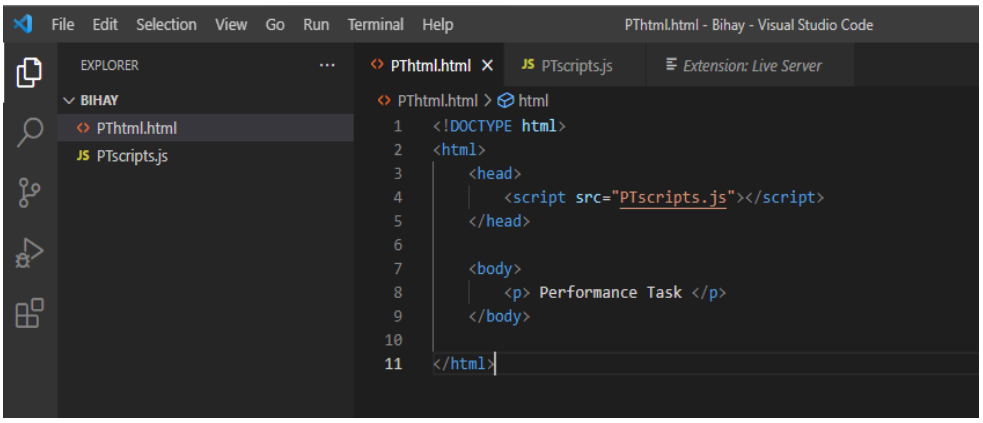
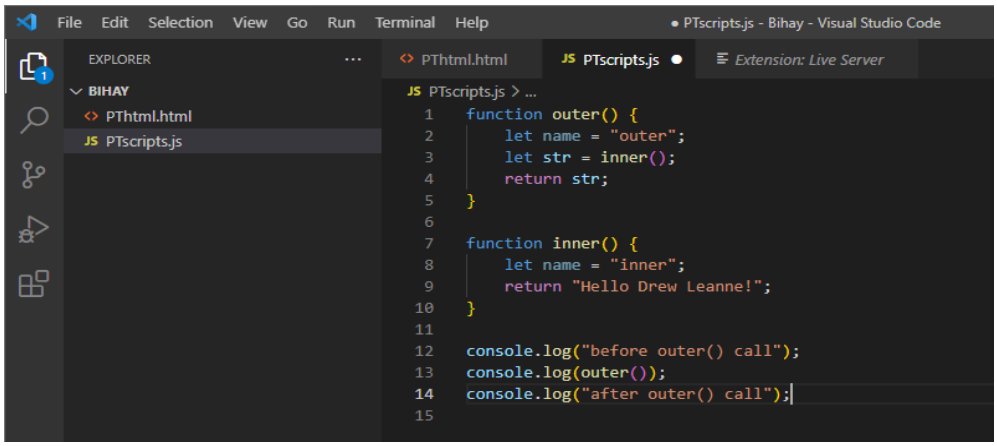
PERFORMANCE TASK: DEBUGGING

1. Create a simple HTML file with the following syntax. Use PThtml as the file name



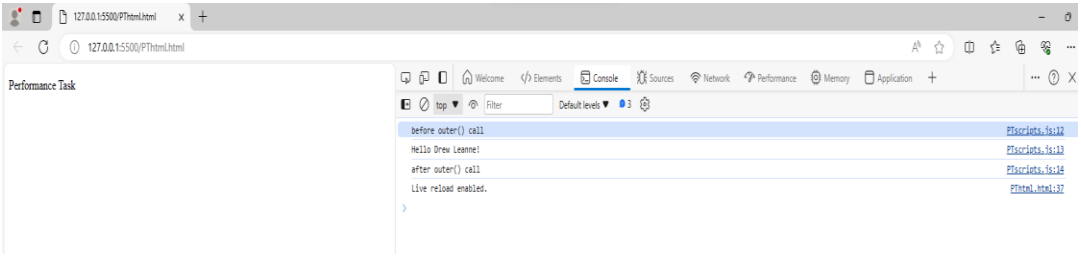
2. Then, create a JS file with the following code. Use the correct file name



