

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.

```
Task_2 > JS assignment-1.js > printType

1 function printType(value) {
2   | console.log(value, "=>", typeof value);
3   | }
4   |
5   | printType("Hello");
6   | printType(100);
7   | printType(true);
8   | printType(undefined);
9   | printType(null);
10  | printType({ name: "Sakshi" });
11  | printType([1, 2, 3]);
12  | printType(function () {});

Problems Output Debug Console Terminal Ports
● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2$ node assignment-1.js
Hello => string
100 => number
true => boolean
undefined => undefined
null => object
{ name: 'Sakshi' } => object
[ 1, 2, 3 ] => object
[Function (anonymous)] => function
○ sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2$
```

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_2 > task3 > JS assignment1.js > ...
1  let str = "hello world";
2
3  let titleCase = str
4    .split(" ")
5    .map(word => word[0].toUpperCase() + word.slice(1))
6    .join(" ");
7
8  console.log(titleCase);
9

Problems  Output  Debug Console  Terminal  Ports
• sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task3$ node assignment1.js
Hello World
```

Task - 4] Objects

Concepts: object creation, access, iteration

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

```
Task_2 > task4 > JS assignment1.js > [?] user > ? age
1  const user = {
2    name: "Sakshi",
3    age: 22,
4    city: "Chalisingaon",
5  };
6
7  console.log(user);
8

Problems  Output  Debug Console  Terminal  Ports
• sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task4$ node assignment1.js
{ name: 'Sakshi', age: 22, city: 'Chalisingaon' }
○ sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task4$
```

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.

```
Task_2 > task4 > JS assignment3.js > ...

1  const user = {
2      name: "Sakshi",
3      age: 22,
4      city: "Chalisgaon",
5  };
6
7  // Add
8  user.mobileNumber = "9876543210";
9  console.log("After adding mobileNumber:", user);
10
11 // Delete
12 delete user.city;
13 console.log("After deleting city:", user);
14
15

Problems  Output  Debug Console  Terminal  Ports
● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task4$ node assignment3.js
After adding mobileNumber: {
  name: 'Sakshi',
  age: 22,
  city: 'Chalisgaon',
  mobileNumber: '9876543210'
}
After deleting city: { name: 'Sakshi', age: 22, mobileNumber: '9876543210' }
```

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 > ...

1  const users = [
2    { name: "Pratik", role: "admin" },
3    { name: "Amit", role: "user" },
4    { name: "Neha", role: "admin" },
5    { name: "Ravi", role: "user" },
6  ];
7  const groupedByRole = users.reduce((acc, user) => {
8    const role = user.role;
9
10     if (!acc[role]) {
11       acc[role] = [];
12     }
13
14     acc[role].push(user);
15     return acc;
16   }, {});
17
18   console.log(groupedByRole);
19
```

| Problems | Output | Debug Console | Terminal | Ports |
|---|--------|---------------|----------|-------|
| <pre>● 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' }] }</pre> | | | | |

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.

```
Task_2 > task7 > JS assignment1.js > ...
1  function createCounter() {
5      count++;
6      console.log("Count:", count);
7  };
8  }
9
10 const counter = createCounter();
11
12 counter();
13 counter();
14 counter(); // 3
15
16 const counter2 = createCounter();
17 console.log("New Variable Value starting from 0")
18 counter2(); // 1
19
```

| Problems | Output | Debug Console | Terminal | Ports |
|--|--------|---------------|----------|-------|
| ● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task7\$ node assignment1.js | | | | |
| Count: 1 | | | | |
| Count: 2 | | | | |
| Count: 3 | | | | |
| New Variable Value starting from 0 | | | | |
| Count: 1 | | | | |
| ○ sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task7\$ | | | | |

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_2 > task8 > JS assignment1.js > runAfter10Seconds

1 function runAfter10Seconds(callback) {
2   console.log("Waiting 10 seconds...");
3
4   setTimeout(() => {
5     callback();
6   }, 10000);
7 }
8 function myCallback() {
9   console.log("✅ Callback executed after 10 seconds!");
10 }
11
12 runAfter10Seconds(myCallback);
13
```

Problems Output Debug Console Terminal Ports

○ sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task8\$ node assignment1.js
Waiting 10 seconds...
█

