

Laboratorio práctico: Depurar una aplicación JavaScript en las herramientas de desarrollo de Chrome



Tiempo estimado necesario: 15 minutos

Lo que aprenderás

En este laboratorio, adquirirás comprensión sobre cómo depurar código en JavaScript. Aprenderás sobre bloques try y catch y obtendrás información sobre cómo y dónde utilizar estos bloques para asegurarte de que tu código funcione sin problemas.

Objetivos de aprendizaje

Después de completar este laboratorio, podrás:

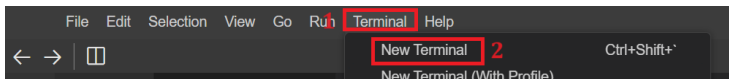
- **Entrada del usuario:** Proporciona dos campos de entrada para que los usuarios ingresen valores numéricos.
- **Ejecución de operaciones:** Multiplica los dos valores de entrada cuando se hace clic en el botón "Realizar operación".
- **Manejo de errores:** Verifica si los valores ingresados son números válidos antes de realizar la operación de multiplicación. Si los valores no son números válidos, muestra un mensaje de error en lugar del resultado.
- **Depuración:** Incluye una declaración de depurador dentro de la función multiply() para pausar la ejecución y permitir a los desarrolladores inspeccionar el código, las variables y el flujo de ejecución utilizando las herramientas de desarrollo del navegador.

Requisitos previos

- Conocimientos básicos de HTML.
- Comprensión básica de la consola y la depuración.
- Navegador web con consola (Chrome DevTools, Firefox Console, etc.).

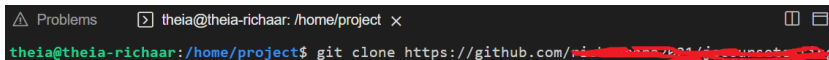
Paso 1: Configurando el entorno

1. Primero, necesitas clonar tu repositorio principal en el **Entorno de Skills Network** que creaste en el primer laboratorio y donde has subido todos tus archivos y carpetas de laboratorios anteriores. Sigue los pasos dados:
 - Haz clic en la terminal en la esquina superior derecha y luego selecciona **Nueva Terminal**.

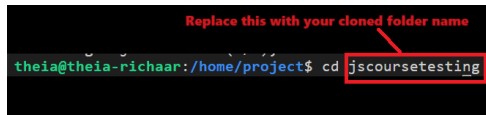


- Realiza el comando git clone escribiendo el comando dado en la terminal.
git clone <github-repository-url>

Nota: Coloca tu propio enlace de repositorio de GitHub en lugar de <github-repository-url>.



- El paso anterior clonará la carpeta de tu repositorio de GitHub dentro de la carpeta del proyecto en el explorador. También verás múltiples carpetas dentro de la carpeta clonada.
- Ahora, necesitas navegar dentro de la carpeta clonada. Para esto, escribe el comando dado en la terminal:
cd <repository-folder-name>



Nota: Escribe el nombre de tu carpeta clonada en lugar de <repository-folder-name>. Realiza git clone si has cerrado sesión en el **Entorno de Skills Network** y no puedes ver ningún archivo o carpeta después de iniciar sesión.

