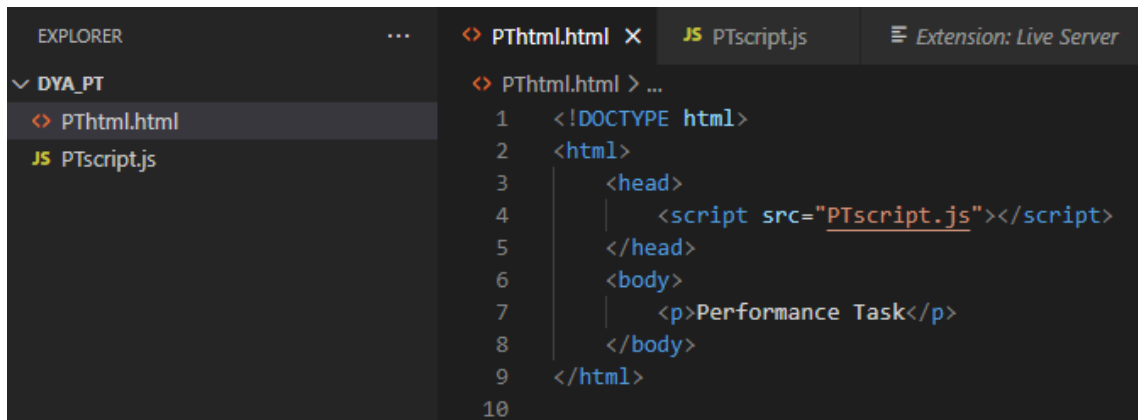


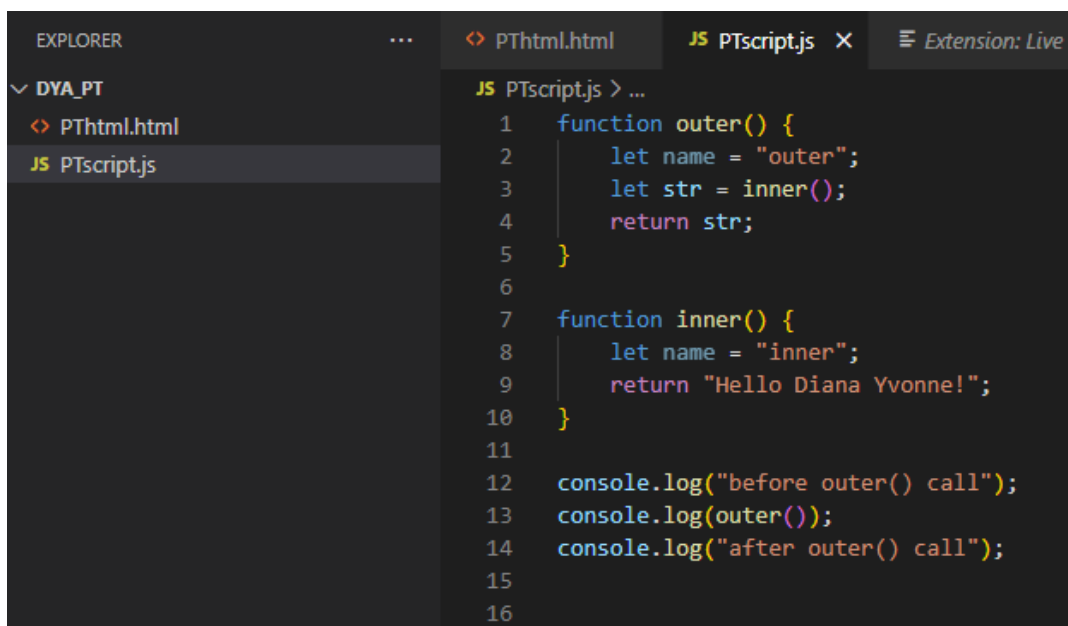
1. Create a simple HTML file with the following syntax. Use PThtml as the file name.



The screenshot shows the VS Code interface with the Explorer sidebar on the left. Under the 'DYA_PT' folder, there is a file named 'PThtml.html' and a file named 'PTscript.js'. The main editor area shows the content of 'PThtml.html' with the following code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script src="PTscript.js"></script>
5   </head>
6   <body>
7     <p>Performance Task</p>
8   </body>
9 </html>
```