```
Task_2 > task8 > JS assignment1.js > runAfter10Seconds

1 function runAfter10Seconds(callback) {
2   console.log("Waiting 10 seconds...");
3
4   setTimeout(() => {
5     callback();
6   }, 10000);
7 }
8 function myCallback() {
9   console.log("✅ Callback executed after 10 seconds!");
10 }
11
12 runAfter10Seconds(myCallback);
13
```

Problems Output Debug Console Terminal Ports

● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task8\$ node assignment1.js
Waiting 10 seconds...
✅ Callback executed after 10 seconds!
○ sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task8\$ █

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(() => {

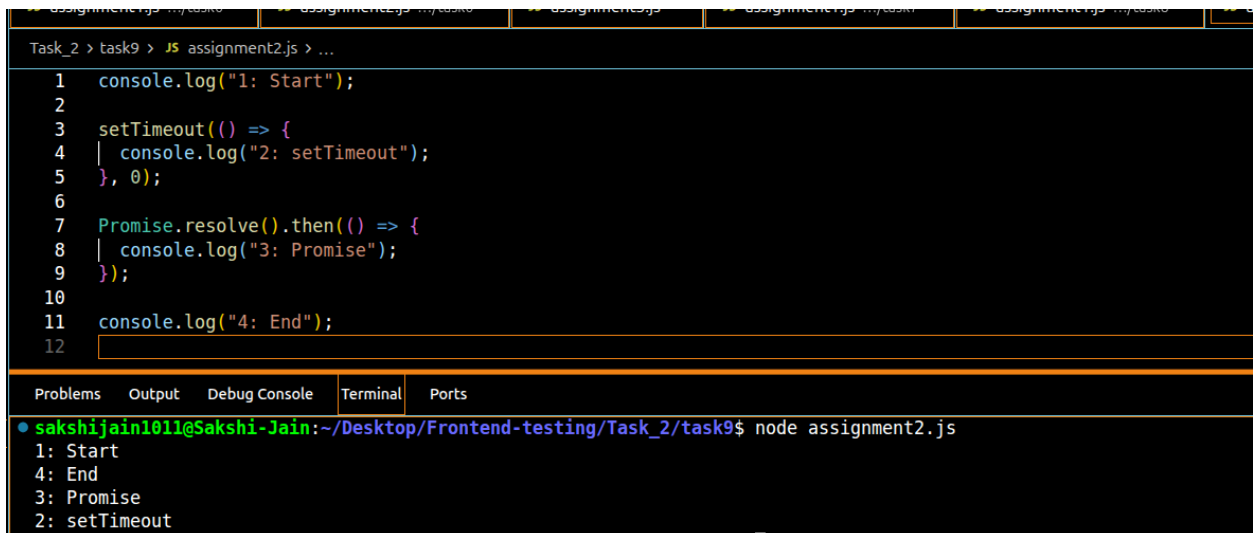
console.log("3: Promise");

```
});
```

```
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)



The screenshot shows a VS Code editor with a file named `assignment2.js` containing the following code:

```
1 console.log("1: Start");
2
3 setTimeout(() => {
4 |   console.log("2: setTimeout");
5 | }, 0);
6
7 Promise.resolve().then(() => {
8 |   console.log("3: Promise");
9 | });
10
11 console.log("4: End");
12
```

Below the editor, the `Terminal` tab is active, showing the command `node assignment2.js` and its output:

```
1: Start
4: End
3: Promise
2: setTimeout
```

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_2 > task10 > JS assignment1.js > ...
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 async function fetchUserData(userId) {
18   try {
19     const data = await getUserData(userId);
20     console.log("User Data:", data);
21   } catch (error) {
22     console.log("Caught Error:", error);
23   }
24 }
25
26 fetchUserData(1);
27 fetchUserData(0);
28
```

Problems Output Debug Console Terminal Ports

