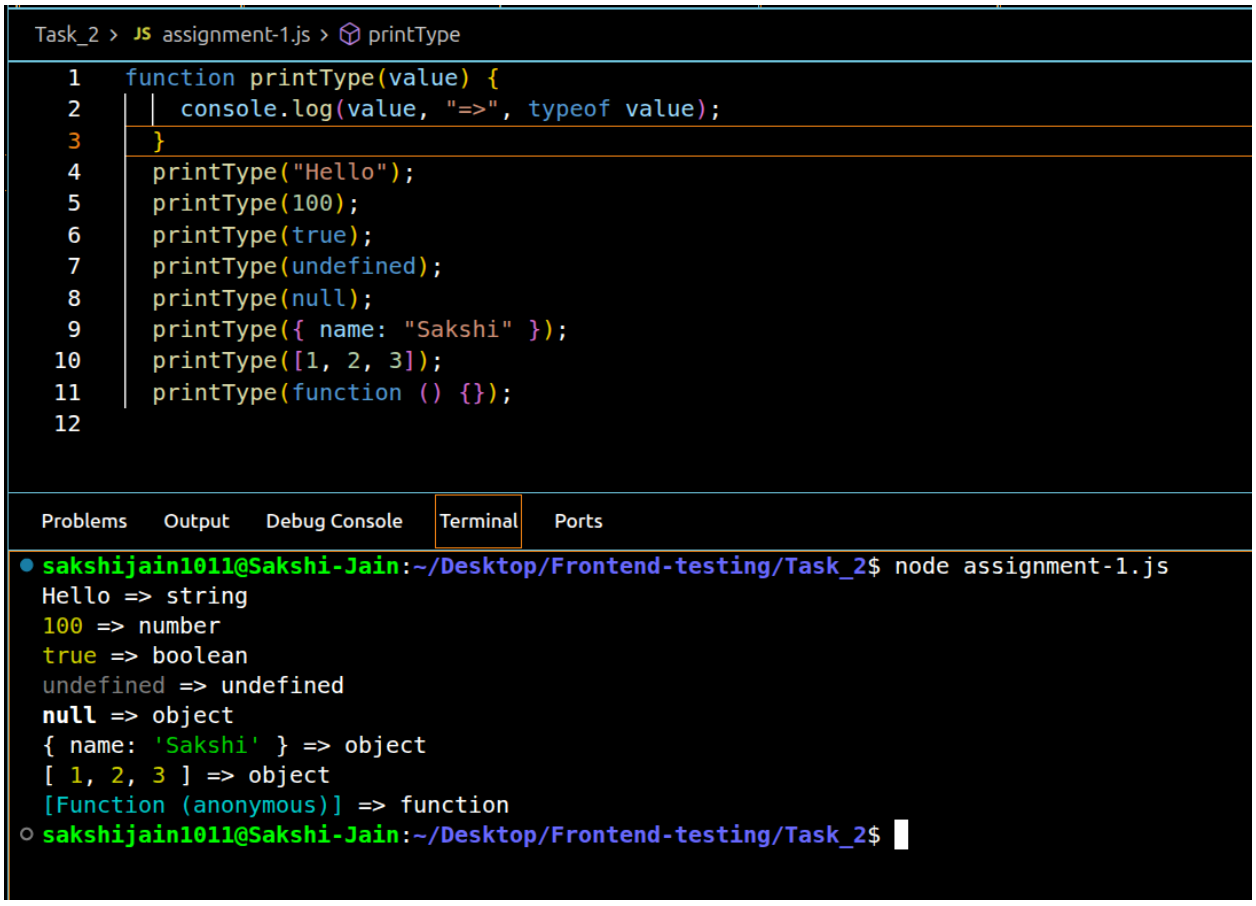
Assignment - 2

Task - 1] Variables & Data Types
Concepts: var, let, const, primitive types

1. Write a function that accepts different data types and prints their type using typeof.



2. Explain the difference between null and undefined with code.
    - **Undefined:** value is not assigned yet
    - **Null:** value is intentionally empty

Code:

Task_2 > task1 > JS assignment2.js > ...

```
1  let user1;          // undefined coz not set
2  let user2 = null;   // null coz empty intentionally
3
4  console.log(user1);
5  console.log(user2);
6
```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task1$ node assignment2.js
undefined
null
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task1$
```

Task -2] Functions

Concepts: normal functions, arrow functions

1. Write a normal function to add two numbers.

Task_2 > task2 > JS assignment-1.js > ...

```
1  function add(a, b) {
2      return a + b;
3  }
4  console.log(add(10, 20));
5
```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task2$ node assignment-1.js
30
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task2$
```

2. Convert the above function into an arrow function.

Task_2 > task2 > JS assignment-2.js > ...

```
1  const addArrow = (a, b) => {
2      return a + b;
3  };
4  console.log(addArrow(10, 20)); // 30
```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task2$ node assignment-2.js
30
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task2$
```

Task - 3] Strings

Concepts: string methods

1. **Write the difference between == and === in java script with examples.**

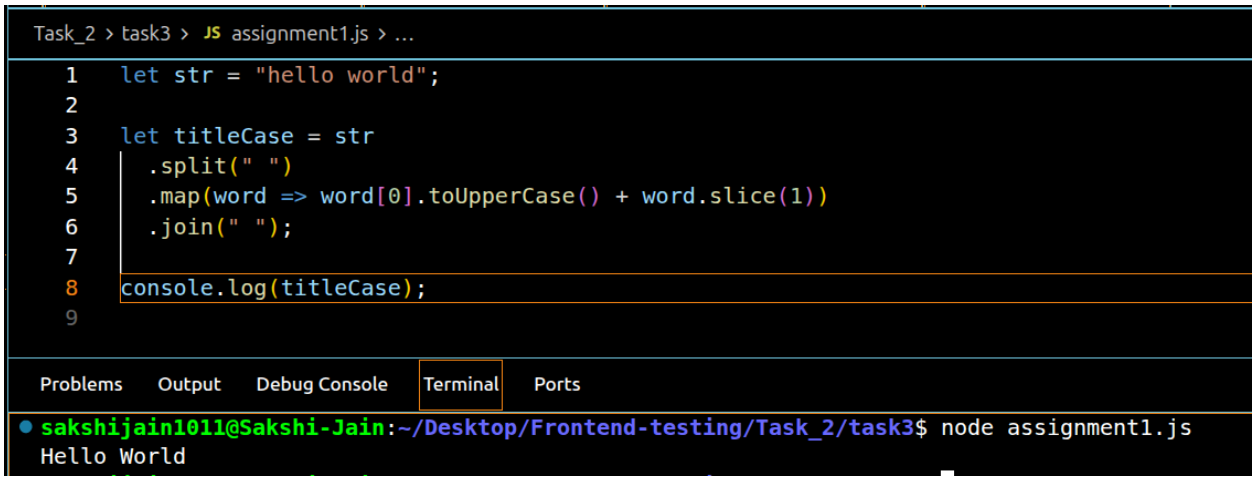
== (Loose Equality):

- Compares **values only**
- Converts types automatically (**type coercion**)

=== (Strict Equality) :

- Compares **value + type**
- No automatic conversion

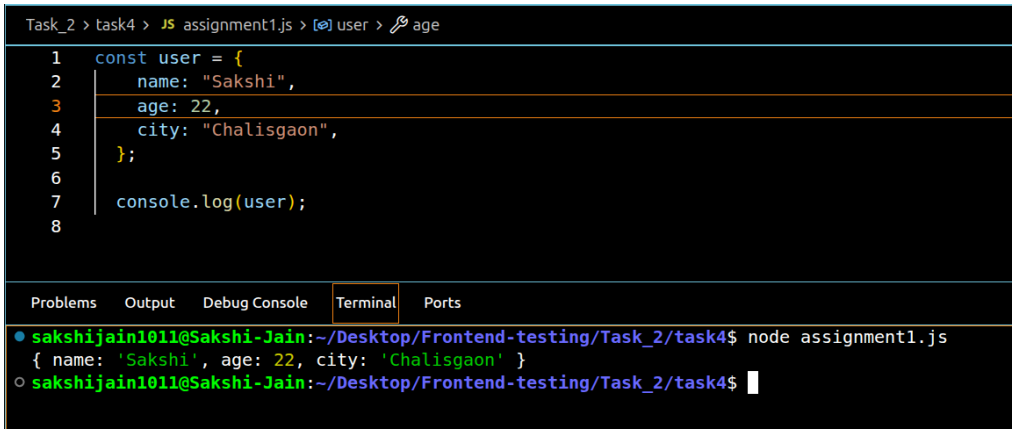
2. **"hello world" convert to title case.**



Task - 4] Objects

Concepts: object creation, access, iteration

1. **Create a user object with properties name, age, and city.**



2. **Print all keys and values using methods Object.keys, Object.values and forEach loop**

Task_2 > task4 > JS assignment2.js > ...

