

https://youtu.be/z7H7WC-s8B4?list=PLIAzIZzCjtLJQzoK-1n6R_KVTXm4tHZqP

https://youtu.be/7SaeZKo6l08?list=PLIAzIZzCjtLJQzoK-1n6R_KVTXm4tHZqP

https://youtu.be/7yW-JO42-9s?list=PLIAzIZzCjtLJQzoK-1n6R_KVTXm4tHZqP

https://youtu.be/Jyrs3BwwbJA?list=PLIAzIZzCjtLJQzoK-1n6R_KVTXm4tHZqP

這份教學原始檔來自國立臺灣大學的**網頁程式設計課程**，主要由教授引導學生進入開發環境的設置與**AI 輔助寫程式的實務操作**。課程重點強調了**安全性與環境管理**，要求學生使用環境變數 (.env) 來保護 API 金鑰等敏感資訊，並指出能讓團隊成員快速**重現在地端開發環境**是專業軟體工程的重要指標。技術層面上，教授親自示範如何利用**Cursor 開發工具**搭配不同 AI 模型（如 Claude），透過精確的**Prompt 指令與一鍵生成 (One-shot)** 技術，在短短幾分鐘內完成包含 HTML、CSS 與 JavaScript 的待辦事項應用程式。此外，課程也涵蓋了實體與線上學習的資源分配，展現出結合**生成式 AI** 轉型網頁開發教學的新趨勢。

這份教學資源旨在引導學生利用**AI 輔助開發技術**，特別是透過與 AI 沟通來快速構建網頁專案。內容重點介紹了如何利用**提示詞工程**與截圖輸入，要求 AI 生成純前端的應用程式，並強調開發者應專注於**需求描述與邏輯審核**而非手動編寫程式碼。針對第一份作業，學生需挑選喜愛的風格並透過 AI 實作個人網頁，練習使用**TypeScript** 與套件，同時學習如何微調 AI 生成的結果以符合設計預期。最後，課程說明了**同儕互評機制**與成績調整方式，鼓勵學生透過實際操作掌握 AI 工具帶來的開發效率變革。

這份教學原始碼主要介紹了**網頁程式設計的基礎架構**，特別著重於建立學生對**前端與後端分工**的正確觀念。內容詳盡說明了瀏覽器如何透過**文件物件模型 (DOM)** 將 HTML 代碼轉換為具備層次感的樹狀結構，並以此作為動態更新網頁內容的核心機制。透過現場示範**動態更換網頁元素**的實驗，講者展示了 JavaScript 如何即時操控 DOM 節點來改變使用者介面。此外，課程也涵蓋了**版本控制工具 Git 與 AI 輔助編程**的應用，旨在讓學習者在掌握 HTML、CSS 與 JavaScript 三大核心語言的同時，能結合現代工具處理更複雜的網路服務開發任務。

這份教學原始碼主要探討**JavaScript 的核心特性與演進**，說明其從早期野心過大的失敗到如今成為網頁開發主流語言的歷程。文中特別強調了**弱型別系統**在網路環境中具備高度容錯性的優點，但也因此延伸出維護困難的問題，進而促使具備強型別保護的**TypeScript** 崛起。此外，作者詳細解構了**DOM 節點操作**、事件監聽以及**非同步函數與 Array 方法**（如 map 與 filter）的實務應用，幫助學習者理解動態網頁的運作機制。最後，教學延伸至**AI 輔助開發**，分享如何利用提示詞與設計圖引導 AI 產出符合期待的 UI 介面，展現了現代軟體開發中**人機協作**的重要趨勢。

Discuss 環境變數設定。

在 Web 程式開發中，**環境變數設定**是確保系統安全、靈活性與團隊協作效率的關鍵技術。根據提供的來源，以下是關於環境變數設定的核心要點：

1. 保護敏感資料與安全管理

在開發網頁服務時，經常需要介接第三方服務（例如 GPT 的 API），這類服務通常需要 API 金鑰（Key）才能運作。為了安全起見，不應將這些金鑰直接寫在程式碼中，而是應該儲存在專門的**環境設定檔**（如 .env 檔案）中。

- **嚴禁「綁死」金鑰**：將金鑰直接寫死在程式碼（Hardcoding）中被視為一種錯誤的做法，因為這會讓任何閱讀程式碼的人都能直接看到你的私密資訊。
- **讀取機制**：程式應該設計成從環境變數中讀取金鑰，而不是直接包含金鑰內容。
- **版本控制規範**：在上傳程式碼到 GitHub 或繳交作業時，絕對不要上傳包含金鑰的環境設定檔。這樣做是為了防止私密資訊外洩給大眾。在實際評分或測試時，助教會在自己的電腦（localhost）上使用他們自己的金鑰來運行你的程式。

2. 開發環境的可重現性

良好的環境變數與設定管理，對於**重現開發環境**至關重要。

- **團隊協作**：一個成熟的服務應該能讓新加入的成員快速地在自己的本地端（Local host）重新安裝並執行。
- **避免設定泥淖**：如果環境設定混亂，新成員可能需要花費大量時間（甚至一個月）才跑得起程式，這會嚴重影響開發效率。

3. 在 AI 開發工具中的應用

在使用如 **Cursor** 這種 AI 驅動的整合開發環境（IDE）時，開發者可以透過設定選取不同的 AI 模型（如 Claude, GPT-4, 或 DeepSeek）。

- **模型選擇與費用管理**：在設定中可以開啟「Max mode」使用更強大的推理模型，但這會消耗較多 token（成本較高）。對於一般作業，開發者可以利用環境設定選擇自動模式（Auto mode）來平衡效能與成本。

比喻理解：環境變數就像是房屋的配電盤。電器（程式碼）本身不帶電，它只需要知道去「配電盤」插上電源。當你把房子從台灣移到美國（更換開發環境），你只需要更換配電盤的電壓（環境變數），而不需要拆掉電器重新製造。這樣既保證了電器（程式碼）的通用性，也保護了電力來源（金鑰）的安全。

助教在 local host 批改期末專案的流程為何？

為什麼建議將 API 金鑰設定在環境變數檔案中？

Cursor 的 Max mode 與一般模式有何主要差異？

Discuss 本地開發環境。

在 Web 程式開發中，**本地開發環境（Local Development Environment）** 是指開發者在自己的電腦（即 **localhost**）上建立的運行與測試空間。根據來源，建立一個良好且可重現的本地環境是開發流程中不可或缺的一環，其核心概念與運作方式如下：

1. 本地開發環境的重要性

- **環境重現性（Reproducibility）**：一個成熟的服務必須能夠讓團隊成員（如新進人員）在自己的本地機器上**快速地重新安裝並重現（Reproduce）** 開發環境。如果環境設定過於複雜，導致新人需要花費大量時間才能讓程式跑起來，會嚴重影響團隊協作效率。

- **作業評分基準**：在學術或教學場景中，助教會將學生的程式碼下載到自己的 **localhost** 環境中安裝並執行，以此作為批改與測試的標準。

2. 開發工具與實務操作

- **專案目錄管理**：開發的第一步通常是在電腦中建立一個適當的目錄（例如名為 DEMO 的資料夾），並使用開發工具（如 **Cursor**）開啟該專案目錄。

- **IDE 介面組成**：一個典型的本地開發介面通常包含三個部分：

1. **檔案瀏覽器**：管理目錄與檔案架構。

2. **程式碼編輯區**：撰寫與修改 HTML、JavaScript 或 CSS 的地方。

3. **終端機（Terminal）**：開發者透過下達指令直接與電腦溝通的介面，例如檢查目錄內容或執行程式。

- **AI 輔助開發**：現代工具如 Cursor 允許在本地環境中整合 AI 聊天室，開發者可以透過 Prompt 讓 AI 直接在本地專案中生成或修改程式碼。

3. 本地環境的安全性規範

- **金鑰管理**：在本地開發時，若涉及第三方服務（如 GPT API），應透過環境變數讀取金鑰，**嚴禁將金鑰寫死（Hardcoding）在程式碼中**。

- **隔離敏感資訊**：上傳程式碼至版本控制系統（如 GitHub）時，應確保不包含本地的私密環境設定檔。助教會在他們的本地環境中使用自己的金鑰來運行你的程式。

比喻理解：本地開發環境就像是廚師的**私人廚房**。你在這裡準備食材、調整配方（寫程式碼），並確保所有的調味料（環境變數）都放在對的地方。一個好的廚房應該要有清楚的動線（良好的專案架構），讓其他廚師進來時，也能立刻找到瓦斯爐（終端機）並煮出同樣味道的料理（重現服務），而不是還要花一個月研究你的廚房怎麼開火。