```
● sakshijain1011@Sakshi-Jain:~/Desktop/Frontend-testing/Task_2/task10$ node assignment1.js
User Data: { name: 'Akshay', age: 25, city: 'Pune' }
Caught Error: Error: Invalid userId (0)
```

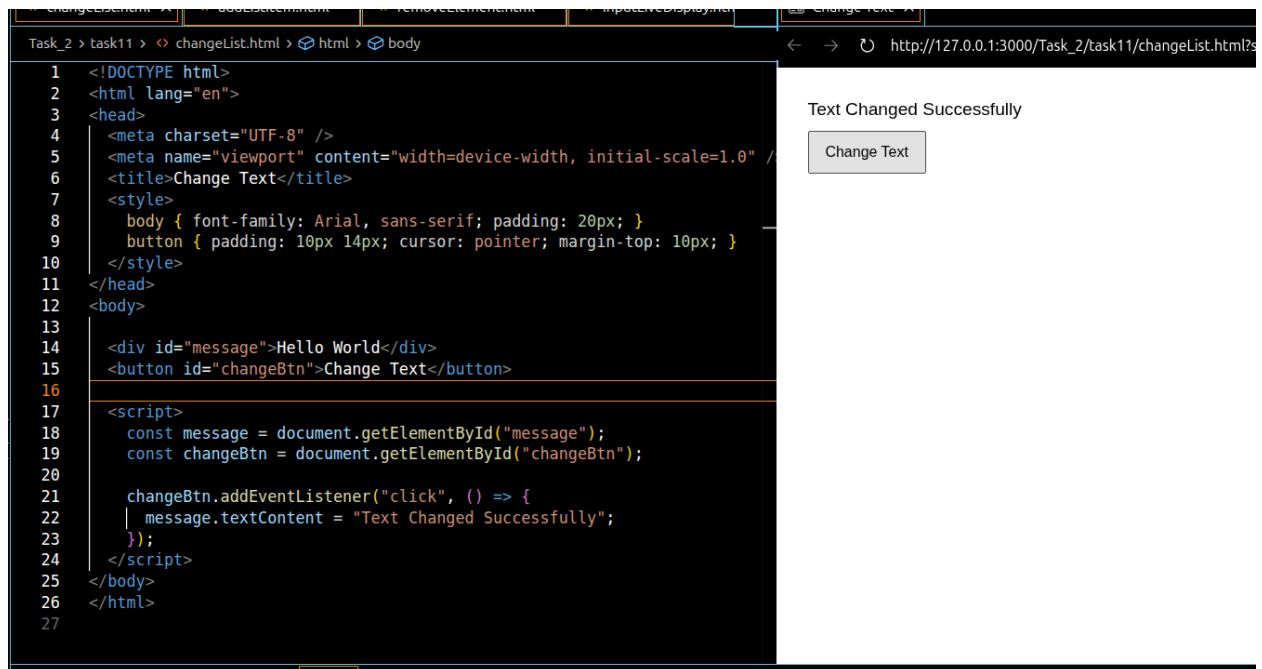
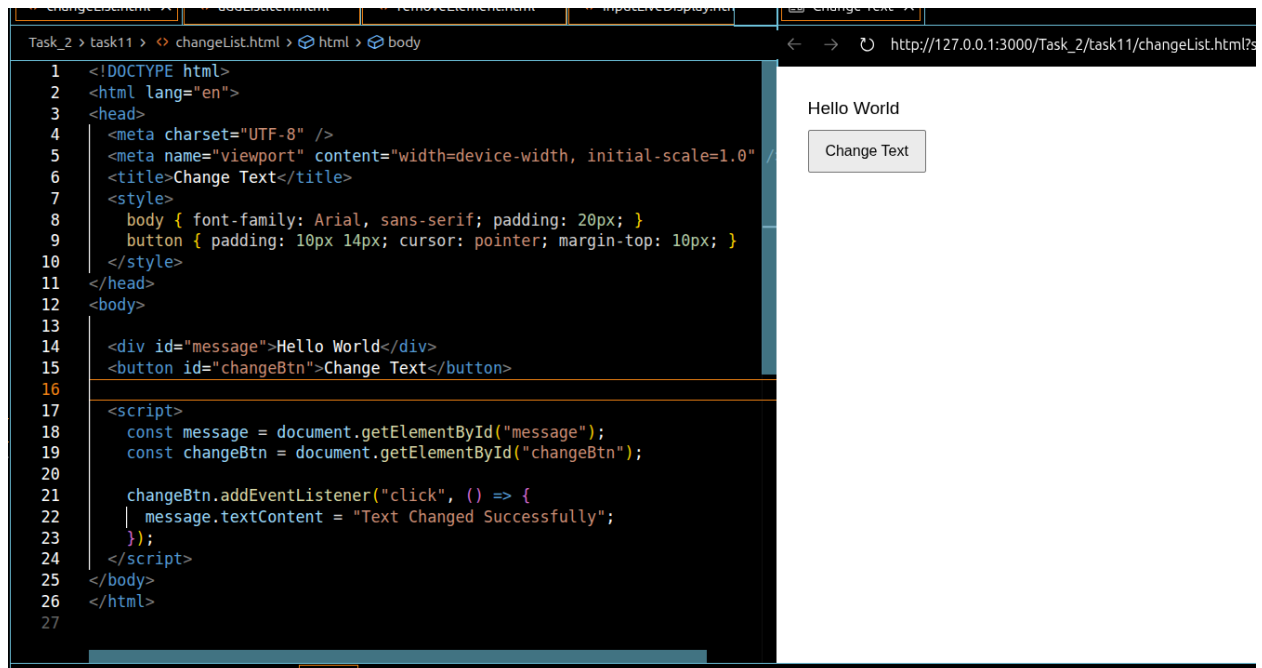
Task - 11] DOM Manipulation

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>`

