使用 Webpack 作為模組打包工具

主要要點

- 使用 webpack 版本 1.13.2 和 webpack-dev-server 版本 1.14.0，似乎需要設置一個具有特定配置的 Node.js 專案。
- 研究建議通過 npm 安裝這些版本，創建一個配置文件，並運行一個開發伺服器以進行即時更新。
- 證據表明確保有一個 index.html 文件，並且設置包括適當的文件路徑以進行打包。

安裝和設置

要開始，請確保從 nodejs.org 安裝 Node.js。創建一個專案目錄，用 npm 初始化它，並安裝指定的版本：

- 運行 `npm init -y` 以設置一個 package.json 文件。
- 安裝 `npm install webpack@1.13.2` 和 `npm install webpack-dev-server@1.14.0`。

配置

創建一個 `webpack.config.js` 文件來定義文件的打包方式。基本設置包括：- 入口點（例如 `./src/index.js`）。- 輸出路徑（例如 `dist` 目錄中的 `bundle.js`）。- 開發伺服器設置，例如 `contentBase` 用於靜態文件。

運行開發伺服器

使用 `npx webpack-dev-server` 或 `./node_modules/.bin/webpack-dev-server`（如果 npx 不可用）啟動伺服器。訪問 http://localhost:8080 以查看您的應用程序，它將在更改時自動更新。

意外細節

這些較舊的版本需要特定的配置，例如 `contentBase` 而不是現代的 `static`，設置可能需要手動調整文件服務，而不像新版本那樣有更多的自動化。

調查筆記：使用 Webpack 1.13.2 和 Webpack-Dev-Server 1.14.0 的詳細指南

這個全面的指南提供了設置和使用 webpack 版本 1.13.2 以及 webpack-dev-server 版本 1.14.0 的詳細步驟，專注於適合 JavaScript 專案的開發環境。由於這些版本的年齡，某些配置和行為與現代標準不同，這個筆記旨在涵蓋所有必要的步驟，確保清晰和完整。

背景和上下文 Webpack 是一個 JavaScript 模塊打包工具，歷史上用於編譯和打包網絡應用程序的文件，管理依賴並優化生產。Webpack-dev-server 是一個伴隨工具，提供一個具有即時重新加載功能的開發伺服器，適合迭代開發。指定的版本，1.13.2 版的 webpack 和 1.14.0 版的 webpack-dev-server，來自 2016 年，表示較舊但仍然功能的設置，可能用於舊版專案的兼容性。

逐步安裝和設置 首先，確保安裝了 Node.js，因為它是我們將使用的 npm 包管理器所需的。您可以從 nodejs.org 下載它。當前時間，2025 年 3 月 3 日星期一 09:45 AM PST，與設置無關，但為了上下文而記錄。

1. **創建專案目錄**：打開您的終端並創建一個新目錄，例如“myproject”：
   - 命令：`mkdir myproject && cd myproject`
2. **初始化 npm 專案**：運行 `npm init -y` 以創建一個具有默認設置的 `package.json` 文件，設置您的專案以進行 npm 依賴。
3. **安裝特定版本**：使用 npm 安裝所需的版本：
   - 命令：`npm install webpack@1.13.2`
   - 命令：`npm install webpack-dev-server@1.14.0`
   - 這些命令將指定的版本添加到您的 `node_modules`，並更新 `package.json` 中的 `dependencies`。

目錄結構和文件創建 為了讓開發伺服器正常運行，您需要一個基本的目錄結構：- 創建一個 `public` 目錄用於靜態文件：`mkdir public` - 創建一個 `src` 目錄用於您的應用程序代碼：`mkdir src`

在 `public` 中，創建一個 `index.html` 文件，這對於提供您的應用程序是必需的：

```
<html>
<body>
<script src="/bundle.js"></script>
</body>
</html>
```

