Tarea no. 2 . JS

Nombre: Christian Hernandez Sardina

Asignatura: Aplicaciones Web

Docente: Dr. Mario Humberto Rodríguez Chávez

Matricula: 2530113

Septiembre 2026

Introducción y contexto

1. ¿Qué es JavaScript y cuáles son sus principales usos en la web?

JavaScript es un lenguaje de programación de alto nivel, interpretado, orientado a objetos y guiado por eventos. Sus principales usos en la web incluyen crear interactividad en el lado del cliente (frontend), manipular el DOM para cambiar el contenido o diseño dinámicamente, validar formularios antes de enviarlos al servidor, crear animaciones, y desarrollar aplicaciones web completas.

2. ¿En qué año y con qué propósito se creó JavaScript?

JavaScript fue creado en 1995 por Brendan Eich mientras trabajaba en Netscape Communications. Su propósito original era agregar interactividad dinámica y comportamientos simples a las páginas web estáticas de la época, permitiendo que el navegador respondiera a las acciones del usuario sin necesidad de recargar la página.

3. ¿Por qué JavaScript no es lo mismo que Java?

A pesar de la similitud en el nombre (la cual fue principalmente una estrategia de marketing), son lenguajes completamente distintos. Java es un lenguaje compilado, fuertemente tipado y orientado a objetos que requiere una máquina virtual para ejecutarse. JavaScript es un lenguaje interpretado (scripting), de tipado dinámico, diseñado principalmente para ejecutarse directamente en los navegadores web.

4. ¿Cómo se complementa JavaScript con HTML y CSS?

En el desarrollo web moderno, estos tres lenguajes forman los pilares fundamentales: HTML proporciona la estructura y el contenido semántico de la página web; CSS se encarga de la presentación, diseño y estilos visuales; y JavaScript añade el comportamiento, la lógica y la interactividad, permitiendo que la página reaccione a las acciones del usuario y cambie dinámicamente HTML o CSS.

Fundamentos del lenguaje

5. ¿Cuál es la sintaxis básica de JavaScript y cómo se escriben comentarios?

La sintaxis de JavaScript está influenciada por C y Java, utilizando llaves '{}' para bloques de código y punto y coma ';' al final de las instrucciones (aunque muchas veces son opcionales). Los comentarios de una sola línea se escriben utilizando dos barras diagonales '//' (ej. // Esto es un comentario). Para comentarios de múltiples líneas, se utiliza '/*' al inicio y '*/' al final (ej. /* Comentario largo */).

6. ¿Qué diferencias hay entre var, let y const?

'var' es la forma tradicional de declarar variables; tiene un alcance global o de función y permite ser redeclarada, lo que puede causar errores. 'let' tiene un alcance de bloque (limitado a las llaves {} donde se define) y no puede ser redeclarada en el mismo alcance. 'const' también tiene alcance de bloque, pero se utiliza para declarar constantes, lo que significa que su valor no puede ser reasignado una vez inicializado.

7. ¿Cuáles son los tipos de datos primitivos en JavaScript? Da un ejemplo de cada uno.

Los datos primitivos incluyen:

- String para texto (ej. "Hola")
- Number para números (ej. 42 o 3.14)
- Boolean para valores lógicos (ej. true o false)
- Undefined para variables declaradas pero sin valor asignado (ej. let x;)
- Null para representar la ausencia intencional de valor (ej. let y = null;)
- Symbol para identificadores únicos (ej. Symbol('id'))
- BigInt para números enteros extremadamente grandes (ej. 90071992547409911).

8. ¿Qué diferencia hay entre un array y un objeto en JavaScript?

Un array (arreglo) es una lista ordenada de valores indexada numéricamente comenzando desde el cero (ej. [1, 2, 3]).
Un objeto es una colección de pares clave-valor que no tienen un orden específico estricto, indexada por cadenas de texto o símbolos (ej. { nombre: 'Juan', edad: 25 }).
Los arrays se usan para listas de elementos similares, los objetos para representar entidades con características.

9. **¿Qué son y cómo se usan los operadores aritméticos, lógicos y relacionales?**

Los operadores aritméticos realizan cálculos matemáticos (+, -, *, /, % para el módulo, ** para exponente).
Los operadores lógicos evalúan condiciones múltiples (&& para AND, || para OR, ! para NOT).
Los operadores relacionales (o de comparación) comparan dos valores y devuelven un booleano (== igualdad abstracta, === igualdad estricta que compara tipo y valor, !=, !==, >, <, >=, <=).

Control de flujo

10. **¿Cómo funcionan las sentencias condicionales if, else y switch?**

La sentencia 'if' ejecuta un bloque de código si una condición es verdadera (true). 'else' proporciona una alternativa para ejecutar código si la condición del 'if' es falsa. 'switch' evalúa una sola expresión y compara su valor frente a múltiples casos posibles (case), ejecutando el bloque de código correspondiente al primer caso que coincida, siendo muy útil cuando hay muchas condiciones sobre una misma variable.

11. **¿Cuáles son las diferencias entre los bucles for, while y do...while?**

El bucle 'for' se usa cuando se conoce el número exacto de iteraciones; incluye la inicialización, condición y actualización en una sola línea.
'while' repite un bloque de código mientras su condición se evalúe como

verdadera, pudiendo no ejecutarse nunca si la condición es falsa desde el inicio. 'do...while' es similar a while, pero garantiza que el bloque de código se ejecute al menos una vez antes de evaluar la condición.