```
1   const user = {
2       name: "Sakshi",
3       age: 22,
4       city: "Chalisgaon",
5     };
6
7   console.log(user);
8
9   const keys = Object.keys(user);
10  console.log("Keys:", keys);
11
12  const values = Object.values(user);
13  console.log("Values:", values);
14
15  Object.keys(user).forEach((key) => {
16      console.log(key, ":", user[key]);
17    });
18
```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task4$ node assignment2.js
{ name: 'Sakshi', age: 22, city: 'Chalisgaon' }
Keys: [ 'name', 'age', 'city' ]
Values: [ 'Sakshi', 22, 'Chalisgaon' ]
name : Sakshi
age : 22
city : Chalisgaon
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task4$
```

3. **Add new property mobileNumber and delete city properties dynamically.**

4. Convert below array of object group by role

```
const users = [
  { name: "Pratik", role: "admin" },
  { name: "Amit", role: "user" },
  { name: "Neha", role: "admin" },
  { name: "Ravi", role: "user" },
];
```

Output:

```
{
  admin: [
    { name: "Pratik", role: "admin" },
    { name: "Neha", role: "admin" }
  ],

  user: [
    { name: "Amit", role: "user" },
    { name: "Ravi", role: "user" }
```

```
  ]
}
```

Task_2 > task4 > JS assignment4.js > ...

```
const users = [
    { name: "Pratik", role: "admin" },
    { name: "Amit", role: "user" },
    { name: "Neha", role: "admin" },
    { name: "Ravi", role: "user" },
  ];
const groupedByRole = users.reduce((acc, user) => {
    const role = user.role;

    if (!acc[role]) {
      acc[role] = [];
    }

    acc[role].push(user);
    return acc;
  }, {});

  console.log(groupedByRole);

```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task4$ node assignment4.js
{
  admin: [
    { name: 'Pratik', role: 'admin' },
    { name: 'Neha', role: 'admin' }
  ],
  user: [ { name: 'Amit', role: 'user' }, { name: 'Ravi', role: 'user' } ]
}
```

Task -5] Array Methods (Important)

Concepts: map, filter, reduce

[20, 4, 23, 56, 1, 23, 65, 78, 45, 3, 9, 6, 23, 1, 50]

1. **Use map to multiply each array element by 2.**

Task_2 > task5 > JS assignment1.js > ...

```
1   const arr = [20, 4, 23, 56, 1, 23, 65, 78, 45, 3, 9, 6, 23, 1, 50];
2
3   const doubled = arr.map((num) => num * 2);
4   console.log("Original Array:", arr);
5   console.log("Modified Array:", doubled);
6
```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task5$ node assignment1.js
Original Array: [
  20,  4, 23, 56, 1, 23,
  65, 78, 45,  3, 9,  6,
  23,  1, 50
]
Modified Array: [
   40,   8,  46, 112,  2, 46,
  130, 156,  90,   6, 18, 12,
   46,   2, 100
]
```

2. Use filter to find numbers greater than 10.

Task_2 > task5 > JS assignment2.js > [] arr

```
1   const arr = [20, 4, 23, 56, 1, 23, 65, 78, 45, 3, 9, 6, 23, 1, 50];
2
3   const greaterThan10 = arr.filter((num) => num > 10);
4   console.log("Original Array:", arr);
5   console.log("Numbers > 10:", greaterThan10);
6
```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task5$ node assignment2.js
Original Array: [
  20,  4, 23, 56, 1, 23,
  65, 78, 45,  3, 9,  6,
  23,  1, 50
]
Numbers > 10: [
  20, 23, 56, 23, 65,
  78, 45, 23, 50
]
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task5$
```

3. Use reduce to find the sum of array elements.

Task_2 > task5 > JS assignment3.js > ...

```
1   const arr = [20, 4, 23, 56, 1, 23, 65, 78, 45, 3, 9, 6, 23, 1, 50];
2
3   const sum = arr.reduce((acc, num) => acc + num, 0);
4   console.log("Sum:", sum);
```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task5$ node assignment3.js
Sum: 407
```

4. **Reverse an array.**

Task_2 > task5 > JS assignment4.js > ...

```
1   const arr = [20, 4, 23, 56, 1, 23, 65, 78, 45, 3, 9, 6, 23, 1, 50];
2
3   const reversed = [...arr].reverse();
4   console.log("Reversed Array:", reversed);
5
```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task5$ node assignment4.js
Reversed Array: [
  50,  1, 23,  6, 9,  3,
  45, 78, 65, 23, 1, 56,
  23,  4, 20
]
```

Task-6] ES6+ Features

Concepts: destructuring, spread, rest

1. **Destructure an object and console name and age from it.**

```
const user = {
  name: "Akshay",
  age: 25,
  city: "Pune"
};
```

output :

```
console.log(name); // Akshay
console.log(age); // 25
```

Task_2 > task6 > JS assignment1.js > ...