2. Ahora selecciona la carpeta **cloned Folder Name**, haz clic derecho sobre ella y selecciona **Nueva Carpeta**. Ingresas el nombre de la carpeta como **DebugCode**. Esto creará la carpeta para ti. Luego selecciona la carpeta **DebugCode**, haz clic derecho y selecciona **Nuevo Archivo**. Ingresas el nombre del archivo como **debug_code.html** y haz clic en Aceptar. Esto creará tu archivo HTML.
3. Crea una estructura de plantilla básica de un archivo HTML agregando el contenido proporcionado a continuación.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Ejemplo de Depuración</title>
</head>
<body>
  <h1>Ejemplo de Depuración</h1>
  <label for="input1">Ingresa Número 1:</label>
  <input type="number" id="input1"><br><br>
  <label for="input2">Ingresa Número 2:</label>
  <input type="number" id="input2"><br><br>
  <button onclick="performOperation()">Realizar Operación</button>
  <p id="result"></p>
</body>
</html>
```

Nota: Después de pegar el código, guarda tu archivo.

4. Ahora selecciona la carpeta **DebugCode** nuevamente, haz clic derecho y selecciona **Nuevo Archivo**. Ingresas el nombre del archivo como **debug_code.js** y haz clic en Aceptar. Esta acción generará tu archivo JavaScript.
5. Para incluir el archivo js en **debug_code.html**, utiliza la etiqueta de script en el archivo HTML por encima de la etiqueta </body>. Puedes usar el código dado para incluir y guardar el archivo de script.
<script src="/debug_code.js"></script>

Paso 2: Definiendo variables

1. Incluye el código proporcionado en el archivo **debug_code.js**.

```
function performOperation() {
  // Get user input from input fields
  let num1 = parseInt(document.getElementById('input1').value);
  let num2 = parseInt(document.getElementById('input2').value);
  // Check if inputs are valid numbers
  if (!isNaN(num1) && !isNaN(num2)) {
    // Perform the operation
    let result = multiply(num1, num2);
    // Display the result
    displayResult(result);
  } else {
    displayResult('Please enter valid numbers');
  }
}

function multiply(a, b) {
  // Introduce a debugger statement to pause execution
  debugger;
  // Multiply the numbers
  return a * b;
}
```