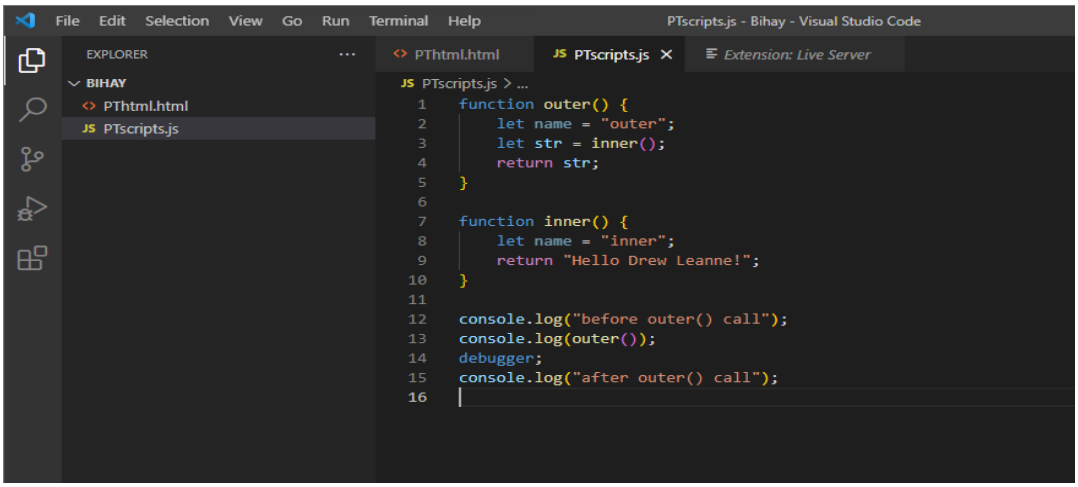
3. Run the HTML file on Google Chrome. If the steps above are done correctly, “Performance Task” should be displayed.

4. Open the browser's Developer Tools using CTRL + Shift + I or by clicking the Customize and Control button on the top right of the browser > More tools > Developer tools.

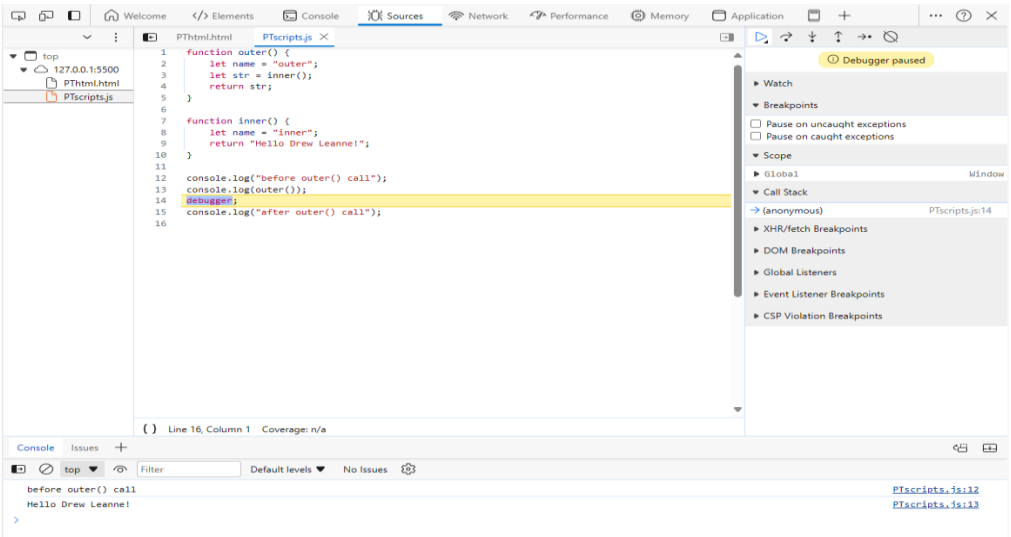
5. Screenshot the message on the console of the page.