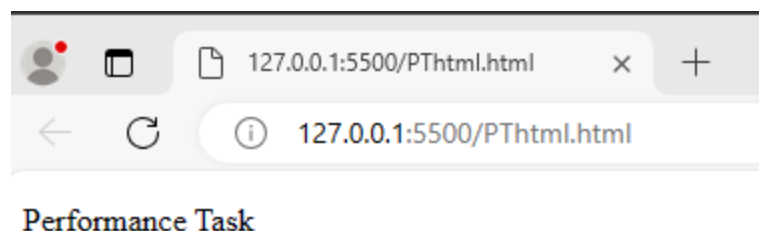
2. Then, create a JS file with the following code. Use the correct file name.



The screenshot shows the VS Code interface with the Explorer sidebar on the left. Under the 'DYA_PT' folder, there is a file named 'PThtml.html' and a file named 'PTscript.js'. The main editor area shows the content of 'PTscript.js' with the following code:

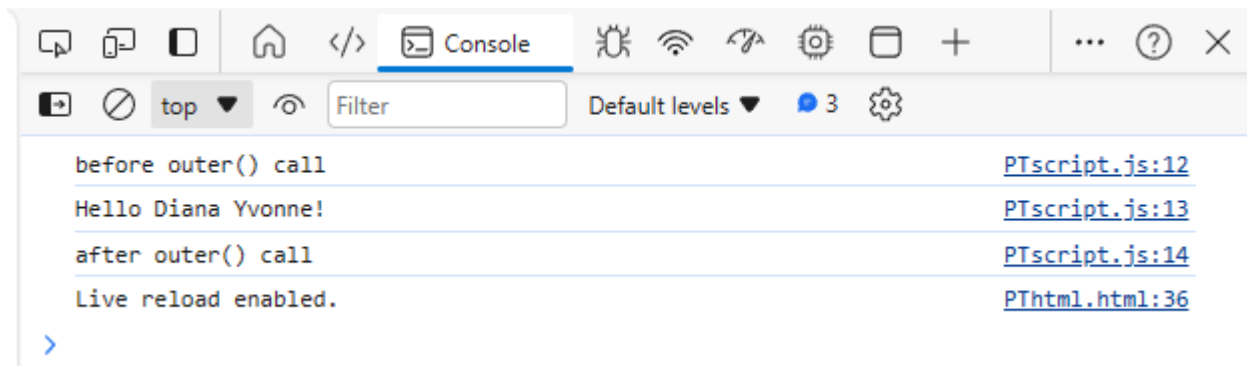
```
1 function outer() {
2   let name = "outer";
3   let str = inner();
4   return str;
5 }
6
7 function inner() {
8   let name = "inner";
9   return "Hello Diana Yvonne!";
10 }
11
12 console.log("before outer() call");
13 console.log(outer());
14 console.log("after outer() call");
15
16
```

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

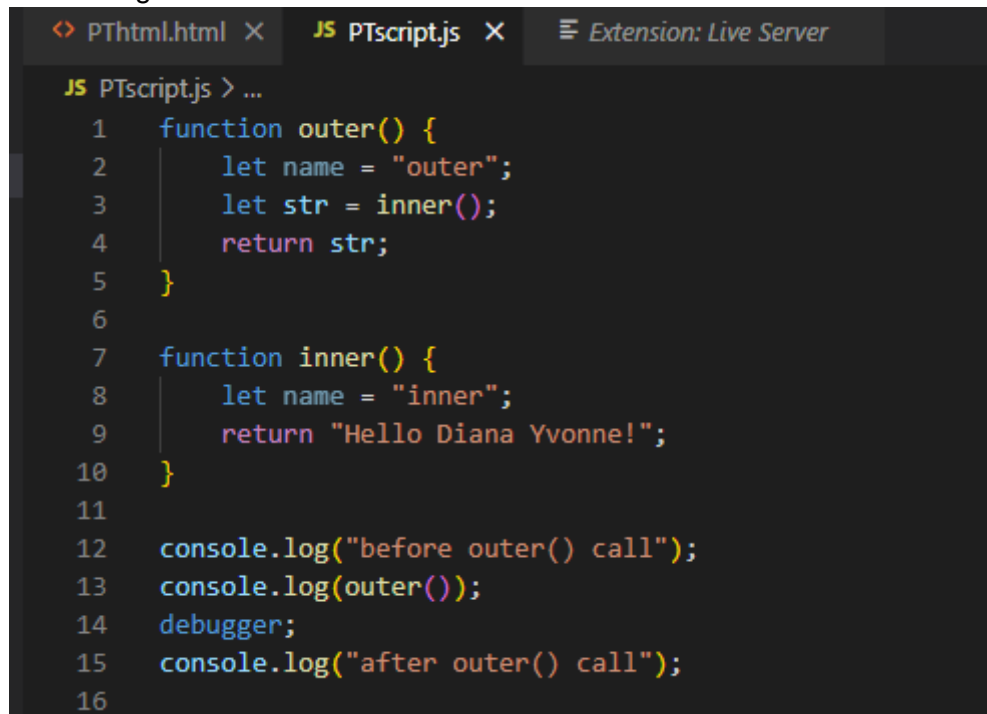


4. Open the browser's Developer Tools using CTRL + Shift + I or by clicking the Customize and Control button on the top right of the browser > More tools > Developer tools.