如何妥善管理專案中的 API 金鑰與敏感資訊？

在 Cursor 中開啟 Max Mode 有何功能與缺點？

開發者應如何與 AI 溝通以精準達成需求？

Discuss 課程錄影上傳。

關於課程錄影的上傳與存取方式，根據來源，主要分為**官方自動系統**與**教師個人錄製**兩個管道：

1. 官方系統：台大演講網

教室後方設有專門的錄影機進行即時錄製。該設備已設定為**自動上傳至「台大演講網」**。老師提到，台大、師大與台科大的同學應有對應的管道可以進入該網站觀看，且其更新速度相對及時。

2. 教師個人錄製：GitHub 開源分享

除了官方系統，老師也會使用自己的筆記型電腦（透過 QuickTime 軟體）同步錄製電腦畫面與現場收音。

- **上傳平台**：這些個人錄製的檔案會上傳至 GitHub。
- **開放性**：老師選擇將這些資源開源，因此任何有興趣的人（不限於修課同學）都可以收看。
- **處理時間**：個人版本的上傳可能會稍有延遲。老師表示整理與上傳大約需要一小時的時間，但這在老師的工作事項中不一定是最高優先權，因此可能不會立即更新。

3. 錄影對課堂出席的影響

老師觀察到，由於提供錄影檔，學生在現場的出席率往往不高，根據經驗可能難以超過 50%，甚至可能僅有 20% 的學生會到課。許多同學會選擇在家睡覺或是事後觀看影片。不過，老師仍鼓勵同學盡量參與實體課程，因為「課堂無法快轉」，這與獨自看影片的學習體驗是不同的。

4. 錄製過程中的特殊情況

在實際操作示範時，錄製過程可能會根據需求調整。例如，老師在準備現場示範 AI 工具（如 Cursor）的過程中，若需要截取特定畫面，可能會暫時暫停錄影以利操作。

比喻理解：這套錄影機制就像是體育賽事的**官方直播與教練觀景窗**。台大演講網提供的是官方標準視角（即時且自動）；而老師的 GitHub 錄影則像是教練的特寫鏡頭，專注於電腦畫面與細節，雖然需要一點時間剪輯整理，但對外公開，讓校外的觀眾也能像「旁聽」一樣獲取知識。

助教在批改作業時，如何在本地端環境運行程式？

在開發過程中，環境設定檔 (.env) 的主要用途為何？

學生在使用 Cursor 的 Max mode 時需注意什麼限制？

Discuss AI 輔助編程.

在 Web 程式開發領域，**AI 輔助編程（AI-assisted programming）** 已成為現代開發者的核心技能。根據來源，這種開發模式不僅能大幅提升生產力，也改變了開發者與程式碼互動的方式。

以下是關於 AI 輔助編程的核心討論點：

1. 強大的開發工具與模型選擇

目前課程中主要使用的工具是 **Cursor**，這是一個整合了 AI 聊天室的集成開發環境（IDE）。

• **多樣化模型**：開發者可以根據需求選取不同的 AI 模型，例如 **Claude**（被認為是目前產生程式碼最強大的模型）、**GPT-4**、或是 **DeepSeek** 等。

• 模式切換（Auto vs. Max Mode）：

◦ **Auto Mode**：預設模式，會自動幫開發者挑選合適的模型，足以應付大部分的作業需求。

◦ **Max Mode**：更進階的模式。當開啟時，AI 會進行更深層的思考，甚至會自動產生測試案例並根據錯誤自行修復。雖然速度較慢且消耗的 Token（點數）較多，但具備極強的邏輯推理能力。

2. 「一鍵生成」（One-shot）的開發潛力

AI 輔助編程展現了極高的效率，例如在示範中，透過簡單的指令或一張設計圖，AI 就能以 "**One-shot**"（一次性）的方式寫出包含 HTML、JavaScript 與 CSS 的完整功能。

- 即便是一個簡單的待辦清單（To-do App），AI 產生的 JavaScript 與 CSS 程式碼也可能各達 100 多行，這在過去需要手動撰寫許久。
- AI 甚至能協助理解 Web 服務的各個層面，包括**前端**、**後端**的技術勾稽。

3. 指令工程（Prompting）與溝通藝術

來源強調，AI 輔助編程的關鍵在於**如何與 AI 溝通**，而不僅僅是下指令。

- **管理者的隱喻**：與 AI 溝通就像擔任一名主管。如果你只是丟下一份 300 頁的規格書叫下屬去做，最後成品可能與想像完全不同。
- **精確描述**：一個好的開發者需要懂得如何精確地與 AI 溝通需求。由於 AI 背後是**統計模型**，結果具有一定的隨機性（類似樂透），因此清晰的 Prompt 才能降低誤差。

4. 成本與實務考量

雖然 AI 功能強大，但開發者仍需管理成本與安全性：

- **Token 管理**：在學習階段，通常使用免費額度或固定的點數（如 \$20 額度）即可應付多次作業，不建議隨意開啟昂貴的 Max mode 以免點數耗盡。
- **安全性與環境變數**：雖然 AI 可以幫你寫 code，但開發者仍須遵循安全規範。例如，**API 金鑰（Key）絕不能直接寫在程式碼中讓 AI 處理**，而是應透過環境變數讀取，並避免將私密設定檔上傳。

比喻理解：AI 輔助編程就像是擁有一位**極其博學但缺乏主見的實習生**。他讀過所有的程式碼教科書（統計模型），你叫他寫什麼他都能寫得飛快。然而，你必須給予他非常清晰的指示（Prompt），並為他準備好安全的辦公桌（環境變數設定），他才能真正幫你打造出一棟穩固的數位大樓。

如何妥善管理與保護 API 金鑰等敏感資訊？

助教在批改作業時會如何測試學生的程式碼？

教師將透過哪些管道提供課程的錄影檔案？

Discuss 網頁開發基礎。

在網頁開發的學習過程中，掌握基礎架構、開發工具以及現代的 AI 協作模式是至關重要的。根據來源，網頁開發基礎可以從以下幾個核心維度來理解：

1. 網頁開發的核心技術組成

一個完整的網頁服務（Web Service）主要由**前端（Front-end）**與**後端（Back-end）**構成，而其最基礎的技術支柱包括：

- **HTML、JavaScript 與 CSS**：這是網頁開發的三大基本元素。

- **DOM (Document Object Model)**：這是一個關鍵概念，讓開發者能夠動態地改變網頁內容。
- **實作練習**：初學者通常從製作簡單的個人網頁或 Todo App 開始，以此熟悉程式碼與技術的結合。

2. 本地開發環境的建立

開發網頁的第一步是在自己的電腦（localhost）建立一個可運作的環境：

- **專案目錄管理**：需要在電腦中建立適當的目錄（例如名為 DEMO 的資料夾）來存放所有檔案。
- **IDE（整合開發環境）介面**：現代開發工具（如 Cursor）通常包含三個部分：**檔案瀏覽器**（管理目錄）、**程式碼編輯區**、以及**終端機（Terminal）**。
- **終端機的作用**：它是開發者直接下達指令與電腦溝通的地方。
- **環境重現性**：確保服務能在本地端快速重新安裝並執行，是團隊協作中非常重要的指標。

3. AI 輔助開發與溝通（Prompt Engineering）

現代網頁開發已大量引入 AI 工具（如 Cursor）來提升效率：

- **AI 模型選擇**：開發者可以根據需求選擇不同的模型（如 Claude, GPT-4, 或 DeepSeek）。
- **Prompt 的重要性**：與 AI 溝通就像主管與部屬溝通。僅僅提供一張圖是不夠的，開發者需要透過清晰的 **Prompt（提示詞）** 來描述需求，否則產出的結果可能與想像不符。
- **自動化能力**：進階的 AI 模式（如 Max mode）甚至能自動產生測試並自行修復錯誤，但一般基礎開發使用自動模式（Auto mode）已綽綽有餘。

4. 安全性與環境設定

在開發過程中，必須建立正確的安全意識：

- **環境設定檔（.env）**：敏感資訊（如 GPT 的 API Key）應儲存在環境設定檔中，而不是直接寫死在程式碼裡。
- **版本控制安全**：在將程式碼上傳至 GitHub 等平台時，嚴禁將包含私密金鑰的設定檔一併上傳，以防資訊外洩。

比喻理解：網頁開發基礎就像是蓋房子。**HTML** 是房屋的結構（樑柱），**CSS** 是室內設計與裝潢，**JavaScript** 是水電系統（讓電燈會亮、水龍頭有水）。而 **DOM** 就像是屋內的自動感應系統，能根據人的行為動態調整燈光。至於 **AI 工具**，則像是一位資深的建築助理，只要你給出精確的指令（Prompt），他就能幫你快速搭建出房屋的雛形。

