延遲,高負載時甚至**整次被丟棄不執行**。對「每天換句標語」影響不大,但如果你拿它做「開賣那一刻自動上架」這種要準的事,會出包。三種應對,由簡到難:

招式 1:把內容寫成「冪等」,讓漏跑自動痊癒(最重要)

冪等 = 跑一次跟跑十次結果一樣、晚跑也算對。做法是讓腳本**從日期算出今天該顯示什麼**,而不是「在昨天基礎上 +1」。

看上面範本的這行:

```
i = now.date().toordinal() % len(slogans)   # 從「今天是哪天」直接算出索引
```

這就是冪等:不管這個排程今天跑了一次、跑了三次、還是中午才補跑,算出來的標語都一樣、都正確。對照之下,如果你寫成「讀上次的索引 +1」,漏跑一天就永遠錯一格。**先把內容設計成冪等,一半的排程可靠性問題就消失了。**

招式 2:跑得比需要的更頻繁 +「沒變就不做事」(catime 的做法)

我們的 catime 專案排程是**每小時**跑一次(不是每天),搭配兩個設計: - 每次都重新計算「現在該是什麼內容」(冪等),所以某個整點漏跑,下個整點自動補上,使用者最多看到一小時的延遲,不會永久卡住。 - 範本裡的 `git diff --cached --quiet || git commit` ——**內容沒變就不 commit**,所以跑再多次也不會洗版、不會產生一堆空 commit。

把 daily.yml 的 cron 從 `0 0 * * *`(每天一次)改成 `0 */6 * * *`(每六小時一次),漏跑的風險就從「整天沒更新」降到「最多晚六小時」,而且因為冪等 + 沒變不 commit,不會有任何副作用。這是 GitHub 排程最務實的加固法。

招式 3:乾脆不靠排程——能在前端算的就別放後端

最穩的「排程」是根本不排程。我們 demo 07 的**倒數天數**就是純前端用今天的日期即時算出來的(看 `index.html` 裡的 countdown),不依賴任何排程、永遠準。凡是「能從當下時間直接推算」的內容(倒數、第幾天、營業中/已打烊),都讓瀏覽器自己算,比任何排程都可靠。

更準時的選擇:Cloudflare Cron Triggers

如果你已經在用 Cloudflare Worker(模組 6、9),它內建的 **Cron Triggers** 比 GitHub 排程準時得多,而且免費方案就有。設定:在 `wrangler.toml` 加幾行,Worker 裡用 `scheduled` handler 接:

```
# wrangler.toml
[triggers]
crons = ["0 0 * * *"]   # 一樣是 UTC,台北 08:00
```

```
// worker.js ── 排程一到,Cloudflare 自動呼叫這個 scheduled
export default {
  async scheduled(controller, env, ctx) {
    const slogans = ["霧還沒散,咖啡先香了。", "今天的山霧,配今天的拿鐵。", /* … */];
    const i = Math.floor(Date.now() / 86400000) % slogans.length;
    await env.DB.prepare("UPDATE site SET slogan=? WHERE id=1").bind(slogans[i]).run();
  },
};
```

差別:Cloudflare 在它自己的基礎設施上跑,準時性遠勝 GitHub 排程(GitHub 那套本來就標明「不保證」);代價是內容得存在它能存的地方(D1、KV),不像 GitHub 是直接改 repo 裡的檔案。**決策很簡單:已經在用 Cloudflare、又在意準時 → 用 Cron Triggers;只是每天換句無關緊要的文案、想跟 repo 綁一起 → GitHub Actions + 上面三招加固就夠。** 注意:剛改完 cron 設定,Cloudflare 端最長要 15 分鐘才生效。

延伸玩法(原理完全相同)

- **倒數天數**:腳本算「距離活動結束還剩 N 天」寫進 json(或如招式 3,直接前端算)。
- **每日 AI 圖**(catime 模式):排程腳本呼叫生圖 API(key 放 repo Settings → Secrets,跟模組 6 同一個觀念),存圖 commit。
- **自動抓資料**:每天抓天氣寫進頁面:「山上 16 度,適合熱拿鐵」。

動手做

任務:給你的頁加一個 `daily.json` 加一個 GitHub Actions 排程檔,到 repo 的 Actions 頁籤按 Run workflow 手動跑一次。 **預期成果**:`daily.json` 內容變了,過一兩分鐘頁面也跟著變。 **卡關怎麼辦**:push 失敗、看到 Permission denied,在 workflow 裡加上 `permissions: contents: write`。

對照成品

`demos/07-auto-update/` ── 「今日山霧」頁,標語來自 `daily.json`;本 repo 的 `.github/workflows/daily.yml` 每天台北 08:00 自動改寫它(跟上面範本同一套)。到 repo 的 Actions 頁籤可以看到每天的執行紀錄,也可以按 Run workflow 現場示範。

常見坑

1. **cron 是 UTC 時區**:`0 0 * * *` 是台北早上八點,不是半夜。要台北時間 X 點,寫 X-8(負數就 +24)。
2. **以為排程一定會跑**:GitHub 排程會延遲、會跳過(見上面整節)。別承諾「整點準時」;用冪等 + 高頻率 + 沒變不 commit 加固,或要準就用 Cloudflare Cron。
3. **忘了 `permissions: contents: write`**:Actions 預設不能改 repo,push 會失敗,紅勾勾點進去看 log 就會看到 Permission denied。

4. **活動結束忘了關**:排程會一直跑下去。活動收攤時把 `daily.yml` 刪掉或在 Actions 頁面 Disable workflow。
5. **排程靜悄悄壞掉沒人發現**:Actions 失敗預設會寄信給 repo 擁有者;別把通知關掉,不然某天 key 過期、排程連續失敗你卻不知道。重要的排程值得偶爾去 Actions 頁籤看一眼最近幾次是不是綠的。