6. Write the debugger statement in the console.log block in the PTscript.js file. Place it before calling the function outer. Save the file

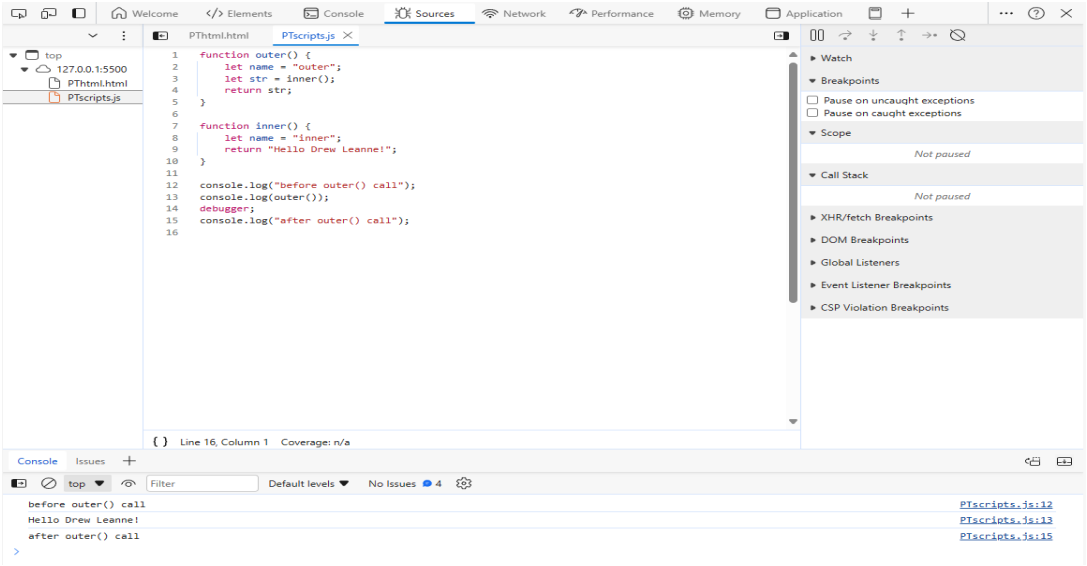


7. Refresh the browser and screenshot the console. Explain how the console displayed such output.



- Firstly, debugging is a part of troubleshooting in programming. It causes the program to stop or halt its execution on the line where the statement “debugger” is place. So since I placed the statement is present in the browser, the console only displayed the output ‘before outer() call’ and ‘Hello Drew Leanne!’.

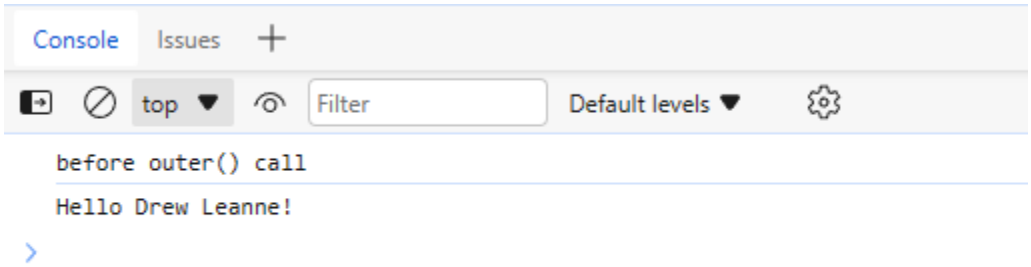
8. On the browser, click the Resume button (the blue triangle icon rotated to the right).



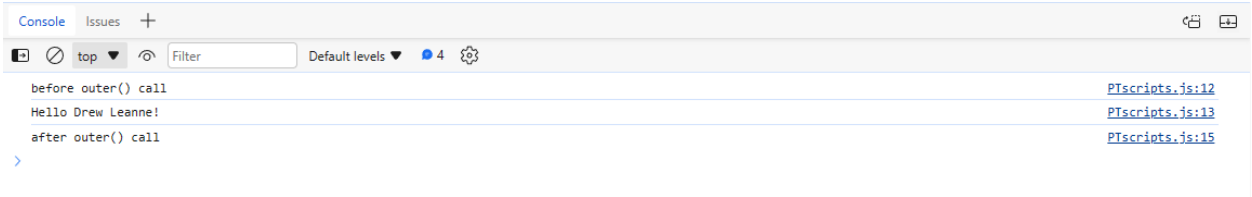
9. Look at the console and explain why the console displayed such output

- “Resuming" execution means telling the debugger to continue from up where the program left off when it was paused. With the help of this instruction, the program can continue running until it reaches another breakpoint or finishes. "Resuming" execution is essentially the same as allowing the program permission to carry on after a brief stop. So that’s why the outputs are ‘before outer() call’ ‘Hello Drew Leanne!’ and ‘after outer() call’ it’s because the resume button just continued the program.