如何妥善管理與保護第三方服務的 API 金鑰？

為什麼開發環境必須具備在本地端重現的功能？

在 Cursor 中，使用 Max Mode 與 Auto Mode 有何差異？

Discuss AI 輔助程式編寫。

在現代網頁開發中，AI 輔助程式編寫（AI-assisted programming）已從單純的自動補完演變為一種「協作開發」模式。根據來源，這種開發方式的核心在於工具的選用、模型特性的理解，以及與 AI 溝通的藝術。

1. 核心工具與 AI 模型選擇

課程中主要使用的開發工具是 **Cursor**，它是一款整合了 AI 聊天功能的整合開發環境（IDE）。

• **多樣化模型**：Cursor 支援多種 AI 模型，包括被認為在程式碼生成上最為強大的 **Claude**，以及 GPT-4、DeepSeek 和 XAI Grok 等。

• **運行模式**：

- **Auto mode**：預設模式，會自動為開發者挑選合適的模型，足以應付大多數的作業需求。

- **Max mode (Composer)**：這是一種進階模式，AI 會進行深層思考，並具備**自動產生測試、執行程式碼、發現錯誤並自我修復**的能力。然而，此模式消耗的 Token（點數）較多，建議在處理複雜邏輯時再開啟。

2. 「一鍵生成」（One-shot）的開發效能

AI 展現了極高的開發效率，特別是在處理標準化的功能時：

• **快速生成**：開發者可以透過「一鍵生成」（One-shot）的方式，從一張截圖或一段簡單的描述中，直接產生超過 100 行的 JavaScript 與 CSS 程式碼。

• **跨領域應用**：除了網頁前端（HTML/CSS/JS），AI 也能協助處理 TypeScript 轉換，甚至能根據影片內容重現經典遊戲（如小蜜蜂、五子棋），並處理音效生成。

3. 指令工程：溝通的藝術（Prompt Engineering）

來源強調，AI 輔助開發的成功關鍵在於**如何精確地與 AI 溝通**。

• **管理者的隱喻**：與 AI 溝通就像擔任主管。如果你只是丟下一份厚重的規格書而不解釋期望，AI 生成的結果可能與想像不符。

• **多模態輸入**：提供截圖（Screenshot）或手繪圖是極佳的溝通方式，能幫助 AI 理解網頁佈局（Layout）與風格。

• **細節描述**：開發者需要詳細說明動態行為（例如：捲動時輸入框是否固定、點擊按鈕後是否彈出確認視窗），以彌補 AI 可能出現的「腦補」偏差。

4. 實務考量與安全性

儘管 AI 非常強大，開發者仍需負起管理與查核的責任：

• **安全性原則**：嚴禁將 API 金鑰（API Key）交給 AI 寫死（Hardcoding）在程式碼中。正確做法是將金鑰儲存於環境設定檔（.env）並透過環境變數讀取。

• **查核與修改**：AI 可能會產生不符合預期的佈局或邏輯，開發者必須學會看懂 AI 產生的程式碼，並在必要時要求 AI 重作或手動修補。

- **成本管理**：Cursor Pro 的 20 美元額度通常足以完成整學期的作業，但開發者需注意 Token 的使用情況，避免過度依賴昂貴的 Max mode。
-

比喻理解：AI 輔助程式編寫就像是擁有一位**動作極快但缺乏直覺的資深實習生**。他可以在幾秒鐘內幫你砌好一面牆（寫好 100 行程式碼），但他不知道你想要的是什麼顏色的磚塊，除非你畫圖給他看或清楚交代。身為開發者的你，則從「苦力工」轉變成了「工頭」，你的責任是確保藍圖（Prompt）準確無誤，並在完工後檢查結構（Code Review）是否安全。

助教在批改作業時，如何處理程式碼中的 API 金鑰？

在 Cursor 中，使用 Max Mode 與 Auto Mode 有何差異？

作業一要求學生使用哪種程式語言，以及如何運用 AI 創作？

Discuss 前端網頁開發。

在 **前端網頁開發 (Front-end Web Development)** 的領域中，重點在於建立使用者能直接看見並互動的介面。根據來源，以下是前端開發的核心組成、技術演進以及現代化的開發流程：

1. 前端開發的三大支柱與核心概念

網頁前端主要由三種語言構成，開發者需要理解它們如何相互協同工作：

- **HTML**：負責定義網頁的**結構與內容**（例如：文字框、預設的列表項）。
- **CSS**：負責控制網頁的**樣式 (Style) 與外觀**（例如：圓角弧度、顏色、佈局）。
- **JavaScript (JS)**：負責處理**動態互動與邏輯**（例如：點擊按鈕後的行為、刪除項目、音效播放）。
- **DOM (Document Object Model)**：這是前端開發中極為關鍵的概念，它讓程式能夠**動態地改變網頁內容**，使網頁不再只是靜態頁面。

2. 從 JavaScript 到 TypeScript 的演進

雖然初學者可以從 JavaScript 開始，但現代前端開發（如課程作業要求）傾向於使用 **TypeScript (TS)**。

- **轉換過程**：開發者可以先用 JavaScript 撰寫功能，再利用 AI 工具將其轉換為 TypeScript，這通常可以「一鍵完成」。
- **進階技術**：未來還會涉及更複雜的前端框架與套件，例如 **React** 和 **Shadcn**。

3. 「純前端」開發的實務應用

在學習初期，通常會強調**「純前端 (Pure Front-end)」**的開發模式，這意味著不涉及後端資料庫。

- **特性**：這類應用程式（如 Todo App）的所有資料都存在瀏覽器端。一旦重新整理網頁（Reload），資料通常會回到初始狀態。
- **練習目標**：透過製作個人網頁、Todo App 甚至是重現經典遊戲（如小蜜蜂、五子棋），來熟悉前端邏輯。

4. 現代前端開發流程：AI 協作與設計

現代前端開發已不再單純依賴手寫程式碼，而是進入了 **AI 協作時代**：

- **多模態輸入與溝通**：開發者可以透過截圖（Screenshot）或手繪草圖來告訴 AI 理想的佈局（Layout）與風格（如「霓虹風格」）。
- **精確描述動態行為**：前端開發中最容易出現落差的是「動態細節」。例如：當列表過長時，輸入框是否應固定在上方而不隨之捲動。若未明確指示，AI 可能會出現「腦補」偏差，導致產出不符合預期。
- **細節修補與互動**：包括按鈕的 Mouse-over 換色、點擊後的確認視窗、甚至是產生不具版權問題的環境音效，這些都是前端開發中提升使用者體驗（UX）的環節。

5. 評分與呈現

在前端專案中，除了程式碼本身，**README 檔案**也至關重要，用於向他人（如助教或用戶）解釋網頁的設計理念與內容。而在評價前端作品時，通常更看重開發者的**用心程度與設計概念**，而非單純的視覺美觀。

比喻理解：前端網頁開發就像是裝修一間店面的櫥窗。**HTML** 是櫥窗裡的貨架架構；**CSS** 是燈光顏色與裝飾背景；**JavaScript** 則是櫥窗裡的自動旋轉台或互動按鈕。而 **TypeScript** 就像是更嚴謹的施工藍圖，確保所有的零件（資料型態）都能精確對接。現代的 **AI 工具** 則是一位全能的裝修助手，你給他一張參考圖，他就能幫你快速佈置，但你仍需親自檢查燈光會不會刺眼、按鈕按下去會不會卡住。

助教在批改程式碼時，會如何處理第三方 API 的金鑰？

課程中提到的環境設定檔（.env）主要用途為何？

學生在使用 Cursor 的 Max Mode 時應注意什麼限制？

Discuss 提示詞工程技巧。

在網頁開發中，**提示詞工程（Prompt Engineering）** 的技巧決定了 AI 產出程式碼的品質與精確度。根據來源，AI 本質上是基於統計的模型，具有一定的隨機性（類似「抽樂透」），因此開發者必須學會如何有效地與其溝通。

以下是根據課程內容整理的提示詞工程核心技巧：

1. 角色定位與管理思維

- **主管與部屬的隱喻**：與 AI 溝通應採取「管理者」的視角。開發者不應只是丟出厚重的規格書而不加解釋，因為當 AI 產出的結果與想像不符時，通常是因為溝通不夠精確。
- **明確技術限制**：在提示詞中應清楚定義技術棧（Tech Stack）。例如，要求 AI 使用「純前端」技術（HTML, JavaScript, CSS），並明確指示「不要套用框架（如 React）」或「將 JavaScript 轉為 TypeScript」。