模組 8 — Prompt 兵法:讓 Claude / Codex 做出本課等級成品的完整方法

> 這個模組回答一個問題:本課的 demo(活動頁、拉霸、測驗、那張山嵐窗景熱拿鐵)到底是怎麼被 prompt 出來的?把方法拆開給你,讓你對任何 AI(Claude、ChatGPT、Gemini、Codex)都能複製這個品質。

學完能做什麼

寫出讓 AI 一次到位率高的 prompt;知道丟什麼 context 給 AI 結果會差十倍;準備與使用圖片素材;分得清「聊天式 AI」與「agent 式 AI(Claude Code / Codex CLI)」各自怎麼下指令。

一、prompt 的四層結構

好 prompt 不是寫得長,是四層都有交代。缺一層,AI 就自己腦補一層——腦補的那層通常就是你不滿意的地方。

層	在做什麼	範例(本課 demo 01 實際用法)
1 角色	借一個專業視角	「你是資深網頁設計師」
2 任務 + 交付格式	做什麼、交出什麼形狀	「做一個**單檔**活動網頁,CSS/JS 全內嵌」
3 規格	內容、風格、技術三種規格	區塊清單 / 「沉穩、霧感、留白多;深墨綠+奶油白」 / 「零依賴、手機優先、要 OG tags」
4 驗收標準 + 禁止事項	你會怎麼檢查、不要什麼	「文案不要 AI 樣板腔」「不要外部框架」

幾個實戰心法:

- **風格用形容詞 + 色票,不要只說「好看」**。「質感」每個人想的不一樣;「沉穩、霧感、留白多、深墨綠 #1d4438」沒有歧義空間。
- **把你會踩的雷預先寫進 prompt**。模組 3 的範本要求「先 loadAll() 再 load」「fullScreen 設 false」,就是把已知的坑在 prompt 層堵住——你不需要懂原理,只需要照抄這幾行。
- **負面清單跟正面需求一樣重要**。「不要 emoji、不要紫色漸層、不要置中大標題配三欄圖示的罐頭版型」,AI 的預設審美需要你明確推開。
- **要求把「之後會改的東西」放在好改的地方**。demo 02 的「中獎機率表放在程式碼最上方」就是這條——prompt 時多一句話,活動中調整機率不用再求人。

二、context 準備:結果差十倍的關鍵

AI 不是讀心術。同一句指令,有沒有附對 context,輸出是兩個世界。每次開工前準備這四種:

1. 品牌規格卡(一次寫好,所有 prompt 重複用)

```
品牌:山霧咖啡 SANWU COFFEE
色票:墨綠底 #10221d / 松綠 #1d4438 / 霧灰綠 #9db8ae / 奶油白 #f4efe4 / 銅棕 #c98f4e
字體:標題襯線(Noto Serif TC),內文黑體
語氣:沉穩、有畫面感、短句;不用驚嘆號轟炸、不用 emoji
一句話人格:把第一杯的儀式感分給你
```

每次跟 AI 開新對話,先貼這張卡再說需求。本課三個 demo 風格一致,靠的就是同一張卡——不是 AI 記性好,是你每次都餵。

2. 現有檔案

要 AI 改頁面,**把整個 HTML 檔貼回去**(或 agent 模式直接給檔案路徑),不要用嘴巴描述「就是上次那個」。AI 沒有上次,每段對話開始時它只知道你貼了什麼。

3. 錯誤現場

回報問題的品質決定修復的速度:

- 壞:「沒效果耶」
- 好:F12 Console 紅字**原文整段**貼上 + 「我做了什麼 → 我預期什麼 → 實際看到什麼」

4. 參照範例

「我要像 https://yazelin.github.io/web-effects-collector/ 裡『星座連線』那種粒子」比「要有科技感的點點」精準一百倍。看到喜歡的頁面,丟網址或截圖描述給 AI 當座標。

三、圖片的 context:兩個方向

方向 A:讓 AI 生圖(文生圖)

用模組 2 的六欄規格卡(主體 / 風格 / 光線 / 構圖 / 配色 / 負面)。本課 hero 的實際生成 prompt 就是規格卡直譯:

```
山嵐間的咖啡店窗景,一杯熱拿鐵在窗台,蒸氣清楚上升/日系極簡商業攝影,
底片顆粒/清晨窗光從左側來/16:9,杯子置右三分之一,左側留白疊標題/
深墨綠、奶油白、木質棕,低飽和/不要文字、人臉、logo、過度 HDR
```

關鍵在**構圖欄寫了用途**(「左側留白疊標題」)——生圖時就想好這張圖要進版面的哪裡,後面網頁層疊字、疊蒸氣動畫才有位置。

方向 B:拿圖片當輸入(圖生文 / 圖生圖)

- **給 AI 看截圖改版面**:把現在頁面的截圖貼給 AI +「左邊太空、卡片不齊」,比文字描述版面問題有效得多。Claude / ChatGPT / Gemini 都吃圖。
- **給 AI 看參照圖定風格**:「照這張圖的氛圍幫我寫 hero 區的 CSS 配色」。
- **圖片進頁面前的紀律**:壓到 500KB 以下、檔名英文小寫、需要進 canvas 的圖考慮 base64 內嵌(demo 03 的做法,避免本機直接開啟時下載功能失效)。