```
Task_2 > task11 > addListItem.html > html > head > style
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <title>Add List Item</title>
7   <style>
8     body { font-family: Arial, sans-serif; padding: 20px; }
9     button { padding: 10px 14px; cursor: pointer; margin-top: 10px; }
10    ul { margin-top: 10px; }
11  </style>
12 </head>
13 <body>
14
15   <ul id="list">
16     <li>Item 1</li>
17   </ul>
18   <button id="addItem">Add Item</button>
19
20   <script>
21     const list = document.getElementById("list");
22     const addItemBtn = document.getElementById("addItem");
23
24     let count = 1;
25
26     addItemBtn.addEventListener("click", () => {
27       count++;
28     });
29     // const li = document.getElementById("li");
```

• Item 1

Add Item

```
Task_2 > task11 > addListItem.html > html > head > style
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <title>Add List Item</title>
7   <style>
8     body { font-family: Arial, sans-serif; padding: 20px; }
9     button { padding: 10px 14px; cursor: pointer; margin-top: 10px; }
10    ul { margin-top: 10px; }
11  </style>
12 </head>
13 <body>
14
15   <ul id="list">
16     <li>Item 1</li>
17   </ul>
18   <button id="addItem">Add Item</button>
19
20   <script>
21     const list = document.getElementById("list");
22     const addItemBtn = document.getElementById("addItem");
23
24     let count = 1;
25
26     addItemBtn.addEventListener("click", () => {
27       count++;
28     });
29     // const li = document.getElementById("li");
```

• Item 1
• Item 2
• Item 3
• Item 4