2. 多模態（Multi-modal）輸入技巧

- **利用截圖與草圖**：文字描述佈局（Layout）有時很困難，建議直接提供網頁風格截圖、手繪草圖或投影機製作的設計圖。
- **利用影片重現功能**：對於複雜的動態效果（如老遊戲的彈幕或動作），可以直接提供一段遊戲或功能的影片，讓 AI 根據影片內容來重現邏輯。

3. 精確描述動態行為與隱藏邏輯

這是減少 AI 「腦補」偏差的關鍵，應包含：

- **滾動行為（Scrolling）**：明確指示哪些元件應固定（如輸入框），哪些應隨頁面捲動。
- **互動細節**：例如「刪除按鈕按下去後直接刪除，不需彈出詢問視窗」。
- **隱藏狀態**：描述畫面上看不到的邏輯，例如「點擊後說明文字會出現在下方」。
- **預設資料**：告知 AI 初始狀態應包含哪些內容（如預設的待辦事項）。

4. 抽象風格與資源生成

- **風格定義**：可以使用抽象詞彙定義視覺感受，例如要求「霓虹風格（Neon style）」或「具備太空感的聲效」。
- **非程式碼資源**：透過描述頻率與時間，AI 可以產生不具版權問題的音效資源。

5. 迭代修復與防雷技巧

- **面對錯誤的對策**：當 AI 產生的結果「很醜」或佈局跑掉時，不應只是抱怨。正確做法是針對錯誤點進行具體描述，或直接畫圖告訴它正確的位置並要求重做。
- **One-shot 的限制**：雖然 AI 具備「一鍵生成（One-shot）」的能力，但開發者仍需檢查其產生的程式碼（如 CSS 的圓角弧度或元件寬度）是否符合原始設計。

比喻理解：提示詞工程就像是點菜。如果你只跟廚師說「我要吃麵」（模糊提示），他可能會送上一碗你不喜歡的牛肉麵。好的提示詞工程則是告訴廚師：「我要一碗細麵，湯頭要清燉，不要蔥，加一個半熟蛋，且麵條要硬一點」（精確描述）。雖然廚師手藝很好（AI 運算強大），但只有明確的點餐指令，才能確保送上桌的料理符合你的胃口。

如何妥善管理與保護開發中的 API 金鑰？

課程作業對於使用 TypeScript 有何具體要求？

Cursor 的 Max mode 與一般模式有何主要差異？

Discuss 個人網頁作業。

「個人網頁作業」是本課程的第一個正式作業（作業一），其核心目的並非單純考驗程式撰寫能力，而是讓學生熟悉 AI 輔助開發工具（如 Cursor）的應用與溝通技巧。

以下是關於這項作業的詳細說明：

1. 技術要求與限制

- **純前端開發**：此作業僅限於**前端技術**，不涉及後端資料庫。這意味著所有資料都儲存在瀏覽器端，重新整理後通常會恢復原狀。
- **語言規範**：雖然基礎是 HTML 與 CSS，但老師特別要求使用 **TypeScript (TS)**。
 - **開發技巧**：學生可以先請 AI 用 JavaScript 撰寫，再透過 AI 「一鍵轉換」成 TypeScript。
- **工具應用**：強烈建議利用 AI 工具進行開發。對於熟悉工具的人來說，這項作業可能在 10 分鐘內就能完成。

2. 設計流程與 AI 溝通 (Prompting)

由於每個人喜好的風格不同，老師並未規定具體內容，而是鼓勵學生發揮創意：

- **參考風格**：學生可以搜尋喜歡的網頁風格並截圖提供給 AI 參考，或是直接下指令要求特定風格（如「霓虹風格 (Neon style)」）。
- **佈局規劃**：建議畫出 **Layout 圖**（手繪或用投影片製作均可），並指定顏色與區塊位置，因為純文字描述有時難以精確傳達。
- **動態行為描述**：必須清楚向 AI 說明網頁的**互動邏輯**。例如：點擊按鈕後畫面如何切換、是否需要彈出確認視窗等，避免讓 AI 自行「腦補」。

3. 繳交與呈現

- **README 檔案**：作業必須包含一個 README 檔案。雖然 AI 可以協助撰寫 70%-80% 的內容，但學生應親自修改，以解釋網頁的**設計理念與內容**。
- **版本控制**：需將程式碼上傳。助教會將學生的程式碼下載到 **localhost (本地環境)** 進行測試與批改。

4. 評分標準與同儕互評

- **用心程度為主**：評分標準建議為「**80% 看用心程度，20% 看美觀**」。重點在於是否展現了設計概念，而非單純的視覺美醸。
- **尊重多元風格**：作業採用同儕互評機制。老師強調應尊重同學的設計理念，即使風格過於「前衛」也不應因此給予低分。
- **分數分布**：表現優異者可得 6 分，否則一般以滿分（通常為 5 分）計。老師最後會根據全班平均進行線性調整（通常往上調）。

比喻理解：這項作業就像是**打造一間數位樣品屋**。你不需要親手鋸木頭或拉電線（手寫底層程式碼），而是擔任**總設計師**的角色。你提供參考圖（截圖）與設計圖（Layout），並明確告訴你的全能助手（AI）哪扇門要怎麼開（互動邏輯）。最後，你還需要寫一份導覽手冊（README），向訪客（助教與同學）介紹這間房子的設計精髓。

如何妥善管理與保護程式碼中的 API 金鑰？

課程中提及 Cursor 介面包含哪三大主要區域？

學生製作個人網頁作業時，應如何運用 AI 工具？

Discuss 同儕評分機制。

在網頁程式設計課程中，**同儕評分機制（Peer Review）** 是作業評核的重要環節，旨在透過同學間的互相檢視來促進學習。根據來源，這套機制的運作核心與規範如下：

1. 評分的核心標準

同儕評分並非單純比較網頁的美醜，其權重分配有明確的建議：

- **用心程度為主（80%）**：評分者應觀察同學在作業中投入的精力與創意。
- **視覺美觀為輔（20%）**：雖然極端不美觀的情況下可以酌情扣分，但這不應是評分的主要基準。
- **尊重多元風格**：老師強調評分時必須**尊重每位同學的設計理念**。即便某個設計風格顯得過於「前衛」或不符合個人審美，也不應因此給予極低分或零分。

2. 分數計算與彈性

- **分數區間**：在一般的同儕評分中，**5 分即代表滿分**。
- **優異加分**：如果你認為同學的作品真的非常出色，可以給予**6 分**的高標評價。
- **誤差處理**：由於每位評分者的主觀感受不同，系統會透過多人評分（例如三人）來平均誤差。

3. README 的輔助作用

為了讓評分者能更準確地理解作品，README 檔案扮演了溝通橋樑的角色：

- **設計理念說明**：同學應在 README 中解釋網頁的設計概念與內容，這有助於評分者了解作者的用心之處。
- **AI 協助撰寫**：雖然 AI 可以協助產生大部分的 README 內容，但學生仍需親自修整，以確保能真實呈現自己的想法。

4. 後續的分數調整

老師針對最終成績有一套配套措施，以確保公平性：

- **線性調整**：由於期末評分可能較為嚴格，老師最後會對學期總成績進行**線性調整（通常是往上調）**。
- **平均基準**：例如，若原始平均分為 75 分，老師可能會將其整體調整至 80 分以上，因此同學不需過度擔心同儕評分過程中的主觀誤差會嚴重打擊最終成績。

目前，老師與助教仍在持續細化這套機制的具體設計細節，以確保其運作順暢。

比喻理解：同儕評分就像是一場創意市集的互評活動。每位攤主（學生）都展示自己的產品，而評分者（同學）不應該只因為不喜歡某種顏色就給差評，而是要觀察這件產品的工法是否紮實、創意是否獨特。最後，老師則像是一位大會總監，會根據當天整體的氣氛與水準，統一微調大家的成績，確保每位用心的攤主都能得到合理的報酬。

助教在批改作業時，會如何處理開發環境與金鑰？

課程錄影會上傳到哪些平台供學生觀看？

第一項作業的內容要求與使用的技術為何？

Discuss 網頁開發語言。

在網頁程式設計中，**HTML、CSS 與 JavaScript** 被視為建構網頁服務最核心的三種語言。隨著技術演進，現代開發流程也納入了 **TypeScript** 以增加程式的嚴謹性，並使用 **Markdown** 進行專案文件管理。