四、聊天式 vs agent 式:同一套兵法,兩種下法

	聊天式(ChatGPT / Claude / Gemini 網頁版)	agent 式(Claude Code / Codex CLI)
檔案	你貼程式碼進去、複製出來存檔	它直接讀寫你資料夾裡的檔案
適合	單檔頁面、本課主線全部	多檔專案、要它自己驗證的時候
context 給法	每次對話重新貼規格卡 + 檔案	規格卡存成檔案(如 BRAND.md),叫它先讀
驗收	你自己開瀏覽器看	可以叫它「開瀏覽器檢查 console 有沒有紅字、截圖給我看」

agent 式的 prompt 多一個殺手鐧:**把驗收清單寫進指令,讓它自己驗完才交**。本課 demo 的實際做法就是這樣:

```
做完之後你要自己驗證:
1. 起本機 server 開頁面,console 不能有紅字
2. 特效 canvas 要真的有畫東西(數非透明像素,不是看元素存在)
3. 手機寬度 390px 檢查無水平捲動
4. 互動流程(抽獎/答題)完整走一遍
驗證通過才回報,並附截圖。
```

demo 01 手機版的兩個 bug(霧氣層溢出、日期折行)就是這樣被抓出來修掉的——不是 AI 第一次就寫對,是驗收清單逼它(或我)發現。**一次到位是迷思,一輪驗收內收斂才是實力。**

五、迭代紀律(不分聊天式或 agent 式)

1. **一次只改一兩件事**。一口氣下十個修改,壞了不知道是哪個造成的。
2. **重要版本先另存再改**。`index-v2.html`,隨時有路可退。
3. **改壞了別戀戰**。連續兩三輪修不好,把「最後一個好版本」貼回去,換個說法重新下指令,常常比繼續補丁快。

4. **滿意的 prompt 存起來**。本課每個模組的範本就是這樣累積的——你的 prompt 庫是你的資產,比成品更值錢。

六、實戰回放:demo 01 的完整 prompt 鏈

把本課質感最高的成品拆成它實際經歷的指令序列:

1. **brief → 整頁**(模組 1 範本):一段四層結構的 brief,拿到第一版單檔 HTML。
2. **規格卡 → hero 圖**(模組 2):六欄規格卡生成窗景熱拿鐵,壓成 108KB 的 hero.jpg。
3. **點菜 → 特效**(模組 3):「星座連線粒子,密度低、優雅不要吵、手機減半」。
4. **疊層指令**:「hero 改成左標題右照片;照片維持原始比例;在杯口位置(約 76%、60%)疊三縷 CSS 蒸氣絲,錯拍上升」。
5. **驗收回饋**:「手機 390px 有水平捲動、日期在詞中間折行」→ 修霧氣層裁切 + 日期整詞換行。

每一步都是本模組的兵法:四層結構、規格卡 context、點得出名字、驗收清單。沒有一步需要會寫程式。

常見坑

1. **prompt 寫成許願池**:「幫我做一個很棒的活動頁」= 把所有決定權讓給 AI 的預設品味。規格不寫,就等著驗收時逐條吵。
2. **context 只給一次**:聊天式 AI 對話一長就忘。重要規格(品牌卡)在關鍵指令前重貼一次。
3. **把 AI 的自信當正確**:AI 永遠語氣肯定,包括它寫錯的時候。驗收清單存在的理由就是不信任何「做好了」。
4. **prompt 裡貼了不該貼的**:API key、客戶名單、內部價格表不要進對話——尤其用公司外的 AI 服務時。

動手做

任務:拿一個你以前下過、結果不滿意的 prompt,用四層結構(角色 / 任務+格式 / 規格 / 驗收+禁止)重寫一次,把新舊兩版都丟給 AI 比較。 **預期成果**:新版結果更可控、更接近你要的,廢話更少。 **卡關怎麼辦**:還是不準,檢查是不是缺了「規格」層——風格用形容詞 + 色票寫清楚,把已知的雷預先寫進去。

對照成品

整個 repo 就是本模組的對照成品:`demos/` 三個頁面、`demos/01-landing/NOTES.md` 的 brief 與規格卡、各模組講義裡的 prompt 範本,全部按這套兵法產出。

模組 9 — 收 email 名單:行銷人最該先做的一件事

為什麼這章最實用

社群觸及是租來的(平台改演算法,你的貼文就沒人看);**email 名單是你自己的**。一個會持續長大的名單,是行銷人手上最保值的資產。這章教你做一個會員/開幕通知報名頁,而且名單是**真的存起來、看得到、匯得出**——不是填完就消失。

學完能做什麼

做出一個報名頁(收 email + 稱呼),後端把名單存進資料庫,再做一個只有你進得去的後台看名單、一鍵匯出 CSV(就能丟進 email 工具群發)。全部免費 tier。

兩條路:先選你的難度

	A. 不部署任何東西	B. 自己的 Worker + 資料庫
做法	Google 表單 / Formspree / Tally 等現成服務	Cloudflare Worker + D1 資料庫(本章主線)
名單存哪	在那個服務的後台 / Google Sheet	你自己的資料庫,完全掌控
適合	只想快、量不大、不介意掛別人服務	要客製報名頁外觀、名單要自己擁有、之後要接其他系統
成本	免費(有筆數/功能上限)	免費 tier(D1 每天五百萬次讀寫,活動綽綽有餘)

先說 A,因為它可能就夠你用了。

路線 A:零部署,把表單塞進你的頁面

1. **Google 表單**:建一個表單(email + 稱呼)→ 取得連結 → 在你的活動頁放一顆「立即報名」按鈕連過去。名單自動進 Google Sheet,Sheet 就是你的後台,還能直接「下載成 CSV」。最省事。
2. **Formspree / Tally / Typeform**:想讓報名框長在自己頁面裡(不跳轉),這類服務給你一段 form 程式碼貼上即可,送出的資料進它們的後台。