```
function displayResult(result) {  
  // Display the result in the paragraph element  
  const resultElement = document.getElementById('result');  
  resultElement.textContent = `The result is: ${result}`;  
}
```

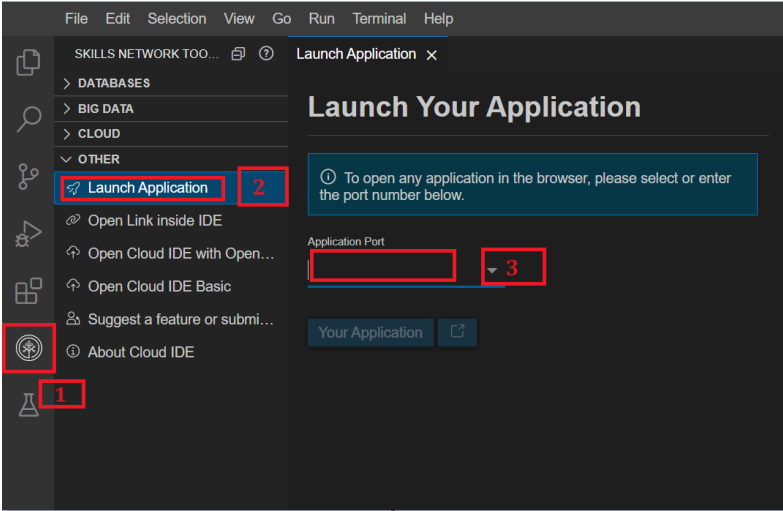
2. El código anterior tiene las siguientes partes:
- Función `performOperation()`:
 - Recupera los valores numéricos ingresados por el usuario desde los campos de entrada HTML (`input1` y `input2`).
 - Valida que los valores ingresados sean números válidos.
 - Si ambos valores son números válidos, llama a la función `multiply()` pasando estos valores; de lo contrario, muestra un mensaje de error.
 - Función `multiply()`:
 - Incluye una instrucción de depuración para pausar la ejecución del código en este punto con fines de depuración.
 - Multiplica dos números de entrada (`a` y `b`) y devuelve el resultado.
 - Función `displayResult()`:
 - Muestra el resultado de la multiplicación o un mensaje de error en un elemento de párrafo designado (`resultElement`) en la página web.

Paso 3: Depurar el código para ver el flujo del código

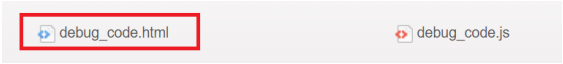
1. Para ver tu página HTML, haz clic derecho en el archivo `debug_code.html` después de seleccionarlo, luego selecciona "Abrir con Live Server".