以下是根據來源對這些開發語言的詳細討論：

1. HTML：網頁的結構與內容

HTML (HyperText Markup Language) 是網頁的基礎，負責描述網頁的**結構 (Structure)**。

- **組成元素**：HTML 由各式各樣的「標籤」(Tags) 組成，通常成對出現（包含開始與結束標籤），並可帶有屬性 (Attributes)。
- **層次結構**：一份標準文件包含 `<head>` (描述背景資訊，如標題、Metadata) 與 `<body>` (網頁主要顯示內容)。
- **常用標籤**：包括標題 (H1-H6)、段落 (P)、超連結 (A)、圖片 (IMG)、清單 (UL/OL) 與表格 (Table)。
- **排版特性**：瀏覽器會根據標籤層次自動由上而下、由左而右排版，開發者不應試圖透過原始碼中的空白來控制間距。

2. CSS：網頁的樣式與美化

CSS (Cascading Style Sheets) 負責控制網頁的**樣式 (Style)** 與外觀。

- **視覺控制**：用於定義顏色、圓角弧度 (Radius)、元件寬度以及佈局排版。
- **進階技術**：現代開發常使用 **Flexbox** 這種佈局模式來輔助排版。
- **AI 協助**：利用 AI 工具（如 Cursor）可以快速生成上百行的 CSS 程式碼，實現複雜的視覺效果（如霓虹風格）。

3. JavaScript：動態互動與邏輯

JavaScript (JS) 是讓網頁具備**動態功能**的關鍵語言。

- **DOM 操作**：JavaScript 透過 **DOM (Document Object Model)** 介面來動態改變網頁內容。開發者可以利用特定方法（如 `innerHTML` 或 `innerText`）即時修改網頁元件，而不需要重新整理整個頁面。
- **事件監聽**：它會「監聽」使用者的行為（如滑鼠點擊、鍵盤輸入），並做出相對應的反應。
- **應用實例**：用於開發如待辦事項（Todo App）或網頁遊戲（如五子棋、小蜜蜂）的互動邏輯。

4. TypeScript：更嚴謹的開發標準

TypeScript (TS) 是 JavaScript 的超集，也是本課程作業指定的語言。

- **開發流程**：在 AI 輔助下，開發者可以先請 AI 用 JavaScript 撰寫邏輯，再「一鍵轉換」為 TypeScript，這在現代 Web 開發中是非常常見的實務做法。

5. Markdown：文件說明語言

在 GitHub 等專案平台上，**README.md**（Markdown 格式）是標準的說明文件。

- **功能**：用於撰寫技術簡介、安裝流程、使用說明以及個人網頁的設計理念。
- **重要性**：在同儕互評中，README 的內容是展現開發者「用心程度」的重要依據。

比喻理解：網頁開發語言的關係就像**建造一棟智慧型大樓**。**HTML** 是大樓的鋼筋水泥與隔間（結構）；**CSS** 是外牆的磁磚、室內裝潢與燈光（風格）；**JavaScript** 則是電梯感應器與自動化系統（互動邏輯）。而 **TypeScript** 就像是更為精確的工程藍圖，確保所有的管線配置都符合嚴格的規格。

為什麼環境設定檔（.env）不應上傳至 GitHub？

根據講義，動態網頁與靜態網頁的主要差異為何？

瀏覽器為何預設禁止存取本地硬碟的資源？

Discuss 前端與後端。

在 Web 服務（Web Service）的開發中，「前端」與「後端」是建構完整服務的兩大核心架構，它們分別承擔不同的職責並協同運作。根據來源，以下是關於兩者的詳細討論：

1. 前端（Front-end / Client-side）

前端又被稱為「**客戶端**」，是使用者直接接觸並進行互動的介面。

- **角色與功能**：前端負責利用 HTML、CSS 與 JavaScript 等語言，在瀏覽器上「渲染」出畫面，並透過監聽（Listen）滑鼠或鍵盤的輸入事件，對使用者的行為做出即時反應。

• 技術組成：

- **HTML** 定義結構。
- **CSS** 控制樣式。
- **JavaScript** 處理動態邏輯。

- **DOM (Document Object Model)** 是前端與網頁互動的基礎介面，讓程式能動態改變網頁內容。
- **本地特性**：在前端進行的修改（例如透過瀏覽器開發者工具 hack 網頁內容）僅存在於使用者的本地機器中。一旦重新整理（Reload），網頁會重新從伺服器抓取原始程式碼，導致本地的暫時性修改消失。

2. 後端 (Back-end / Server-side)

後端又被稱為「**伺服器端**」，是在使用者看不見的幕後運作的服務。

- **角色與功能**：後端主要負責執行**消耗資源的運算**以及**資料的儲存與管理**。當前端提出要求時，後端處理完畢後會將結果回傳給前端。
- **技術彈性**：與前端主要依賴瀏覽器能解讀的語言不同，後端程式不限語言，只要能透過 HTTP 等通訊協定與前端溝通即可。
- **資料庫 (Database)**：這是後端的重要機制，像是一個專門存放龐大或機密資料的「倉庫」。資料庫具備高效的存取效能、安全性，且能確保資料不會因為伺服器關機而流失。

3. 前後端的互動與動態網頁

現代網頁多屬於「**動態網頁**」。在這種模式下，前端 JavaScript 會根據特定事件，向後端要求「部分」資料，並僅針對網頁的特定區塊進行更新，而不需要重新讀取整個頁面，這大大提升了操作的流暢度。

4. 安全性限制

瀏覽器（前端）基於安全考量，通常**不允許存取使用者的本地硬碟資源**。這種限制（與 CORS 跨資源共享概念相關）是為了防止惡意網頁程式碼在使用者瀏覽時偷取或修改本地端磁碟的檔案。

比喻理解：前端與後端的關係就像是一家餐廳。前端是「用餐區與菜單」，客人（使用者）在這裡看裝潢、點菜並與服務生互動；後端則是「廚房與倉庫」。廚房負責繁重的烹飪（運算），而倉庫存放著所有食材（資料庫）。客人不需要看到廚房如何運作，只需透過服務生（通訊協定）點餐，廚房處理完畢後，料理就會被送到客人面前（回傳資料）。

為什麼不應將 API 金鑰直接寫死在程式碼中？

根據講義，DOM 這種資料結構的邏輯特性為何？

瀏覽器無法直接讀取本地硬碟檔案的原因為何？

Discuss 瀏覽器歷史.

根據來源內容，瀏覽器的發展史是一段從文字介面轉向圖形介面，並伴隨著多次市場大戰與法律訴訟的動態過程。以下是瀏覽器歷史的核心轉折點：

1. 圖形介面的革命：Mosaic 與 Netscape
- **早期階段**：最早的網路介面主要是**文字介面**，視覺上較為簡陋。
 - **Mosaic (1992年)**：由當時在 UIUC 咸書的學生 Marc Andreessen 開發。他因覺得文字瀏覽器太醜，利用寒假期間製作了具備**圖形介面**的 Mosaic 瀏覽器，這成為了現代瀏覽器的雛形。

- **Netscape 的崛起**：隨後 Marc Andreessen 與研究機構合作成立了 **Netscape** 公司。在 90 年代末期，Netscape 曾是網路界的巨人，以瀏覽器產品稱霸市場。

2. 第一次瀏覽器大戰：Microsoft IE 的介入

- **微軟的轉向**：最初微軟（Microsoft）認為網路並不重要，專注於作業系統與 Office。但隨著網路熱潮興起，微軟推出了 **Internet Explorer (IE)**。
- **綁定策略與壟斷爭議**：微軟利用 Windows 作業系統的優勢，將 IE **預設綁定在系統中**。由於 Windows 是當時最普及的系統，這項策略讓 IE 迅速崛起並打敗了 Netscape。
- **反托拉斯法訴訟**：此舉導致微軟被美國政府控告違反「反托拉斯法」（壟斷），微軟為此支付了巨額賠償。

3. 多元化發展與 Firefox 的誕生

- **Netscape 的沒落與開源**：受到 IE 打擊，Netscape 公司逐漸沒落，後來決定將瀏覽器程式碼**開源**，這成為了後來 **Firefox (火狐)** 瀏覽器以及 Mozilla 機構的起源。
- **其他競爭者**：此時期也出現了如 Opera 以及 Apple 推出的 Safari 等瀏覽器。

4. 現代霸主：Google Chrome 的出現

- **Chrome (2008年)**：Google 較晚才進入市場，但 Chrome 推出後迅速獲得極高的市佔率。
- **當前局勢**：Chrome 由於目前市佔率過高，同樣也面臨了美國政府的反托拉斯法訴訟。而微軟在 IE 被嘲笑多年後，推出了評價較好的 **Edge** 瀏覽器。

5. 瀏覽器的角色定義

從歷史演進來看，瀏覽器本質上是一個**用來解讀指定資源（網址）的程式**。它負責抓取伺服器傳來的 HTML、CSS 與 JavaScript 檔案，並將其「渲染」（Render）成我們看到的畫面，同時監聽使用者的互動事件。

