DEV.F::

Clase 1

Módulo 5

Objetivos de la clase (180 min)

Asincronía en JavaScript (30-40 min)

- ➔ Entender event loop, call stack y task queue
- ➔ Distinguir entre tareas síncronas y asíncronas
- ➔ Visualizar cómo se ejecuta una función asincrónica paso a paso

Arquitectura Cliente-Servidor (20-25 min)

- ➔ Comprender el flujo de una petición HTTP desde el cliente al servidor
- ➔ Identificar qué es un endpoint y cómo se estructura una API REST
- ➔ Relacionar formularios web con operaciones reales de cliente-servidor

Uso de Postman para probar APIs (20-25 min)

- ➔ Realizar peticiones GET y POST a una API pública desde Postman.
- ➔ Explorar y entender la respuesta en formato JSON.
- ➔ Simular errores comunes para entender el ciclo request-response.s

DEV.F::

Objetivos de la clase (180 min)

Asincronía en JavaScript (30-40 min)

- ➔ Event loop, call stack y task queue
- ➔ Tareas síncronas y asíncronas
- ➔ Función asincrónica paso a paso

Arquitectura Cliente-Servidor (20-25 min)

- ➔ Flujo de una petición HTTP desde el cliente al servidor
- ➔ Endpoint y cómo se estructura una API REST
- ➔ Relacionar formularios web

Uso de Postman para probar APIs (20-25 min)

- ➔ GET y POST a una API pública
- ➔ Respuestas en formato JSON
- ➔ Simular errores comunes para entender el ciclo request-response

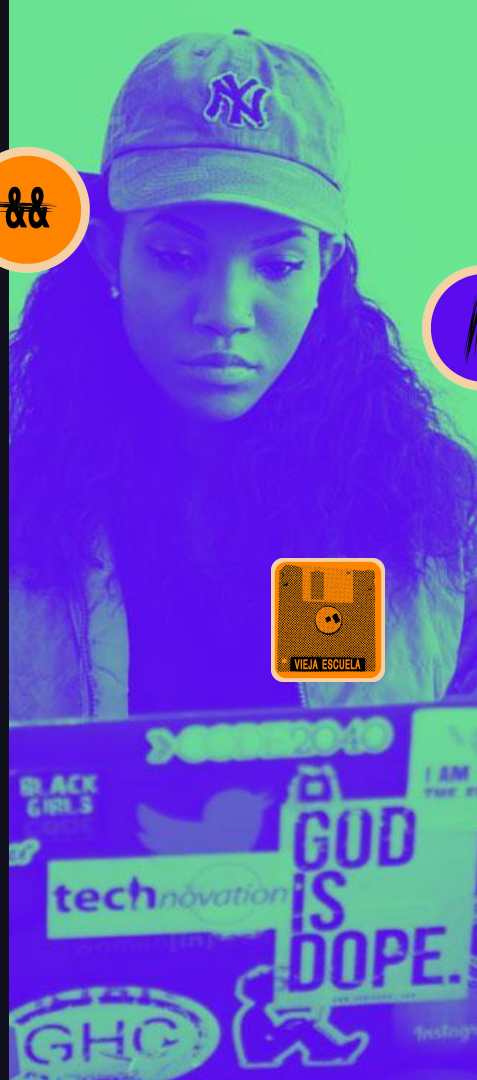
DEV.F::

