02 Learning Activity: JavaScript Debugging

Overview

Debugging is the process of finding and fixing errors in your code. This is an essential skill for any developer. You will learn how to use the browser's DevTools to debug your JavaScript code.

Prepare

Types of Errors

There are three types of errors that you will encounter when writing JavaScript code:

- **Syntax Errors** – These are errors that occur when you write code that does not follow the rules of the JavaScript language. These errors are caught by the browser and will prevent your code from running.
- **Runtime Errors** – These are errors that occur while your code is running. These errors can be caused by a variety of issues, such as trying to access a property of an object that does not exist or calling a function that does not exist.
- **Logical Errors** – These are errors that occur when your code does not produce the expected result. These errors can be difficult to find because they do not cause the browser to throw an error.

Activity Instructions

This activity focuses on debugging techniques for the browser.

Step 1: Setup Files

1. In your **week02** subfolder, add a file named "**debugging.html**".
2. In the **debugging.html** file, type `!` and then press the `tab` key on your keyboard. This action will automatically add the basic components of an HTML document.
3. Add a subfolder named "**scripts**" to the **week02** folder, add a file named "**debugging.js**".

4. In **debugging.html** file, add a `script` reference to the **debugging.js** JavaScript file.

▶ Check Your Understanding

Step 2: HTML

1. Add the following elements to the HTML `body` of your **debugging.html** page:

```
<h1>JavaScript Debugging using DevTools</h1>
<p>Area of circle with radius <span id="radius"></span>: <span id="area"
```

Step 3: JavaScript

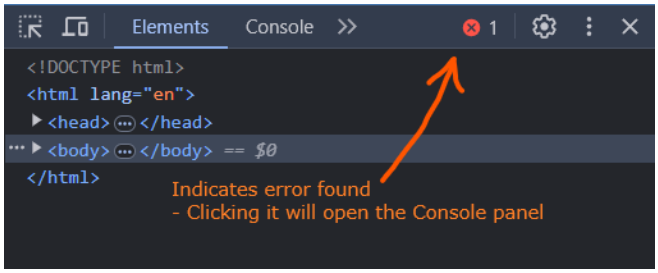
1. Add the following JavaScript statements to your **debugging.js** file:

```
const radiusOutput = document.getElementById('radius');
const areaOutput = document.querySelector('area');

let area = 0;
const PI == 3.14159;

const radius = 10;
area = PI * radius * radius;
radiusOutput = radius;
areaOutput = area;

radius = 20;
area = PI * radius * radius;
radiusOutput = radius;
areaOutput = area;
```

Step 4: Debugging

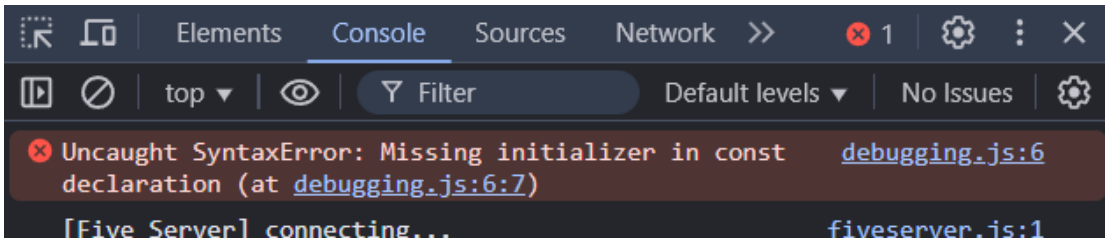
Video Demonstration: ▶ Debugging JavaScript using DevTools Activity Walkthrough – [5.56 minutes]

1. Using Live/Five Server, open your **debugging.html** file in a browser.
2. Open the browser's DevTools and open the the **Console** tab.
3. Note any redlined errors in the DevTools console and outlined in Console window. These are syntax or runtime errors. This errors will stop your code from running until they are fixed.



4. Fix the first error outlined in the **debugging.js** file. The image below indicates that the error is on line 6 (**`debugging.js:6`**)

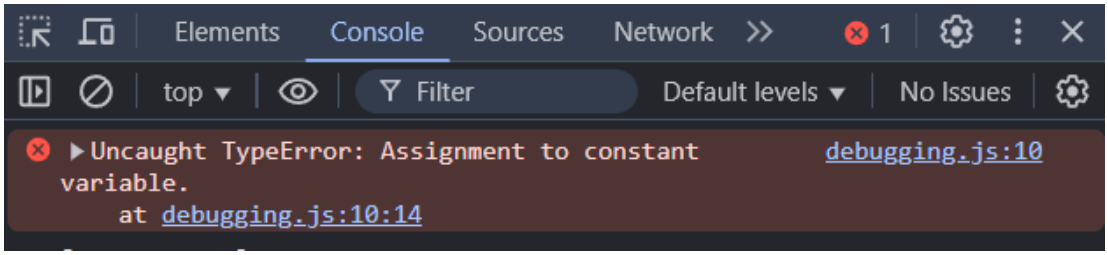
DevTools Console



The comparison operator is confusing the compiler. In VS Code, remove the extra equal sign.

```
const PI = 3.14159;
```