比喻理解：瀏覽器的歷史就像是一場數位世界的交通工具演進史。最早我們只能在泥濘的小徑上步行（文字介面），隨後 **Mosaic** 打造了第一台有擋風玻璃的汽車（圖形介面）。**Netscape** 曾是最大的車商，但微軟直接在每條新蓋的馬路（Windows 系統）旁免費停放一台 **IE**，逼得 Netscape 退出市場。直到後來 **Chrome** 憑藉更強大的引擎（運算效能）出現，才再次改變了大家的駕駛習慣。

如何利用環境設定檔（.env）保護 API 金鑰？

試述動態網頁與靜態網頁在更新機制上的差異。

簡述文件物件模型（DOM）的階層式資料結構概念。

Discuss 動態網頁技術。

在現代網頁開發中，**動態網頁技術（Dynamic Web Technology）** 是提升使用者體驗與功能性的核心。根據來源，這項技術讓網頁不再只是靜止的資訊展示，而能與使用者進行即時互動。

以下是動態網頁技術的關鍵討論點：

1. 靜態網頁與動態網頁的分野

- **靜態網頁 (Static Webpage)**：早期網頁在更新內容時，必須從伺服器重新抓取整個 HTML 檔案，導致網頁在載入過程中會出現「刷白」現象。
- **動態網頁 (Dynamic Webpage)**：現代網頁透過在瀏覽器執行的程式（如 JavaScript）監聽（Listen）使用者的行為事件（Events），並僅針對網頁的**特定部分進行更新**，無需重新讀取整個頁面，操作更加流暢。

2. 動態網頁的核心：DOM (Document Object Model)

要實現網頁的動態改變，必須理解 **DOM** 這個關鍵概念。

- **樹狀結構 (Tree Structure)**：瀏覽器會將 HTML 文件解析成一個名為 DOM 的邏輯樹。這個樹從根節點（Root）開始，每一層分支代表不同的網頁元件，最終到達葉子節點（如文字或圖片），。
- **程式介面**：DOM 提供了一種介面，讓 JavaScript 能夠動態地管理與修改網頁內容，。當開發者透過程式碼改變這棵樹的節點（Node）時，網頁看到的樣子也會隨之改變。

3. JavaScript 的操作實務

JavaScript 是驅動動態網頁的靈魂，它能直接操作 DOM 來實現各種效果：

- **內容修改**：透過特定的方法（如 `innerHTML` 或 `innerText`），開發者可以即時替換網頁上的文字或圖片，。
- **即時互動**：來源中示範了如何在不更動伺服器檔案的情況下，透過瀏覽器的控制台（Console）直接修改台大校長網頁的顯示內容，這證明了網頁的動態展現是在**前端（客戶端）**執行的，。
- **複雜應用**：動態技術不僅能用於簡單的內容替換，還能用來開發如「小蜜蜂」或「五子棋」等具備音效與即時反饋的網頁遊戲。

4. 前端與後端的協作

雖然前端（瀏覽器）能處理許多動態邏輯，但完整的服務通常需要後端支援：

- **部分資料抓取**：動態網頁會向**後端（伺服器端）**要求所需的「部分」資料，後端處理完複雜的運算或從資料庫存取資料後回傳，前端再據此更新畫面，。
- **純前端限制**：若為「純前端」應用（如作業中的 Todo App），所有動態資料僅存在於瀏覽器記憶體中，一旦重新整理（Reload），網頁就會回到初始狀態，。

5. 安全性與存取限制

動態網頁技術在瀏覽器端受到嚴格的安全限制。例如，瀏覽器基於安全考量（如跨資源共享限制 CORS），通常**不允許動態網頁程式存取使用者的本地磁碟資源**，以防止惡意網頁讀取或竄改使用者電腦中的檔案。

比喻理解：動態網頁技術就像是一塊可以隨時更換內容的智慧型白板。靜態網頁像是印好的考卷，要改一個字就得整張換掉；而動態網頁則是由 JavaScript 擔任「執筆者」，它監控著你的動作（事件），根據你的要求擦掉白板上的某個區塊並重新畫上新內容（修改 DOM 樹），而不需要把整塊白板搬走換新的。

助教批改作業時，如何處理程式碼中的 API 金鑰？

為什麼瀏覽器預設禁止存取本地硬碟的資源？

什麼是 DOM？它在網頁運作中扮演什麼角色？

Discuss 文件物件模型 (DOM).

文件物件模型 (Document Object Model，簡稱 DOM) 是網頁開發中極其核心的基礎概念，它讓網頁從靜態的資訊展示轉變為可互動的動態服務。以下根據來源詳細討論 DOM 的定義、結構與運作方式：

1. DOM 的定義與核心功能

- **程式介面**：DOM 是針對網頁文件的 **程式介面 (Programming Interface)**，它將 HTML 文件的結構化內容轉譯為程式可以操作的物件。
- **動態基礎**：它是讓開發者能夠**動態改變網頁內容**的基本手段。透過 JavaScript 呼叫 DOM 提供的各種方法 (Methods)，可以即時指揮瀏覽器修改頁面的樣式或內容。

2. 邏輯樹狀結構 (Logical Tree)

DOM 最重要的特性在於它將網頁表示為一個**邏輯樹狀結構**：

- **階層化**：就像電腦裡的檔案目錄或是自然界的樹木，DOM 有一個**樹根 (Root)**，從中長出**樹枝 (Branches)**，最後終止於**葉子 (Leaves)**。
- **節點 (Node) 的概念**：網頁上的每一個元素，無論是按鈕、標題、一段文字還是一張圖片，在 DOM 結構中都是一個**節點 (Node)**。
- **層次排佈**：在 DOM 樹的最頂層是 `document` 物件，接著依序向下展開為 `HTML`、`head` 與 `body` 等層級。

3. 動態更動與 JavaScript 的協作

網頁之所以能「動起來」，是因為程式碼動態地管理了 DOM：

- **監聽與反應**：瀏覽器會渲染出畫面，並「監聽」使用者的互動事件（如滑鼠點擊），隨後透過 JavaScript 修 改對應的 DOM 節點來更新頁面，。
- **常用操作方法**：開發者可以使用如 `innerHTML` 或 `innerText` 等方法來替換節點內容，。例如，在不改動伺服器原始檔的情況下，可以透過瀏覽器的開發者工具（Console）直接把網頁上的圖片或姓名替換掉，。
- **局部更新**：現代動態網頁技術讓程式只需更新 DOM 樹的某個特定部分，而不需要重新讀取整份 HTML 文 件，這避免了早期網頁跳轉時常見的「畫面刷白」現象。

4. 執行環境與安全性限制

- **客戶端執行 (Client-side)**：對 DOM 的所有即時修改都是在**使用者本地的瀏覽器 (前端)** 執行的，。如果重新整理網頁（Reload），瀏覽器會重新從伺服器抓取原始的程式碼，先前的動態修改就會消失，。
- **安全隔離**：基於安全考量（與 CORS 跨資源共享限制相關），瀏覽器通常**不允許**網頁上的 JavaScript 透過 DOM 操作去存取使用者的本地硬碟資源，以防惡意程式竄改本地檔案。

比喻理解：DOM 就像是一座智慧型樂高大樓的藍圖與接孔。HTML 是大樓已經蓋好的初始結構，而 DOM 則提供了這棟大樓的「動態控制面板」。JavaScript 是操作員，他不需要拆掉整棟大樓重蓋（重新整理網頁），只需透過控制面板（DOM Methods），就能精確地更換某個房間的窗戶（修改圖片節點）或改變招牌上的文字（修改文字節點）。

為什麼上傳專案到 GitHub 時，不應包含 `.env` 檔？

課程作業要求使用哪種程式語言來撰寫網頁邏輯？

瀏覽器為何預設禁止網頁程式存取本地硬碟資源？

Discuss JavaScript 語言特性。

JavaScript 是一門極其強大且主宰現代網頁生態的語言，它是實現前端動態互動最核心的工具。根據來源，JavaScript 的語言特性可以從其發展歷史、類型系統、物件模型及函數設計等面向來深入討論：

1. 弱型別與高度容錯性

- **弱型別（Weakly Typed）**：JavaScript 在不同型別之間的轉換非常「積極」且自動化。例如，將數字與布林值相加時，它會自動將其轉換為字串而不報錯。
- **錯誤容忍度高**：為了確保網頁瀏覽不會因為一點小錯誤就導致整個瀏覽器當掉或當機，JavaScript 被設計成能高度容忍錯誤。雖然這有利於網頁穩定，但有時會讓錯誤隱藏在程式碼深處，增加除錯（Debug）的難度。

