Question: Find and fix the bug from below code

Prob - 1] function printName() {

if (true) {

var name = "Akshay";

}

console.log(name);

}

printName();

Output: Akshay

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

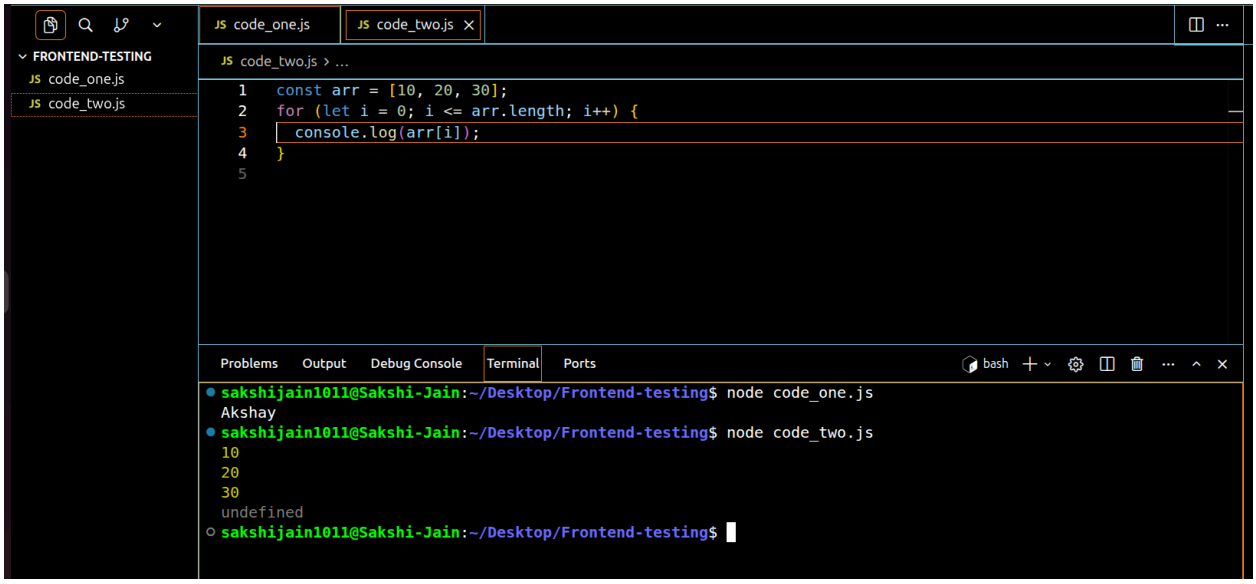
Prob - 2] const arr = [10, 20, 30];

for (let i = 0; i <= arr.length; i++) {

console.log(arr[i]);

}

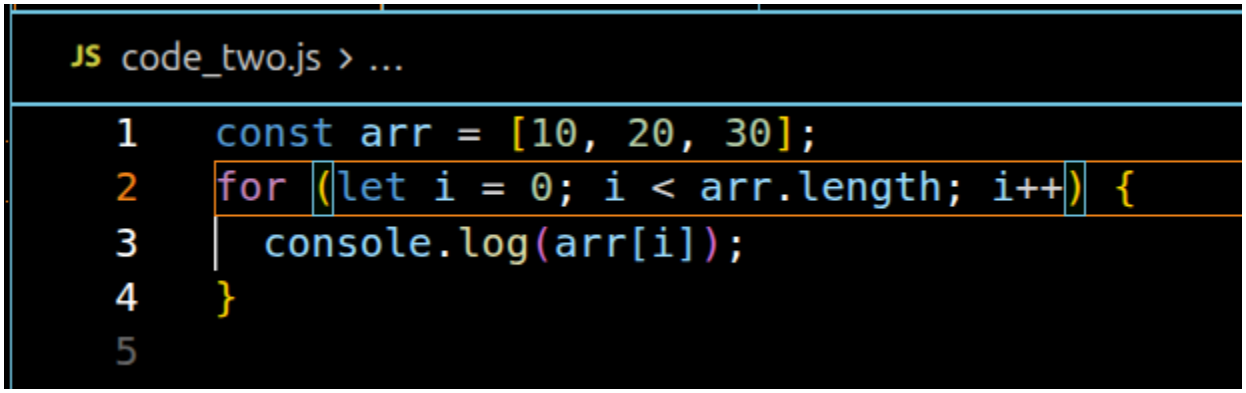
Output:10 20 30 undefined



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:



Prob -3] let data;

setTimeout(() => {

data = "Loaded";

}, 1000);

console.log(data);

Output: 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 > ...

```js
// 1. Print inside setTimeout
let data;
setTimeout(() => {
  data = "Loaded";
  console.log(data);
}, 1000);

//2. Use a callback function
function loadData(callback) {
    setTimeout(() => {
      callback("Loaded");
    }, 1000);
  }

  loadData((result) => {
    console.log(result); // ✅ Loaded
  });

  // 3. Use a promise
  function loadData() {
    return new Promise((resolve) => {
      setTimeout(() => resolve("Loaded"), 1000);
    });
  }

  loadData().then((result) => console.log(result));

  // 4. Use async/await
  function loadData() {
    return new Promise((resolve) => {
      setTimeout(() => resolve("Loaded"), 1000);
    });
  }

  async function run() {
    const data = await loadData();
    console.log(data);
  }
  run();
```

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

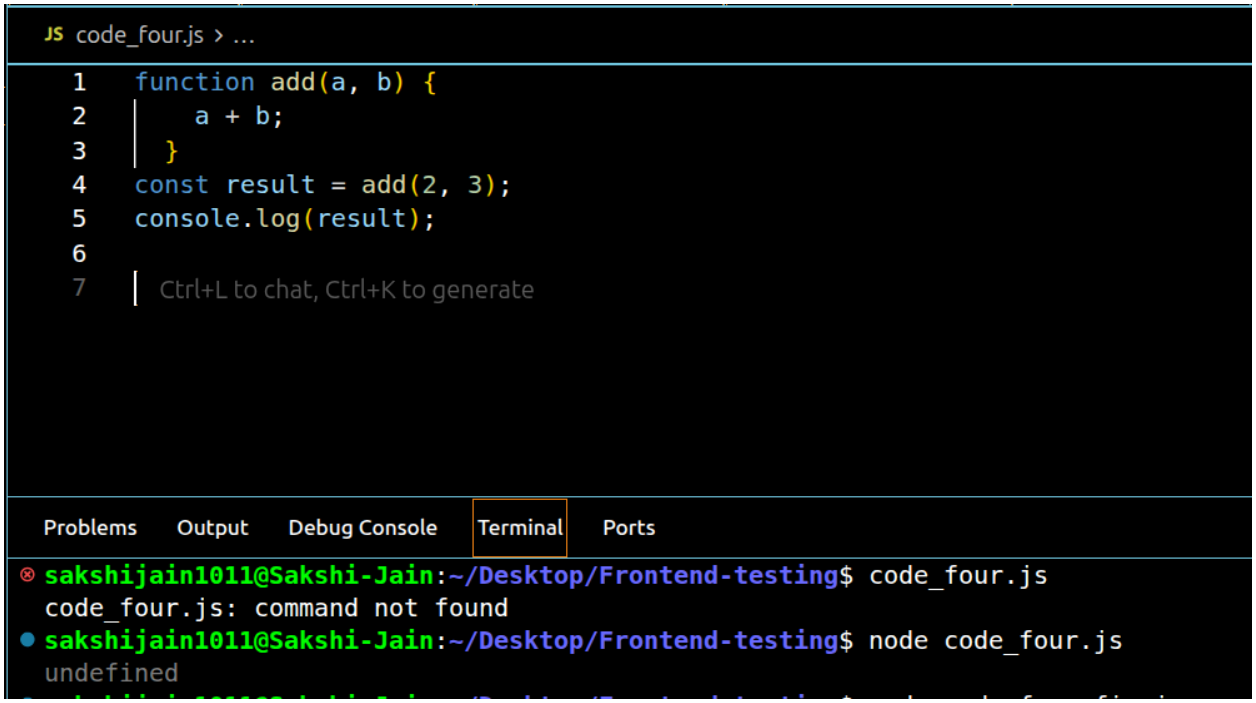
a + b;