A 路線的代價:報名框外觀受限於服務、名單放在別人家、量大或要進階功能要付費。對多數小活動,這完全夠。

路線 B:自己的報名後端(本章主線)

當你想要「報名頁完全照自己的設計、名單百分百自己擁有、之後能接 AI 或其他系統」,就走這條。架構是模組 6 的延伸——多了一個資料庫:

```
報名頁(GitHub Pages)──POST email──> Worker ──寫入──> D1 資料庫
                                          ↑
你的後台頁 ──GET ?token=管理密碼──────┘ ──回傳名單──> 看名單 / 匯 CSV
```

三個零件: - **報名頁**:純前端,把 email 用 fetch POST 給 Worker。 - **Worker**:驗證 email、擋機器人、限流、去重,寫進 D1;另開一個帶密碼的讀取入口給後台。 - **D1**:Cloudflare 的免費 SQL 資料庫,名單就存這。

給 AI 的 prompt(照模組 8 的四層結構 + agent 驗收清單)

```
做一個「開幕通知報名」的 Cloudflare Worker(worker.js)+ D1:
1. D1 一張表 signups:id、email(唯一)、name、created_at、ip
2. POST 收 {email, name, company}:
   - company 是 honeypot,有值就當機器人,假裝成功但不寫入
   - email 格式驗證、轉小寫;每 IP 每分鐘最多 5 次
   - email 重複(UNIQUE 衝突)當「已報名」處理,不報錯也不洩漏
3. GET /list?token=xxx:token 比對 Secret ADMIN_TOKEN,對了才回名單 JSON
4. 全程加 CORS
做完用 curl 測:正常報名、重複報名、honeypot、爛 email、錯 token,
五種都符合預期才算完成。
```

部署(wrangler,跟模組 6 同一套)

```
wrangler d1 create sanwu-signups            # 建資料庫,記下印出的 database_id
# 把 database_id 填進 wrangler.toml 的 [[d1_databases]]
wrangler d1 execute sanwu-signups --remote --file ./schema.sql   # 建表
wrangler deploy                             # 部署 Worker
wrangler secret put ADMIN_TOKEN             # 設後台密碼(自己想一組長字串)
```

`ADMIN_TOKEN` 是你後台的鑰匙,藏在 Worker 的 Secret 裡,永遠不寫進任何網頁。

後台頁

一個 `admin.html`:輸入密碼 → 帶著密碼打 `GET /list` → 把名單畫成表格 → 「匯出 CSV」按鈕(前端把 JSON 轉 CSV 下載)。密碼記在你瀏覽器的 localStorage,下次免再輸。CSV 拿到後就能匯進任何 email 群發工具。

後台登入怎麼防護:兩種做法,從基本到實名

後台等於你整份名單的入口,這道門怎麼鎖很重要。本章給兩層,按名單的敏感度選。

做法一:token 走 Authorization 標頭(基本功,本章已採用)

最先要做對的一件事:**密碼別放進網址**。`GET /list?token=密碼` 看起來會動,但網址會被瀏覽器歷史、CDN 日誌、甚至 referer 標頭記下來——等於把鑰匙到處留印子。正確做法是放進 HTTP 的 `Authorization` 標頭:

```
// 後台頁:把密碼放標頭,不放網址
fetch(API + "/list", { headers: { Authorization: "Bearer " + token } });
```

```
// Worker:從標頭讀、跟 Secret 裡的 ADMIN_TOKEN 比對
const auth = request.headers.get("Authorization") || "";
const token = auth.startsWith("Bearer ") ? auth.slice(7) : "";
if (token !== env.ADMIN_TOKEN) return json({ error: "unauthorized" }, 401);
```

這叫 **token gate**:一把共享鑰匙,誰有誰能看。對「活動報名名單、自己一個人看」這種情境**夠用**。它的本質限制要心裡有數:只有一把鑰匙、不會自己過期、外洩了要手動換(`wrangler secret put ADMIN_TOKEN` 設一組新的,然後重新發給該知道的人)。

做法二:Cloudflare Access 實名門禁(進階,放真實客戶資料才需要)

當名單裡是真實客戶個資、或要給團隊多人看,一把共享鑰匙就不夠了。Cloudflare Access 把「鑰匙」換成「實名門禁」:每個人各自用自己的 email(收一次性碼)或 Google 帳號登入,你指定誰能進、誰要踢出即時生效、而且每次存取都有紀錄可查。

但有個前提一定要先講清楚:Cloudflare Access 只能保護「**在 Cloudflare 經手的域名底下**」的頁面。本章的 `admin.html` 放在 GitHub Pages(`yazelin.github.io`),那不是你的 Cloudflare 域名,**Access 罩不住它**。所以要用 Access,得先把後台搬到 Cloudflare,兩步:

1. **把後台移上 Cloudflare**:讓 Worker 自己吐出後台 HTML(加一個 `GET /` 回傳 admin 頁),或用 Cloudflare Pages 放後台。
2. **掛一個 DNS 託管在 Cloudflare 的自訂網域**給它(例如 `admin.你的品牌.com`)——這是 Access 能介入的條件。

接著在 Cloudflare 後台設定(Zero Trust 是免費 tier 就有的產品):

1. Cloudflare Dashboard → **Zero Trust → Access → Applications → Add an application → Self-hosted**
2. 填你後台的那個 Cloudflare 網域
3. 加一條 **Policy**:Action 選 `Allow`,Include 選 `Emails`,填你自己的 email(要給團隊就填多個或用 Email domain)
4. 登入方式選 **Email OTP**(寄一次性碼,零設定)或接 Google / Microsoft
5. 存檔。之後任何人打開那個後台網址,都會先被 Cloudflare 攔一道登入,只有你授權的 email 收得到碼、進得來