2. El servidor debería comenzar en el puerto 5500, indicado por una notificación en la parte inferior derecha.



3. Haz clic en el botón de Skills Network a la izquierda (consulta el número 1), se abrirá la "Caja de herramientas de Skills Network". Luego haz clic en Lanzar Aplicación (consulta el número 2). Desde allí, ingresa el número de puerto como 5500 en el número 3 y haz clic en este botón 



4. Se abrirá tu navegador predeterminado donde verás el nombre de la carpeta `cloned-folder-name`. Haz clic en ese nombre de carpeta `cloned-folder-name`. Después de hacer clic, verás múltiples nombres de carpetas, entre esos nombres de carpetas haz clic en la carpeta `DebugCode`. Verás archivos relacionados con esta carpeta donde nuevamente harás clic en el archivo `debug_code.html` como se muestra a continuación.



Nota: Guarda tu archivo después de pegar el código. Si editas tu código, actualiza tu navegador en el puerto 5500. No es necesario relanzar la aplicación.

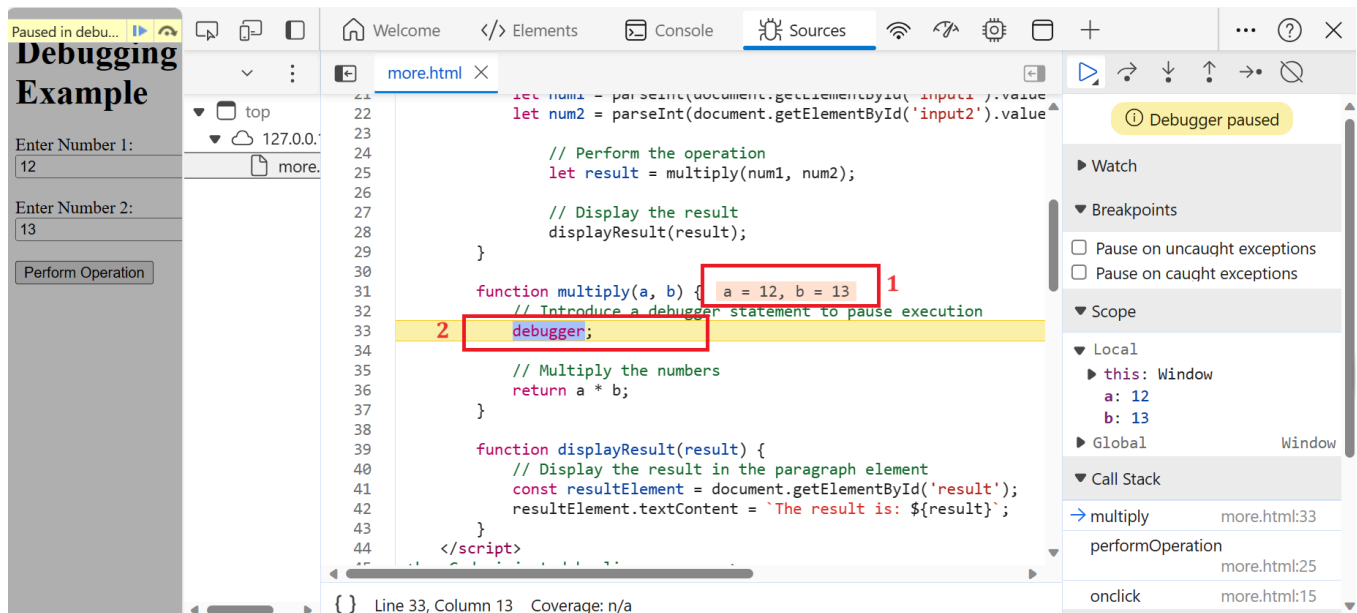
5. Verás la salida como a continuación.

Debugging Example

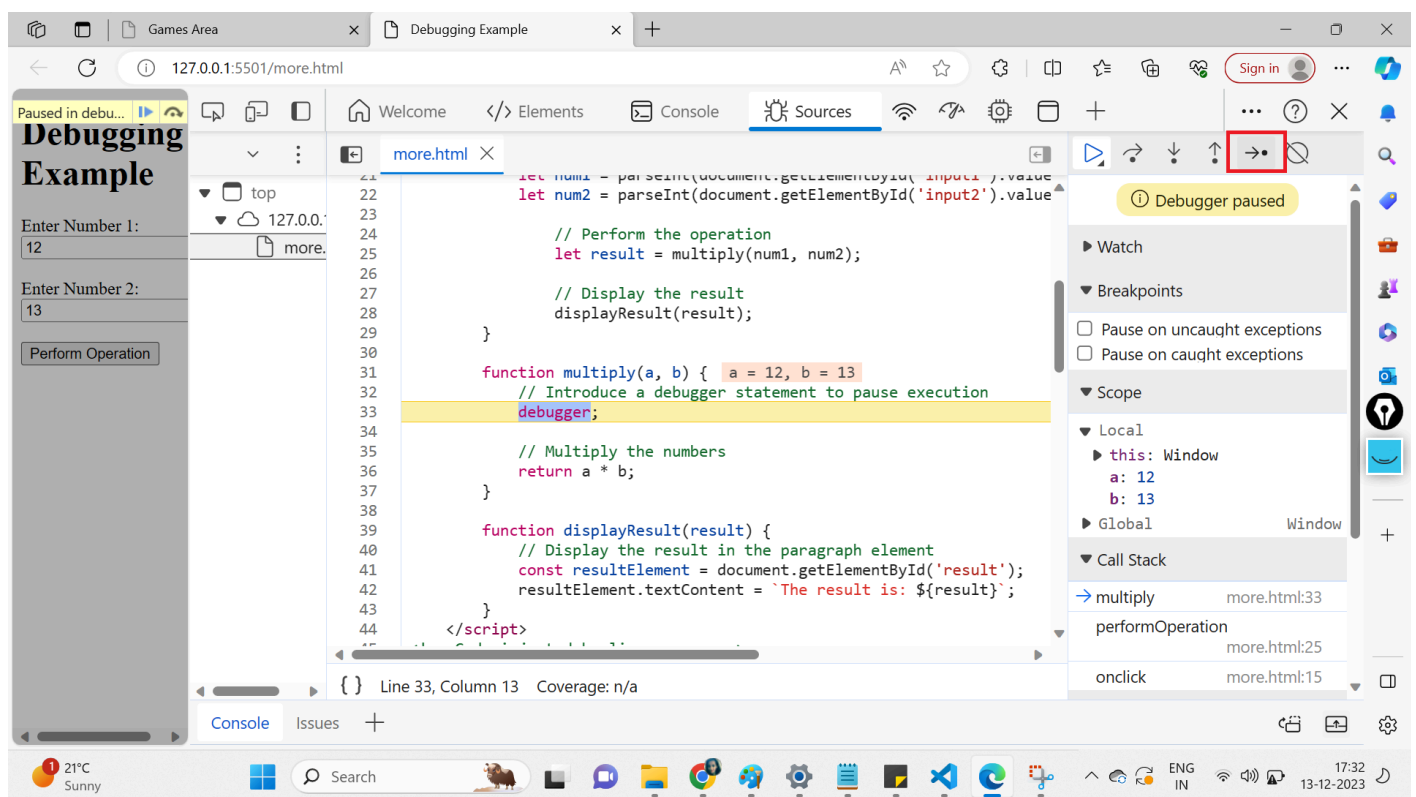
Enter Number 1:

Enter Number 2:

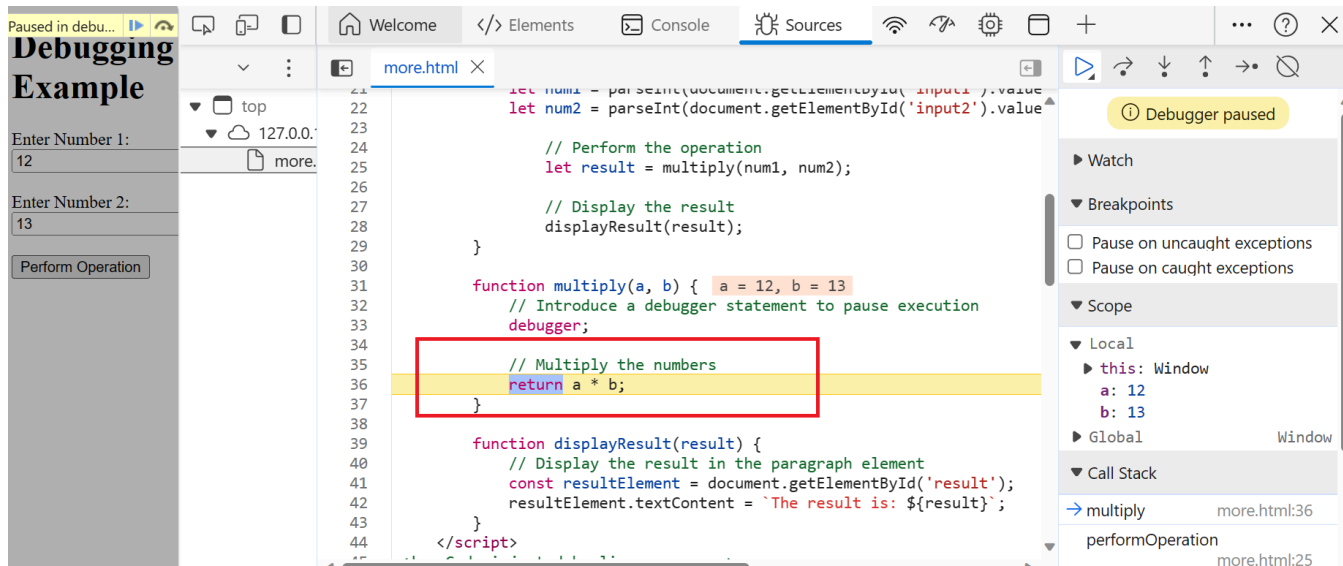
6. Ahora ingresa los números en los cuadros de entrada disponibles y haz clic en el botón "Realizar operación".