```
1  const user = {
2      name: "Akshay",
3      age: 25,
4      city: "Pune",
5  };
6  const { name, age } = user;
7  console.log(name);
8  console.log(age);
9
```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task6$ node assignment1.js
Akshay
25
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task6$
```

2. **Merge two arrays using spread operator.**

Const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
Output:
console.log(mergedArray);
// [1, 2, 3, 4, 5, 6]

Task_2 > task6 > JS assignment2.js > ...

```
1  const arr1 = [1, 2, 3];
2  const arr2 = [4, 5, 6];
3
4  const mergedArray = [...arr1, ...arr2];
5  console.log(mergedArray);
6
```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task6$ node assignment2.js
[ 1, 2, 3, 4, 5, 6 ]
```

3. **Create a function accepting 5 numbers using rest parameters and display sum of all numbers from function.**

Task_2 > task6 > JS assignment3.js > sumOfNumbers

```
1  function sumOfNumbers(...numbers) {
2      const sum = numbers.reduce((acc, num) => acc + num, 0);
3      console.log("Sum:", sum);
4  }
5  sumOfNumbers(10, 20, 30, 40, 50);
6
```

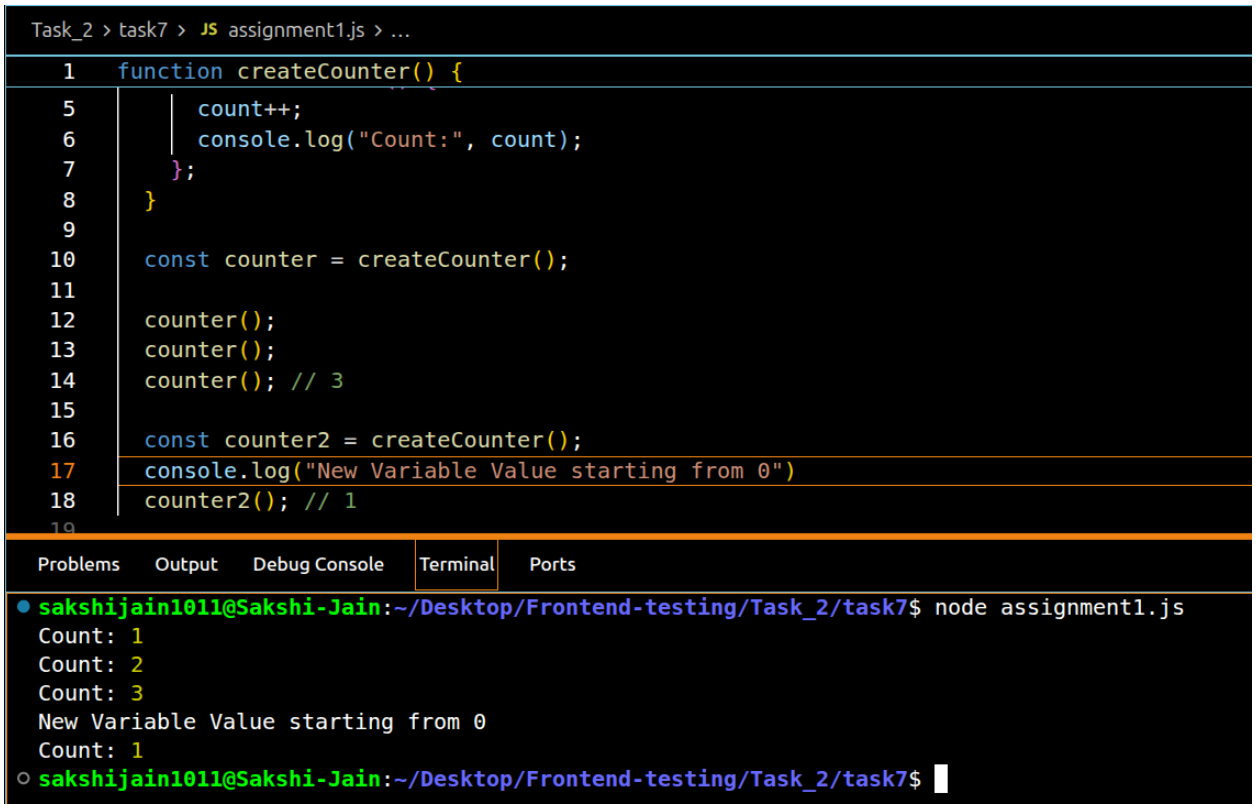
Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task6$ node assignment3.js
Sum: 150
```

