# 現代前端技術與 AI 開發

FROM AJAX TO VUE & AI GENERATION

HORAZON / 互動媒體

# 網頁發展史：從靜態到動態

1. **靜態網頁 (Static)**：純 HTML/CSS，內容寫死。
2. **動態網頁 (Dynamic)**：Server 產生 HTML (PHP, ASP.NET)，換頁要重載。
3. **單頁應用 (SPA)**：前端負責跟 Server 要資料，**只有局部更新**。

**關鍵技術：AJAX**

# 什麼是 AJAX?

- **A**synchronous **J**ava**S**cript **A**nd **X**ML
- (現在大部分是傳送 **JSON** 格式，不是 XML 了)

**重點：**

網頁**不需要重新整理**，就可以跟伺服器 (Server) 拿資料並更新畫面。

> 想像你去餐廳：
> **傳統網頁**：加點飲料，服務生把整張桌子收走，再重新擺盤一次。
> **AJAX**：服務生直接把飲料端上來，原本的菜不動。

# AJAX 現代寫法：FETCH API

以前用 `XMLHttpRequest` (很難寫)，現在用 `fetch` 。

```
// 向伺服器要資料
fetch('https://api.example.com/data')
    .then(response => response.json()) // 把回應轉成 JSON
    .then(data => {
        console.log(data); // 拿到資料了！
        document.querySelector("#box").innerText = data.name;
    });
```

這就是現代網頁 (Facebook, Gmail) 順暢的原因。

# 現代前端框架 (FRONTEND FRAMEWORKS)

隨著網頁越來越複雜，純寫 JavaScript (Vanilla JS) 變得難以維護。
於是誕生了三大框架：

1. **Vue.js** (好學、靈活、台灣很紅) 🟢
2. **React** (Facebook維護、生態系最大) 🔵
3. **Angular** (Google維護、架構嚴謹) 🔴

它們的共同點：**組件化 (Component)** 與 **資料驅動 (Data-Driven)**。

# 為什麼選 VUE.JS?

- **直覺**：HTML/CSS/JS 寫在一起，很像原本的網頁開發方式。
- **雙向綁定**：資料變了，畫面自己會變 (不用一直寫 `document.querySelector`)。

```
<script setup>
import { ref } from 'vue'
const count = ref(0) // 定義變數
</script>

<template>
  <h1>目前數字：{{ count }}</h1> <!-- 自動更新 -->
  <button @click="count++">+1</button> <!-- 事件 -->
</template>
```

# 時代變了：AI 與前端開發

前端技術更新太快 (Webpack, Vite, Tailwind, TypeScript...)。
學習曲線變得很陡峭。

**但是，AI 非常擅長寫前端！**

為什麼？

1. **結構明確**：HTML 是樹狀結構，CSS 是規則集，邏輯清晰。
2. **視覺化**：描述「我要一個藍色的按鈕，滑過去會變大」，AI 聽得懂。
3. **重複性高**：很多組件 (Navbar, Card, Form) 寫法都很像。

# 如何利用 AI 做前端？

**不要手刻 CSS，不要死背語法。**

**流程 (Workflow)：**

1. **Prompt (詠唱)**：清楚描述你的需求。
    - "幫我做一個 RWD 的產品卡片，要有陰影，圖片是圓形的。"
    - "用 Vue 寫一個待辦事項清單，可以新增和刪除。"
2. **Generate (生成)**：AI 給你程式碼。
3. **Refine (微調)**：你只需要看得懂程式碼，並做小修改。

# AI 實戰：VS CODE + GITHUB COPILOT

VS Code 是目前最流行的程式碼編輯器，搭配 GitHub Copilot 更強大。

- **智慧補全 (Ghost Text)**：
  - 當你打字時，AI 會猜你想寫什麼，按 `Tab` 即可完成。
- **Copilot Chat**：
  - 直接在編輯器內跟 AI 對話：「這段程式碼在做什麼？」、「幫我修好這個 Bug」。
- **Inline Chat (Cmd/Ctrl + I)**：
  - 選取程式碼，叫 AI 直接修改：「幫我加上註解」、「改成 Vue 的寫法」。

> **未來工程師的核心能力，不是「寫 code」，而是「描述需求」與「審查 code」。**

# 結論

1. **AJAX** 讓網頁不用重整就能更新資料 (Fetch API)。
2. **Vue.js** 等框架讓開發大型網頁變簡單 (資料驅動)。
3. **AI 是最強的助手**：前端開發已經進入「AI 輔助」時代。
4. 善用工具 (**VS Code + Copilot**)，專注在**創意**與**邏輯**，髒活交給 AI。

# THE END

Happy Coding with AI!