GRUPO 2

Explorando el Entorno de JavaScript en el navegador

Actividad M4C2

Paz Beltrán – Israel Lagos – Joaquín Leppe – Camilo Olivares – Sophia López – Aldo Celle

Grupo 2 Desarrollo de aplicaciones front-end trainee v2.0

Contenido

Explorando el Entorno de JavaScript en el navegador

1. Investigación sobre el Entorno de JavaScript en el Navegador

a. Investiga qué es el entorno de ejecución de JavaScript y su relación con los navegadores web.

Este entorno es el conjunto de componentes que permiten que el código JavaScript funcione correctamente, donde el JS se ejecuta, incluyendo:

1. **Motor de JavaScript (JavaScript Engine)**: Traduce el código a lenguaje de máquina. Ejemplos: V8 (Chrome y Edge), SpiderMonkey (Firefox) y JavaScriptCore (Safari).
2. **APIs del entorno**: Proveen funcionalidades adicionales como manipulación del DOM, manejo de eventos, temporizadores, peticiones HTTP, almacenamiento local y geolocalización.
3. **Modelo de concurrencia (Event Loop)**: Gestiona tareas asíncronas sin bloquear el hilo principal.

Los navegadores web integran en el entorno del JS, permitiendo que JavaScript modifique contenido dinámico, interactúa con usuarios, se comunique con servidores y almacene datos localmente. El navegador actúa como un sistema operativo miniatura para Java. Esto hace posible el desarrollo de aplicaciones web avanzadas.

Además, los navegadores modernos ofrecen herramientas de desarrollo (DevTools) para trabajar con JavaScript, como:

- **Consola**: Ejecutar código, ver errores y depurar.
- **Depurador**: Colocar breakpoints, inspeccionar variables y analizar el flujo del código.
- **Inspector del DOM**: Modificar HTML y analizar interacciones con JavaScript.
- **Network**: Monitorear peticiones HTTP y rendimiento.
- **Performance**: Evaluar tiempos de ejecución y optimizar el código.
- **Application/Storage**: Gestionar almacenamiento local y cookies.

Es por esto, que el entorno de ejecución de JavaScript, junto con las herramientas de desarrollo de los navegadores, es esencial para crear y optimizar aplicaciones web modernas.

b. Describe las herramientas de desarrollo que los navegadores ofrecen para trabajar con JavaScript.

Las herramientas de desarrollo no solo nos permiten lograr la visualización del código, sino que además nos permite su correcta operación y ejecución en vivo.

Para esto los navegadores como chrome o edge nos facilitan las siguientes herramientas:

1. Consola o console : herramienta que nos permite interacción con el entorno de ejecución del código incluyendo el motor que interpreta el código

1.1. REPL Lectura-Evaluación-Impresión al escribir javascript se ejecuta al instante

1.2. Visualización de errores sintaxis o de ejecución (en rojo) con el enlace directo a la línea de código que falló.

1.3.Depuración básica utilizamos console.log (), console.table() o console.error permitiendo imprimir mensajes y rastrear el flujo de datos.

2. Panel de fuentes Sources / Debugger: Permite gestionar archivos, establecer *breakpoints* (puntos e interrupción) para pausar código y analizar el estado de variables en tiempo real sin usar console.log()

2.1. Breakpoints (Puntos de interrupción): se logra pausar la ejecución del código en una línea específica para ver qué está ocurriendo exactamente en ese momento.

2.2. Watch & Scope: Permite observar el valor de las variables en tiempo real y entender qué hay en el "Scope" (alcance) global o local.

2.3. Call Stack (Pila de llamadas): muestra el rastro de breadcrumbs o elementos de navegación secundaria en sitios web donde se logra visualización jerárquica desde la página inicial principal hasta la web acctual ej: Inicio > Categoría > Producto

3. Panel de Red (Network): Formato crucial para entender la relación de JavaScript con el mundo exterior (APIs y Servidores).