這道門擋在最前面,後面的 token gate 可留可拿。

做法三:Google 登入白名單(每人實名、免域名,推薦的中間選項)

做法二的 Cloudflare Access 很完整,但要有自訂網域。如果你**沒有域名、又不想用一把共享密碼**,有個甜蜜點:讓後台用 **Google 帳號登入**,只放行你白名單裡的 email。每個人各自實名、隨時加人/踢人、不用域名、免費,直接在 GitHub Pages 就能跑。

原理:前端用 Google 的登入元件(Google Identity Services),使用者按「用 Google 登入」→ 拿到一個 Google 簽名的身分憑證(ID token,一個 JWT)→ 傳給你的 Worker → Worker 驗證簽名 + 比對 email 是否在白名單 → 對了才回名單。

```
管理員 ──Google 登入──> 拿到 Google 簽的 ID token ──> Worker 驗簽 + 查白名單 ──> 回名單
```

關鍵:前端只需要一個 **Client ID(公開、可放網頁原始碼,不是密碼)**;這個流程**不需要 client secret**。

先搞懂「專案 = 品牌」(不然會搞混)

一個 Google 帳號底下可以開**很多個專案**,彼此獨立;**每個專案只有一個「OAuth 同意畫面」,那就是一個品牌**——你在同意畫面填的「App 名稱」,就是使用者按「用 Google 登入」時,彈窗上顯示的「**登入以繼續使用 ___**」那個名字。

所以:

- 想要幾個品牌,就開幾個專案,各設各的同意畫面。
- **如果你「選到」一個以前建過的舊專案,登入彈窗就會顯示那個舊品牌名**(功能照常,只是顯示舊名)。
- 想要這個活動有乾淨、專屬的品牌名 → **為它新開一個專案**,別跟舊品牌混在一起。

步驟一:建一個 Google OAuth Client ID(這就是「在哪建」的答案)

1. 開 https://console.cloud.google.com/apis/credentials ,左上角點專案選單 →「新增專案」,**為這個品牌開一個新專案**(別沿用舊品牌的專案,否則登入彈窗會顯示舊品牌名)
2. 第一次會要你先設「**OAuth 同意畫面**」:User Type 選**外部**、「**App 名稱**」**填你要的品牌名**(這就是登入彈窗顯示的名字)、填你的 email、其餘一路下一步(不用送審)
3. 回「憑證」→「**+ 建立憑證**」→「**OAuth 用戶端 ID**」
4. 應用程式類型選「**網頁應用程式**」
5. 「**已授權的 JavaScript 來源**」加上你後台所在的網域(**只填協定+網域,不含路徑**),例如 `https://你的帳號.github.io`;要本機測再加 `http://localhost:8000`
6. 按建立 → 跳出視窗顯示 **用戶端 ID** `xxxx.apps.googleusercontent.com` → 複製起來。**這串是公開的,放進前端沒關係**。

步驟二:前端放 Google 登入

```
給 AI:在我的後台頁用 Google Identity Services 做登入:
- 引入 https://accounts.google.com/gsi/client
- 用我的 Client ID 渲染「用 Google 登入」按鈕
- 登入成功拿到 credential(ID token),帶 Authorization: Bearer <token> 去打 Worker 的 /list
- 被擋(401)就顯示「你的 Google 帳號不在白名單」
```

步驟三:Worker 驗 token + 白名單

Worker 收到 Bearer token 後:從 Google 的公鑰驗 JWT 簽名、檢查 `aud` 等於你的 Client ID、檢查 email 在白名單(白名單放 Worker 環境變數 `ALLOWED_EMAILS`,逗號分隔)。**改白名單就能即時加人/踢人**,不用動程式。

怎麼選

	做法一:token 標頭	做法三:Google 登入	做法二:Cloudflare Access
門檻	一把共享密碼	每人用 Google 實名	每人實名
要域名嗎	不用	**不用**	要(Cloudflare 託管)
加人/踢人	換 token 全員重給	改白名單即時	後台移除即時
成本	零,本章已內建	零(只要建個 Client ID)	要把後台搬上 Cloudflare

一句話:**自己看就做法一;沒域名又想實名登入用做法三(Google);有 Cloudflare 域名、要完整稽核用做法二。** 別把真實客戶個資長期只靠一把共享 token 守著。

名單拿到了,然後呢

收名單只是第一步,真正的行銷是「寄信給他們」。CSV 匯出後可以: - 匯進 **MailerLite / Mailchimp**(免費 tier 都能群發數百到數千封)做開幕通知。 - 或在 Worker 端接 email 寄送 API(Resend、Postmark 等),報名當下就自動寄一封歡迎信——這是進階,本章先做到「收得到、看得到、匯得出」。

常見坑

1. **裸表單沒擋機器人**:公開的 email 收集端點一定會被灌垃圾。本章三道防線:honeypot 隱形欄位、每 IP 限流、email 格式驗證。要更強可加 Cloudflare Turnstile(免費人機驗證)。
2. **去重沒做**:同一人按三次送出,名單就三筆。靠資料庫的 UNIQUE 約束擋,而且回應不要洩漏「這個 email 在不在名單」(隱私)。
3. **管理密碼放進前端 / 放進網址**:後台密碼一旦寫進 admin.html 就等於公開;放進網址 `?token=` 則會被瀏覽器歷史與 CDN 日誌記下。密碼只存在 Worker 的 Secret,前端是「使用者輸入 → 放 Authorization 標頭傳給 Worker 比對」(見上面「後台登入怎麼防護」)。

