網頁前端工程概述

網頁前端工程（Frontend Web Development）是指負責使用者在瀏覽器中所見介面的開發與實作工作。前端工程師的主要目的是將設計轉化為可互動的網頁介面，並確保良好的使用者體驗（UX）與效能。

一、前端工程的核心技術

1. HTML（超文件標記語言）

HTML 是構建網頁的基本骨架。它定義了網頁中的元素結構，例如標題、段落、圖片、連結等。

使用規則與要點說明

1. **結構性語意標籤**：使用如 `<header>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<footer>` 等語意化標籤，有助於提升 SEO 與可讀性。
2. **元素巢狀**：HTML 元素可以巢狀，但必須正確開關標籤，遵守 DOM 階層邏輯。
3. **屬性設置**：元素可以帶有多種屬性，例如 `id`, `class`, `href`, `src`, `alt`，可供 CSS 與 JavaScript 操作。
4. **文件宣告與編碼**：每份 HTML 檔案應以 `<!DOCTYPE html>` 開頭，並在 `<head>` 中設定 `charset="utf-8"` 以支援多國語言。
5. **可存取性考量**：使用 `alt` 描述圖片、適當標題階層（如 `<h1>` 到 `<h6>`）與表單 `label` 可提升無障礙設計。

範例：

```
<!DOCTYPE html>
<html>
  <head>
    <title>我的第一個網頁</title>
  </head>
  <body>
    <h1>歡迎光臨</h1>
    <p>這是一段簡單的介紹文字。</p>
  </body>
</html>
```

2. CSS（層疊樣式表）

CSS（Cascading Style Sheets）是用來描述 HTML 文件外觀與樣式的語言。它將內容與樣式分離，能讓同一份 HTML 結構在不同設備或情境下呈現不同風格。CSS 主要控制文字、顏色、背景、邊框、間距、排版與動畫等。

基本結構

一條 CSS 規則包含三個部分：選擇器、屬性與值。

```
h1 {
  color: red;
  font-size: 24px;
}
```

上述表示將所有 <h1> 元素的文字設為紅色、字體大小為 24 像素。

引用方式

CSS 可透過三種方式套用於 HTML：

1. **行內樣式（inline）**：直接寫在 HTML 元素中（不建議）

   ```
   <p style="color: green;">文字</p>
   ```

2. **內部樣式表（internal）**：寫在 <style> 標籤中，放於 HTML <head> 內。

3. **外部樣式表（external）**：使用 .css 檔案並透過 <link> 引入，為最推薦方式。

   ```
   <link rel="stylesheet" href="styles.css">
   ```

使用規則與要點說明

1. **選擇器（Selectors）**：用來指定要套用樣式的 HTML 元素，可使用元素名稱、類別 .class、ID #id、屬性或組合選擇器。

2. **繼承與層疊（Cascade）**：CSS 遵循優先順序與層疊規則。若多條樣式作用於同一元素，優先順序為：行內樣式 > ID 選擇器 > 類別選擇器 > 元素選擇器。

3. **盒模型（Box Model）**：所有 HTML 元素皆視為矩形區塊，包括 margin, border, padding, content，影響元素在頁面中的尺寸與位置。

4. **單位與顏色系統**：長度單位常見如 px, em, rem, %；顏色支援 HEX、RGB、HSL。

5. **排版與版面配置**：常用的方式包括 display, position, float, flexbox, grid，可建立彈性或響應式版面。

6. **媒體查詢（Media Query）**：用於製作響應式設計，依照螢幕尺寸或裝置特性切換樣式。

7. **CSS 變數（Custom Properties）**：可定義重複使用的值，提升樣式一致性與維護性。

   ```
   :root {
     --main-color: #3498db;
   }
   button {
   ```

```
    background-color: var(--main-color);
  }
```

8. **動畫與轉場（Animation & Transition）**：使用 `transition` 增加平滑過渡，或使用 `@keyframes` 創造動畫效果。

```
.fade-in {
  animation: fadeIn 2s ease-in;
}
@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}
```

SCSS（Sass）簡介與差異

SCSS 是 CSS 的一種語法擴充版本，屬於 Sass（Syntactically Awesome Stylesheets）預處理器的主流格式。

SCSS 與傳統 CSS 的主要差異在於：

- **語法擴充**：SCSS 支援變數（Variables）、巢狀結構（Nested Rules）、Mixin、函式、繼承（Inheritance）等功能，而 CSS 為靜態語法，需重複撰寫相同邏輯。
- **邏輯能力**：SCSS 可撰寫條件語句（@if/@else）、迴圈（@for/@each），提升程式彈性。
- **模組化與可讀性**：SCSS 支援多檔案拆分與模組匯入（@import 或 @use），有利團隊協作與大型專案維護。
- **需編譯**：SCSS 必須經由編譯工具轉譯成純 CSS 才能運作，而 CSS 可直接被瀏覽器解析。

SCSS 範例：

```
$primary-color: #e74c3c;

@mixin button-style {
  padding: 10px 20px;
  border-radius: 5px;
  color: white;
}

.button {
  background-color: $primary-color;
  @include button-style;
  &:hover {
    background-color: darken($primary-color, 10%);
  }
}
```

SCSS 需經由工具（如 Dart Sass、Webpack + sass-loader）轉譯為標準 CSS 後才能在瀏覽器中運作。

範例：

```
<style>
  h1 {
    color: blue;
    font-size: 32px;
    text-align: center;
  }
  p {
    line-height: 1.6;
    color: #333;
  }
</style>
```