3.1. Monitoreo de peticiones: permite visualización cuando el código utiliza usa fetch o AJAX.

3.2. Fetch: API moderna, usa Promesas, sintaxis fetch(url).then(...)

3.3.AJAX: Tecnología más antigua, requiere más código, tradicionalmente usa XMLHttpRequest

3.4. Inspección de respuestas: Permite revisar si el JSON que devuelve el servidor es correcto o si hubo un error de conexión (404, 500).

3.5. Simulación de velocidad: permite limitar la velocidad de internet para ver cómo se comporta el código en conexiones lentas.

4. Panel de Aplicación (Application / Storage)

JavaScript no solo procesa datos, sino también los guarda. Este panel gestiona la memoria persistente del navegador:

4.1. Local Storage y SessionStorage: Para guardar datos que no expiran o que duran solo una sesión.

4.2. Cookies: Para revisar y editar las cookies guardadas. las cookies son pequeños archivos de texto que los sitios envían y almacenan para recordar información previa de visita ejem formularios o contraseñas

4.3. IndexedDB: Para manejar bases de datos más complejas dentro del navegador.

Es una api y un sistema de base de datos noSQL integrado en el navegador diseñado para almacenar grandes cantidades de datos estructurados puede funcionar sin conexión logrando búsquedas de alto rendimiento

5. Panel de Rendimiento (Performance)

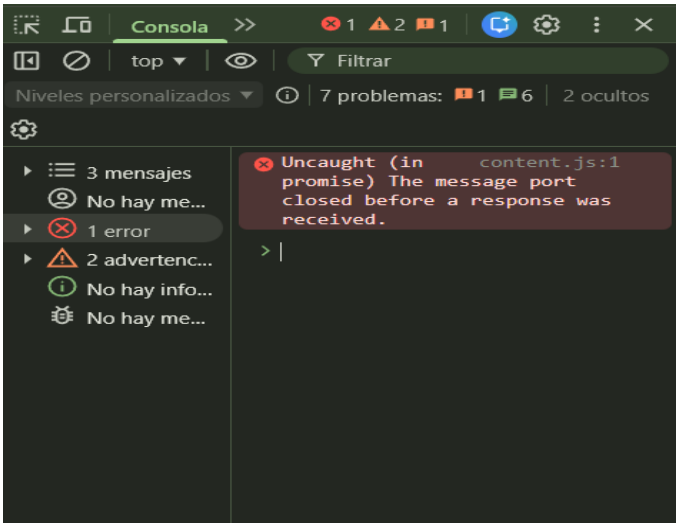
Dentro del panel se analiza el **Evento Loop o bucle de eventos** en acción.

5.1. Grabación de perfil: Registra todo lo que ocurre en el navegador durante unos segundos.

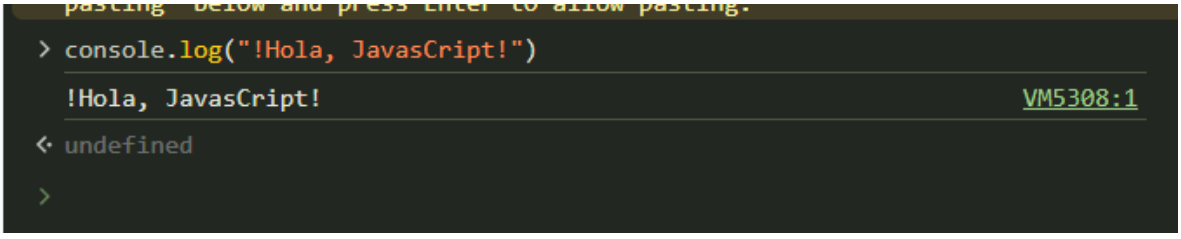
5.2. Análisis de frames: ayuda a detectar por qué una animación se ve entrecortada (si JavaScript está bloqueando el hilo principal por demasiado tiempo).