Task - 7] Closures

Concepts: closures, lexical scope

1. **Create a counter function using closure.**

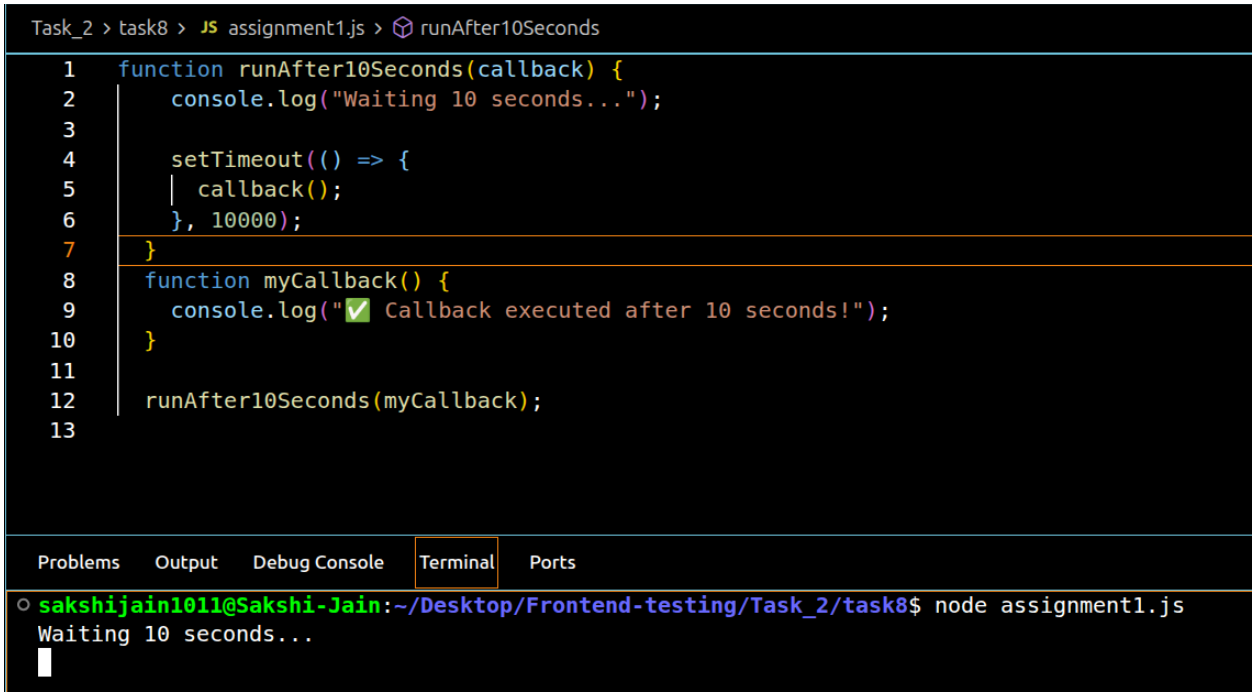


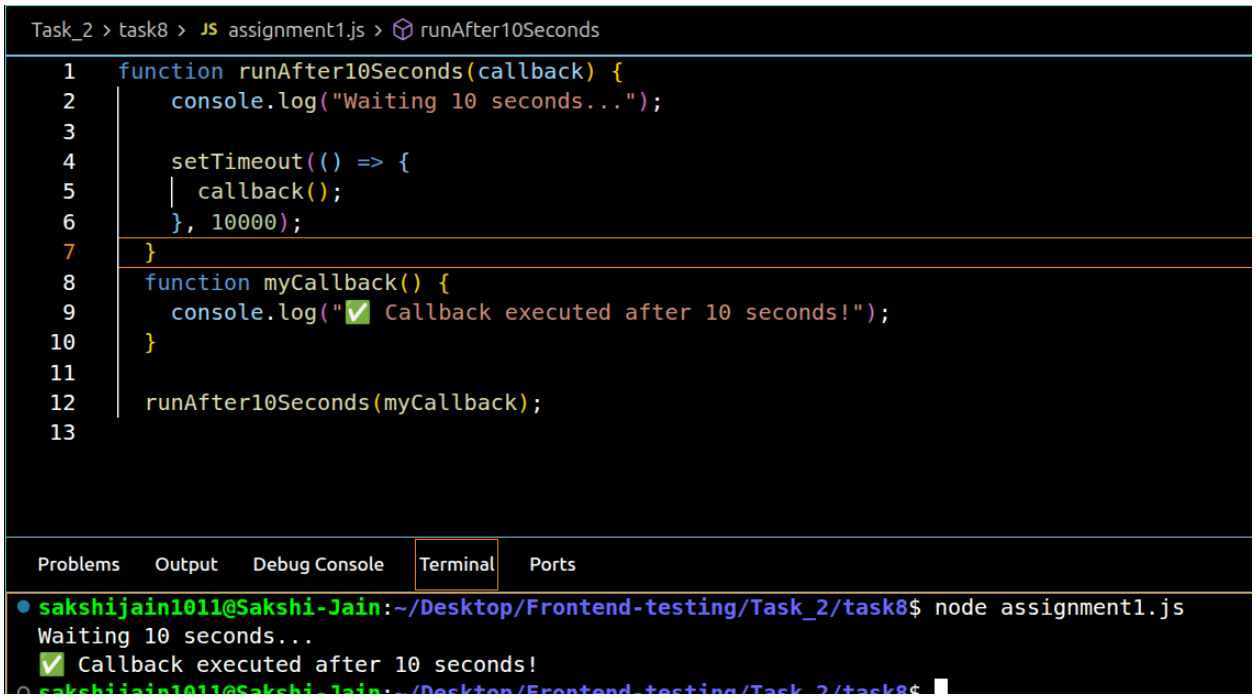
2. **Explain how inner functions access outer variables.**
    - **Due to lexical scope**
    - **Lexical Scope means: functions can access variables from where they are written in code (not where they are called).**
    - **So innerFunction() can use outerValue because it is written inside outerFunction().**

Task -8] Callbacks

Concepts: callback functions

1. **Create a function that accepts a callback and executes it after 10 seconds.**