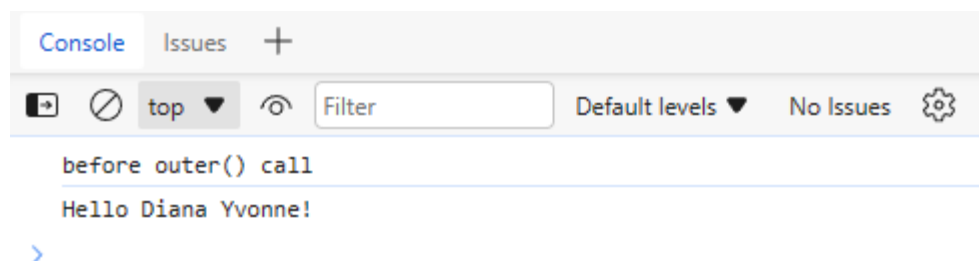
5. Screenshot the message on the console of the page.



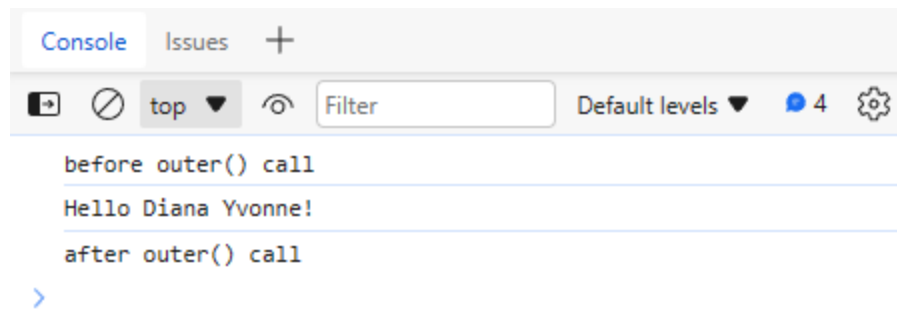
6. Write the debugger statement in the console.log block in the PTscript.js file. Place it before calling the function outer. Save the file.



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



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

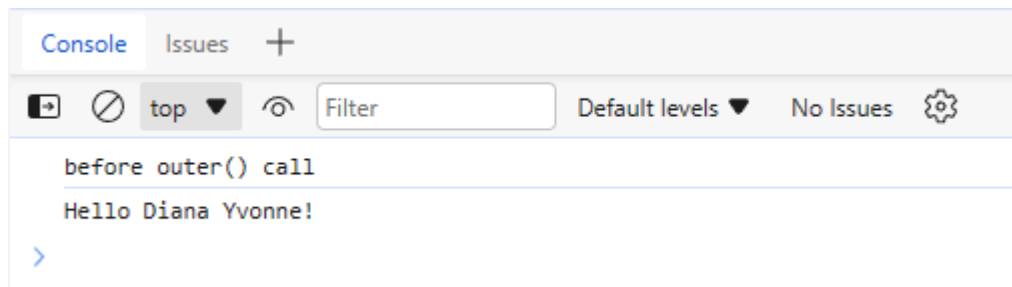


9. Look at the console and explain why the console displayed such output.

~ When you "resume" execution in Java debugging, you're essentially instructing the debugger to let the program continue running from where it was paused, usually at a breakpoint. This enables the program to proceed until it hits another breakpoint or completes its execution. In simpler terms, hitting the "resume" button is like giving the program the green light to carry on after a brief pause. As a result, the outputs 'before outer () call', 'Hello Diana Yvonne!', and 'after outer() call' are displayed because the 'resume' action allows the program to continue executing its code.