}

const result = add(2, 3);

console.log(result);

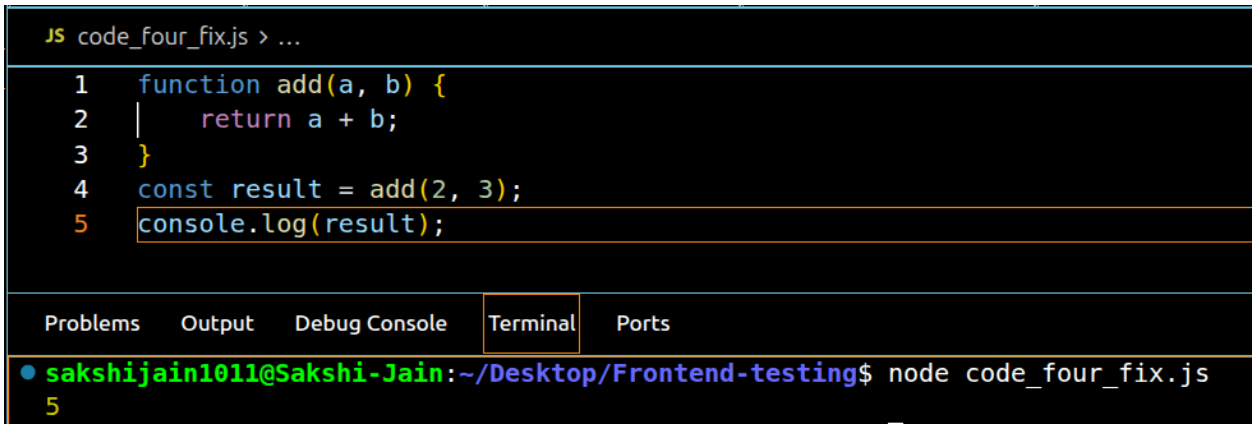
Output:



Why this output?

- **Because inside function you did not return anything**

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



Prob - 5] const user = {

name: "John",

age: 25,

};

function updateAge(u) {

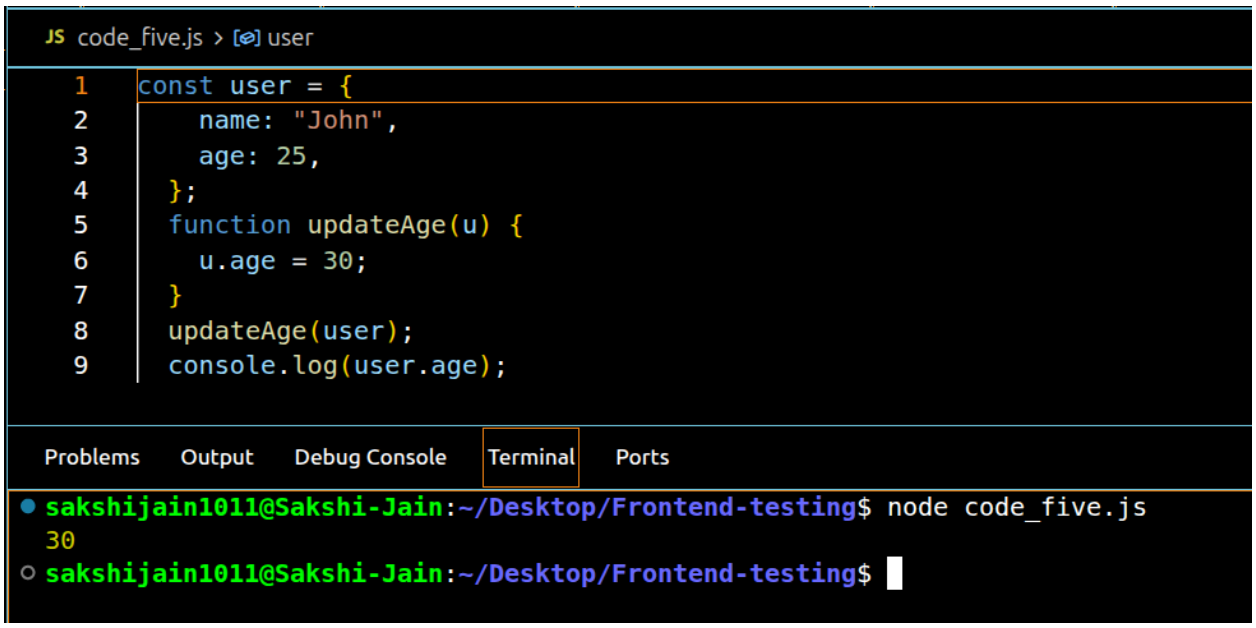
u.age = 30;