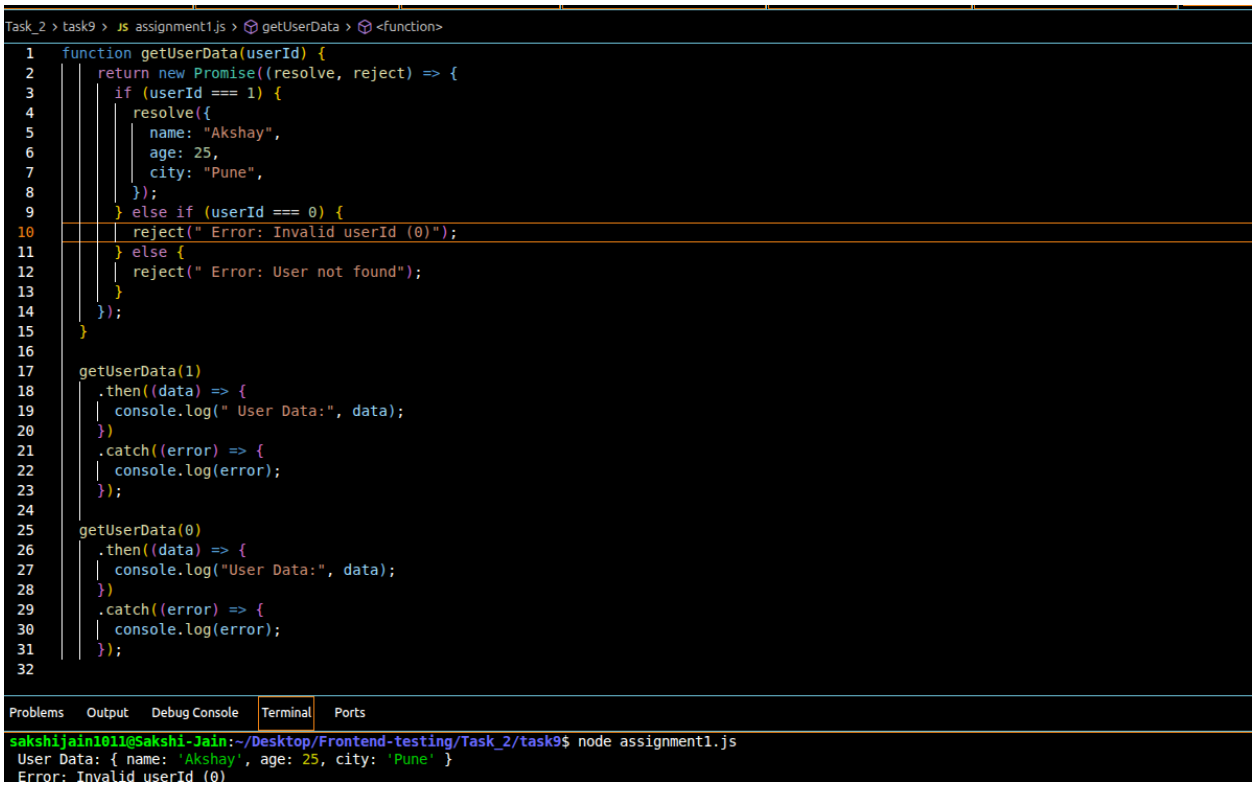




Task - 9] Promises

Concepts: Promise, then, catch

1. **Create a function called getUserData that:**
    - **Returns a Promise**
    - **Resolves with user details object contains name, age, city if userId is 1**
    - **Rejects with an error message if userId is 0**
    - **Handles the response using .then() and .catch()**



```
Task_2 > task9 > JS assignment1.js > getUserData > <function>
1  function getUserData(userId) {
2      return new Promise((resolve, reject) => {
3          if (userId === 1) {
4              resolve({
5                  name: "Akshay",
6                  age: 25,
7                  city: "Pune",
8              });
9          } else if (userId === 0) {
10             reject(" Error: Invalid userId (0)");
11         } else {
12             reject(" Error: User not found");
13         }
14     });
15 }
16
17 getUserData(1)
18     .then((data) => {
19         console.log(" User Data:", data);
20     })
21     .catch((error) => {
22         console.log(error);
23     });
24
25 getUserData(0)
26     .then((data) => {
27         console.log("User Data:", data);
28     })
29     .catch((error) => {
30         console.log(error);
31     });
32

Problems  Output  Debug Console  Terminal  Ports

sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task9$ node assignment1.js
 User Data: { name: 'Akshay', age: 25, city: 'Pune' }
 Error: Invalid userId (0)
```

2. **Guess the execution sequence of below code**

```
console.log("1: Start");

setTimeout(() => {
  console.log("2: setTimeout");
}, 0);

Promise.resolve().then(() => {
  console.log("3: Promise");
```

```
});

console.log("4: End");
```

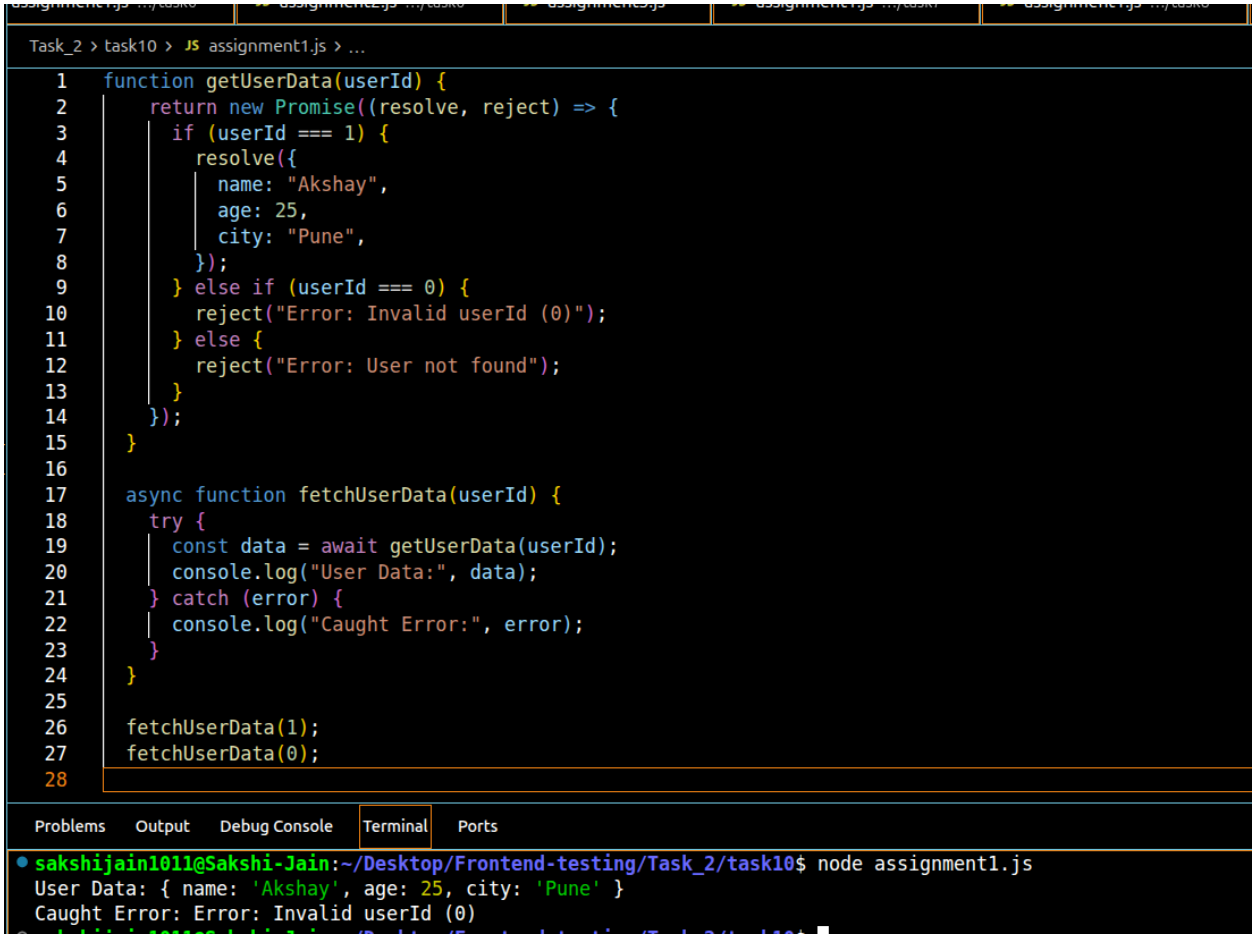
Why This Output?

- **console.log() runs immediately (sync)**
- **Promise.then() goes to Microtask Queue (runs first)**
- **setTimeout() goes to Callback/Task Queue (runs after microtasks)**

Task - 10] Async / Await

Concepts: async functions, error handling

1. **Convert the question 1 from assignment 9, to async/await with try , catch block**

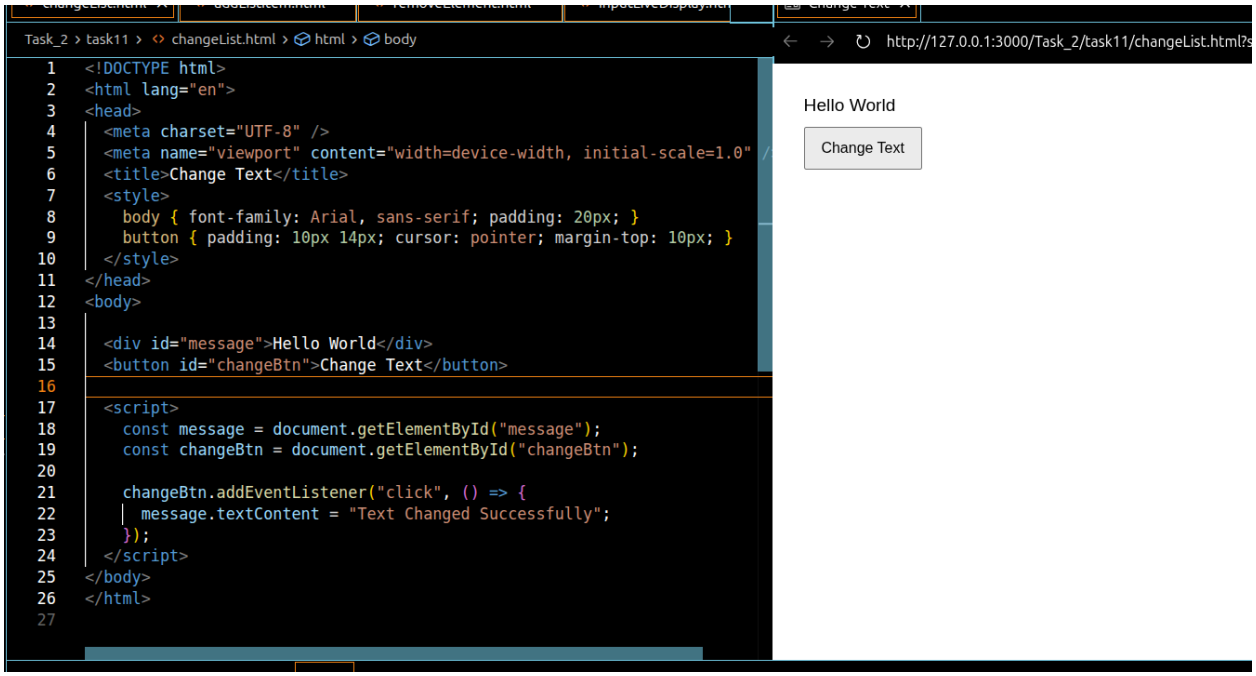


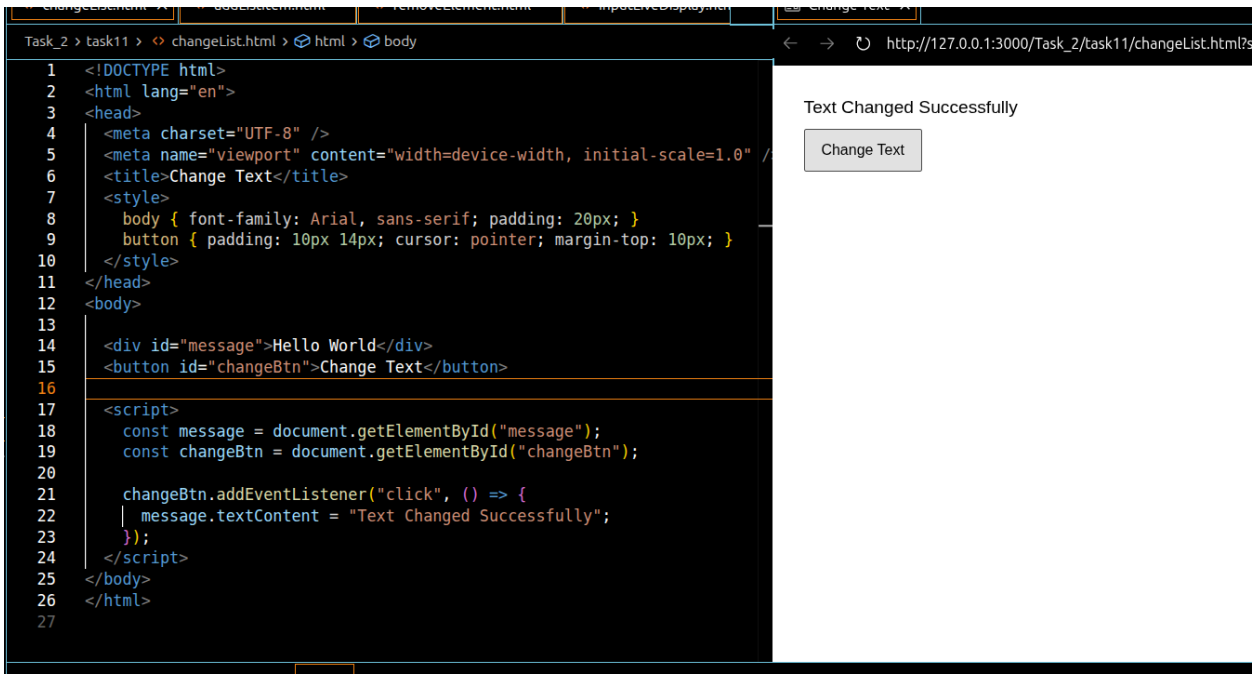
Task - 11] DOM Manipulation

Concepts: DOM selection, events

1. **Take one div with some text and change text of an element on button click.**

```
<div id="message">Hello World</div>
<button id="changeBtn">Change Text</button>
```