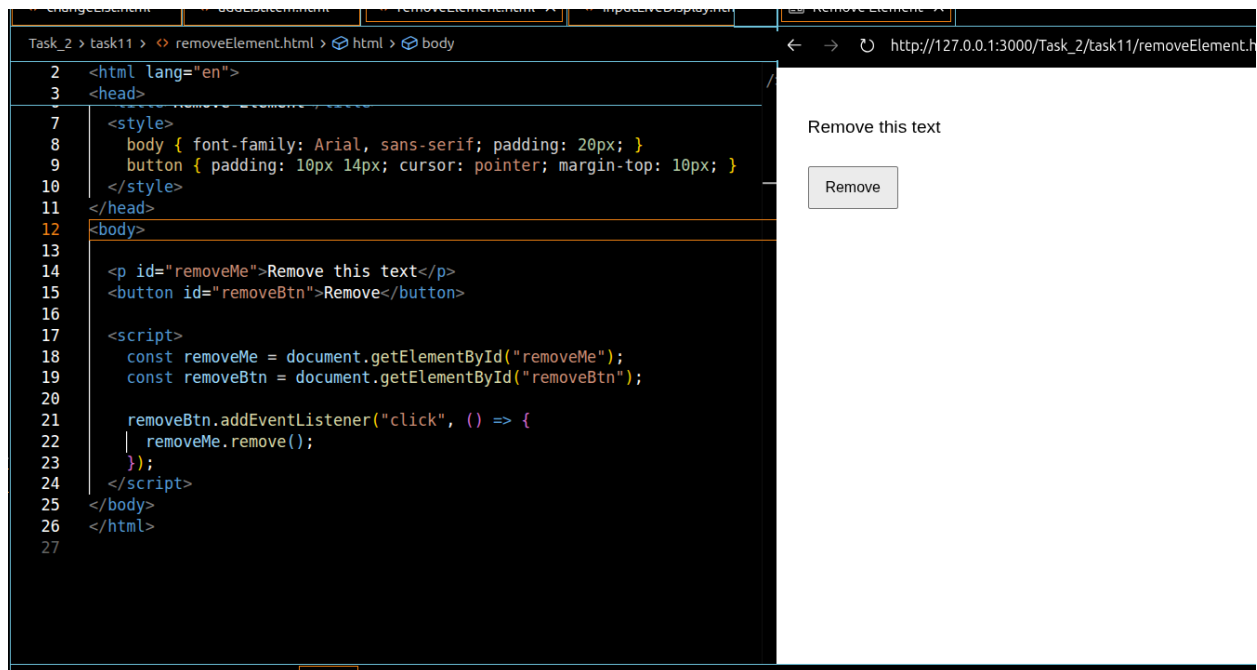
Add Item

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

3. Remove an element from the DOM.

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

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

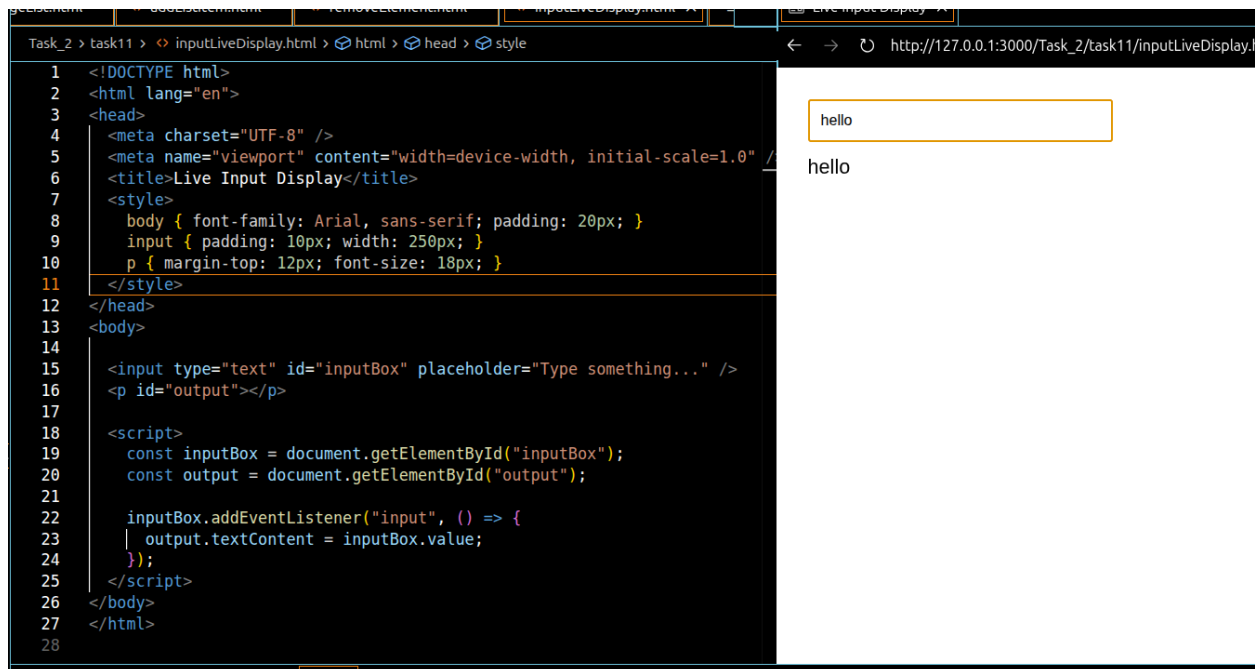
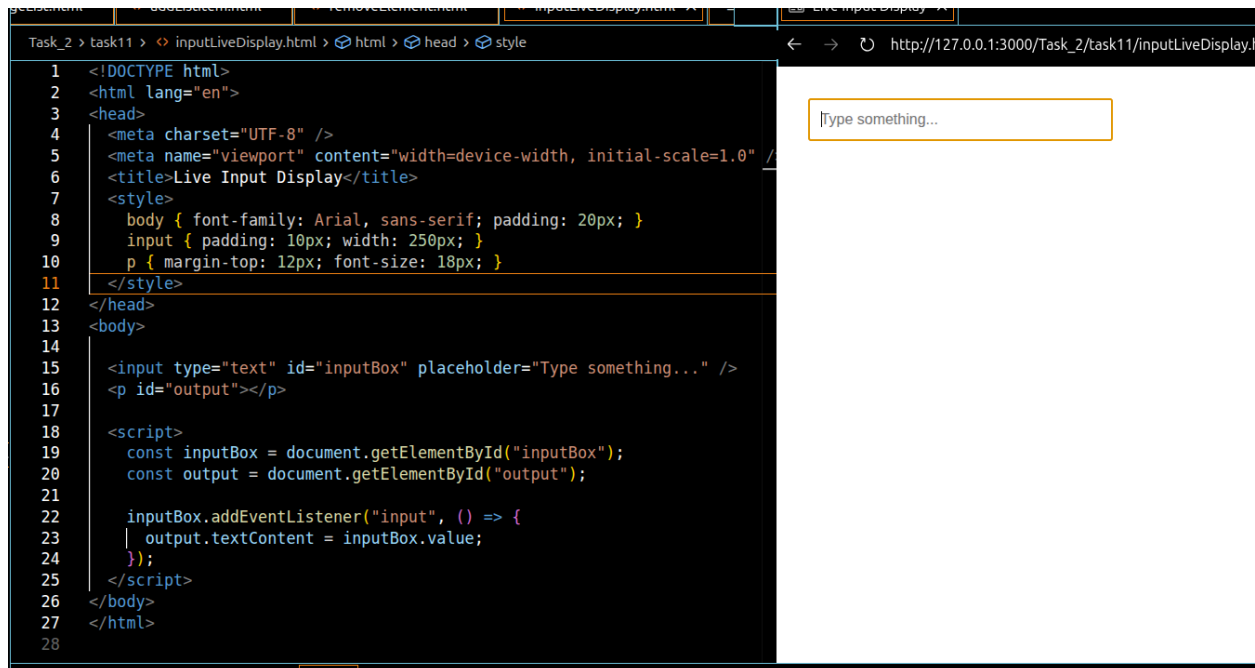


Live Url: https://sakshijain-josh.github.io/remove_element/

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 > ...  
1 let timeLeft = 10;  
2  
3 const intervalId = setInterval(() => {  
4   console.log("Time Left:", timeLeft);  
5  
6   timeLeft--;  
7  
8   if (timeLeft < 0) {  
9     clearInterval(intervalId);  
10    console.log("Countdown Finished");  
11  }  
12 }, 1000);  
13
```

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$
```