7. Ahora, para ver el flujo de este código, haz clic derecho en la misma ventana de tu navegador donde se está mostrando la salida y luego selecciona "Inspeccionar".
8. Tan pronto como hagas clic en el botón, verás la pantalla como si se hubiera pausado, justo como en la captura de pantalla proporcionada. Además, el valor número 1 resaltado en el cuadro rojo en la captura de pantalla muestra que se han recibido dos números en las variables `a` y `b`. Junto con eso, el número 2 indica dónde ha ocurrido la pausa en el punto de depuración, que es el puntero actual de tu código.



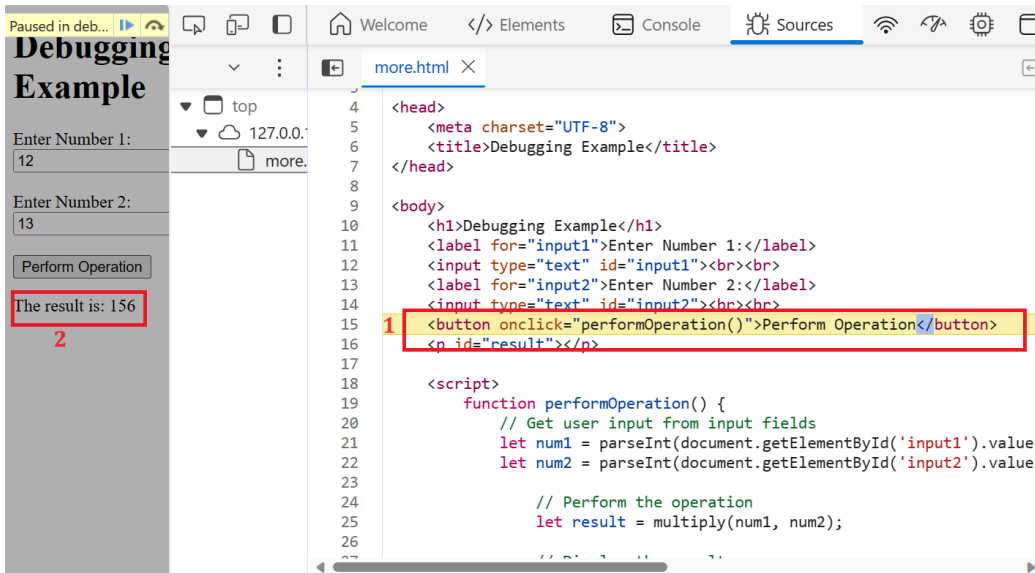
9. Luego, necesitas hacer clic en la flecha hacia adelante resaltada con un cuadro rojo en la captura de pantalla dada.



10. Tan pronto como hagas clic en ese botón, tu puntero actual se moverá a `return a*b`, indicando que el flujo del código ha llegado a este punto.



11. Haz clic en el botón hacia adelante nuevamente, y el puntero se moverá hacia el número 1 en la captura de pantalla dada, indicando que se está llamando a la función `displayResult()`. Además, el número 2 indica que `num1` y `num2` tienen valores de 12 y 13 respectivamente.



15. Después de esto, saldrá de la zona de depuración.

Paso 4: Realizar comandos de Git

1. Realiza `git add` para agregar los últimos archivos y carpetas en el entorno de git.