10. Click line 15, the last line of the code. It is how a breakpoint is set. Refresh the page.

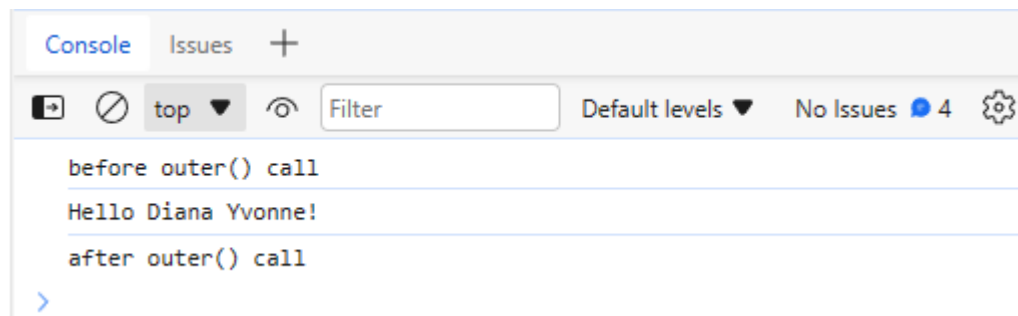
11. With the breakpoint on the 15th line, click the Resume button again. It will display the following:



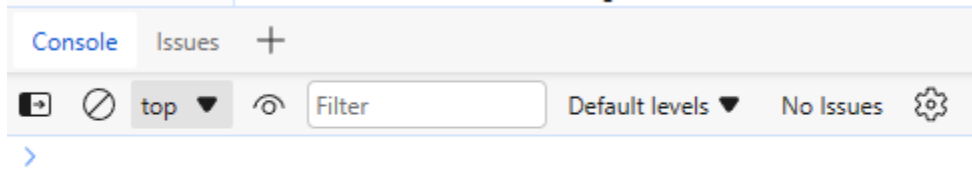
12. Clicking the Resume button again will display the rest of the output. Try it.

13. Re-click the 15th line to remove the breakpoint.

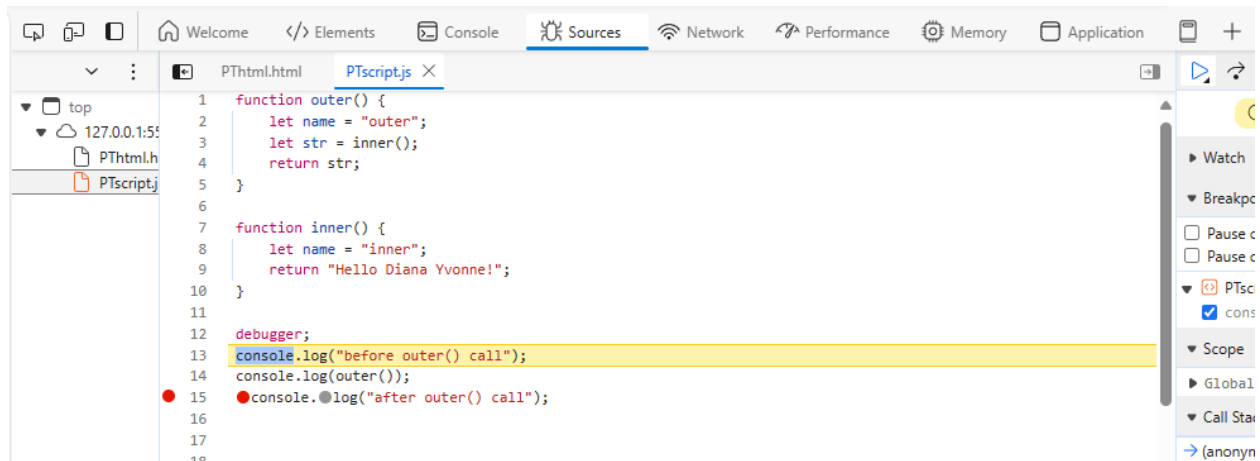
14. Remove the debugger statement on the JS file. Save it and reload the browser.



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).



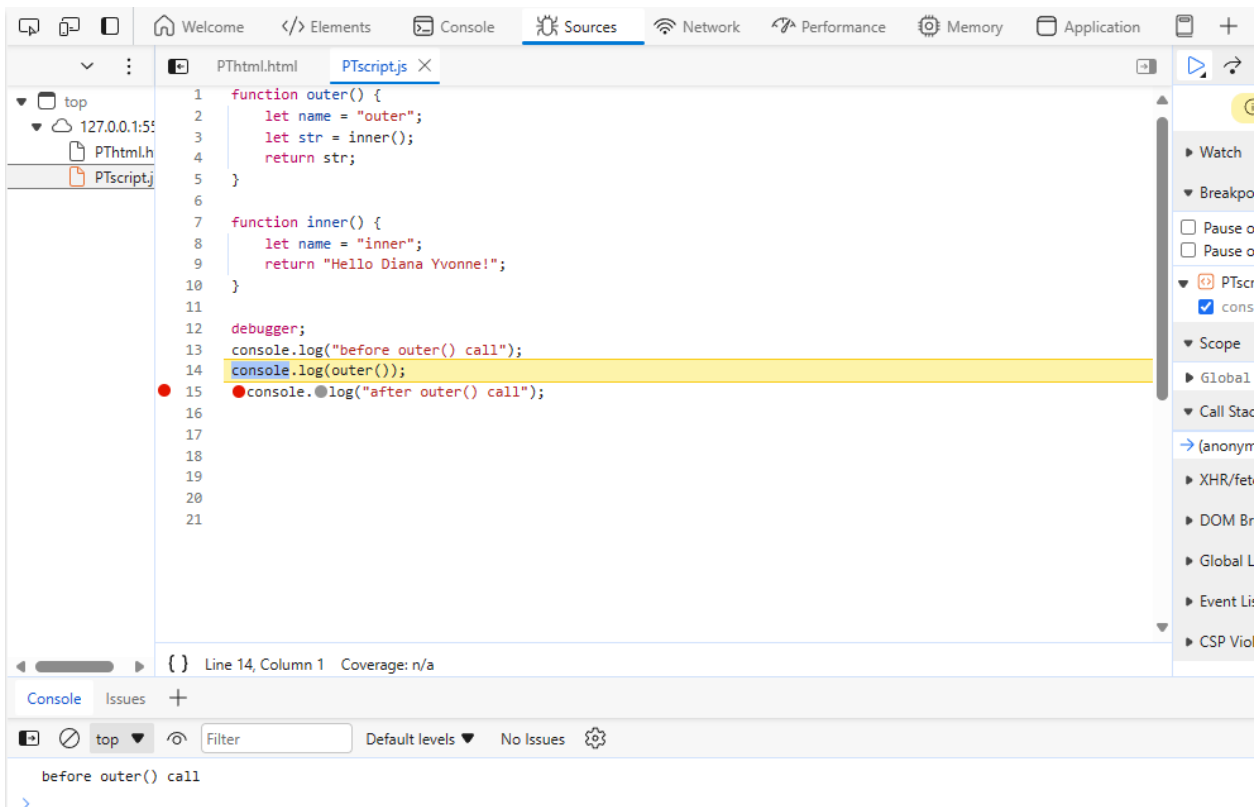
17. What happens to the JavaScript code in the Developer Tools when the Step Over button is pressed?

~ When you press the "Step Over" button in Developer Tools for JavaScript, the debugger skips past the details of the current function being executed and proceeds to the next line of code in the main program after the function call. Whether or not the current line of code includes a function call, the JavaScript debugger within Developer Tools executes that line and moves to the subsequent line in the script upon clicking the "Step Over" button.

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

~ Pressing the "Step Over" button twice more in the JavaScript debugger of Developer Tools causes the debugger to execute the next two lines of the script and progress the code execution. Any outputs specified by console.log statements or other operations sending information to the console will be displayed in the console window. In this scenario, only the output 'before outer () call' will be visible in the console.

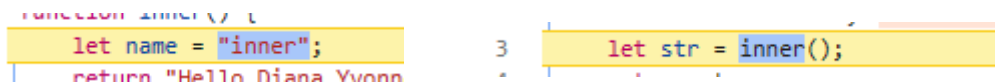
19. Reload the page and click the Step Over button once.



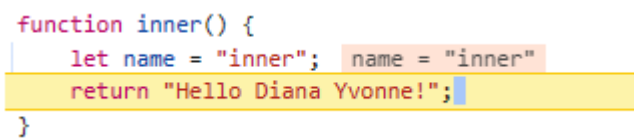
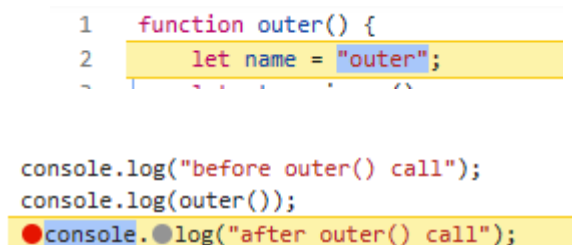
20. Now, click the Step Into button (next to the Step Over button) and list the highlighted words each time the button is clicked.

~ 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.



22. Submit the final version of the HTML and JavaScript files in a ZIP file.



```
1 function inner() {  
2   |   let name = "inner";   name = "inner"  
3   return "Hello Diana Yvonne!";  
4 }  
5
```

```
2 debugger;  
3 console.log("before outer() call");  
4 console.log(outer());  
5 ●console.●log("after outer() call");  
6
```

```
1 function outer() {  
2   |   let name = "outer";   name = "outer"  
3   let str = inner();   str = "Hello Diana Yvonne!"  
4   return str;  
5 }  
6
```