debugging in JavaScript

It is common to have errors w due to syntax or logical. These errors create a lot of ambiguity in the logic and understanding of both users and programmers. There can also be errors in the code which can remain invisible to the programmer's eye and can create havoc. To identify these errors we need Debuggers that can go through the entire code or program, identify the errors and also fix them.

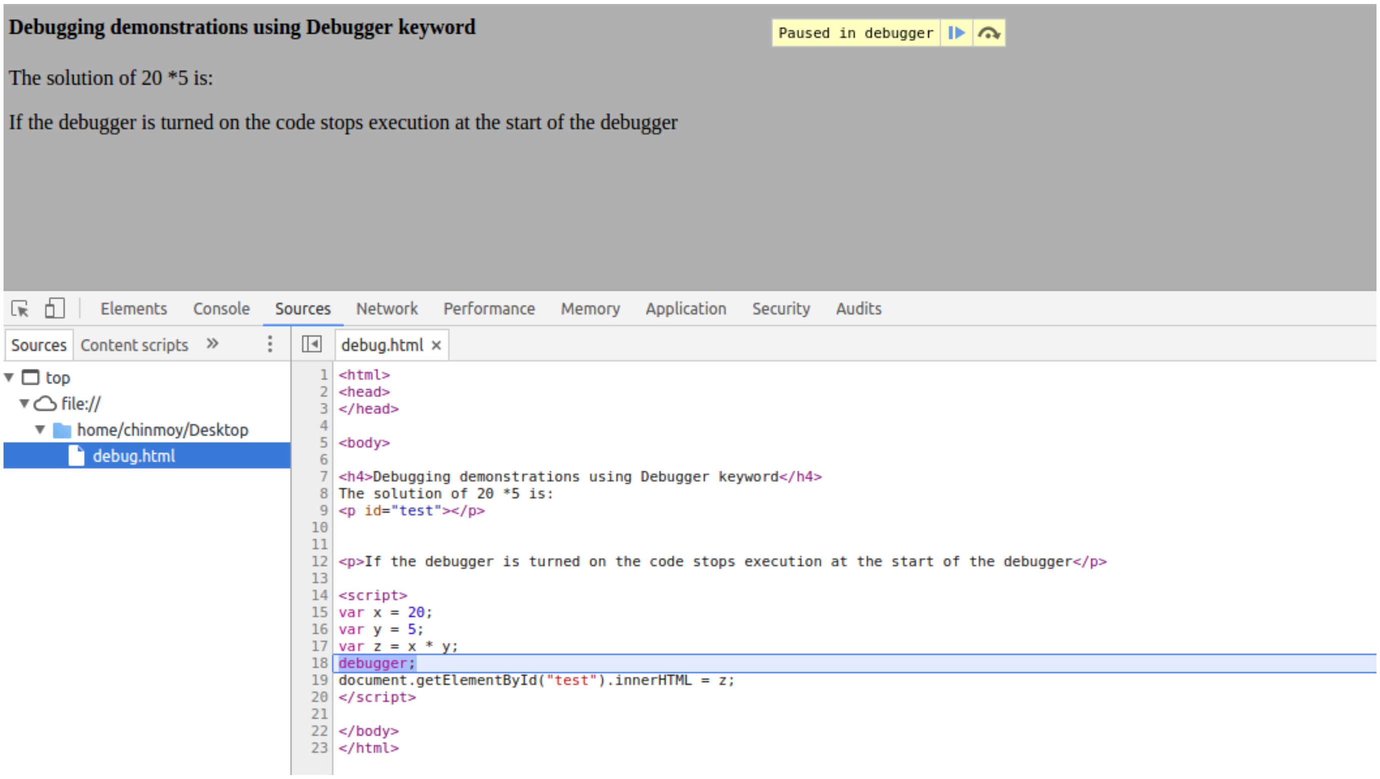
1) debugger
The debugger keyword is used in the code to force stop the execution of the code at a breaking point and calls the debugging function. The debugger function is executed if any debugging is needed at all else no action is performed.
Let's see JavaScript program on debugging:

```
<h4>Debugging demonstrations using Debugger keyword</h4>
    The solution of 20 * 5 is:
        <p id="test"></p>


<p>If the debugger is turned on the code stops
execution at the start of the debugger</p>


        var x = 20;
var y = 5;
var z = x * y;
debugger;
document.getElementById("test").innerHTML = z;
```

Output:



Debugging demonstrations using Debugger keyword

The solution of 20 *5 is:

If the debugger is turned on the code stops execution at the start of the debugger

```
<html>
<head>
</head>

<body>

<h4>Debugging demonstrations using Debugger keyword</h4>
The solution of 20 *5 is:
<p id="test"></p>


<p>If the debugger is turned on the code stops execution at the start of the debugger</p>

<script>
var x = 20;
var y = 5;
var z = x * y;
debugger;
document.getElementById("test").innerHTML = z;
</script>

</body>
</html>
```

Previously implementing debuggers was tough but with time and with the advent of the modern browsers, various built-in debuggers came into implementation. As previously mentioned that setting up of breakpoints breaks the line of code and flows the control to another area from where it is called. This can be done in the debugging window of the browser. Also setting up of breakpoints in javascript acts similar to the breakpoints in Java where the execution of the code stops and examination of the values is performed by the browser. One can resume the flow control after the examination is done. The debugging can be turned on or off depending upon the user's convenience. This all can be done through the "Console" of the debugger menu.

2) Use of console.log() method

There is another way in which the JS values can be displayed in the debugger window.

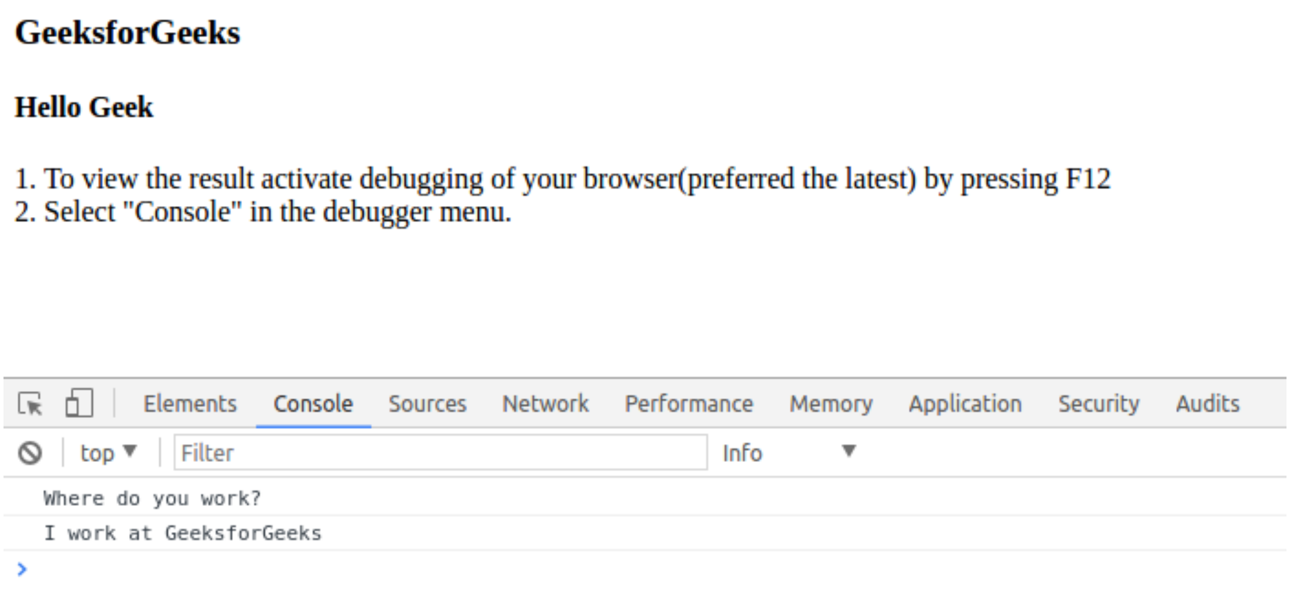
Let’s see JavaScript program using console.log():

```
<h3>GeeksforGeeks</h3>
<h4>Hello Geek</h4>

<p>
1. To view the result activate debugging of your
browser(preferred the latest) by pressing F12 <br>
2. Select "Console" in the debugger menu.
</p>


x = "Where do you work?"
y = "I work at GeeksforGeeks"
console.log(x);
console.log(y);
```

Output:



GeeksforGeeks

Hello Geek

1. To view the result activate debugging of your browser(preferred the latest) by pressing F12
2. Select "Console" in the debugger menu.

```
Where do you work?
I work at GeeksforGeeks
>
```