2. Explorando la Consola del Navegador

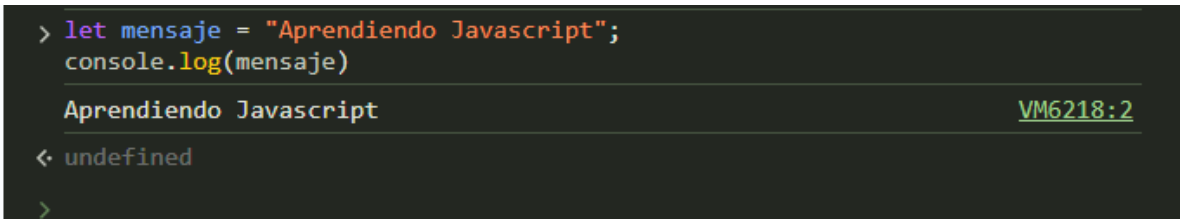
a. Abre las herramientas de desarrollador del navegador (F12 o Ctrl + Shift + I) y accede a la pestaña Console.



b. Escribe y ejecuta un console.log("¡Hola, JavaScript!") en la consola, toma una captura de pantalla.



c. Declara una variable en la consola (let mensaje = "Aprendiendo JavaScript";) y muestra su contenido con console.log(mensaje), toma una captura de pantalla.



3. Identificación de Capacidades y Limitaciones

a. Investiga y anota al menos tres cosas que JavaScript puede hacer dentro del navegador.

1. Manipulación del DOM (Modificar contenido y estilo en tiempo real)

Esta es quizás la función más común. El navegador crea una estructura de árbol de todos los elementos HTML de tu página, llamada DOM (Document Object Model). JavaScript tiene el poder de acceder a este árbol y modificarlo sobre la marcha.

2. Comunicación Asíncrona con el Servidor (Fetch / AJAX)

Antiguamente, si querías ver datos nuevos, tenías que recargar toda la página web. JavaScript permite enviar y recibir datos desde un servidor en "segundo plano" (asíncronamente) sin interrumpir lo que el usuario está haciendo.

3. Generación de Gráficos y Animaciones (API Canvas)

JavaScript no solo maneja texto y botones; también puede dibujar píxeles directamente en la pantalla utilizando la etiqueta <canvas> de HTML5.

b. Investiga y anota al menos dos cosas que JavaScript no puede hacer en el navegador y explica por qué.

1. No puede acceder directamente a archivos del sistema del usuario

JavaScript no puede leer ni modificar archivos del computador del usuario sin su permiso. Esto se debe a razones de seguridad, ya que permitir acceso libre podría poner en riesgo la privacidad y la información personal del usuario.

2. No puede ejecutar programas o acceder al sistema operativo

JavaScript en el navegador no puede instalar programas, ejecutar comandos del sistema ni acceder a configuraciones del equipo. Esto también es una medida de seguridad para evitar que páginas web puedan dañar el sistema del usuario.

4. Reflexión Final

a. Explica la importancia de la consola de desarrollador para los programadores.

Es muy importante, ya que permite generar interactividad entre el usuario, el lenguaje y la máquina. ya que tiene una importancia tanto empírica, como pedagógica. JavaScript es una herramienta muy poderosa para crear páginas web dinámicas e interactivas dentro del navegador, pero está limitado por medidas de seguridad que protegen la privacidad y el sistema del usuario, evitando accesos directos al hardware o archivos personales.

b. Describe tu experiencia utilizando la consola y qué descubriste en el proceso.

Nuestra experiencia en si no ha sido la más cómoda a la hora de ejecutar una línea de comando, nos ha dado como resultado undefined, forzándola a reiniciar por defecto y cargar nuevamente el devtools.

Se comprende que viene con una configuración por defecto. y quizás sea esta la causa del undefined. Investigando en la red nos ha dado una solución, pero no se mantiene en el tiempo. Necesitamos más investigación al respecto para comprobar sus funciones. La consola en sí tiene distintas opciones de configuración. por lo tanto no solo es ejecutar una línea de código. también deberíamos investigar en la configuración correcta para sus ejecuciones.