2. 變數宣告與作用域

- **宣告方式**：早期使用 `var`，但現代 JavaScript（ES6 之後）強烈建議使用 `let`（一般變數）與 `const`（唯讀變數/常數）。
- **const 的保護作用**：使用 `const` 宣告的變數是 **Read-only**（唯讀）的，這能保護程式碼不被自己或團隊成員意外修改，並在編譯階段（Compile time）就抓出錯誤。
- **抬升（Hoisting）**：JavaScript 允許**先使用再宣告**變數或函數，這與許多嚴謹的程式語言不同。

3. 以物件為核心的模型

- **萬物皆物件**：除了五種內建的基本型別（數字、布林、字串、null、undefined）外，**其餘所有東西都是物件**，包括陣列與函數。
- **原型基礎（Prototype-based）**：JavaScript 不需要先定義類別（Class）即可產生物件，且物件產生後仍可動態地增加新的屬性或方法。
- **傳址（Copy by Reference）**：當物件進行賦值時，它是透過 **Reference**（引用/指標）的方式運作。這意味著兩個不同的變數名稱可能指向同一個實體物件，修改其中一個會影響另一個。

4. 函數與現代語法

- **一等公民函數（First-class Function）**：函數被視為一般物件，可以像變數一樣被傳遞、賦值或當作參數傳給另一個函數。

- **箭頭函數 (Arrow Function)**：現代開發建議使用箭頭函數（例如 `() => {}`），它能讓語法更簡潔，並避免傳統函數中常見的 **Scope** 或 **This** 綁定問題。
- **回呼函數 (Callback Function)**：常用於處理事件監聽（Event Listener）或陣列操作（如 `map`, `filter`, `reduce`），讓程式能在特定事件發生時執行對應動作。

5. 其他獨特細節

- **數值處理**：JavaScript 沒有整數概念，所有數字都是浮點數。這導致了精確度限制，例如 `0.1 + 0.2` 並不精確等於 `0.3`。
- **相等性判斷**：建議使用 `===""`（嚴格相等）而非 `==`，因為後者會進行令人困惑的自動型別轉換，導致邏輯錯誤。
- **非同步處理**：包含 `Promise` 與 `Async/Await` 等機制，用於處理網頁中不需同步等待的操作（如資料抓取）。

比喻理解： JavaScript 就像是一個性格隨和且反應靈敏的翻譯官。他為了讓對話（網頁運行）能持續下去，即便你說錯幾個字或用錯文法，他也會試著「腦補」並幫你圓場（弱型別與容錯）。他身上帶了很多標籤貼紙（物件與屬性），可以隨時在對話中幫任何東西貼上新標籤（動態增加屬性）。然而，正因為他太過隨和，有時他幫你腦補的內容可能不是你原本的意思，所以你得用更精確的合約（如 TypeScript 或 `const`）來規範他。

網頁開發中 HTML、JavaScript 與 CSS 各自的角色為何？

為什麼 JavaScript 被設計成一種「弱型別」語言？

在開發過程中，使用 `const` 宣告變數有何優點？

Discuss TypeScript 型別安全。

在現代網頁開發中，**TypeScript (TS)** 的崛起主要是為了解決 JavaScript 在型別處理上的先天缺陷。根據來源，TypeScript 被視為一種「具備型別的 JavaScript」，其核心價值在於提供**型別安全 (Type Safety)** 與開發時的保護機制。

以下是關於 TypeScript 型別安全的詳細討論：

1. 弱型別與強型別的對比

- **JavaScript 的弱型別問題**：JavaScript 為了高度容忍錯誤、避免網頁輕易掛掉，會進行「雞婆的型別轉換」。例如，將數字與字串相加時，它會自動轉換型別而不報錯，這導致錯誤可能在數萬行程式碼之後才爆發，極難除錯（Debug），。
- **TypeScript 的強型別保護**：TypeScript 要求開發者嚴格遵守型別規範，「該是什麼型別就是什麼型別」。它像是在 JavaScript 外面套了一層殼，確保資料流動時的型別一致性。

2. 編譯階段 (Compile-time) 與執行階段 (Runtime)

型別安全最大的好處是將錯誤攔截在開發初期：

- **及早發現錯誤**：在 JavaScript 中，許多錯誤要到程式「執行時」點擊了按鈕才發現。而 TypeScript 會在編譯階段就進行檢查，如果型別亂搞（例如預期傳入數字卻傳了字串），編譯就不會通過，。
- **防止錯誤發布**：這種機制確保了開發者不會將帶有明顯型別錯誤的程式碼發布（Publish）到實際的服務中，大幅提升了服務的**強固性**。

3. 可維護性與團隊協作

- **維護成本**：隨著應用程式變得複雜，型別安全變得必不可少。雖然寫 TypeScript 初始時會覺得比 JavaScript 麻煩，但它能顯著提升程式碼的**可維護性**，。
- **保護開發者**：透過明確的型別宣告（如使用 `const` 告訴唯讀變數），可以防止自己或團隊成員意外修改資料，這是一種保護開發假設不被破壞的措施，。

4. 實務應用技巧

根據課程建議，初學者在面對 TypeScript 的型別要求時，可以善用 AI 工具：

- **自動轉換**：開發者可以先使用 JavaScript 撰寫功能邏輯，再利用 AI 工具（如 Cursor）進行「一鍵轉換」，將其改寫為符合 TypeScript 規範的程式碼，。
- **作業規範**：課程中的個人網頁作業明確要求使用 TypeScript，目的就是讓學生習慣這種更嚴謹、更現代的開發標準。

比喻理解：型別安全就像是電器產品的插頭設計。JavaScript 像是萬用插座，什麼形狀的插頭（資料）都試著讓它通電，雖然方便但容易導致短路火災；而 TypeScript 則為每一種插座設計了專屬形狀，如果你拿錯了插頭，在插進去之前（編譯階段）就會發現合不起來，這雖然增加了一點麻煩，卻能保證整棟大樓（網頁服務）的電力系統安全無虞。

為什麼建議將 Script 標籤放在 Body 的最下方？

在 JavaScript 中，使用三個等號與兩個等號有何差異？

請解釋 CSS 中 Padding 與 Margin 的主要區別。

Discuss DOM 事件處理。

在網頁開發中，**DOM 事件處理（Event Handling）** 是實現網頁互動的核心機制。它讓網頁能夠「聽取」使用者的動作並做出相對應的反應，從而將靜態的頁面轉化為動態的服務。

以下是根據來源整理的 DOM 事件處理核心概念與實作技巧：

1. 事件監聽的基本原理

- **事件（Events）**：指使用者透過輸入裝置（如滑鼠點擊、鍵盤按壓）與網頁產生的互動訊號。
- **事件監聽器（Event Listener）**：JavaScript 利用 `addEventListener` 方法將特定的事件「綁定」在 DOM 元件（Node）上。

- **綁定步驟**：通常會先透過 `document.getElementById` 取得元件的引用（Reference），再對該變數設定監聽器。例如，可以在 `document` 上監聽 `DOMContentLoaded` 事件，確保網頁內容載入完成後才執行初始化程式碼。

2. 常見的事件類型

根據不同的互動需求，開發者可以監聽多種事件：

- **鍵盤事件**：如 `keypress` 或 `keydown`。常見的應用是在輸入框監聽使用者是否按下 **Enter 鍵**（透過 `event.key === 'Enter'` 判斷），藉此觸發送出功能。
- **滑鼠事件**：包括 `click`（點擊）、`mouseenter`（滑鼠移入）及 `mouseleave`（滑鼠移出）。
- **狀態與表單事件**：如 `focus`（進入輸入框）、`blur`（離開輸入框）或 `change`（如勾選框狀態改變）。

3. 回呼函數（Callback Function）的應用

- **匿名函數與箭頭函數**：事件發生後要執行的動作通常封裝在回呼函數中，來源建議使用 **箭頭函數（Arrow Function）**，以避免傳統函數在作用域（Scope）或 `this` 綁定上的困擾。
- **常見錯誤預防**：在綁定事件時，應傳遞函數本身而非函數的執行結果。若寫成 `addEventListener('click', myFunction())`，會導致事件尚未發生就先執行了函數並綁定其回傳值；正確的做法是使用箭頭函數包裝，如 `() => myFunction(id)`。

4. 實現動態更新

當事件被觸發時，程式會透過操作 DOM 來改變網頁狀態：

- **局部更新（Rendering）**：在事件處理過程中，開發者可以修改元件的 `innerHTML` 或 `innerText`，或是在陣列中新增資料後重新「渲染」（Render）畫面。
- **效能優化**：常見的作法是在更新資料後，先將該區域的內容清空，再根據更新後的資料重新繪製，這讓網頁無需重新整理（Reload）即可展現新內容。

