Ch1 認識 JavaScript

陳宏益, 朝陽科技大學資管系

1 JavaScript 特色與用途

- 直譯式 (Interpret) 的描述語言
- 跨平台、物件導向、輕量
- 主要應用於 Web 開發, 製作動態網頁
  - 也可用於後端開發 (Node.js)
- 採用 ECMAScript 標準 (如 ES5、ES6、ES9)

JavaScript 的用途

- 操作 HTML DOM
- 開發網頁遊戲
- HTML5 前端資料儲存
- Node.js 後端開發

設置 JavaScript 開發環境

2 JavaScript 運行環境

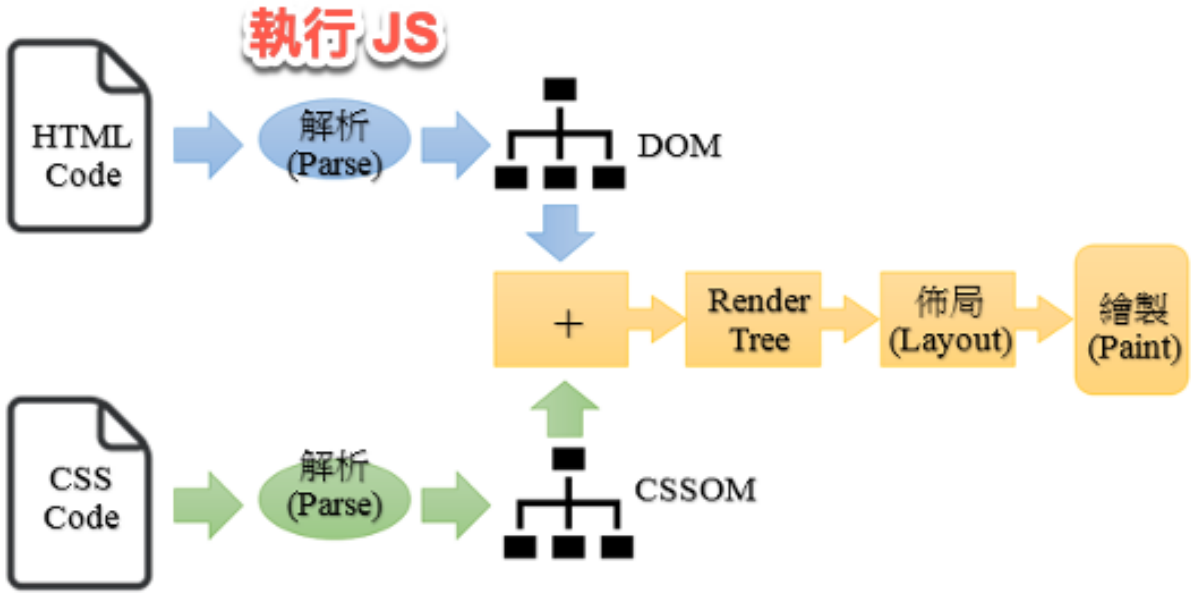
- 瀏覽器：在 Browser 中執行(用戶端)
- Node.js：讓 JavaScript 也能在後端 (伺服器端) 執行

3 核心組成

- **ECMAScript**：定義語法、流程控制、物件、函數、錯誤處理
- **DOM API**：操作網頁文件結構與內容、事件處理
- **WEB API**：操作瀏覽器功能, 如網路請求、計時器、地理位置、用戶端儲存

JavaScript 在用戶端執行

1. HTML 與 CSS 交給渲染引擎處理
2. HTML 解析並建構 DOM 樹
3. CSS 解析並建構 CSSOM 樹
4. 組合成渲染樹 (Render Tree)
5. 透過 Layout() 方法安排版面
6. Paint() 方法繪製網頁



JavaScript 在後端執行

- **Node.js** 採用 Google V8 引擎
- 官方網站: Node.js
- LTS (Long Term Support) 版本較穩定
- 提供 REPL 交互式開發環境
  - REPL: Read-Eval-Print Loop

