

Programación Web Avanzada

Del lenguaje a producción en 4 módulos

16 lecciones · 4 ejercicios prácticos · Proyecto final · Rúbrica y glosario

-
- 01 JavaScript y TypeScript profesional
 - 02 Frontend avanzado: React y rendimiento
 - 03 Backend, APIs y datos
 - 04 Calidad, DevOps y arquitectura

Creado por saberaclick.com

Edición 2026 · Contenido original · Todos los derechos reservados

Índice

Introducción al curso

4

A quién va dirigido

4

Requisitos previos

4

Estructura del curso

4

Metodología: aprende, construye, mide

4

Cómo sacar el máximo provecho

5

Módulo 1: JavaScript y TypeScript de nivel profesional

6

Lección 1.1 · El motor por dentro: event loop y colas de tareas

6

Lección 1.2 · Asincronía avanzada y concurrencia controlada

6

Lección 1.3 · Closures, módulos y patrones de diseño

7

Lección 1.4 · TypeScript avanzado

8

Ejercicio práctico del Módulo 1

9

Módulo 2: Frontend avanzado: React, rendimiento y accesibilidad

10

Lección 2.1 · Arquitectura de componentes y hooks personalizados

10

Lección 2.2 · Gestión de estado: cada dato en su lugar

11

Lección 2.3 · Rendimiento web y Core Web Vitals

12

Lección 2.4 · Renderizado en servidor y accesibilidad

13

Ejercicio práctico del Módulo 2

13

Módulo 3: Backend, APIs y datos

15

Lección 3.1 · Diseño de APIs REST robustas

15

Lección 3.2 · Bases de datos relacionales con PostgreSQL

16

Lección 3.3 · Autenticación y autorización

17

Lección 3.4 · Seguridad y caché

17

Ejercicio práctico del Módulo 3

18

Módulo 4: Calidad, DevOps y arquitectura

19

Lección 4.1 · Estrategia de pruebas

19

Lección 4.2 · Contenedores con Docker

20

Lección 4.3 · Integración y entrega continua (CI/CD)

21

Lección 4.4 · Observabilidad, escalabilidad y arquitectura

22

Ejercicio práctico del Módulo 4

22

Proyecto final: plataforma de reservas

24

MÓDULO 1

JavaScript y TypeScript de nivel profesional

Todo el desarrollo web avanzado descansa sobre un conocimiento profundo del lenguaje. En este módulo abrirás la caja negra: cómo decide JavaScript qué ejecutar y cuándo, cómo manejar la asincronía sin crear código frágil, qué patrones ordenan un proyecto grande y cómo TypeScript convierte errores de ejecución en errores de compilación.

Al terminar este módulo podrás:

- Predecir el orden de ejecución de código síncrono, microtareas y macrotareas.
- Controlar la concurrencia y cancelar operaciones asíncronas.
- Aplicar closures, módulos y patrones de diseño con criterio.
- Modelar dominios con genéricos, uniones discriminadas y tipos utilitarios.

Lección 1.1 · El motor por dentro: event loop y colas de tareas

JavaScript ejecuta tu código en un único hilo, pero la plataforma (navegador o Node.js) le entrega trabajo a través de colas. El **call stack** ejecuta las funciones en curso. Cuando termina, el **event loop** revisa primero la cola de **microtareas** (promesas, `queueMicrotask`) y la vacía por completo; solo entonces toma **una** macrotaska (temporizadores, eventos de usuario, red) y repite el ciclo.

Entender esto explica errores clásicos: un `setTimeout(fn, 0)` nunca se ejecuta antes que una promesa ya resuelta, y un bucle de microtareas infinito puede congelar la página porque el navegador nunca llega a pintar.

```
console.log("1. inicio");

setTimeout(() => console.log("2. macrotaska (setTimeout)"), 0);

Promise.resolve().then(() => console.log("3. microtaska (promesa)"));

queueMicrotask(() => console.log("4. microtaska (queueMicrotask)"));

console.log("5. fin del script");

// Salida real: 1, 5, 3, 4, 2
```

Práctica rápida

Predice la salida antes de ejecutar. Después añade un `await` dentro de una función `async` y razona en qué cola cae la continuación. Anota tus conclusiones: serán tu mapa mental del resto del curso.

Lección 1.2 · Asincronía avanzada y concurrencia controlada

`async/await` es azúcar sintáctico sobre promesas, y como tal hereda sus reglas. Lanzar cien peticiones con `Promise.all` puede saturar la red o el servidor; esperar una por una es innecesariamente lento. La solución profesional es la **concurrencia limitada**: un grupo fijo de trabajadores que consumen tareas de una cola.