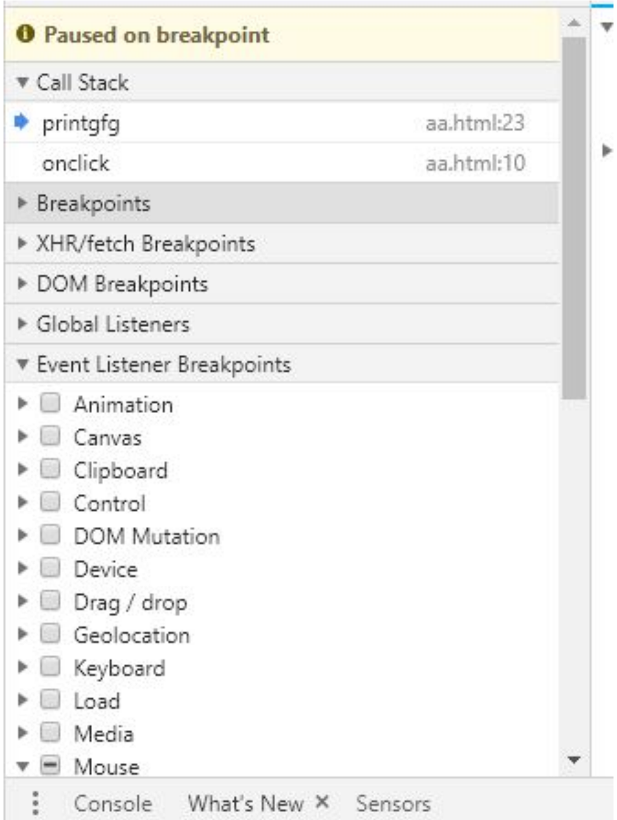
3) Setting Break Points:

The console.log() is a good way to debug errors but setting breakpoint is a faster, efficient and better method. In this method, Breakpoints are set in code which stops the execution of code at that point so that the values of variables can be examined at that time.

Here are some advantages of using Breakpoints over console.log() method:

- In console.log() method user has to understand the code and manually put console.log() at points in the code. But in breakpoints method, we can set breakpoints through Developers tool anywhere in code without even knowing it.
- In console.log() method we have to manually print values of different variables but at a certain breakpoint, all the values of all variables are displayed automatically in Developers tool.

Enter Developers tool section by pressing F12 key and go to sources.
In the source section, select a javascript file and set breakpoints by either selecting from the provided list like DOM breakpoints or Event listener breakpoints which stops the execution of code whenever an event occurs