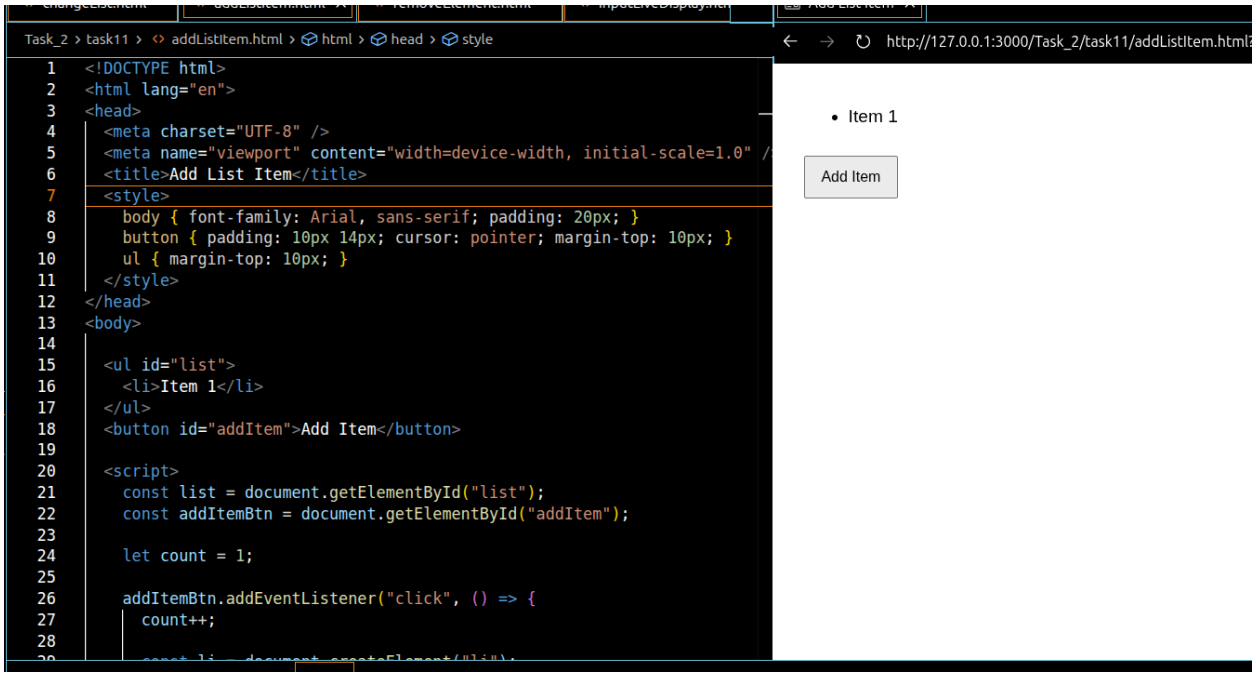


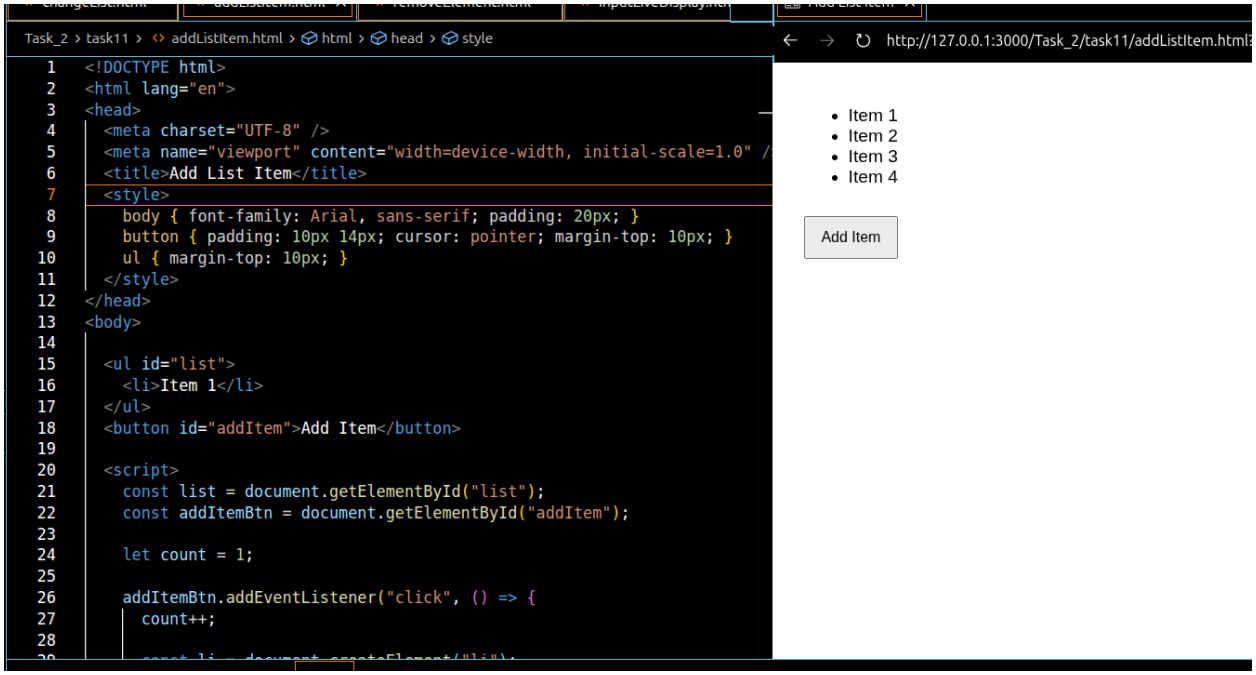
Live Url: https://sakshijain-josh.github.io/Change_Text/