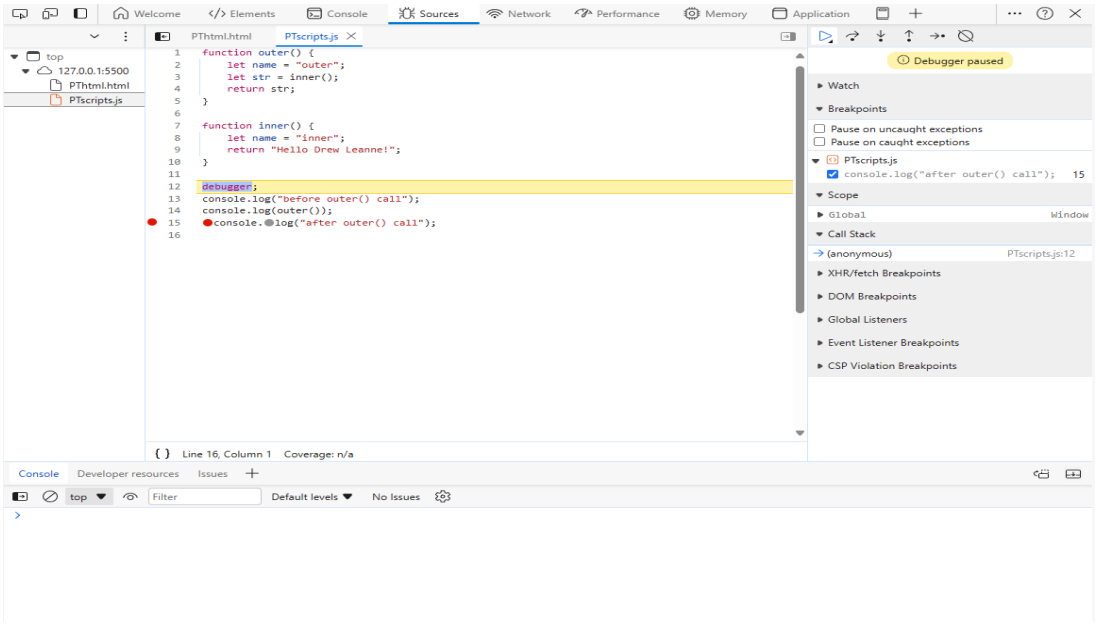
10. Click line 15, the last line of the code. It is how a breakpoint is set. Refresh the page
11. With the breakpoint on the 15th line, click the Resume button again. It will display the following:



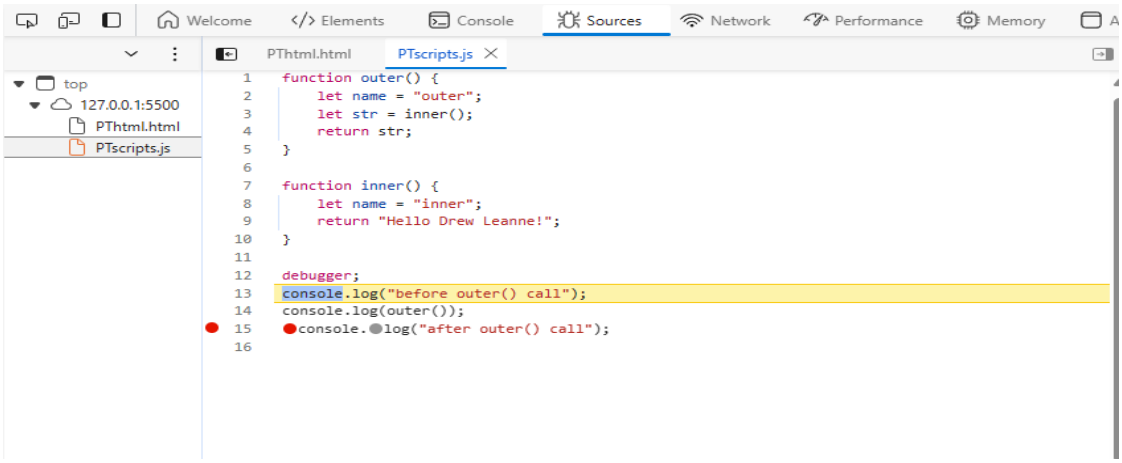
12. Clicking the Resume button again will display the rest of the output. Try it.
13. Re-click the 15th line to remove the breakpoint
14. Remove the debugger statement on the JS file. Save it and reload the browser