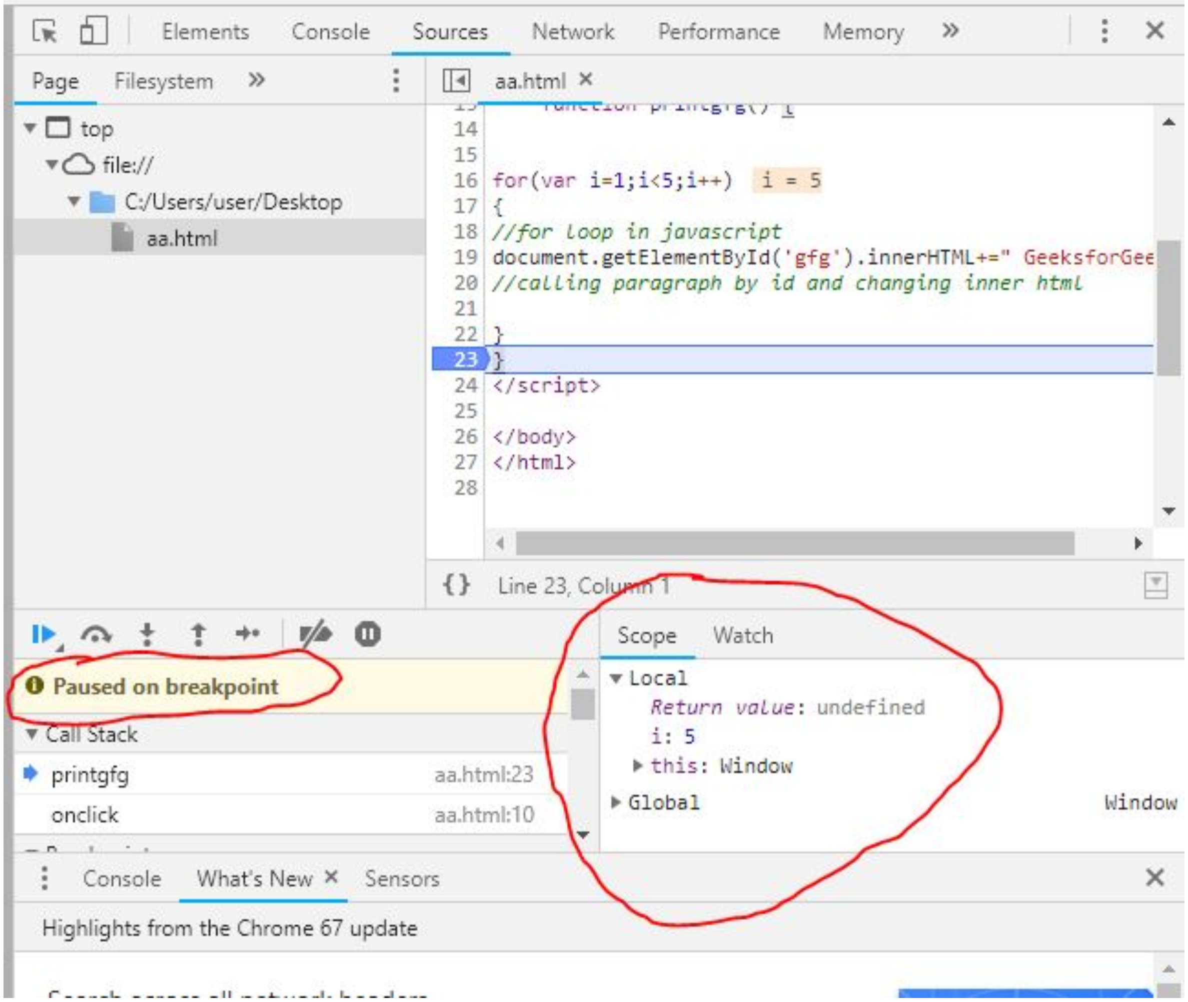
Paused on breakpoint

▼ Call Stack

printgfg	aa.html:23
onclick	aa.html:10

▶ Breakpoints
▶ XHR/fetch Breakpoints
▶ DOM Breakpoints
▶ Global Listeners
▼ Event Listener Breakpoints
▶ □ Animation
▶ □ Canvas
▶ □ Clipboard
▶ □ Control
▶ □ DOM Mutation
▶ □ Device
▶ □ Drag / drop
▶ □ Geolocation
▶ □ Keyboard
▶ □ Load
▶ □ Media
▼ ⊟ Mouse

OR set a breakpoint by simply clicking line number shown in the code. In the following image, a breakpoint is set at line 23. (The code used is same as the above example)



```
13       function printgfg() {
14
15
16  for(var i=1;i<5;i++)  i = 5
17  {
18  //for loop in javascript
19  document.getElementById('gfg').innerHTML+=" GeeksforGee
20  //calling paragraph by id and changing inner html
21
22  }
23  }
24  </script>
25
26  </body>
27  </html>
28
```

Line 23, Column 1

Paused on breakpoint

Call Stack

printgfg	aa.html:23
onclick	aa.html:10

```
▼ Local
    Return value: undefined
    i: 5
  ▶ this: Window
▶ Global                                  Window
```

Highlights from the Chrome 67 update

Search across all network headers

As shown in the image, the code stopped at line 23 and showed all the values of variables in the code at that point in time.

Since we can see the value of I was initially 1 and at the breakpoint, it is 5. Hence GeeksForGeeks was printed 5-1=4 times.

Method 3 is contributed by Kartikay Bhutani.

Recommended Posts:

How to get the first key name of a JavaScript object ?

What are decorators and how are they used in JavaScript ?

JavaScript | JSONP

How to parse URL using JavaScript ?

Loops in JavaScript

Functions in JavaScript

JavaScript | Know the value of GET parameters from URL

JavaScript | Events

JavaScript | Objects

How to use goto in Javascript ?

What is JavaScript >>> Operator and How to use it ?

JavaScript | Generator

Map.entries( ) In JavaScript

JavaScript | Iterator

What does javascript:void(0) mean?

Chinmoy Lenka

Check out this Author's contributed articles.

Article Tags : Misc JavaScript-Misc

Practice Tags : Misc Misc

1

□ To-do □ Done

0

No votes yet.

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos