Cueto Louigi

BSIT3.2A

1.

```html
<!DOCTYPE html>
<html>
    <head>
        <script src="script.js"></script>
    </head>
    <body>
        <p>Performance Task</p>
    </body>
</html>
```

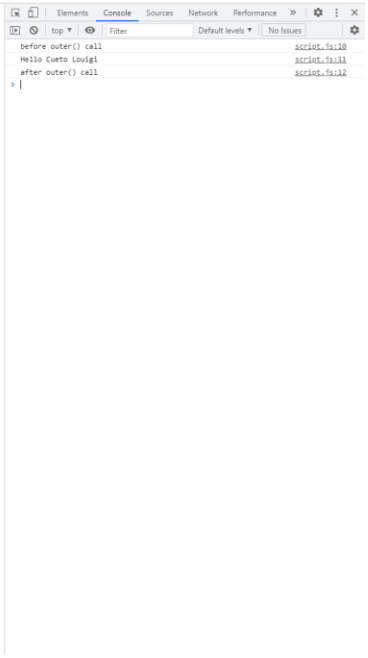
2.

```javascript
function outer(){
    let name = "outer";
    let str = inner();
    return str;
}
function inner(){
    let name = "inner";
    return "Hello Cueto Louigi";
}
console.log("before outer() call");
console.log(outer());
console.log("after outer() call");
```

3.

4 and 5.

Performance Task

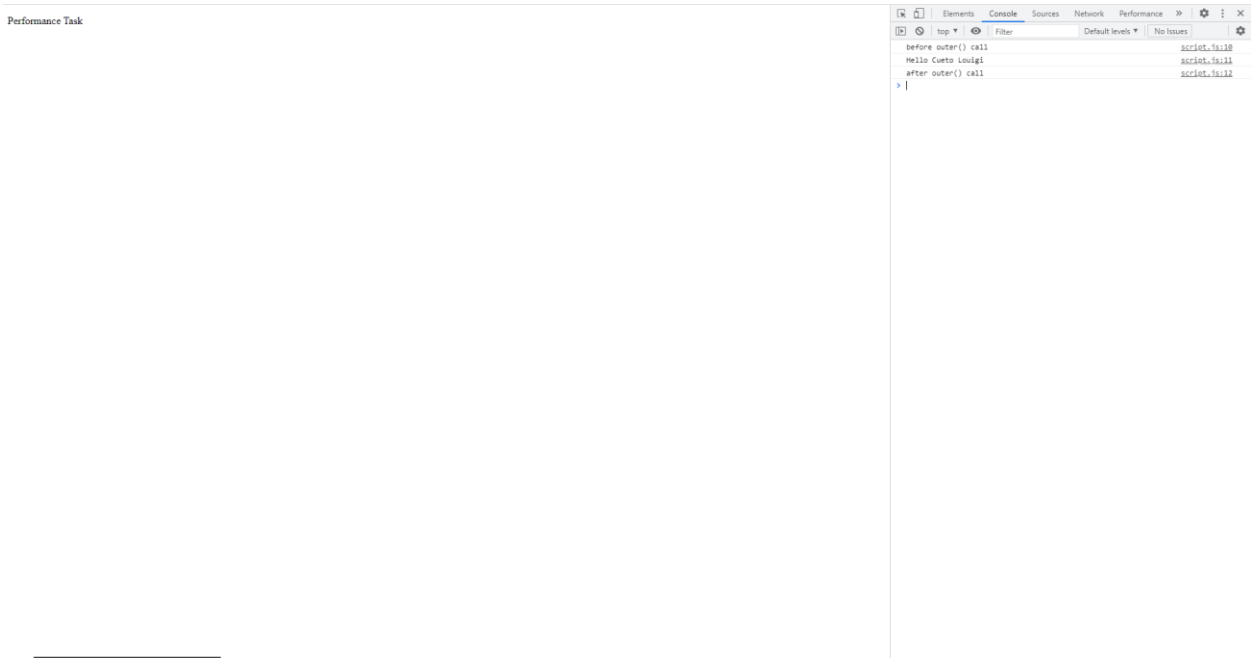


6.

```javascript
function outer(){
    let name = "outer";
    let str = inner();
    return str;
}
function inner(){
    let name = "inner";
    return "Hello Cueto Louigi";
}
console.log("before outer() call");
console.log(outer());
console.log("after outer() call");
```

7. The debugger statement pauses JavaScript execution, allowing inspection in Developer Tools. The console displays only the output before the debugger (console.log("before outer() call")). Execution waits for user interaction

8. RESUMED



9. The debugger paused execution, but Resume let the script complete, executing all console.log statements in order.

10.

TP.html script.js

```
function outer(){
    let name = "outer";
    let str = inner();
    return str;
}
function inner(){
    let name = "inner";
    return "Hello Cueto Louigi";
}
console.log("before outer() call");
console.log(outer());
console.log("after outer() call");
```

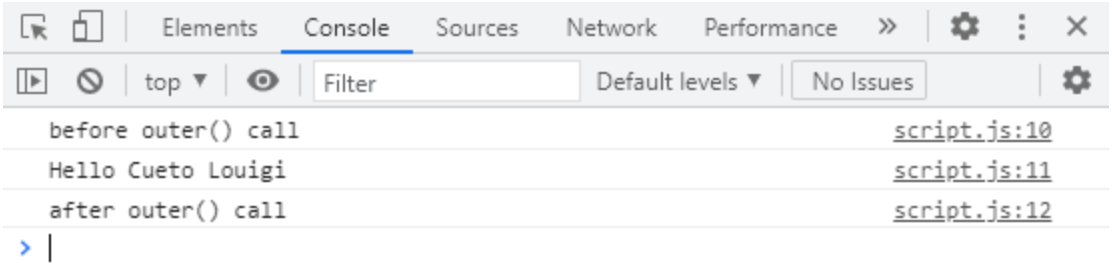
11.

Elements Console Sources Network Performance

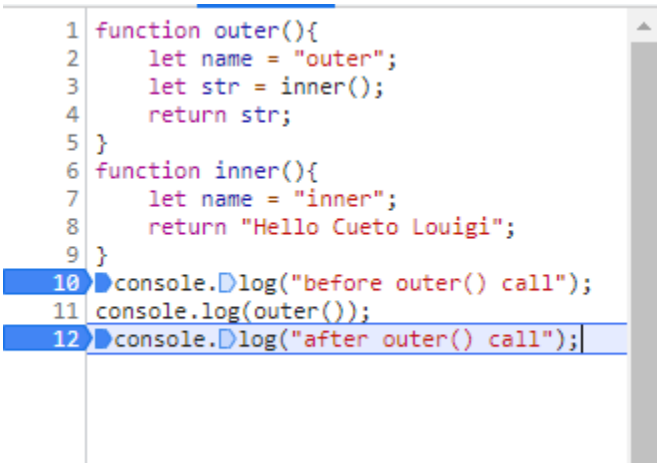
top Filter Default levels No Issues

```
before outer() call                    script.js:10
Hello Cueto Louigi                     script.js:11
>
```

12.