4. **個資責任**:收了 email 就有保管責任。報名頁寫清楚用途與退訂方式;不需要的個資不要收;名單別外流。
5. **沒有退訂機制**:正式營運的名單要能退訂(法規與商譽)。本章 demo 未做,正式上線前補上。

動手做

任務:把報名表單接起來(最快用現成的 Google 表單,或照本章自架 Worker + D1),自己報名一筆,再進後台看到那筆。 **預期成果**:後台名單出現你剛剛報的那筆,而且能匯出成 CSV。 **卡關怎麼辦**:後台看不到,確認報名頁與後台連到同一個後端、管理密碼 / 登入對得上。 **自檢**:把你的頁貼進行銷頁健檢器 (https://yazelin.github.io/marketing-page-checker/)看相關項目是否變綠

對照成品

- 報名頁:`demos/09-email-signup/index.html`(已接真的 Worker + D1,送出會真的進名單)
- 後台:`demos/09-email-signup/admin.html`(輸入管理密碼看名單、匯 CSV)
- Worker + schema:dev repo 的 `worker-deploy/email/`

報名頁送一筆,再開後台輸入密碼,就會看到自己剛剛那筆——完整的「收集 → 儲存 → 後台」一條龍。

模組 10 — 被看見:SEO、社群分享卡(OG)、給 AI 看的頁面(AEO / GEO)

為什麼這章

做得再好的活動頁,**沒人看見等於沒做**。頁面被看見有三條路,這章一次補齊:

1. 有人 Google 搜尋 → **SEO**(搜尋引擎優化)
2. 有人把連結貼到 LINE / FB / IG → **OG**(社群分享卡)
3. 有人問 ChatGPT / Claude / Perplexity「OO 有什麼活動」→ **AEO / GEO**(讓 AI 找得到、引用得到你)

第三條是 2026 越來越重要的新管道:很多人已經不 Google 了,直接問 AI。你的頁面要讓 AI 講得出來。

> **名詞釐清**:這條線業界有兩個常見說法,目標相近、常被合稱: - **AEO(Answer Engine Optimization,答案引擎優化)**:讓內容被 AI 的直接答案、精選摘要、語音助理引用。 - **GEO(Generative Engine Optimization,生成引擎優化)**:讓 ChatGPT / Claude / Perplexity 這類大模型在生成回答時引用你。
>
> 兩者吃的訊號高度重疊(結構化資料、清楚文字、可被爬),所以下面的做法對兩者都有用。(沒有「Agent EO」這個正式術語——若看到,當它是 AEO/GEO 的口語說法即可。)

一、SEO 基本功(給搜尋引擎)

- `<title>`:每頁不同,放關鍵字 + 品牌名
- `<meta name="description">`:一兩句,是搜尋結果裡的摘要,寫得讓人想點
- 語意化 HTML:一頁一個 `<h1>`、標題層級清楚
- `sitemap.xml`:列出你所有公開頁,丟進 Google Search Console
- `robots.txt`:允許爬、指向 sitemap;不想被收錄的頁(後台、感謝頁)用 `<meta name="robots" content="noindex">` 擋

二、社群分享卡 OG(給 LINE / FB / IG)

活動頁最常被從 LINE / FB 點開,分享出去長什麼樣,由 OG tags 決定:

```
<meta property="og:title" content="山霧咖啡 開幕週 — 整週買一送一">
<meta property="og:description" content="6/20 至 6/26,霧還沒散,咖啡先香了。">
<meta property="og:image" content="https://你的網域/og.jpg">   <!-- 完整網址! -->
<meta property="og:url" content="https://你的網域/活動頁/">
<meta name="twitter:card" content="summary_large_image">
<meta name="twitter:image" content="https://你的網域/og.jpg">
```

鐵則:

- og:image 必須是**完整 https 網址**,不能相對路徑(LINE / FB 抓不到)
- 圖 **1200x630**、壓到 200KB 內
- 別留 `example.com` 假網址——本課第一版 demo 就犯過這個,分享出來是破圖(這就是為什麼有了這一章)
- 改了 OG,平台會快取舊的;FB 有 **Sharing Debugger** 可強制重抓:https://developers.facebook.com/tools/debug/ (貼網址 → 按 Scrape Again)

怎麼做 OG 圖,不用設計軟體:寫一個 1200x630 的 HTML(品牌底圖 + 標題字),用瀏覽器截圖即可。本課的兩張 OG 圖就是 headless 瀏覽器截出來的,中文字靠瀏覽器字體,品質穩定。

三、AEO / GEO:給 AI 看的頁面(新管道)

越來越多人問 AI 而不是 Google。要讓 AI 正確講出你的活動,給它讀得懂的訊號:

1. 結構化資料 JSON-LD(schema.org)

用機器讀得懂的格式,明白告訴 AI 與搜尋引擎「這頁是什麼」。活動頁用 `Event`,課程用 `Course`,實體店用 `LocalBusiness`:

```
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@type": "Event",
  "name": "山霧咖啡 開幕週",
  "startDate": "2026-06-20T08:00:00+08:00",
  "endDate": "2026-06-26T18:00:00+08:00",
  "location": { "@type": "Place", "name": "山霧咖啡", "address": "霧峰山路 36 號" },
  "offers": { "@type": "Offer", "description": "全品項買一送一" }
}
</script>
```

AI 與 Google 都吃這個。Google 會拿它做「複合式搜尋結果」(rich result),AI 會用它精準回答你的活動何時、在哪、有什麼優惠。

2. llms.txt

放在網站根目錄的純文字說明書,**專門寫給 LLM**:一句話講你是誰 + 重要頁面連結。AI 抓你的站時先看它。格式很簡單:一個標題、一段 `>` 摘要、幾組分節連結。