Lecturas del día

Event loop y ejecución asincrónica en JavaScript

Modelo Cliente-Servidor y ciclo de vida de una petición HTTP

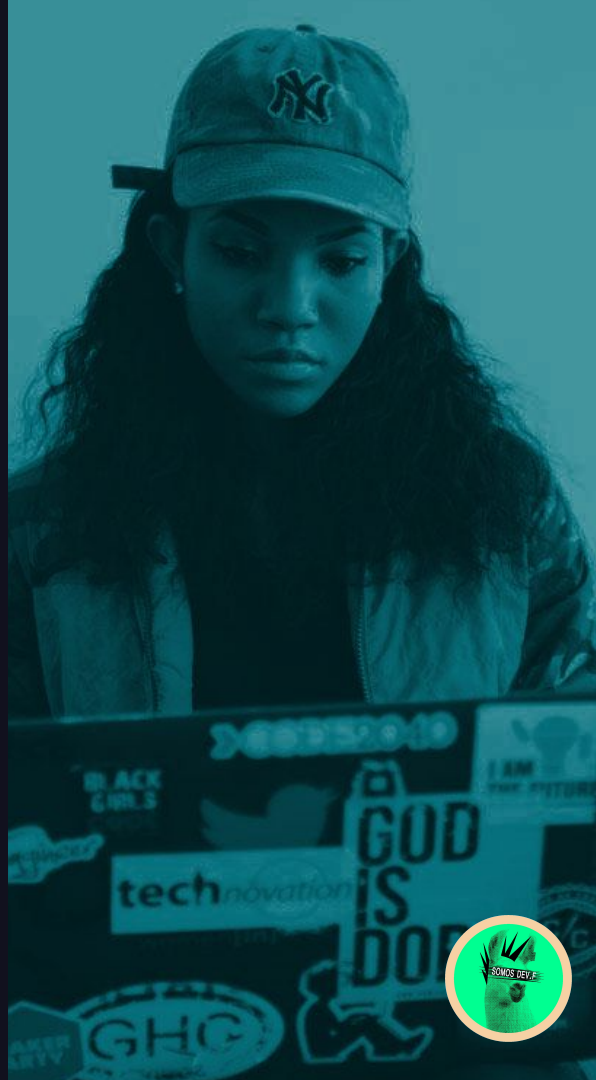
Pruebas de endpoints con Postman y análisis de respuestas JSON



DEV.F::

Event Loop y asincronía en JavaScript

El Event Loop es el mecanismo que permite a JavaScript ejecutar código, manejar eventos y realizar operaciones asíncronas. Funciona como un ciclo que revisa constantemente si hay tareas pendientes (como `setTimeout`, fetch, etc.) y las ejecuta cuando el *call stack* está libre.



DEV.F::

¿Cómo entender la asincronía?

Piensa en una sola persona en una cocina. Solo puede preparar un plato a la vez (call stack), pero si algo necesita tiempo (como calentar agua), lo pone a un lado (web APIs) y sigue cocinando otras cosas. Cuando el agua hierve, alguien le avisa que ya está lista (task queue).

DEV.F::

Modelo Cliente-Servidor

En la web, el cliente (navegador) hace solicitudes a un servidor, que responde con datos o acciones. Es un flujo constante de petición-respuesta que permite cargar páginas, enviar formularios, y más.

Como cuando pides comida en una taquería. Tú (cliente) haces el pedido, el taquero (servidor) lo recibe, lo prepara y te lo entrega. No sabes cómo lo cocina, solo confías en que llegará tu taco.

SYSTEM EDUCATION

Postman y pruebas de API

Postman es una herramienta para simular peticiones HTTP sin necesidad de escribir código. Permite ver cómo se comporta un endpoint: qué datos recibe, qué devuelve y cómo responde ante errores.

DEV.F::

Postman y pruebas de API

Es como usar un walkie-talkie para probar si otra persona (el servidor) te escucha. Dices algo (envías un mensaje) y esperas una respuesta. Si la respuesta no llega o llega mal, sabes que hay un problema en la comunicación.

Postman es una herramienta para simular peticiones HTTP sin necesidad de escribir código. Permite ver cómo se comporta un endpoint: qué datos recibe, qué devuelve y cómo responde ante errores.

Proyecto del día

Crear un pequeño programa en JavaScript que simule cómo funciona el Event Loop, combinando operaciones síncronas y asíncronas (setTimeout, console.log, fetch opcional), y visualizar el orden en que se ejecutan.

📌 Objetivos del proyecto

- Observar el comportamiento del call stack y la cola de tareas.
- Ver cómo las operaciones asíncronas se manejan fuera del flujo principal.
- Reforzar la diferencia entre tareas que bloquean y no bloquean.

DEV.F::

ESCUELA DE HACKERS
commodore
CBM
CONST : DEV.F
GENIO O HACKER?
DEVF FUNCTION ()
404
SOMOS DEV.F
DEV.F
DEV.F = EDUCA
DEV.F

Usar con discreción

DEV.F ~~FUNCTION~~ ()

DEV.F::