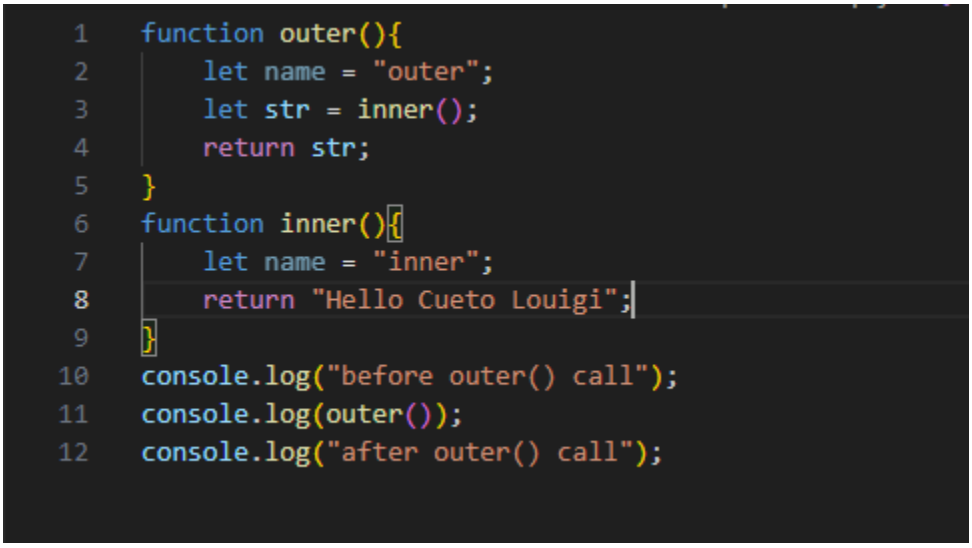


13.



```
function outer(){
    let name = "outer";
    let str = inner();
    return str;
}
function inner(){
    let name = "inner";
    return "Hello Cueto Louigi";
}
console.log("before outer() call");
console.log(outer());
console.log("after outer() call");
```

14.



```
function outer(){
    let name = "outer";
    let str = inner();
    return str;
}
function inner(){
    let name = "inner";
    return "Hello Cueto Louigi";
}
console.log("before outer() call");
console.log(outer());
console.log("after outer() call");
```

15.

```
1  function outer(){
2      let name = "outer";
3      let str = inner();
4      return str;
5  }
6  function inner(){
7      let name = "inner";
8      return "Hello Cueto Louigi";
9  }
10 console.log("before outer() call");
11 console.log(outer());
12 console.log("after outer() call");
```

16.

```
before outer() call                                    script.js:10
>
```

17. The JavaScript code in Developer Tools executes the current line (console.log(outer())), logging "Hello Cueto Louigi" to the Console. The debugger advances to the next executable line (console.log("after outer() call")). In the Sources tab, the highlighted line changes, indicating the debugger's new paused position. The code itself remains unchanged, but the visual indication updates to reflect the next line to be executed.

18. First press: Executes logs "after outer() call", and moves to the end of the script no more executable lines. Second press: No further lines, so the debugger exits, and the Console shows the full output.

```
before outer() call                                    script.js:10
Hello Cueto Louigi                                     script.js:11
after outer() call                                     script.js:12
>
```

19.

TP.html script.js

```
1  function outer(){
2      let name = "outer";
3      let str = inner();
4      return str;
5  }
6  function inner(){
7      let name = "inner";
8      return "Hello Cueto Louigi";
9  }
10 console.log("before outer() call");
11 console.log(outer());
12 console.log("after outer() call");
```

20. let name = "outer" let str = inner let name = "inner" return "Hello Louigi Cueto" let str = inner (after returning from inner()) return str

```
1  function outer(){
2      let name = "outer";   name = "outer"
3      let str = inner();
4      return str;
5  }
6  function inner(){
7      let name = "inner";
8      return "Hello Cueto Louigi";
9  
10 log("before outer() call");
11 console.log(outer());
12 console.log("after outer() call");
```

log

f log()

21. After clicking Step Into six times, the Developer Tools pauses at line 4 (return str;). Highlighted (return str;). Console output: before outer() call. Debugger controls visible

TP.html script.js

```
1  function outer(){
2      let name = "outer";
3      let str = inner();
4      return str;
5  }
6  function inner(){
7      let name = "inner";   name = "inner"
8      return "Hello Cueto Louigi";
9  }
10 console.log("before outer() call");
11 console.log(outer());
12 console.log("after outer() call");
```

Line 8, Column 5 Coverage: n/a

Scope Watch

```
▼ Local
    name: "inner"
  ▶ this: Window
▶ Global                    Window
```

all...

l"...

js:8

js:3