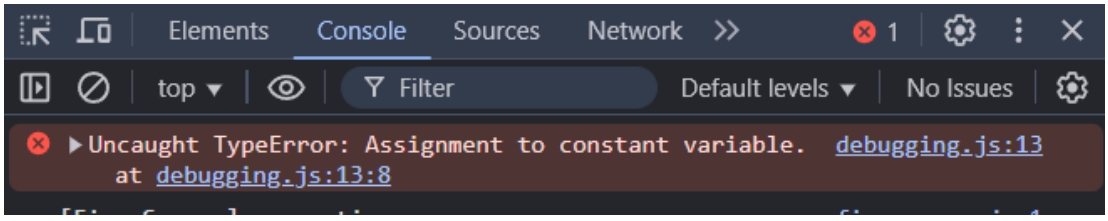
5. In your localhost browser session, refresh the page and check the console for the next error.



This error is referring to the issue of trying to assign a value to an HTML element. This might be confusing given the error message says "Assignment to constant variable". Change the line of code for both elements to use the **`textContent`** property.

```
radiusOutput.textContent = radius;
areaOutput.textContent = area;
```

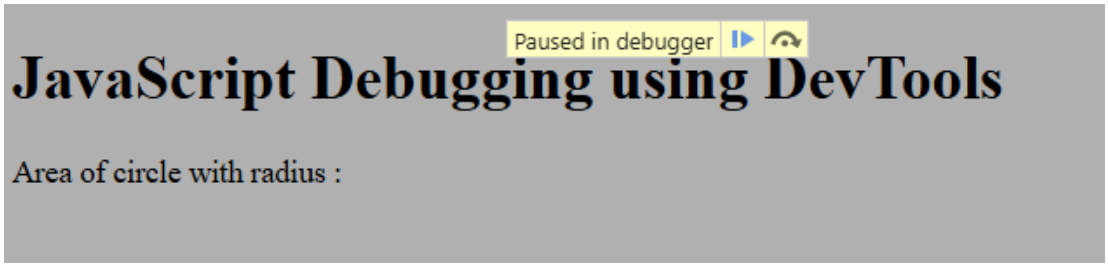
6. In your localhost browser session, refresh the page and check the console for the next error.



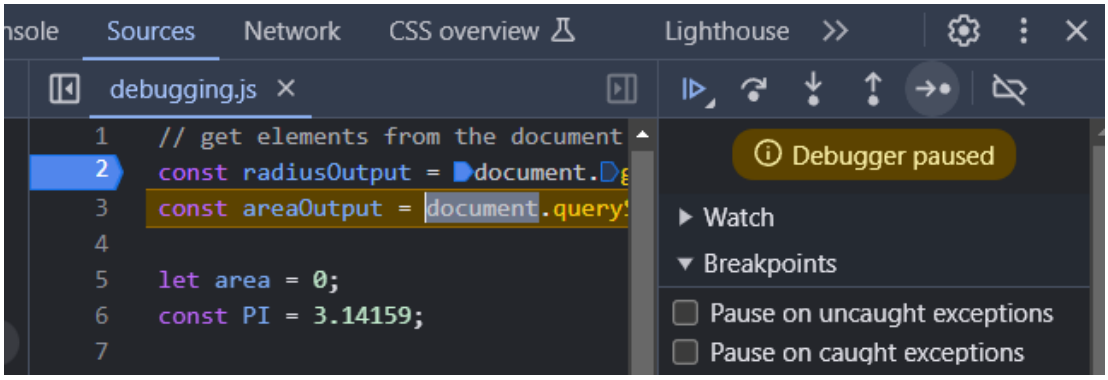
This error's message is accurate. The code is attempting to assign a new value to the **`radius`** variable which was declared as a constant. Change the variable declaration to a **`let`** instead of **`const`**.

```
let radius = 10;
```

7. Fix any other errors found in the console.
8. After you get output to the page that seems correct, open the **Sources** panel in DevTools.
9. Practice "step debugging" through the JavaScript code by setting a breakpoint – click on the line number where you want to execute a pause in the program.
10. Refresh the page to trigger the code to run again. The body will be grayed out and indicate that the page load was "Paused in debugger"



11. Use the **Step** →• button to move through the code line by line.



12. At any time you can pass your mouse over a variable to view its current assignment value.
13. Continue to step through the code until you have a good understanding of how the code is executing.

Week 02 Home