比喻理解：DOM 事件處理就像是為大樓安裝「感應自動門」。HTML 標籤是大樓的結構（門），JavaScript 是控制邏輯，而** `addEventListener` 就是那個感應器**。當有人走近（觸發事件）時，感應器會通知控制中心執行「開門」的動作（回呼函數）。如果沒有這個感應機制，門就只是一塊動不了的木板，網頁也就失去了與人對話的能力。

網頁開發中 HTML、JavaScript 與 CSS 各自的角色為何？

試述靜態網頁與動態網頁在處理資料更新時的差異。

撰寫 AI Prompt 時，為何需明確描述 UI 的細節？

Discuss CSS 樣式排版。

在網頁開發中，**CSS (Cascading Style Sheets)** 負責定義網頁的**外觀與排版**。與負責結構的 HTML 不同，CSS 決定了元件的顏色、大小、位置以及各種視覺特效。

以下根據來源詳細討論 CSS 樣式排版的關鍵概念：

1. CSS 語法與選擇器 (Selectors)

CSS 的核心運作邏輯是透過「選擇器」抓取 HTML 元件，並為其套用指定的「樣式規則」。

• 基本選擇器：

- **萬用選擇器 (***)****：套用到頁面上所有元件。
 - **標籤選擇器 (Tag)**：直接針對特定的 HTML 標籤（如 `body`, `p`, `div`）進行定義。
 - **類別選擇器 (Class, .)**：最常用的方式，透過元件的 `class` 屬性來抓取，例如 `.container`。
 - **ID 選擇器 (**#**)**：針對具有唯一 `id` 屬性的元件進行樣式設定。
- **偽類 (Pseudo-classes)**：用於定義元件在特定行為下的樣式。常見的包括 `:hover`（滑鼠移過）、`:active`（點擊中）以及 `:focus`（取得焦點，如點擊輸入框時邊框變色）。
 - **優先權順序**：後寫的規則通常會覆蓋先寫的規則，且**越精確的選擇器（如 ID 選擇器）優先權高於較籠統的選擇器（如標籤選擇器）**。

2. 盒模型 (Box Model)

這是理解 CSS 排版的基礎，每個網頁元件都可以被視為一個方框：

- **內容 (Content)**：文字或圖片本身。
- **內邊距 (Padding)**：內容與邊框之間的距離。
- **邊框 (Border)**：包圍在內容外的線條。可以透過 `border-radius` 設定圓角，當數值達到 **50%** 時可將元件變為圓形。
- **外邊距 (Margin)**：元件與元件之間的間距。
- **寬度計算**：透過 `box-sizing` 屬性，可以指定寬度是指內容本身 (`content-box`) 還是包含邊框 (`border-box`)，這對精確排版至關重要。

3. 排版邏輯與佈局技術

瀏覽器預設會由上而下、由左而右排列元件。為了達成更複雜的佈局，開發者會使用以下技術：

- **Flexbox (彈性盒佈局)**：這是現代網頁最主流的排版引擎。它允許開發者輕鬆設定元件的**對齊方式（如居中）、排列方向（橫向或縱向）以及如何分配剩餘空間**。
- **容器技術 (Container)**：實務上常會在多個元件外包覆一個「空的殼子」（即 Container），並對這個容器設定 Flexbox 規則。例如，若想讓三行文字同時居中，應對包覆它們的容器下指令，而非個別對文字下指令。
- **響應式設計 (Responsive)**：調整樣式使網頁在不同螢幕尺寸下（如手機或桌機）都不會跑版。

4. 實務應用與 AI 輔助

在現代開發流程中，CSS 檔案通常獨立存在，並在 HTML 的 `<head>` 區塊中連結引用。

- **AI 自動化**：利用 AI 工具（如 Cursor），開發者只需提供設計理念（如「霓虹風格」或截圖），AI 就能快速生成數百行 CSS 程式碼。
 - **動態操作**：雖然 CSS 定義初始樣式，但 JavaScript 可以動態修改元件的類別名稱，進而即時切換 CSS 樣式（如點擊按鈕後改變背景顏色）。
-

比喻理解：CSS 樣式排版就像是室內裝潢與擺設。HTML 提供了房屋的毛胚結構（牆壁、門框），而 CSS 則是壁紙顏色（背景）、家具與牆壁的距離（Margin/Padding）、家具本身的尺寸（寬度/高度），以及如何在大廳中整齊排列沙發與茶几（Flexbox）。當你希望客人進門時（滑鼠移過）燈光自動變亮，那就是 CSS 偽類的作用。

根據老師經驗，台大課程出席率通常難以超過多少？

Web Programming 作業一的主要開發目標與技術要求為何？

什麼是 DOM？其邏輯結構類似於哪種資料結構？

Discuss AI 輔助開發技巧。

在現代網頁開發中，**AI 輔助開發（AI-Assisted Development）** 已成為提升效率與創造力的核心技巧。根據來源，這種開發模式不再強調死記硬背語法，而是專注於與 AI 的溝通協作與架構理解。以下是來源中提到的關鍵技巧與實務：

1. 核心工具與介面操作：以 Cursor 為例

來源推薦使用 **Cursor** 作為主要的開發環境，其介面整合了 AI 聊天的功能。

• **介面組成**：包含檔案瀏覽器、程式碼編輯區，以及最關鍵的 **AI 聊天室**，讓開發者能直接與 AI 對話生成程式碼。

• **模型選擇與模式**：

◦ **Auto 模式**：適合處理大多數基礎工作，AI 會自動挑選適合的模型。

◦ **Composer (Maxmode)**：具備更強的推理能力，能自動產生測試、檢查錯誤並自我修復，適合處理更複雜的邏輯，但消耗的 Token（點數）較多。

• **上下文感知**：直接在編輯器內問 AI 技術問題非常有效，因為 AI 已掌握你的程式碼脈絡，能提供最精確的解答。

2. 多模態提示詞（Prompting）技巧

與 AI 溝通不僅限於文字，**圖形與影像是極為強大的輸入方式**：

• **以圖生 Code**：開發者可以將手繪草圖、簡報截圖甚至其他網頁的截圖餵給 AI，讓它根據視覺畫面直接生成 HTML、CSS 與 JavaScript 程式碼。

• **動態規格說明**：單靠圖片可能會有「腦補」偏差，因此需搭配文字描述動態行為（例如：點擊按鈕後隱藏某區塊、清單過長時不要跟著捲動）。

- **影片引導**：甚至可以提供遊戲影片（如小蜜蜂），讓 AI 根據影片內容還原遊戲邏輯。

3. 開發流程：從 One-shot 到 迭代優化

- **一鍵生成（One-shot）**：對於簡單的應用（如待辦事項、五子棋），AI 往往能一次就寫出 80% 以上正確的程式碼。

- **迭代修正（Iteration）**：若 AI 產出的結果與預期有差距（例如按鈕位置跑掉或 UI 太醜），開發者應透過對話持續微調，甚至「打掉重練」給予更清晰的引導。

- **型別轉換**：可以先用 JavaScript 寫出功能，再請 AI 一鍵轉換為更嚴謹的 **TypeScript** 以符合現代工程標準。

4. 高階策略與最佳實務

- **抽象化與模組化**：面對複雜專案，應將大功能分割成小模組，並學習如何用簡潔的邏輯（抽象化）向 AI 表達需求。

- **別讓 Prompt 成為天花板**：來源觀察到，有時給予過於精確、死板的限制反而會限制 AI 的發揮。先讓 AI 自由發揮（**Freely play**），往往能得到比預期更完善的功能，事後再針對不滿意之處修改即可。

- **自動化資產生成**：除了程式碼，AI 也能協助產生**README 說明文件**，甚至是利用程式邏輯產生無版權的音效。

比喻理解：AI 幫助開發就像是聘請了一位極速打字且博學的「實習工程師」。你不需要親自去搬每一塊磚頭（寫每一行標籤），你更像是一位**專案經理或建築師**，你的職責是畫好設計圖（截圖）並下達清晰的指令（Prompt）。雖然他偶爾會理解錯誤，但只要你懂得如何與他對話與驗收，他能幫你在海邊喝咖啡的同時，就在幾分鐘內蓋好一棟智慧大樓。

網頁開發中「前端」與「後端」的主要功能差異為何？

為什麼 JavaScript 被設計成「弱型別」的程式語言？

在 Cursor 中開啟「Max Mode」會對開發過程產生什麼影響？