2. **Add a new list item dynamically.**

   **<ul id="list">
Item 1
**

   <button id="addItem">Add Item</button>



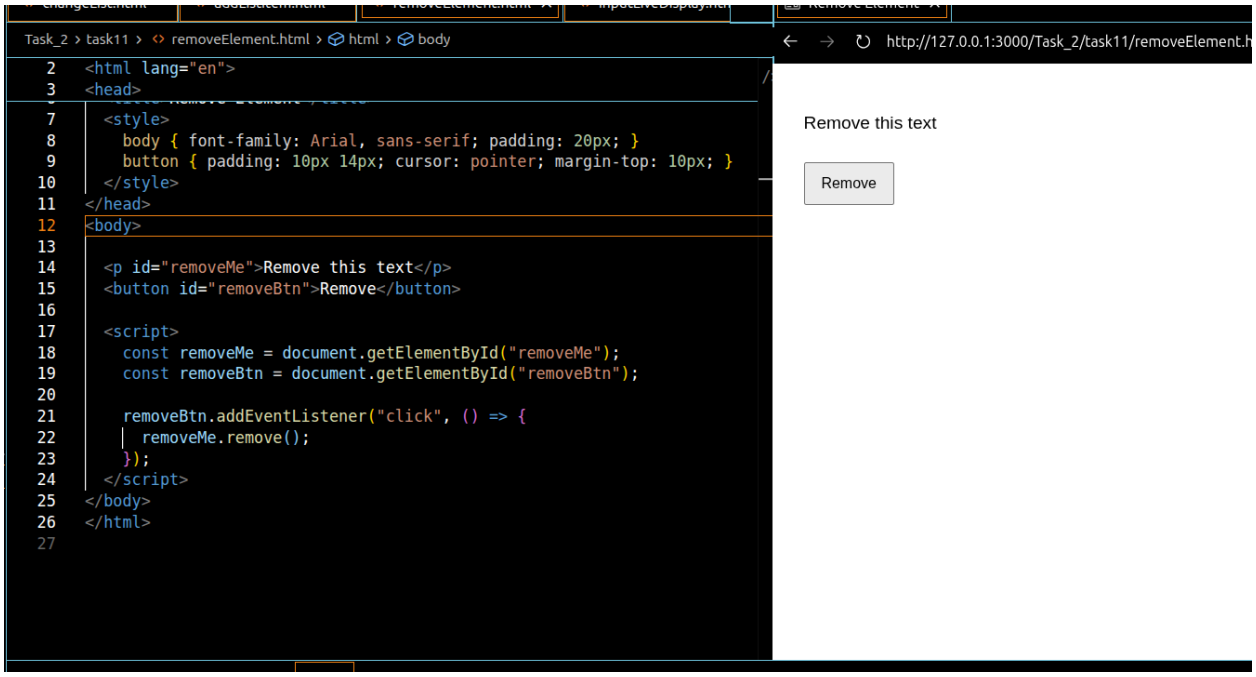


Live Url: https://sakshijain-josh.github.io/Add-Element/

3. **Remove an element from the DOM.**

   ** Remove this text **

<button id="removeBtn">Remove</button>



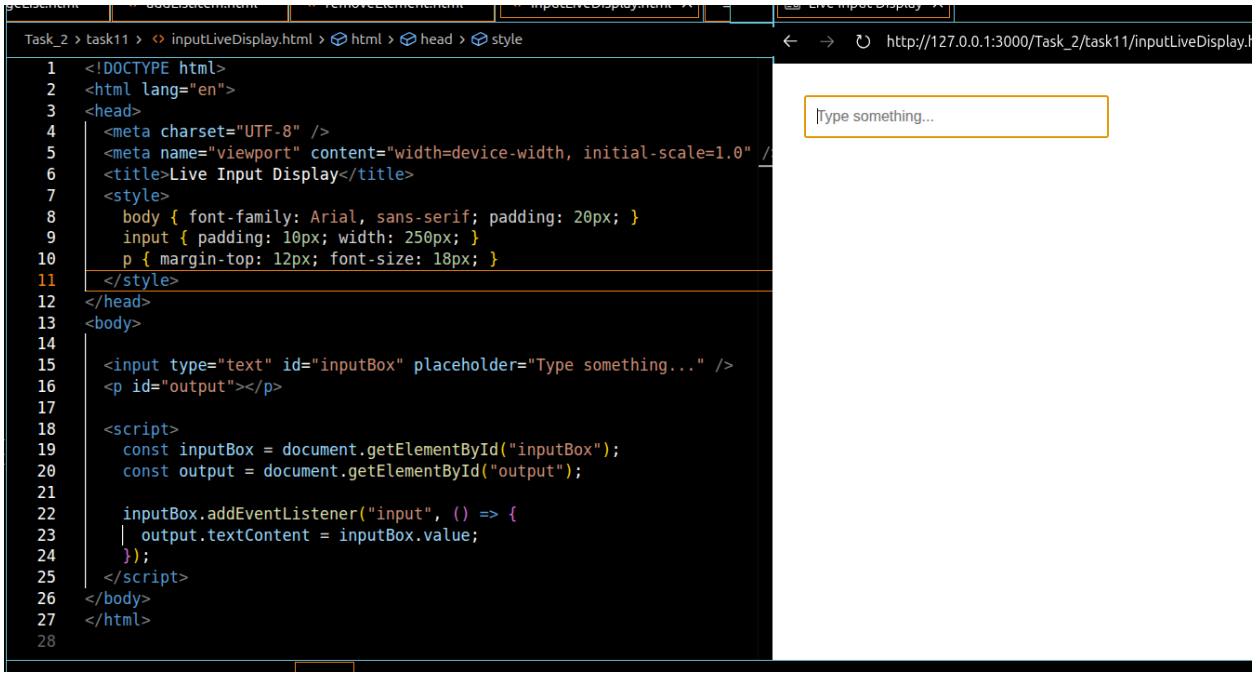

```
<html lang="en">
<head>
  <style>
    body { font-family: Arial, sans-serif; padding: 20px; }
    button { padding: 10px 14px; cursor: pointer; margin-top: 10px; }
  </style>
</head>
<body>

  <p id="removeMe">Remove this text</p>
  <button id="removeBtn">Remove</button>

  <script>
    const removeMe = document.getElementById("removeMe");
    const removeBtn = document.getElementById("removeBtn");

    removeBtn.addEventListener("click", () => {
      removeMe.remove();
    });
  </script>
</body>
</html>
```

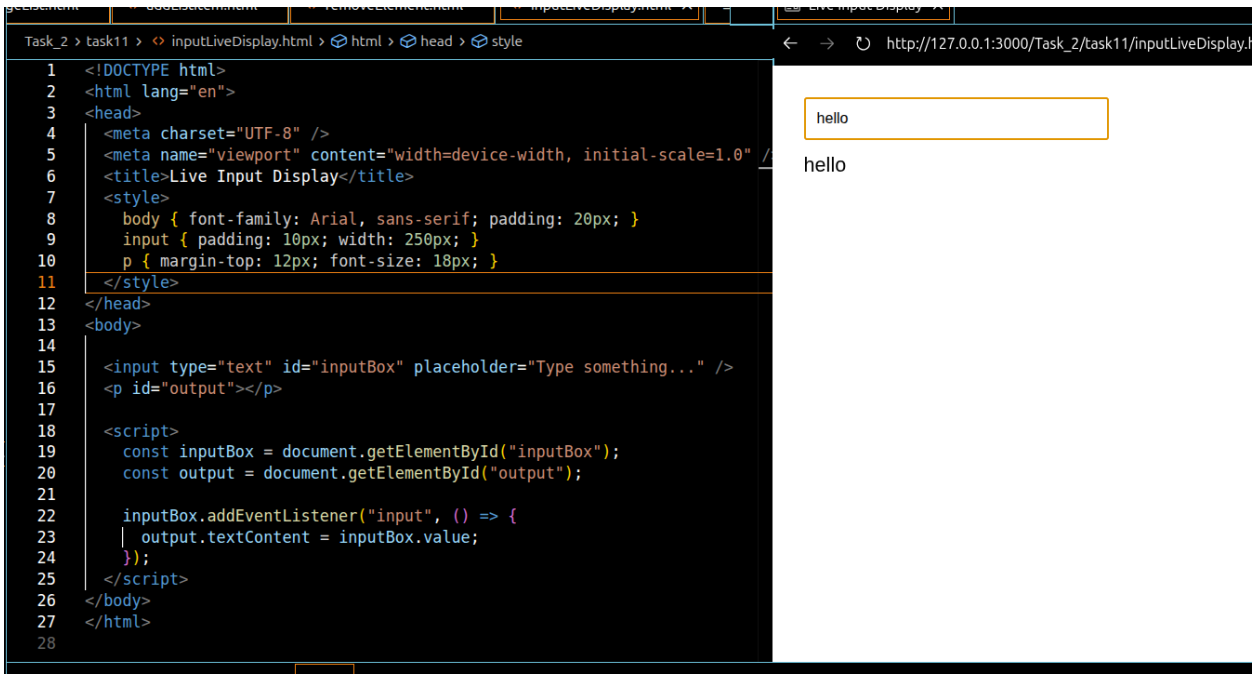

```
<html lang="en">
<head>
  <style>
    body { font-family: Arial, sans-serif; padding: 20px; }
    button { padding: 10px 14px; cursor: pointer; margin-top: 10px; }
  </style>
</head>
<body>

  <p id="removeMe">Remove this text</p>
  <button id="removeBtn">Remove</button>

  <script>
    const removeMe = document.getElementById("removeMe");
    const removeBtn = document.getElementById("removeBtn");

    removeBtn.addEventListener("click", () => {
      removeMe.remove();
    });
  </script>
</body>
</html>
```

Live Url: https://sakshijain-josh.github.io/remove_element/

4. **Display input value on screen while typing.**

```
<input type="text" id="inputBox" />
<p id="output"></p>
```





LIve URL: https://sakshijain-josh.github.io/Input-Display/

Task - 12] Timers

Concepts: setInterval

1. Create a countdown timer using setInterval.

Task_2 › task12 › JS countTimer.js › ...

```
let timeLeft = 10;

const intervalId = setInterval(() => {
  console.log("Time Left:", timeLeft);

  timeLeft--;

  if (timeLeft < 0) {
    clearInterval(intervalId);
    console.log("Countdown Finished");
  }
}, 1000);
```

Problems Output Debug Console Terminal Ports

```
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task12$ node countTimer.js
Time Left: 10
Time Left: 9
Time Left: 8
Time Left: 7
Time Left: 6
Time Left: 5
Time Left: 4
Time Left: 3
Time Left: 2
Time Left: 1
Time Left: 0
Countdown Finished
sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task12$
```