}

updateAge(user);

console.log(user.age);

Output: 30



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**

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

<script>

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

btn.addEventListener("click", handleClick());

function handleClick() {

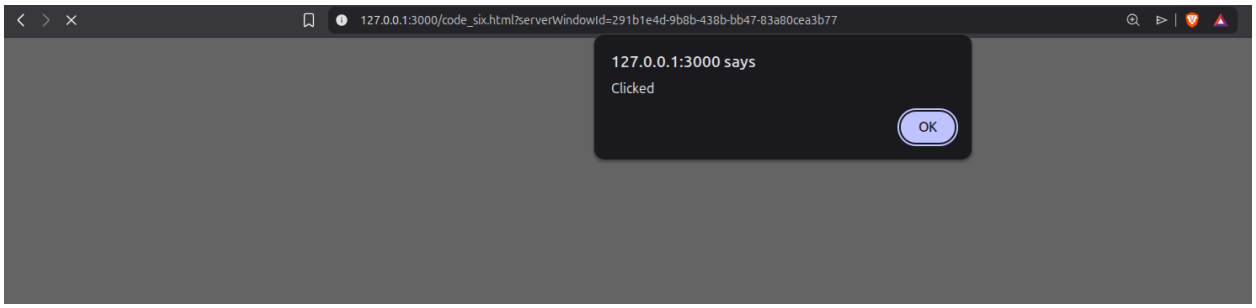
alert("Clicked");

}

</script>

Output:

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



Why This Output?

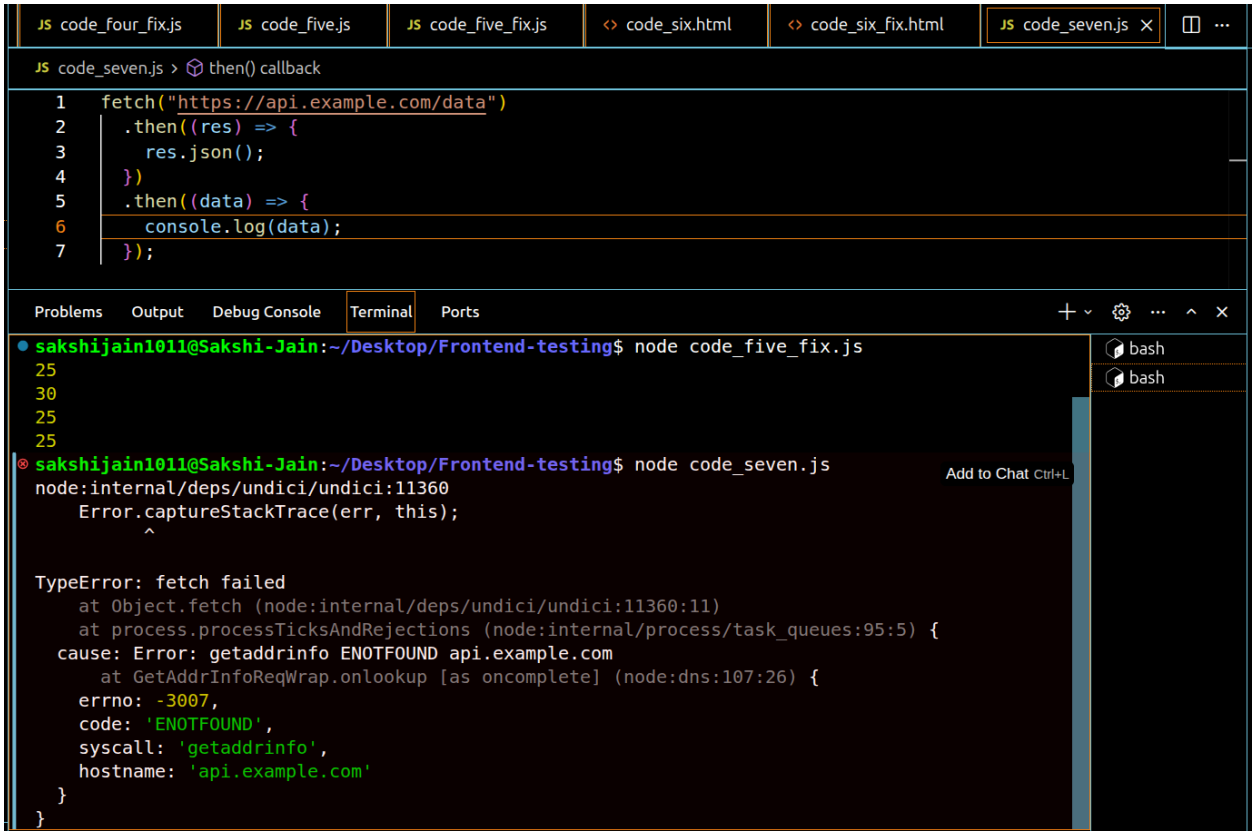
- **You're calling the function immediately because of handleClick().**
- **Instead pass the function reference, not the function call.**

