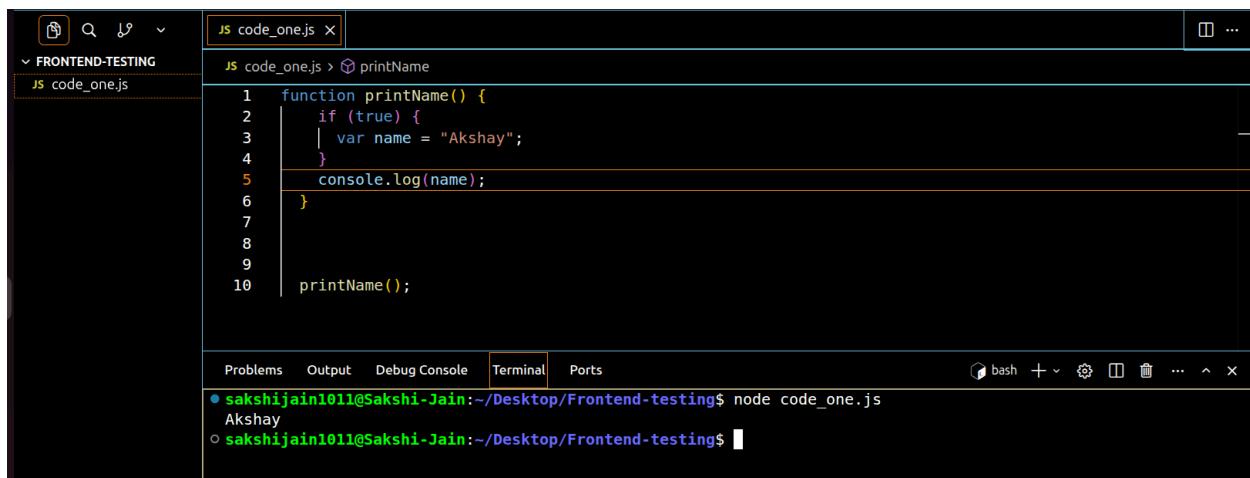


## Question: Find and fix the bug from below code

Prob - 1] function printName() {

```
if (true) {  
    var name = "Akshay";  
}  
console.log(name);  
}  
  
printName();
```

Output: Akshay



The screenshot shows a VS Code interface. On the left, there's a file tree with 'code\_one.js' selected. The main area has a code editor with the following content:

```
JS code_one.js  
JS code_one.js > printName  
1 function printName() {  
2     if (true) {  
3         var name = "Akshay";  
4     }  
5     console.log(name);  
6 }  
7  
8  
9  
10 printName();
```

Below the code editor is a terminal window showing the output of running the script:

```
sakshijain1011@sakshi-Jain:~/Desktop/Frontend-testing$ node code_one.js  
Akshay  
sakshijain1011@sakshi-Jain:~/Desktop/Frontend-testing$
```

Why this Output:

-> var is not block - scoped that means variables created using var is accessible anywhere in a function even if declared in different block

Prob - 2] const arr = [10, 20, 30];

```
for (let i = 0; i <= arr.length; i++) {  
    console.log(arr[i]);  
}
```

**Output:10 20 30 undefined**

The screenshot shows a terminal window with the following content:

```
JS code_one.js JS code_two.js
v FRONTEND-TESTING
JS code_one.js
JS code_two.js

JS code_two.js > ...
1 const arr = [10, 20, 30];
2 for (let i = 0; i <= arr.length; i++) {
3   console.log(arr[i]);
4 }
5

Problems Output Debug Console Terminal Ports
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_one.js
Akshay
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_two.js
10
20
30
undefined
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$
```

**Why this output?**

- Because `arr.length` is 3, but array indexes go from 0 to 2.
- Because in code we wrote `i <= arr.length`, the loop runs till `i = 3` also.
- So when `i = 3`, it prints:
- `arr[3] → undefined` (because that index does not exist)

**Correct Code:**

The screenshot shows a terminal window with the following content:

```
JS code_two.js > ...
1 const arr = [10, 20, 30];
2 for (let i = 0; i < arr.length; i++) {
3   console.log(arr[i]);
4 }
5
```

**Prob -3] let data;**

```
setTimeout(() => {
```

```
  data = "Loaded";
```

```
}, 1000);  
  
console.log(data);
```

**Output: undefined**



The screenshot shows a terminal window within a code editor interface. The terminal tab is selected. The code editor sidebar on the left shows files: code\_one.js, code\_three.js, and code\_two.js. The code in code\_three.js is:

```
1 let data;  
2 setTimeout(() => {  
3   | data = "Loaded";  
4 }, 1000);  
5 console.log(data);
```

The terminal output shows the following sequence of commands and their results:

- node code\_one.js
- Akshay
- node code\_two.js
- 10  
20  
30  
undefined
- node code\_three.js
- ^C
- undefined

## Why this Output?

- **setTimeout() runs after 1 second (it is asynchronous)**
- **But console.log(data) runs immediately (synchronous)**
- **So at the moment console.log(data) executes, data is still undefined**

## Fixes:

1. Using **console.log(data)** inside of the function
2. Using **callback function**
3. Using **promises**
4. Using **async and await**

## Code:

```
js code_three.js > ...
```

```
1 // 1. Print inside setTimeout
2 let data;
3 setTimeout(() => {
4   data = "Loaded";
5   console.log(data);
6 }, 1000);
7
8 //2. Use a callback function
9 function loadData(callback) {
10   setTimeout(() => {
11     | callback("Loaded");
12     }, 1000);
13 }
14
15 loadData((result) => {
16   console.log(result); // ✓ Loaded
17 });
18
19 // 3. Use a promise
20 function loadData() {
21   return new Promise((resolve) => {
22     | setTimeout(() => resolve("Loaded"), 1000);
23   });
24 }
25
26 loadData().then((result) => console.log(result)); |
27
28 // 4. Use async/await
29 function loadData() {
30   return new Promise((resolve) => {
31     | setTimeout(() => resolve("Loaded"), 1000);
32   });
33 }
34
35 async function run() {
36   const data = await loadData();
37   console.log(data);
38 }
39
40 run();
```

```
Prob - 4] function add(a, b) {
```

```
    a + b;
```

```
}
```

```
const result = add(2, 3);
```

```
console.log(result);
```

**Output:**

```
JS code_four.js > ...
1  function add(a, b) {
2  |    a + b;
3  }
4  const result = add(2, 3);
5  console.log(result);
6
7  | Ctrl+L to chat, Ctrl+K to generate

Problems Output Debug Console Terminal Ports
@sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ code_four.js
code_four.js: command not found
● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_four.js
undefined
```

**Why this output?**

- Because inside function you did not return anything

**Fixes:** Just add return in the function `add(a,b)`

```
JS code_fix.js > ...
1  function add(a, b) {
2  |    return a + b;
3  }
4  const result = add(2, 3);
5  console.log(result);

Problems Output Debug Console Terminal Ports
● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_fix.js
5
```

Prob - 5] const user = {

```
name: "John",
```

```
age: 25,
```

```
};
```

```
function updateAge(u) {
```

```
    u.age = 30;
```

```
}
```

```
updateAge(user);
```

```
console.log(user.age);
```

**Output: 30**

```
JS code_five.js > [0] user
1 const user = {
2     name: "John",
3     age: 25,
4 };
5     function updateAge(u) {
6         u.age = 30;
7     }
8     updateAge(user);
9     console.log(user.age);

Problems    Output    Debug Console    Terminal    Ports
● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_five.js
30
○ sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$
```

**Why This Output?**

- Because user is an object, and objects are passed as a reference.
- So u inside updateAge() points to the same object as user, and changing u.age also changes user.age.
- That's why output becomes 30.

**Fixes**

1. Create a copy inside function
2. Copy before passing

```
JS code_fix.js > ...
1  const user = {
2    name: "John",
3    age: 25,
4  };
5
6  // 1. Create copy inside function
7  function updateAge(u) {
8    return { ...u, age: 30 };
9  }
10 const updatedUser = updateAge(user);
11 console.log(user.age);
12 console.log(updatedUser.age);
13
14 // 2. Copy before passing
15 const copyUser = { ...user };
16 updateAge(copyUser);
17 console.log(user.age);
18 console.log(copyUser.age);
19
```

Problems   Output   Debug Console   **Terminal**   Ports

● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing\$ node code\_fix.js  
25  
30  
25

Prob - 6] <button id="btn">Click</button>

```
<script>

const btn = document.getElementById("btn");

btn.addEventListener("click", handleClick);

function handleClick() {

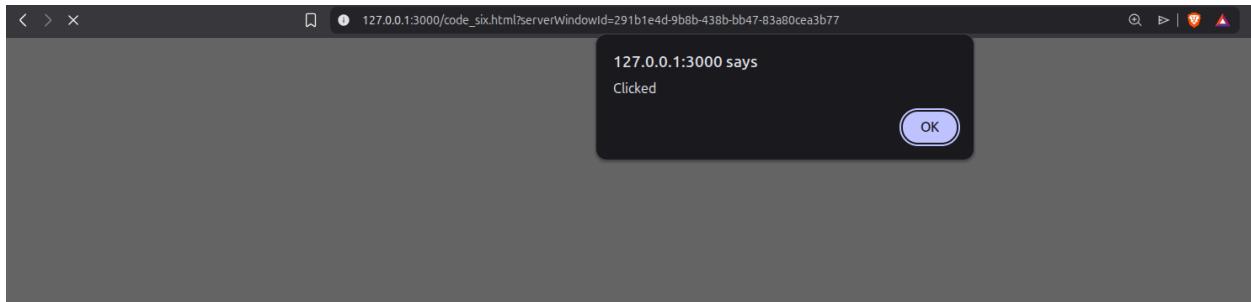
alert("Clicked");

}
```

```
</script>
```

### Output:

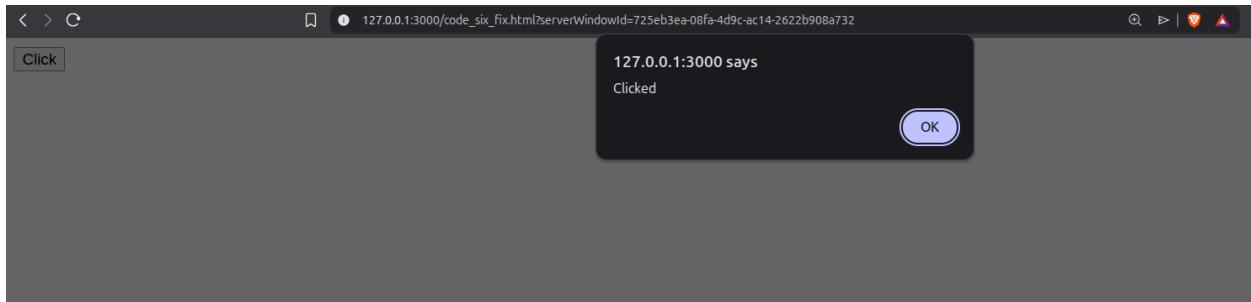
1. When refreshed automatically getting this
2. And Click button do not work



### Why This Output?

- You're calling the function immediately because of handleClick().
- Instead pass the function reference, not the function call.

### Fixes: Button works fine now



Prob - 7] `fetch("https://api.example.com/data")`

```
.then((res) => {  
  res.json();  
})  
.then((data) => {  
  console.log(data);  
});
```

## Output:

The screenshot shows the VS Code interface. In the top bar, there are tabs for 'JS code\_four\_fix.js', 'JS code\_five.js', 'JS code\_five\_fix.js', 'code\_six.html', 'code\_six\_fix.html', 'JS code\_seven.js' (which is currently active), and '...'. Below the tabs, the code editor displays a JavaScript file with the following code:

```
1 fetch("https://api.example.com/data")
2   .then((res) => {
3     res.json();
4   })
5   .then((data) => {
6     console.log(data);
7   });

```

Below the code editor is a tab bar with 'Problems', 'Output', 'Debug Console', 'Terminal' (which is selected), and 'Ports'. The terminal window shows the following output:

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_five_fix.js
25
30
25
25
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_seven.js
node:internal/deps/undici/undici:11360
    Error.captureStackTrace(err, this);
    ^
TypeError: fetch failed
    at Object.fetch (node:internal/deps/undici/undici:11360:11)
    at process.processTicksAndRejections (node:internal/process/task_queues:95:5) {
  cause: Error: getaddrinfo ENOTFOUND api.example.com
      at GetAddrInfoReqWrap.onlookup [as oncomplete] (node:dns:107:26) {
    errno: -3007,
    code: 'ENOTFOUND',
    syscall: 'getaddrinfo',
    hostname: 'api.example.com'
  }
}
```

The terminal also shows two bash sessions listed in the sidebar.

## Why This Output?

- **https://api.example.com/data** is not a real API domain, so DNS can't find it → **ENOTFOUND**
- That's why Node throws: **fetch failed + getaddrinfo ENOTFOUND**

## Fixes: Use real api

Prob - 8]const nums = [1, 2, 3, 4];

```
const result = nums.map(👍 => {
  if (n % 2 === 0) {
    return n * 2;
  }
});
console.log(result);
```

```
JS code_eight.js > [✖] result
1  const nums = [1, 2, 3, 4];
2  const result = nums.map(👍 => {
3    if (n % 2 === 0) {
4      return n * 2;
5    }
6  });
7  console.log(result);
```

Invalid character. ts(1127)  
Fix in Chat (Ctrl+Shift+D)  
Ctrl+click to open in new tab  
[View Problem \(Alt+...\)](#) [Quick Fix... \(Ctr...](#)

**Output: [ undefined, 4, undefined, 8 ] (Output when we used n character instead of emoji)**

```
JS code_eight.js > [✖] result > ⓘ nums.map() callback
1  const nums = [1, 2, 3, 4];
2  const result = nums.map(n => {
3    if (n % 2 === 0) {
4      return n * 2;
5    }
6  });
7  console.log(result);
```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ * History restored
● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_eight.js
[ undefined, 4, undefined, 8 ]
● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_eight.js
```

**Why This Output?**

- Using n without declaring it was the first issue
- Using emoji

**Fixes**

- As we getting undefined value in place of odd values and actually manipulating just even values so we can use **filter** instead

```
JS code_eight_fix.js > [0] result
1 const nums = [1, 2, 3, 4];
2 const result = nums.filter((n) => {
3     if (n % 2 === 0) {
4         return n * 2;
5     }
6 });
7 console.log(result);
8
```

Problems   Output   Debug Console   Terminal   Ports

```
● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_eight_fix.js
[ 2, 4 ]
○ sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ █
```

Prob - 9] const person = {

```
name: "Amar",
greet: () => {
    console.log("Hello " + this.name);
},
};

person.greet();
```

Output:Hello undefined

```
JS code_nine.js > [0] person
1 const person = {
2     name: "Amar",
3     greet: () => {
4         console.log("Hello " + this.name);
5     },
6 };
7 person.greet();
```

Problems   Output   Debug Console   Terminal   Ports

```
● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_nine.js
Hello undefined
○ sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ █
```

## Why This Output?

- This doesn't work inside arrow functions

## Fixes

- use normal function

The screenshot shows a terminal window with the following content:

```
JS code_nine_fix.js > ...
1 const person = {
2   name: "Amar",
3   greet() {
4     console.log("Hello " + this.name);
5   },
6 };
7 person.greet();
8
```

Below the code, there are tabs: Problems, Output, Debug Console, Terminal (which is selected), and Ports.

```
● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_nine_fix.js
Hello Amar
○ sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ █
```

Repository of code tried and fixes I did:

[https://github.com/sakshijain-josh/Training/tree/main/Frontend\\_Training](https://github.com/sakshijain-josh/Training/tree/main/Frontend_Training)