3. JavaScript（JS）

JavaScript 是用來實作網頁互動性的程式語言。它能讓網頁具備動態更新、使用者操作反應、資料驗證等功能。

使用規則與要點說明

1. **插入方式**：可透過 `<script>` 標籤內嵌（inline）或引入外部 `.js` 檔案，通常建議使用外部檔案以利維護。

   ```
   <script src="script.js"></script>
   ```

2. **變數與作用域**：使用 `let`, `const`, `var` 宣告變數，並依照區塊作用域或函式作用域決定其可存取範圍。
3. **事件處理**：透過 `.addEventListener` 或 `onXXX` 屬性註冊使用者互動事件（如點擊、滑鼠移動、表單送出等）。
4. **條件與迴圈**：使用 `if`, `switch`, `for`, `while` 等控制流程語法進行邏輯判斷與重複執行。
5. **函式與模組**：可自訂函式或使用箭頭函式，並透過 ES 模組（`import/export`）建立模組化程式結構。
6. **DOM 操作**：透過 `document.querySelector`, `getElementById` 等方法操作頁面元素內容、屬性與樣式。
7. **非同步處理**：使用 `setTimeout`, `Promise`, `async/await` 處理非同步邏輯（如等待伺服器回應）。
8. **瀏覽器相容性與除錯工具**：開發時應注意不同瀏覽器支援程度，並善用瀏覽器開發者工具（DevTools）進行除錯與效能檢查。

ES6+ 語法與物件導向概念

9. **ES6+ 新語法**：ES6 引入多項語法進化，有助於提高開發效率與可讀性。
   - `let`/`const`：取代 `var`，避免作用域混淆。

- 箭頭函式：簡化函式撰寫，例如 `(x) => x * 2`。
- 解構賦值：快速取得物件或陣列中的值。

```
const person = { name: "Amy", age: 28 };
const { name, age } = person;
```

- 模板字串（Template Literals）：使用反引號（`）可內插變數。

```
console.log(`姓名：${name}，年齡：${age}`);
```

- 展開運算符（Spread）與其餘參數（Rest）：操作陣列與函式參數更便利。

10. **物件導向（OOP）**：JavaScript 支援基於原型的物件導向設計，也可使用 `class` 關鍵字進行類別定義。

```
class User {
  constructor(name) {
    this.name = name;
  }
  greet() {
    console.log(`哈囉，我是 ${this.name}`);
  }
}

const u = new User("John");
u.greet();
```

- 支援繼承（`extends`）與多型（override 方法）。
- 適合建構模組化與結構化應用。

11. **原型鍊與繼承機制**：每個 JavaScript 物件皆繼承自 `Object` 原型，可透過 `prototype` 擴充功能；此特性為許多框架與套件運作基礎。

範例：

```
<script>
  function showMessage() {
    alert("歡迎使用本網站！");
  }
</script>
<button onclick="showMessage()">點我顯示訊息</button>
```

4. jQuery（JavaScript 函式庫）

jQuery 是一個簡化 JavaScript 操作的函式庫，特別適合處理 DOM 操作、事件處理與 Ajax 請求。雖然在現代框架盛行後使用率下降，但在許多傳統專案中仍廣泛應用。

範例：

```
<!DOCTYPE html>
<html>
<head>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <button id="btn">點我</button>
  <p id="text"></p>

  <script>
    $("#btn").click(function() {
      $("#text").text("你點了按鈕！");
    });
  </script>
</body>
</html>
```

5. Node.js（伺服端 JavaScript 執行環境）

Node.js 是一個基於 Chrome V8 引擎的 JavaScript 執行環境，讓 JavaScript 可以在伺服器端執行。雖然 Node.js 本身屬於後端技術，但在前端工程中，它扮演著重要的工具平台角色，例如建構開發伺服器、編譯工具、部署流程等。

範例：啟動簡單的 HTTP 伺服器

```
// server.js
const http = require('http');
const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello from Node.js!');
});

server.listen(3000, () => {
  console.log('伺服器運行於 http://localhost:3000');
});
```

Node.js 常與前端開發工具整合，如 npm 套件管理、Webpack/Vite 打包工具、以及開發環境的建置等，對於現代前端工程師而言，是不可或缺的一環。

二、現代前端開發工具與框架

1. 前端框架與函式庫

- **React**：由 Facebook 推出的 UI 函式庫，支援組件化開發與虛擬 DOM。
- **Vue.js**：輕量易學，適合中小型專案，支援資料綁定與元件系統。
- **Angular**：由 Google 開發，功能完整，適合大型專案。

React 簡單範例：

```
function Welcome() {
  return <h1>Hello, React!</h1>;
}
```

2. 模組打包工具

- **Webpack**：主流打包工具，可整合 JS、CSS、圖片等資源。
- **Vite**：新一代快速開發工具，支援原生 ES 模組與即時更新。

3. 套件管理工具

- **npm / yarn**：用於管理 JavaScript 套件的工具，支援依賴安裝與版本管理。

三、前端開發流程與最佳實踐

1. 開發流程概要

1. 規劃與設計：需求分析、UI/UX 設計
2. 切版與樣式：HTML + CSS
3. 行為開發：JavaScript 加入互動功能
4. 整合與測試：與後端 API 串接、功能測試
5. 打包與部署：使用工具將專案部署至伺服器

2. 最佳實踐

- 元件化開發：提升重用性與維護性
- RWD 響應式設計：支援手機、平板與桌機等不同裝置
- 使用版本控制（如 Git）
- 碼格式化與靜態檢查（如 ESLint, Prettier）
- 測試覆蓋率與效能優化