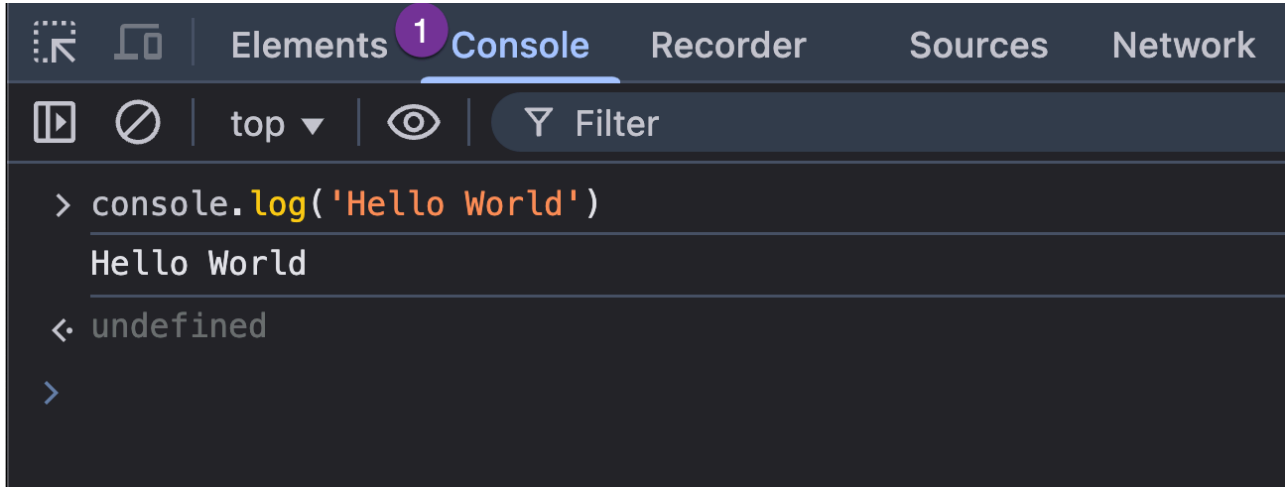
REPL 交互式開發環境

1. 在 CMD 輸入 `node` 進入 REPL
2. 輸入 `console.log("Hello World")`
3. 退出方式：
   - `Ctrl+D`
   - 輸入 `.exit`

```
~ (1m 32.92s)
node

undefined
> console.log('Hello World')
Hello World
undefined
> .exit
```

瀏覽器主控台 Console

- 在用戶端執行 JavaScript, 不需要安裝 Node.js
- 瀏覽器開發工具 `Console`
- 可用於 Debug 與測試 JavaScript
- **開啟方式**:
  - Chrome: `F12` 開啟 DevTools
  - `console.log("測試輸出")`



Console 常用方法

4 1. 基本輸出

- 一般訊息

```
console.log("一般訊息");
```

- 錯誤訊息 `console.error()`

```
console.error("錯誤訊息");
```

- 警告訊息 `console.warn()`

```
console.warn("警告訊息");
```

5 2. 清除輸出

- 清除輸出 `console.clear()`

```
console.clear();
```

6 設置開發環境

安裝 Node.js、VSCode 和 Quokka.js 插件

Node.js 是一個 JavaScript 運行時，允許您在瀏覽器外運行 JavaScript 代碼。

- 下載並安裝 Node.js

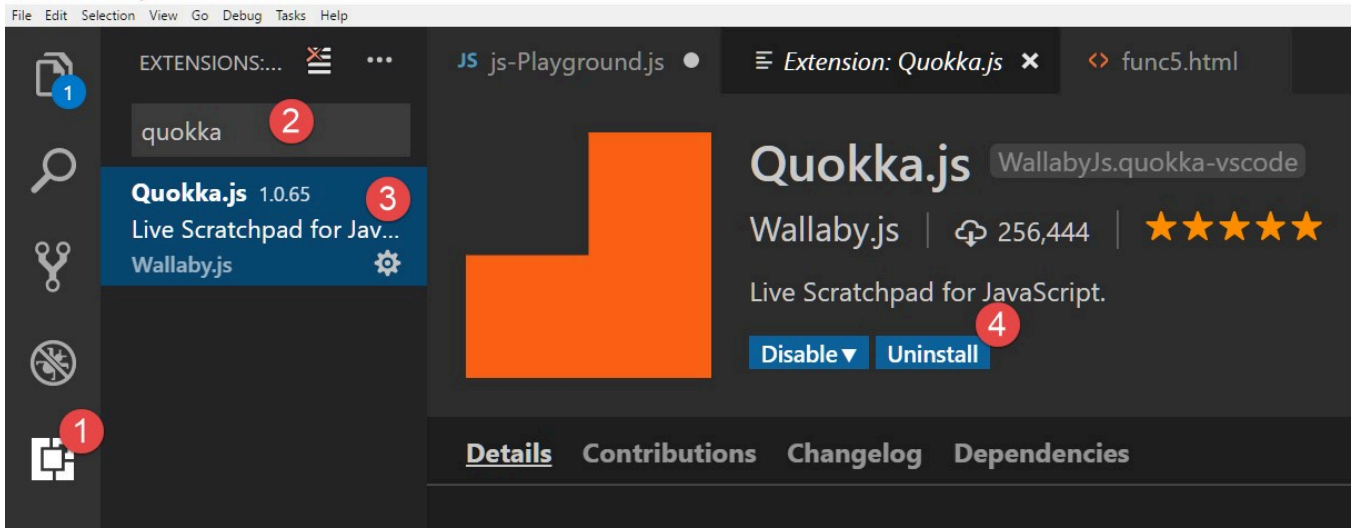
Visual Studio Code (VSCode) 是目前最受歡迎的代碼編輯器之一。

- 下載並安裝 Visual Studio Code (VSCode) for windows

Quokka.js 是一個 VSCode 插件，允許輸入 JavaScript 代碼時立即查看結果。

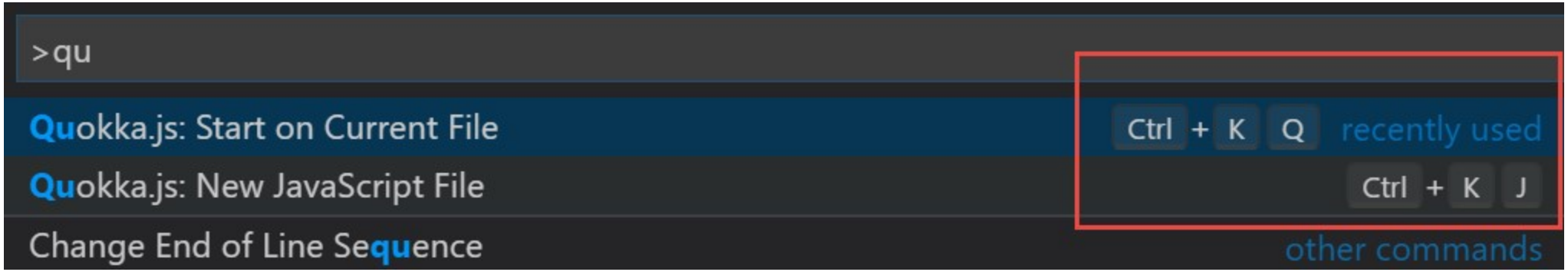
在 VSCode 中安裝 Quokka.js 擴展

在 VSCode Extensions 中搜索 Quokka.js 並安裝它



Quokka 快捷鍵：

- Cmd/Ctrl+K,Q：在現有文件上啟動/重啟 Quokka。
- Cmd/Ctrl+K,J：打開一個新的 Quokka JavaScript 文件。



啟動 Quokka 以在輸入 JavaScript 代碼時立即查看結果。

將 JavaScript 添加到網頁

有三種方式將 JavaScript 添加到網頁：

1. 內部（直接）JavaScript：在 HTML 文件中的 `<script>` 標籤內添加 JavaScript 代碼。
2. 外部 JavaScript：將 JavaScript 代碼寫在外部木檔案，再將其鏈接到 HTML 文件。
3. 行內(Inline) JavaScript：將 JavaScript 代碼直接添加到 HTML 元素的屬性中。

以下分別說明。

7 Internal JavaScript

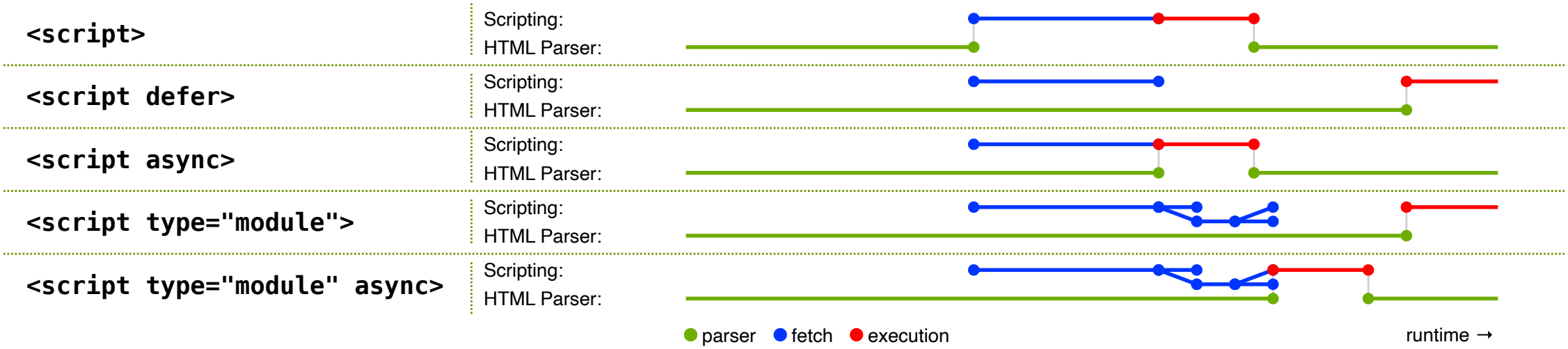
```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>

    <script>
      alert('Hello World!');
    </script>

    <h1>Hello World</h1>
    <p>Welcome to my first web page!</p>

    <script>
      alert('Hello World! 2');
    </script>
  </body>
</html>
```

對於上述代碼：

- JavaScript 代碼的執行順序是從上到下。
- 瀏覽器在遇到 `<script>` 標籤時執行 JavaScript 代碼。
- `alert()` 是一個函數, 用對話框顯示訊息。
- 當頁面載入時，瀏覽器會顯示帶有 "Hello World!" 消息的對話框。
  - 然後，它會渲染頁面的 HTML 內容。
  - 之後, 會再次顯示帶有 "Hello World! 2" 訊息的對話框。

8 Advanced reading: Control when the browsers fetch and execute the JavaScript code

使用 `<script>` 標籤中的 `defer` 和 `async` 屬性來控制瀏覽器何時提取和執行 JavaScript 代碼。



- Green: parser
- Red: fetch

`<script>`: 預設情況下， 會阻止頁面的渲染，直到瀏覽器提取並執行 JavaScript 代碼

`<script defer>`：瀏覽器在解析 HTML 內容時同時抓取 JavaScript 代碼。但它在 HTML 內容解析完成後再執行 JavaScript code。

`<script async>`：瀏覽器在解析 HTML 內容時抓取 JavaScript 代碼。抓取完成後立即執行 JavaScript code，之後再繼續解析 HTML 內容。

9 外部 JavaScript

將 JavaScript 代碼和 HTML 代碼分開的原因:

- 避免內嵌 JavaScript 代碼導致 HTML 頁面過長，不易維護。
- 在多個 HTML 文件中重複使用 JavaScript 代碼。
- 建立獨立於 HTML 文件的 JavaScript 函數庫。

10 引用外部 JavaScript 文件

使用 `<script>` 標籤的 `src` 屬性來鏈接外部 JavaScript 文件到 HTML 文件。

```
<script type="text/javascript" src="your_script.js"></script>
```

注意:

- 檔案名稱區分大小寫。
- 指定文件的相對或絕對路徑。

Example:

- Code Samples

11 Lab 1

Lab_01_01

總結

- JavaScript 可以用於前端 (瀏覽器) 和後端 (Node.js)的程式語言
- 可以在 瀏覽器 Console 中執行 JavaScript 代碼
- 可以在 Node.js REPL 中執行 JavaScript 代碼
- 三種方式將 JavaScript 添加到網頁：內部(internal)、外部(external)、行內(inline)