15. Set the 12th line as the new breakpoint on the browser. Reload the browser.



16. Press the Step Over button (right next to the Resume button, the arrow arching over the dot).



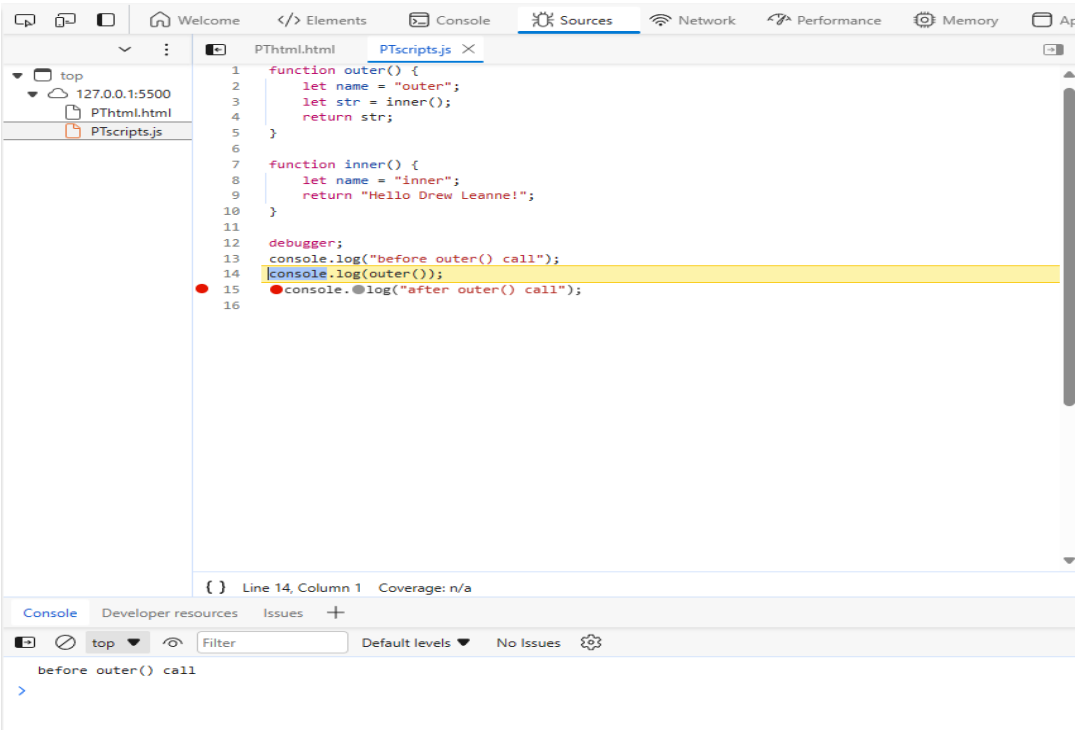
17. What happens to the JavaScript code in the Developer Tools when the Step Over button is pressed?

- "Step Over" allows the debugger to skip over the finer points of the invoked function and move on to the next line of code in the main program that follows the function call. Regardless of whether the current line of code contains a function call or not, the JavaScript debugger in Developer Tools executes the current line of code and advances to the next line in the script when I click the "Step Over" button.

18. What happens to the console if the Step Over button is pressed two (2) additional times?

- In the JavaScript debugger of Developer Tools, if I press the "Step Over" button two more times, the debugger executes the script's following two lines and advances the code execution sequence. The corresponding outputs are printed in the console window as each line of code is performed because these lines of code contain console.log statements or other operations that send information to the console. The output in the console is only the 'before outer() call'

19. Reload the page and click the Step Over button once



20. Now, click the Step Into button (next to the Step Over button) and list the highlighted words each time the button is clicked

- outer
- inner
- inner
- return

21. Repeat the instruction above, but this time, click the Step Into button six (6) times and provide the screenshot of the Developer Tools.