這個文件引用了 `bundle.js`，webpack 將生成並提供它。

在 `src` 中，創建一個 `index.js` 文件，內容為基本內容：

```
console.log('Hello, World!');
```

這是您的入口點，webpack 將打包它。

配置文件設置　在根目錄中創建一個 `webpack.config.js` 文件來配置 webpack：

```
const path = require('path');
module.exports = {
    entry: './src/index.js',
    output: {
        path: path.join(__dirname, 'dist'),
        filename: 'bundle.js'
    },
    devServer: {
        contentBase: path.join(__dirname, 'public')
    }
};
```

- `entry`：指定起點（`src/index.js`）。
- `output`：定義打包文件的位置（`dist/bundle.js`）。
- `devServer.contentBase`：指向 `public` 目錄以提供靜態文件，例如 `index.html`。

請注意，在 1.14.0 版中，使用 `contentBase` 而不是現代的 `static`，反映了較舊的 API。

運行開發伺服器　要啟動開發伺服器，使用：- 喜好：`npx webpack-dev-server` - 替代方案（如果 npx 不可用）：`./node_modules/.bin/webpack-dev-server`

這個命令啟動一個伺服器，通常可以在 http://localhost:8080 訪問。伺服器在內存中運行，這意味著文件不會寫入磁盤，而是動態提供，並啟用了即時重新加載，方便開發。

操作細節和考量

- **訪問應用程序**：打開瀏覽器訪問 http://localhost:8080。您應該會看到您的 `index.html`，它加載 `bundle.js` 並執行您的 JavaScript，將“Hello, World!”記錄到控制台。
- **即時更新**：編輯 `src` 中的文件，伺服器將重新編譯並自動重新加載瀏覽器，這是 webpack-dev-server 迭代開發的關鍵功能。
- **端口衝突**：如果端口 8080 被使用，伺服器可能會失敗。您可以在 `webpack.config.js` 中的 `devServer.port` 下指定不同的端口，例如 `port: 9000`。

文件服務和路徑考量　給定這些版本，`devServer.contentBase` 從指定的目錄（如果未設置則默認為當前目錄）提供文件。確保 `index.html` 在 `public` 中以便在根目錄提供。腳本標籤 `<script src="/bundle.js"></script>` 起作用，因為 `output.publicPath` 默認為 ‘/’，並且 `output.filename` 為 ‘bundle.js’，使其在 `/bundle.js` 可訪問。

需要注意的是，webpack-dev-server 1.14.0 需要一個 HTML 文件來提供，並且它不會自動注入腳本，因此在 `index.html` 中手動包含是必要的。這與現代設置不同，現代設置中插件如 `html-webpack-plugin` 可能會自動化這一點。

兼容性和限制　這些版本來自 2016 年，雖然功能齊全，但缺乏現代功能和安全補丁。應檢查與當前 Node.js 版本（截至 2025 年 3 月）的兼容性，因為較舊的版本可能需要 Node.js 6 或 8，這些版本已不再支持。建議在當前 Node.js 版本（例如 20.x）上進行測試，但要準備好可能的廢棄警告。

表：版本詳細信息和使用說明　以下是總結關鍵版本信息和使用說明的表格：

組件	版本	發布年份	關鍵使用說明
Webpack	1.13.2	2016	模塊打包工具，需要配置入口/輸出
Webpack-Dev-Server	1.14.0	2016	開發伺服器，使用 `contentBase` 靜態文件

高級配置選項　對於需要更多的用戶，考慮通過將 `hot: true` 添加到 `devServer` 來啟用熱模塊替換：

```
devServer: {
    contentBase: path.join(__dirname, 'public'),
    hot: true
}
```

這允許模塊更新而不需要完全重新加載頁面，從而提高開發速度。

結論　這個設置提供了一個基本但功能齊全的開發環境，使用 webpack 1.13.2 和 webpack-dev-server 1.14.0。由於它們的年齡，預期與現代工作流程的差異，例如手動文件服務和與當前工具的潛在兼容性問題。對於生產，這些版本不推薦，但對於舊版專案的維護，這個指南應該足夠。

關鍵引用

- Webpack 文檔詳細設置指南
- Webpack-dev-server 配置選項