Fixes: Button works fine now

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

```
.then((res) => {

res.json();

})

.then((data) => {

console.log(data);

});
```

Output:



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(👍 => {

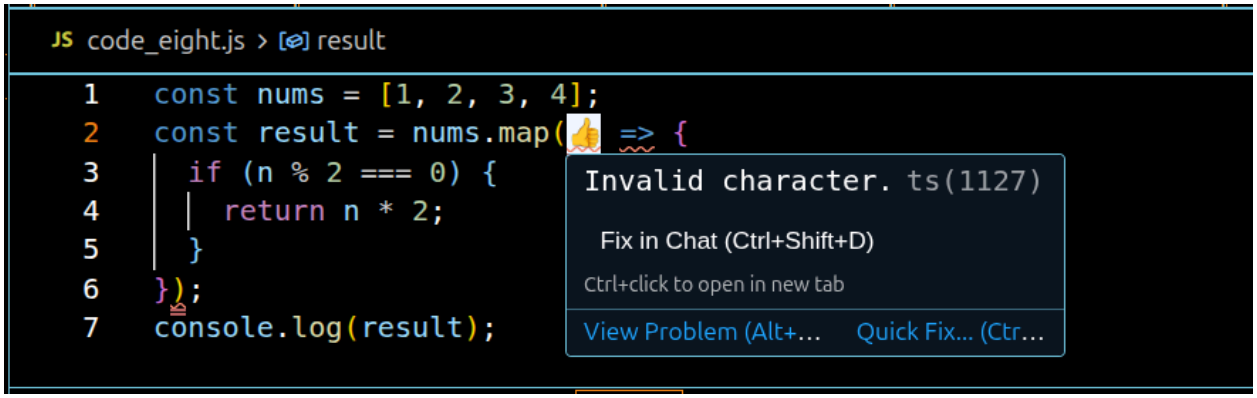
if (n % 2 === 0) {

return n * 2;

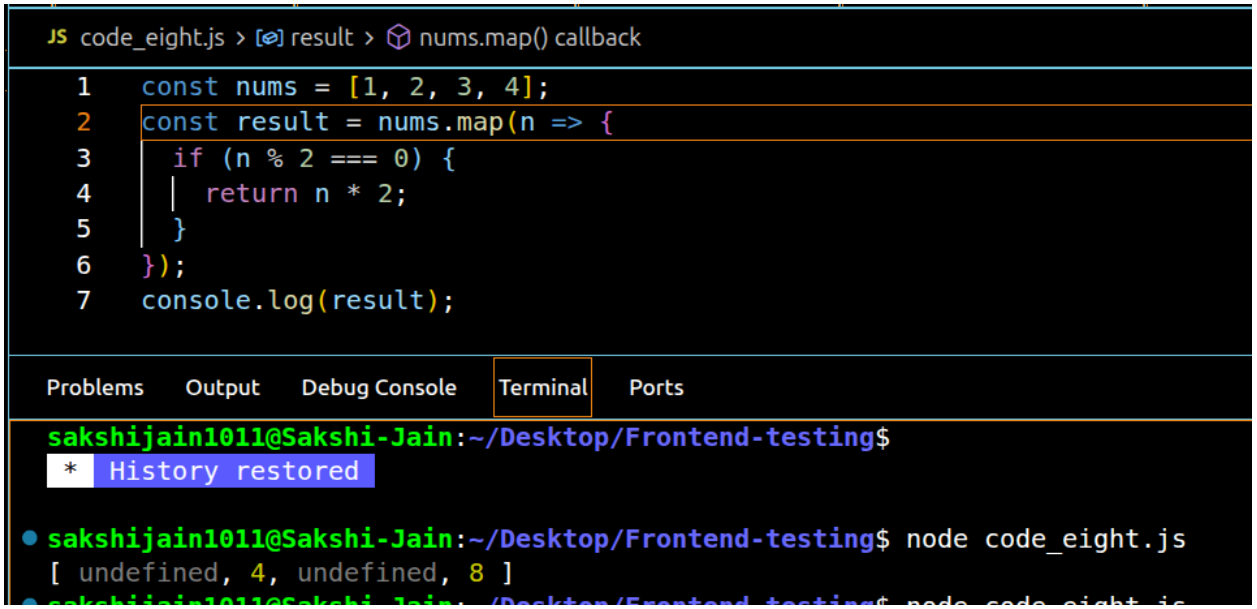
}

});

console.log(result);



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



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 > 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 > 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

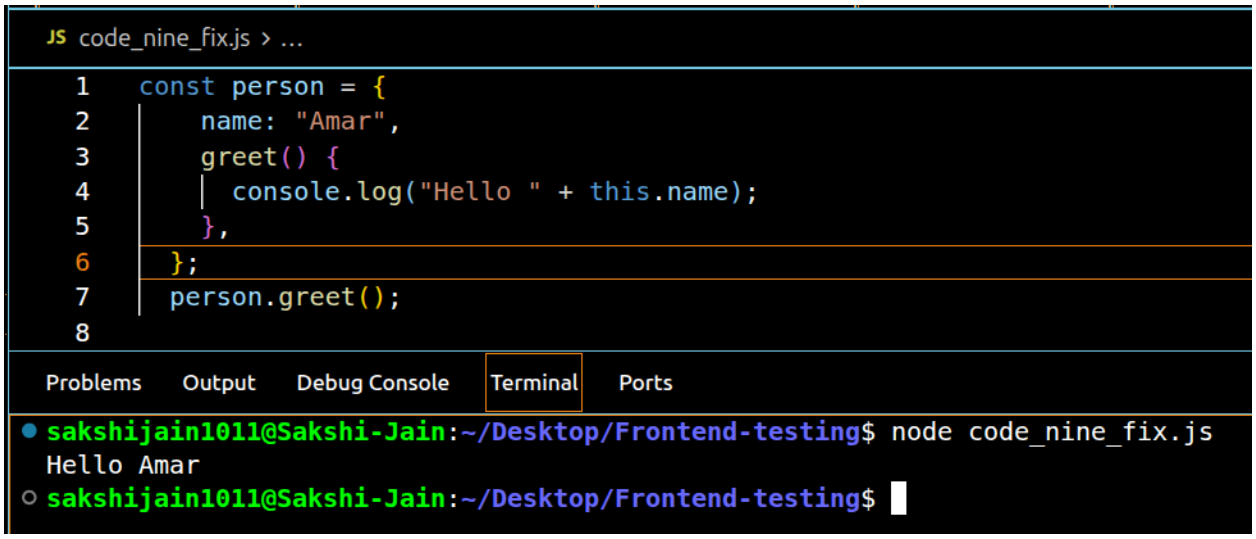
```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$ node code_nine.js
Hello undefined
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing$
```

Why This Output?

- **This doesn’t work inside arrow functions**

Fixes

- **use normal function**



Repository of code tried and fixes I did:
https://github.com/sakshijain-josh/Training/tree/main/Frontend_Training