3. 把重要資訊用「文字」寫出來

AI 讀的是文字。日期、地點、優惠這些關鍵資訊,一定要用**文字**寫在頁面上,別只放在圖片裡——圖裡的字 AI(和搜尋引擎)讀不到。這也呼應模組 2:圖負責氛圍,字用網頁層。

驗收

- **SEO**:Google 搜「site:你的網址」看有沒有被收錄;Search Console 提交 sitemap
- **OG**:把網址貼到 LINE 給自己,看預覽卡有沒有出現你的標題和圖;FB Sharing Debugger(https://developers.facebook.com/tools/debug/)看平台抓到什麼、按 Scrape Again 強制重抓
- **AEO / GEO**:把網址貼給 ChatGPT / Claude:「這個頁面在講什麼活動?」,看它講得對不對、講不講得出日期優惠;用 Google Rich Results Test(https://search.google.com/test/rich-results)驗 JSON-LD 合不合法
- **線上 AI 可見性檢查工具**:有不少免費工具一鍵掃描你的網址,檢查 robots.txt / llms.txt / 結構化資料 / AI 爬蟲存取是否就緒,給一個「AI 健康度」評分。可用的有:
- CrawlerCheck(crawlercheck.com)— 專查 robots.txt / meta robots / 各家 AI 爬蟲 user-agent 是否被擋
- ClayHog GEO Audit、AI Rank Lab、Web Aloha、Ayzeo — 抓首頁 + robots/llms.txt + schema,綜合評 AEO/GEO 就緒度
- 用法:貼上你上線的網址,看它標紅的項目逐條補。(這些是第三方服務,送出等於把你的網址給它們——對公開的活動頁無妨。)
- **後台別外洩**:admin / 感謝頁加 noindex、不進 sitemap

常見坑

1. **og:image 相對路徑或假網址**:分享破圖。必須完整 https 網址。
2. **改了 OG 沒重抓**:平台快取舊的,用 Sharing Debugger(https://developers.facebook.com/tools/debug/)強制重抓。
3. **JSON-LD 跟頁面內容對不上**:亂填會被搜尋引擎懲罰。schema 寫的要跟頁面真的一致。
4. **重要資訊只放在圖裡**:AI 與搜尋引擎讀不到。日期、地點、優惠用文字也寫一遍。
5. **後台 / 感謝頁被索引**:加 `<meta name="robots" content="noindex">`。
6. **專案頁的 robots / llms 在子路徑、不在網域根**:GitHub Pages 專案頁(`github.io/你的repo/`)的 `robots.txt`、`llms.txt` 只能放子路徑,爬蟲讀的是網域根。要完整 SEO,得用自訂網域(呼應模組 5)。

真實案例:為什麼「專案頁」AEO 分數低,根域才是解法

把網址丟進 AI 可見性檢測工具(如 isitagentready.com),如果你的站是 GitHub Pages **專案頁**(`帳號.github.io/repo名/`),分數常常很低——但問題往往不是你沒做優化,是工具掃的是**網域根** `帳號.github.io/`,不是你專案的子路徑。

這門課自己就踩到這個,實測數據(同一個帳號,根域 vs 課程站子路徑):

訊號	網域根(檢測工具掃的)	專案子路徑(你做了優化的)
llms.txt	404 沒有	200 有
首頁 JSON-LD	0 個	有
meta description	空	有

工具掃根域,看到的是你的 Profile / 部落格首頁,沒有你在子專案做的那一套——分數當然低。三條解法:

1. **自訂網域**:把專案掛上自己的網域,優化就在網域根,工具掃得到。最徹底(呼應模組 5)。
2. **把根域站(個人 Profile / 部落格)的 AEO 做好**,並在它的 `llms.txt` 列出你所有專案——讓 AI 從根域一次發現你全部的子站。這是**不買網域時的最佳解**。
3. **接受專案頁的限制**:靠站內 meta(每頁完整 OG / description / JSON-LD)讓「個別頁面」被 AI 讀懂。本課 demo 已驗證:單頁貼給 AI,它讀得懂是什麼活動——即使整站的根域分數不漂亮。

教訓:**AEO 檢測分數要看「它掃的是哪個 URL」**。專案頁拿低分,先確認工具是不是只掃了網域根,再決定要不要為它自訂網域。

動手做

任務:給你的頁補上完整 OG tags 加一段 `Event` 或 `Course` 的 JSON-LD,部署後用 FB Sharing Debugger 重抓,再把網址貼給 ChatGPT / Claude 問「這頁在講什麼活動」。 **預期成果**:Debugger 顯示出你的分享卡;AI 答得出你的活動重點。 **卡關怎麼辦**:Debugger 沒圖,檢查 `og:image` 是不是完整的 https 網址,按 Scrape Again 重抓。 **自檢**:把你的頁貼進行銷頁健檢器(https://yazelin.github.io/marketing-page-checker/)看相關項目是否變綠

對照成品

本課**整站就是這一章的案例**:

- 每個 demo 頁有完整 OG tags + JSON-LD(活動頁 `Event`、課程站 `Course`)
- 兩張真實 OG 圖(headless 瀏覽器截的,不是 example.com 占位)
- 站點有 `sitemap.xml` 與 `llms.txt`(`/ai-marketing-pages-course/llms.txt`)
- 後台頁 `admin.html` / `admin-google.html` 都 noindex

把任一 demo 網址貼給 AI 問「這在講什麼」,它應該答得出是山霧咖啡的什麼活動——那就是 AEO / GEO 生效了。

做完了?用「行銷頁健檢器」自檢

把這一章(以及整門課:OG、SEO、CTA、圖片速度、體質、追蹤)該檢查的事做成了一個免費工具——**行銷頁健檢器**:

> https://yazelin.github.io/marketing-page-checker/

貼上你做好的活動頁網址,30 秒給你一個分數 + 各面向的綠/黃/紅 + 一份「先修這幾個」的白話清單(每項還附可直接複製貼給 AI 的修正 prompt)。**做完你的頁就貼進去自檢,照它的紅燈逐條修。**

而且這個健檢器本身就是用本課教的東西做的:**GitHub Pages 前端 + Cloudflare Worker 後端**(模組 5、6),Worker 抓你的頁、解析、評分。它是「學 → 做 → 自檢」這個閉環的最後一塊,也是 Worker 架構的真實範例。

模組 11 — 成效追蹤:裝分析與廣告 Pixel

學完能做什麼

幫活動頁裝上「網站分析」(知道多少人來、從哪來、有沒有完成報名)和「廣告 Pixel」(要投 FB/IG 廣告時,才追得到轉換、做得了再行銷)。都是貼一段 script 的事,不用寫程式。

為什麼一定要裝

行銷最怕「做了卻不知道有沒有效」。沒有分析,你不知道:這波貼文帶來幾個人?大家從手機還是電腦來?有多少人滑到報名按鈕卻沒按?**裝了分析,下次活動才有依據優化;沒裝,就是憑感覺。** 這也是健檢器會檢查「追蹤」這一項的原因。

一、網站分析(每個活動頁都該裝)

選一個,貼一段 script 到 `<head>`:

選項 A:Google Analytics 4(最主流、免費)

1. 到 analytics.google.com 建一個「資源」,拿到一段 `G-XXXXXXX` 的代碼。
2. 它會給你一段 script,貼進你頁面的 `<head>`:

```
<script async src="https://www.googletagmanager.com/gtag/js?id=G-XXXXXXX"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());
  gtag('config', 'G-XXXXXXX');
</script>
```

1. 換成你自己的 `G-XXXXXXX`。上線後,GA 後台就會看到即時人數、來源、裝置。

選項 B:Plausible / Umami(隱私友善、更簡單)

不想用 Google、或在意隱私,可以用 Plausible(付費,介面極簡)或 Umami(可自架免費)。同樣是貼一段 script。對小活動,GA 免費版已經很夠。

給 AI 的 prompt:

```
幫我把這段 Google Analytics 代碼正確貼進我頁面的 <head>:[貼上 GA 給你的 script]
確認 measurement id 是 G-XXXXXXX,放在 <head> 結束前。
```

二、追「報名成功」這個關鍵動作(事件追蹤)

只知道「有人來」還不夠,你要知道「有幾個人真的報名了」。在報名成功時送一個事件給 GA:

```
<!-- 在報名表單送出成功後執行 -->
<script>gtag('event', 'signup', { event_category: '開幕週' });</script>
```

這樣 GA 後台就能看到「轉換數」,算得出轉換率(報名數 ÷ 來訪數)。給 AI:「我的報名表單送出成功時,幫我呼叫 gtag 送一個名為 signup 的事件。」

三、廣告 Pixel(只有要投廣告才需要)

如果你要在 FB/IG 下廣告,**Meta Pixel** 是必裝——沒有它,你無法知道哪個廣告帶來報名、也無法做「再行銷」(把廣告投給來過但沒報名的人)。

1. 到 Meta 商務管理工具建立 Pixel,拿到一段含 `fbq('init', '你的PixelID')` 的代碼。
2. 貼進 `<head>`(Meta 會給完整 script,照貼)。
3. 在報名成功時加一行 `fbq('track', 'Lead');` 或 `fbq('track', 'CompleteRegistration');` 追轉換。

不投廣告就不用裝(健檢器對 Pixel 缺少只給「提醒」不扣硬分,就是這個道理)。

隱私與法規(別忽略)

- 裝了追蹤就有在蒐集行為資料,頁面該有簡短的隱私說明或 cookie 提示(尤其面向歐盟訪客要考慮 GDPR / cookie 同意)。
- 不要追蹤你不需要的東西;能匿名就匿名。
- email 名單(模組 9)那種個資,保管責任更重——別把分析資料跟個資隨意串接。

常見坑

1. **裝了沒驗證**:貼完 script 要實際開頁面,在 GA「即時」報表看到自己這一次造訪,才算真的通。
2. **只追流量、不追轉換**:知道一千人來、卻不知道幾個報名,等於沒抓到重點。一定要設「報名成功」事件。
3. **Pixel 裝了卻沒在轉換點觸發**:只 init 不 track,廣告後台看不到轉換,再行銷也做不了。
4. **把追蹤碼貼錯位置**:GA/Pixel 主代碼放 `<head>`;事件 `track` 放在「動作真的完成」的那一刻(表單送出成功後),不是頁面一載入就送。
5. **忘了隱私揭露**:蒐集資料卻沒說,法規與信任都有風險。

對照

健檢器(見模組 10 結尾)的「追蹤」類別就是檢查這兩件:有沒有裝分析、有沒有廣告 Pixel。做完本模組,把你的頁貼進健檢器,追蹤那一格應該變綠。

動手做

任務:給你的頁裝 Google Analytics,開頁面後到 GA「即時」報表看到自己這次造訪;在報名成功時送一個名為 `signup` 的事件。 **預期成果**:GA 即時報表看到 1 位使用者;事件清單裡出現 `signup`。 **卡關怎麼辦**:即時沒看到,確認 `G-XXXXXXX` 換成你自己的、追蹤碼貼在 `<head>` 裡。 **自檢**:把你的頁貼進行銷頁健檢器(https://yazelin.github.io/marketing-page-checker/)看相關項目是否變綠