# JavaScript 101：從基礎到進階的全面指南

---

## JavaScript 簡介與歷史 (00:00)

- JavaScript 的定位與特點 (00:00)
  - JavaScript 是一種適合初學者的程式語言
  - 它既是一種奇妙又可怕的語言
  - 可以用於構建幾乎任何東西，並在任何地方找到工作
  - 但同時也被認為是怪異和醜陋的，且被大量框架和庫所包圍
- JavaScript 的起源 (00:40)
  - 1993年由 Brendan Eike 在 Netscape 創建
  - 當時網頁瀏覽器是尖端技術，連接全球人們
  - 最初的網站是靜態的，只有純 HTML
  - JavaScript 被設計為易用的高級語言，幫助開發者製作互動式網站

---

## JavaScript 的現代應用 (01:04)

- JavaScript 的普及性 (01:04)
  - 如今可能是世界上最流行的程式語言
  - 標準實現稱為 ECMAScript
  - 是所有網頁瀏覽器的默認語言
- JavaScript 的運行環境 (01:15)
  - 瀏覽器是主要運行環境，但不是唯一的
  - 可以通過 Node.js 和 Deno 等工具在伺服器上運行
  - 作為腳本語言，可以在瀏覽器開發工具中即時執行代碼

---

## JavaScript 的核心概念 (01:33)

- JavaScript 的解釋與編譯 (01:33)
  - 被認為是解釋型語言，逐行執行
  - 實際上，瀏覽器引擎（如 V8）使用即時編譯（JIT）將代碼轉換為機器碼
  - 這種方式使 JavaScript 運行速度極快
- 基本語法與變量 (02:00)
  - 在 HTML 文檔中使用 <script> 標籤嵌入 JavaScript
  - 使用 console.log() 輸出到控制台
  - 變量聲明方式：let、const、var
  - JavaScript 是動態類型語言，無需顯式聲明數據類型

---

## JavaScript 的數據類型與作用域 (02:25)

- 基本數據類型 (02:25)
  - 七種原始數據類型，包括數字、字符串、undefined、null 等 ⑦
  - 變量可以重新賦值為不同類型
- 變量作用域 (03:31)
  - 全局作用域：在最外層定義的變量
  - 函數作用域：在函數內部定義的變量
  - 塊級作用域：在 if 語句等塊中定義的變量（使用 let 和 const）

---

## 函數與閉包 (04:00)

- 函數定義與使用 (04:00)
  - 函數是 JavaScript 的主要構建塊
  - 可以作為語句或表達式使用
  - 支持高階函數：函數可以作為參數或返回值
- 閉包 (04:26)
  - 嵌套函數可以創建閉包，封裝數據和邏輯
  - 內部函數可以訪問外部函數的變量，即使在初始函數調用後

---

## this 關鍵字與箭頭函數 (04:56)

- this 的使用與綁定 (04:56)
  - this 關鍵字引用基於函數調用方式的對象
  - 在全局作用域中，this 引用 window 對象
  - 可以使用 bind 方法手動綁定函數到特定對象
- 箭頭函數 (05:20)
  - 現代 JavaScript 引入的新函數定義方式
  - 沒有自己的 this 值，總是匿名的
  - 適合用於函數表達式

---

## 對象與原型鏈 (05:44)

- 對象創建與繼承 (05:44)
  - 可以使用對象字面量語法或構造函數創建對象 ⑥
  - 對象包含鍵值對（屬性和值）
  - 通過原型鏈實現對象之間的繼承
- 類與面向對象編程 (06:19)
  - JavaScript 支持使用 class 關鍵字的面向對象編程
  - 類是原型繼承的語法糖
  - 可以定義構造函數、屬性、getter 和 setter

---

## 數據結構與內存管理 (06:45)

- 內置數據結構 (06:45)
  - 數組：用於存儲索引項的動態集合
  - Set：用於存儲唯一項的集合
  - Map：類似對象，但更易於遍歷的鍵值對集合
- 垃圾回收與內存管理 (07:03)
  - JavaScript 是垃圾回收語言，自動釋放不再引用的對象
  - WeakMap 和 WeakSet 用於存儲可被垃圾回收的屬性，以減少內存使用

---

## 異步編程與事件循環 (07:28)

- 事件循環與非阻塞 I/O (07:28)
  - JavaScript 使用事件循環實現非阻塞操作
  - 允許異步代碼在單線程環境中執行多任務
- 回調函數與 Promise (08:04)
  - 回調函數用於處理異步操作，但可能導致回調地獄 ④
  - Promise 提供更好的異步編程方式，處理未來值
- Async/Await (08:39)
  - async 函數自動返回 Promise
  - await 關鍵字用於暫停函數執行，等待 Promise 解決
  - 使用 try/catch 進行錯誤處理

---

## 模塊化與包管理 (09:01)

- ES6 模塊 (09:01)
  - 使用 export 和 import 語句在文件間共享代碼
  - 支持默認導出和命名導出
- NPM 包管理 (09:28)
  - npm 是最大的 JavaScript 包管理器
    - 包管理：npm 提供一個在線庫，其中包含可重用的 JavaScript 模塊和庫，開發者可以輕鬆下載和安裝所需的包。
    - 依賴管理：npm 自動處理項目依賴，幫助開發者在項目中跟蹤和安裝所需的依賴包。
    - package.json：在項目中使用 npm 時，會生成一個 package.json 文件，列出項目的依賴、版本以及其他項目信息。
    - 命令行界面：npm 提供命令行工具，開發者可以通過命令安裝、更新或卸載包，例如：
    - 開源社區：npm 具有龐大的開源社區，開發者可以分享和貢獻自己的包，促進生態系統的成長。
  - 使用 package.json 文件列出項目依賴

---

## 瀏覽器 DOM 操作 (09:42)

- DOM 介紹 (09:42)
  - DOM 將 UI 表示為 HTML 元素樹
  - 瀏覽器提供 API 與這些節點交互
- 元素選擇與事件監聽 (09:58)
  - 使用 querySelector 和 querySelectorAll 選擇元素
  - addEventListener 用於監聽事件，如點擊click、mouseover、keydown、submit 等。

---

## 前端框架與打包工具 (10:30)

- 聲明式編程與組件化 (10:30)
  - 許多開發者使用前端框架進行聲明式編程
  - 將 JavaScript、HTML 和 CSS 封裝為組件
- 模塊打包 (11:14)
  - 使用模塊打包器如 Vite 或 Webpack 合併 JavaScript 文件
  - 支持代碼分割和動態導入以優化性能

---

## 服務器端 JavaScript (11:36)

- Node.js 與服務器端應用 (11:36)
  - Node.js 是流行的服務器端 JavaScript 運行時 ⑤
  - 支持使用 JavaScript 構建全棧應用
- 跨平台開發 (11:45)
  - Electron 結合 Node.js 和瀏覽器創建桌面應用
  - React Native 用於構建 iOS 和 Android 應用

---

## 結語與進階學習 (12:04)

- 代碼質量工具 (12:04)
  - 使用 TypeScript 或 ESLint 進行靜態分析，提高代碼質量
    - TypeScript: ③
    - ESLint: ③
- 深入學習資源 (12:08)
  - 介紹了更深入的 JavaScript 課程
  - 課程包括深入概念講解、實踐項目和測驗