JavaScript Debugging

Back to JS page

Table of Contents

- JavaScript Debugging
  - Using console.log()
    - Example 1
  - The debugger Statement
    - Example 2
  - Try Catch for Debugging
    - Example 3
  - Reading Error Messages
    - Example 4
  - Common Beginner Mistakes
    - Example 5
  - A Simple Debugging Checklist
    - Example 6
  - Document
  - Reference

Using console.log()

The `console.log()` method is the simplest way to debug JavaScript code. It writes messages to the browser's console, allowing you to inspect variable values and track program flow without interrupting execution.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Debugging</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Using console.log()</h4>
<p>Open the browser console (F12) to see the debug output.</p>
<p id="demo"></p>

<script>
let a = 5;
let b = 6;
let c = a + b;

console.log("Value of a: " + a);
console.log("Value of b: " + b);
console.log("Value of c: " + c);

document.getElementById("demo").innerHTML = "Result: " + c;
</script>

</body>
</html>
```

Example 1

Result View Example

The debugger Statement

The `debugger` statement stops the execution of JavaScript and calls the debugging function (if available). It acts like a breakpoint in the code. If no debugging is available, the debugger statement has no effect.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Debugging</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>The debugger Statement</h4>
<p id="demo"></p>

<script>
let x = 15 * 5;

// debugger; will pause execution here if DevTools is open
console.log("x = " + x);

document.getElementById("demo").innerHTML =
  "x = " + x + " (check console)";
</script>

</body>
</html>
```

Example 2

Result View Example

Try Catch for Debugging

Using `try...catch` is a powerful debugging technique. It allows you to catch errors gracefully, display meaningful messages, and continue execution instead of crashing. This is essential for identifying and fixing problems.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Debugging</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Try Catch for Debugging</h4>
<p id="demo"></p>

<script>
let text = "";

try {
  // Intentionally calling an undefined function
  myFunction();
  text += "This line is skipped.";
}
catch(err) {
  text += "Debug Info:<br>";
  text += "Error Name: " + err.name + "<br>";
  text += "Error Message: " + err.message + "<br>";
  text += "Check your code for the error above.";
}

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 3

Result View Example

Reading Error Messages

Error messages in the console are your best debugging tool. They tell you exactly what went wrong and where. Learning to read and understand error messages is an essential debugging skill.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Debugging</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Reading Error Messages</h4>
<p id="demo"></p>

<script>
let text = "";

try {
  let result = undefinedVariable + 5;
}
catch(err) {
  text += "Error Type: " + err.name + "<br>";
  text += "Description: " + err.message + "<br><br>";
  text += "Tip: The error message tells you what and where.<br>";
  text += "Look in the console (F12) for line numbers.";
}

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 4

Result View Example

Common Beginner Mistakes

Some of the most common debugging challenges for beginners include misspelled variable names, using `=` instead of `==` or `===`, and forgetting to close brackets or quotes.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Debugging</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Common Beginner Mistakes</h4>
<p id="demo"></p>

<script>
let text = "";

// Mistake 1: Assignment (=) instead of comparison (==)
let x = 10;
if (x = 5) {   // This assigns 5 to x, always true!
  text += "Mistake: '=' instead of '=='. ";
  text += "x is now: " + x + "<br><br>";
}

// Mistake 2: Misspelled variable name
let total = 100;
try {
  text += "total: " + total + "<br>";
  // text += tota1;  // This would cause a ReferenceError
  text += "(Misspelled variables cause ReferenceErrors)<br><br>";
}
catch(err) {
  text += err.name + ": " + err.message + "<br>";
}

// Mistake 3: Case sensitivity
let myVar = "Hello";
text += "myVar: " + myVar + "<br>";
text += "(JavaScript is case-sensitive: myvar !== myVar)";

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 5

Result View Example

A Simple Debugging Checklist

When your code doesn't work, follow this simple debugging checklist:

1. **Check the console** for error messages
2. **Use** `console.log()` to inspect variable values
3. **Check spelling** of variable and function names
4. **Check data types** — is it a string or a number?
5. **Use** `try...catch` to handle unexpected errors

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Debugging</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>A Simple Debugging Checklist</h4>
<p id="demo"></p>

<script>
let text = "";

// Step 1: Check browser console (F12)
console.log("Debugging Checklist Demo");

// Step 2 & 3 & 4: Inspect variables, check spelling and types
let num1 = "5";
let num2 = 10;

console.log("num1 type:", typeof num1);  // string
console.log("num2 type:", typeof num2);  // number

text += "num1 = \"" + num1 + "\" (type: " + typeof num1 + ")<br>";
text += "num2 = " + num2 + " (type: " + typeof num2 + ")<br><br>";

// Step 5: Try-catch for safety
try {
  let result = Number(num1) + num2;  // Explicit conversion
  text += "Number(num1) + num2 = " + result;
}
catch(err) {
  text += "Error: " + err.message;
}

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 6

Result View Example

Document

Document in project

You can Download PDF file.

Reference

- W3Schools JavaScript Debugging