12. **¿Para qué sirven las palabras clave break y continue?**

La palabra clave 'break' detiene completamente la ejecución de un bucle o una estructura switch, saliendo de ella inmediatamente. Por otro lado, 'continue' detiene únicamente la iteración actual del bucle y salta directamente a la evaluación de la siguiente iteración, omitiendo cualquier código restante dentro del bloque para ese ciclo específico.

Funciones

13. **¿Cómo se declara una función y cómo se llama en JavaScript?**

Una función se declara utilizando la palabra clave 'function', seguida de un nombre, paréntesis para los parámetros, y llaves para el bloque de código (ej. function saludar() { console.log('Hola'); }). Para llamarla y que ejecute su código, se escribe su nombre seguido de paréntesis: saludar();.

14. **¿Qué significa que una función tenga parámetros y un valor de retorno?**

Los parámetros son variables temporales que actúan como entradas o canales de información que la función necesita para realizar su trabajo. El valor de retorno utilizando la palabra 'return', es el resultado de salida que la función entrega de vuelta al lugar donde fue llamada. Si una función no tiene un 'return' explícito, devuelve 'undefined' por defecto.

15. **¿Qué son las funciones flecha (=>) y qué ventajas tienen?**

Las funciones flecha son una sintaxis más corta y moderna para escribir expresiones de funciones (ej. const sumar = (a, b) => a + b;). Sus principales ventajas son la concisión del código (permitiendo omisión de la palabra 'return' y

llaves en expresiones simples) y el hecho de que no vinculan su propio contexto 'this', sino que heredan el 'this' del ámbito circundante (lexical scoping).

16. **¿Qué es el scope en JavaScript y cómo afecta a las variables?**

El scope (alcance o ámbito) determina la visibilidad y accesibilidad de las variables y funciones en distintas partes del código.
El scope global significa que la variable está disponible en todo el script.
El scope de función significa que las variables declaradas dentro de una función solo existen allí.
El scope de bloque (aplicable a let y const) restringe las variables a cualquier bloque delimitado por llaves {}.

Objetos y colecciones

17. **¿Cómo se declara un objeto y cómo se accede a sus propiedades?**

Un objeto literal se declara usando llaves y definiendo pares de clave y valor separados por comas (ej. const persona = { nombre: 'Ana', edad: 28 };). Para acceder a sus propiedades, se puede usar la notación de punto (persona.nombre) o la notación de corchetes con una cadena de texto (persona['edad']).

18. **¿Qué diferencia hay entre una propiedad y un método en un objeto?**

Una propiedad es una variable que está asociada al objeto y guarda datos estáticos o estados (por ejemplo, el color o modelo de un coche). Un método es una función que está asociada al objeto y define su comportamiento o acciones que puede realizar (por ejemplo, arrancar() o frenar()).

19**. Menciona al menos 3 métodos básicos de los arrays (push, pop, map, etc.) y explica cómo funcionan.**

- El push se utiliza para insertar uno o varios valores en un array en la última posicion disponible de este y devuelve la nueva longitud del array.
- Con el pop se puede eliminar el ultimo valor que se ingreso en el array y regresa el valor eliminado.
- Usando unshift se puede agregar uno o varios valores al inicio del array regresando la nueva longitud.

Otros temas básicos

20. **¿Qué es el DOM y por qué es importante en el uso de JavaScript en páginas web?**

El DOM (Document Object Model) es una interfaz de programación que representa la estructura HTML o XML de una página web como un árbol jerárquico de objetos o nodos. Es fundamental para JavaScript porque actúa como el puente que le permite al código leer, modificar, agregar o eliminar elementos de la página y responder a interacciones en tiempo real.

21**. ¿Qué es un evento en JavaScript y menciona al menos 3 ejemplos comunes?**

Un evento es cualquier acción o suceso que ocurre en el navegador, al cual el código JavaScript puede 'escuchar' y reaccionar ejecutando funciones (event listeners). Ejemplos comunes incluyen: 'click' (cuando el usuario hace clic en un elemento), 'keydown' (cuando el usuario presiona una tecla), y 'submit' (cuando se envía un formulario).

22. **¿Cuál es la diferencia entre alert, prompt y console.log?**

'alert' detiene la ejecución y muestra una ventana emergente simple con un mensaje y un botón de aceptar. 'prompt' muestra una ventana similar pero permite al usuario ingresar texto y devuelve ese valor. 'console.log' no interrumpe al usuario ni altera la interfaz web; en su lugar, imprime mensajes de manera invisible en la consola del desarrollador, siendo la herramienta principal para debugging.

23. **¿Cómo se abre la consola del navegador y qué ventajas tiene usarla para depurar código?**

La consola se abre típicamente pulsando F12, o haciendo clic derecho en la página y seleccionando 'Inspeccionar', luego yendo a la pestaña 'Consola'. Sus ventajas incluyen: mostrar advertencias y errores críticos (como sintaxis incorrecta o variables no definidas), permitir la ejecución interactiva de comandos JavaScript en tiempo real, e imprimir el estado de variables para rastrear el flujo del programa.