```
git add --a
```

- Asegúrate de que la terminal tenga la ruta como sigue:



2. Luego, realiza `git commit` en la terminal. Al realizar `git commit`, la terminal puede mostrar un mensaje para configurar tu `git config --global` para `user.name` y `user.email`. Si es así, entonces también necesitas realizar el comando `git config` para `user.name` y `user.email` como se indica.

```
git config --global user.email "you@example.com"
```

```
git config --global user.name "Your Name"
```

Luego realiza el comando de commit como se indica:

```
git commit -m "message"
```

3. A continuación, realiza `git push` escribiendo el comando dado en la terminal.

```
git push origin
```

- Después del comando de push, el sistema te pedirá que ingreses tu nombre de usuario y contraseña. Ingresar el nombre de usuario de tu cuenta de GitHub y la contraseña que creaste en el primer laboratorio. Después de ingresar las credenciales, todas tus últimas carpetas y archivos se enviarán a tu repositorio de GitHub.

Tarea de práctica

1. Necesitas realizar operaciones aritméticas como suma, multiplicación y división simultáneamente utilizando la misma función.
2. Además, necesitas verificar el flujo del código, que dependerá de la operación aritmética que estés realizando primero.
3. También, intenta asignar un valor en forma de caracteres y observa cómo se muestra este valor utilizando el depurador.

Resumen

1. Has aprendido cómo depurar código para ver el flujo paso a paso del programa.
2. También has aprendido que el uso de la palabra clave `debugger` también te permite ver el flujo de los valores de entrada que se almacenan en variables en JavaScript.

© IBM Corporation. Todos los derechos reservados.