

Descubro JavaScript como nuevo terreno

Sección 1 – Reflexión comparativa

1. ¿Qué aspectos me resultan familiares y cuáles extraños?

Como vengo de aprender Python, me resultan familiares cosas como las variables, las condicionales **if**, y los ciclos como **for**. Pero me parecen un poco extrañas cosas como **let**, **const**, y cómo se escriben las funciones en JavaScript, porque son diferentes a **def** en Python.

2. ¿Cómo cambia mi forma de pensar al usar el navegador como entorno?

Cuando uso el navegador para ejecutar el código, siento que tengo que pensar en cómo el usuario va a ver e interactuar con la página. No solo pienso en que el código funcione, sino también en cómo se muestra y responde.

3. ¿Qué es para mí interactuar con el DOM?

Para mí, interactuar con el DOM es como hablarle a una página web y decirle: “cambia esto”, “muéstrame aquello”, o “haz que esto desaparezca”. Es como controlar los elementos de la página desde el código.

Desafío

Explica qué es el DOM como si se lo contaras a un niño de 10 años.

El DOM es como un arbolito donde cada parte de una página web (como los botones, textos o imágenes) son como ramas y hojas. JavaScript puede treparse a ese árbol y cambiar cosas, como pintar las hojas de otro color o quitar ramas.

Metáfora:

El DOM es como una casa, donde cada habitación es una parte de la página (el título, un botón, una imagen). JavaScript es como el dueño que puede cambiar los muebles, prender o apagar la luz, o abrir y cerrar las puertas.

Conexiones profundas

Sección 2 – Conexiones profundas

4. ¿Qué rol tiene JS con HTML y CSS?

JavaScript es el que le da vida y movimiento a la página web. HTML es la estructura, como los huesos del cuerpo. CSS es el estilo, como la ropa. JS es el comportamiento. Si JavaScript falta, la página se ve bonita, pero no reacciona a nada.

5. ¿Qué decisiones impactan la experiencia del usuario?





Hacer que algo aparezca o desaparezca, cambiar colores, mover elementos, mostrar mensajes, o cambiar la forma en que navega el usuario. Todo esto mejora la experiencia.

6. ¿Ventajas y riesgos de manipular el DOM?

- **Ventajas:** puedo cambiar el contenido de la página al instante.
- **Riesgos:** si lo hago mal, puedo borrar cosas o hacer que la página se vuelva lenta.

Desafío

Mapa mental de roles:

-  HTML: estructura
-  CSS: estilo
-  JS: comportamiento
-  Se cruzan cuando JS cambia el HTML o afecta el CSS dinámicamente.

Autoevaluación guiada

7. ¿Qué entendí claramente hoy?

Entendí claramente cómo usar JavaScript para cambiar cosas en la página (DOM).

8. ¿Qué concepto me pareció el más confuso?

Me pareció confuso cómo funcionan las funciones flecha y el scope.

9. ¿Qué necesito para profundizar y afianzar lo que aprendí?

Necesito hacer más ejercicios prácticos para reforzar lo aprendido.

Domino la lógica y empiezo a cuestionarla

Sección 1 – Pensamiento crítico

10. ¿Por qué existen var, let y const? ¿Qué decisiones de diseño hay detrás?

Porque `var` tiene comportamientos impredecibles y `let` y `const` son formas más seguras de declarar variables. Es una evolución del lenguaje.

11. ¿Qué consecuencias trae no entender bien el “scope” en JavaScript?

Podemos crear errores sin darnos cuenta, como sobrescribir variables o usar variables que no existen en ese contexto.

12. ¿Qué beneficios y riesgos trae el hoisting?

- **Beneficio:** podemos usar algunas funciones antes de declararlas.
- **Riesgo:** si no lo entendemos bien, podemos pensar que una variable tiene un valor cuando en realidad es ``undefined``.

Desafío:

– Historia de hoisting

Oye amiga tu conoces a var, let y const? NO.... Quienes son ellos? Bueno deja te cuento, var es un personaje impulsivo, que llega primero a la fiesta aunque nadie lo haya invitado aun. Y bueno let y const son más ordenados y pasientes, ellos esperan su invitacion para llegar a la fiesta. Pero, si tratas de chantajearlos (usarlos) antes de invitarlos formalmente(declararlos), se enojan y no van mas la fiesta hasta que les pidas perdon.

Sección 2 – Enseñanza como forma de aprender

13. ¿Cómo le enseñaría a alguien el concepto de asincronía en JavaScript?

Le diría que en JavaScript las cosas no siempre pasan en orden. Algunas tardan más, como cuando haces un pedido por internet, por lo general uno sigue haciendo sus cosas mientras espera que llegue el pedido. Eso es asincronía: el programa no se queda esperando, sigue haciendo otras cosas.

Eso es asincronía: el programa no se queda esperando, sigue haciendo otras cosas.

14. ¿Qué ejemplo cotidiano puedo usar para explicar qué es una promesa (Promise)?

Colocando el mismo ejemplo anterior una promesa es como hacer un pedido pero supongamos que pedimos una pizza:

- Hacemos el pedido de la pizza.
- Esperamos a que llegue.
- Puede que llegue bien, si es así se acepta o puede que no, entonces se rechaza.

15. ¿En qué casos no usaría arrow functions?

Cuando necesito acceder a this, porque las arrow functions no tienen su propio this. También si quiero declarar una función larga o más estructurada, prefiero usar function.

Desafío – Guion con emojis sobre Promises

👤 : Quiero una 🍕

☎️ : Llamo a pedirla (Promise)

⌚ : Sigo haciendo cosas

📦 : Llega ✅ o no llega ❌

🍕 : ¡Pizza lista! | Algo no salió bien

😋 : ¡A comer! | 😞 Te quejas

Autoevaluación guiada 2

16. ¿Qué error conceptual he detectado en mí o en otros al usar funciones o variables en JS?

Confundir let con const, y creer que var funciona igual que let, o pensar que una variable declarada dentro de un bloque se puede usar en otro lugar.

17. ¿Qué tipo de práctica necesito para afianzar estos conceptos?

Necesito hacer más ejercicios pequeños pero variados, como cambiar textos, usar funciones y hacer clic en botones que reaccionen con JS. También me ayuda ver ejemplos visuales y jugar con ellos.

18. ¿Cómo sabré que entendí la asincronía?

- Cuando use fetch o una API sin trabar la página, y sepa explicarlo.

🚀 Proyecto mi pensamiento hacia el uso profesional

Sección 1 – Abstracción y proyección

19. ¿Qué implicancias tiene usar JavaScript tanto en el frontend como en el backend?

Significa que puedo usar un solo lenguaje para trabajar en las dos partes de una aplicación. Eso me ayuda a tener un flujo más uniforme, pero también debo

entender que el backend tiene otras responsabilidades (como manejar bases de datos o seguridad).

20. ¿Qué debe tener una aplicación moderna para ser robusta y escalable usando JavaScript?

Debe tener código bien organizado, componentes reutilizables, uso de frameworks (como React o Node.js), manejo correcto de errores y seguridad. También debe ser fácil de mantener.

21. ¿Cómo cambia la lógica cuando programo para el navegador vs para el servidor?

En el navegador (frontend), pienso en lo que el usuario ve y toca. En el servidor (backend), pienso en cómo se procesan los datos, cómo responder a peticiones y cómo proteger la información.

Desafío: Describe dos escenarios (frontend y backend fallando).

- **Si falla el frontend:** El usuario ve una página vacía o botones que no funcionan. No puede interactuar con la aplicación.
- **Si falla el backend:** La página carga, pero no se puede iniciar sesión, guardar datos o hacer consultas. El sistema responde con errores o no responde.

Sección 2 – Diseño y autoconciencia

22. ¿Qué decisiones de diseño impactan en la mantenibilidad del código?

Usar nombres claros para variables y funciones, organizar los archivos por carpetas, escribir comentarios útiles, y separar la lógica por módulos para que sea fácil de entender.

23. ¿Qué hábitos debo desarrollar para evitar errores sutiles en JS?

Revisar el código antes de ejecutarlo, usar la consola para probar cosas, escribir pruebas (tests) y tener cuidado con cosas como el hoisting y el scope.

24. ¿Cómo puedo saber si mi código JavaScript es “ético”, accesible y usable?

Si mi código funciona bien en diferentes dispositivos, si respeta la privacidad del usuario, si permite el acceso a personas con discapacidades (lectores de pantalla, teclado), y si no excluye a nadie por su forma de navegar.

Desafío: App que uso a diario.

Uso WhatsApp Web. JavaScript hace posible que los mensajes lleguen en tiempo real, que se muestren las conversaciones y que todo funcione sin recargar.

Mejoraría el sistema de búsqueda para que encuentre mensajes y contactos más rápido y también la velocidad con muchos chats abiertos.

Autoevaluación final

25. ¿Qué habilidades desarrollé esta semana que no tenía antes?

Aprendí a interactuar con el DOM, a usar eventos en JavaScript y a entender cómo funcionan las promesas.

26. ¿Qué tema de JavaScript me gustaría dominar a profundidad?

Me gustaría entender mejor la asincronía y cómo usar ``async`` y ``await``.

27. ¿Cómo puedo explicarle a alguien por qué JavaScript es una herramienta poderosa y versátil?

Porque te permite hacer todo en una aplicación: desde lo que el usuario ve hasta cómo se conectan los datos por detrás. Es rápido, se usa en todas partes, y tiene muchas herramientas que lo hacen práctico.

28. ¿Cuál fue mi mayor obstáculo y cómo lo enfrenté?

Mi mayor obstáculo fue entender el concepto de asincronía. Lo enfrenté viendo ejemplos sencillos, videos y practicando con promesas.

29. ¿Cómo cambiaría mi forma de aprender si tuviera que enseñar JavaScript el próximo mes?

Intentaría explicarlo con dibujos, ejemplos cotidianos y muchos ejercicios simples para que la otra persona no se sienta perdida como yo.

30. ¿Qué tipo de desarrollador quiero ser? ¿Cómo me ayuda JavaScript a acercarme a esa meta?

Quiero ser una desarrolladora frontend que pueda crear páginas interactivas y accesibles. JavaScript me ayuda porque es la herramienta principal para hacer eso realidad.