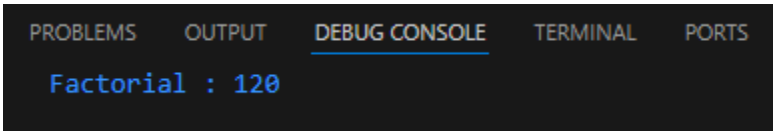
Week 3 and 4:

1.Recursion and stack:

Task 1:

```
<html>
<head>
   <title>Factorial</title>
</head>
<body>
   <script>
      function factorial(n){
         if(n<=1){
            return 1;
         }
         else{
            return n * factorial(n-1);
         }
      }
      console.log("Factorial : "+factorial(5));
   </script>
</body>
</html>
```

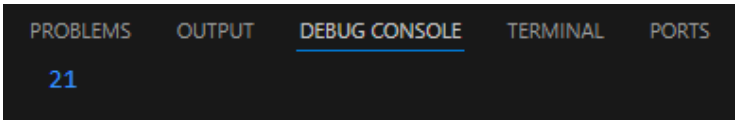
Output:



Task 2:

```
<html>
<head>
   <title>Fibonacci series</title>
</head>
<body>
   <script>
      function f(n){
         if(n==0){
            return 0;
         }
         if(n==1){
            return 1;
         }
         return f(n-1) + f(n-2);
      }
      console.log(f(8));
   </script>
</body>
</html>
```

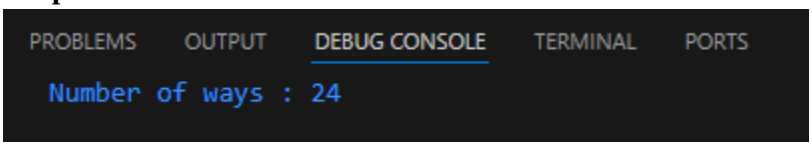
Output:



Task 3:

```
<html>
<head>
   <title>Climbing Stairs</title>
</head>
<body>
   <script>
      function f(n){
         if(n==0){
            return 1;
         }
         if(n<0){
            return 0;
         }
         return f(n-1) + f(n-2) + f(n-3);
      }
      console.log("Number of ways : "+f(6));
   </script>
</body>
</html>
```

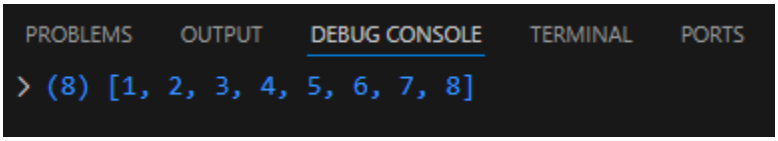
Output:



Task 4:

```
<html>
<head>
   <title>Climbing Stairs</title>
</head>
<body>
   <script>
      function flattenArray(arr){
         var result = [];
         arr.forEach(num => {
            if(Array.isArray(num)){
               result = result.concat(flattenArray(num));
            }
            else{
               result.push(num);
            }
         });
         return result;
      }
      var nestedArray = [1,[2,[3,4],5],[6,7],8];
      console.log(flattenArray(nestedArray));
   </script>
</body>
</html>
```

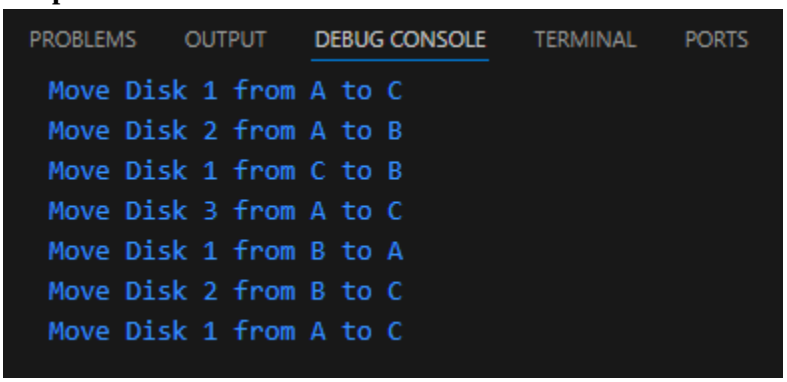
Output:



Task 5:

```
<html>
<head>
   <title>Climbing Stairs</title>
</head>
<body>
   <script>
      function towerOfHanoi(n, source, target, auxiliary) {
      if (n === 1) {
         console.log("Move Disk 1 from " + source + " to " + target);
         return;
      }
      towerOfHanoi(n - 1, source, auxiliary, target);
      console.log("Move Disk "+n+" from " + source + " to " + target);
      towerOfHanoi(n - 1, auxiliary, target, source);
      }
      towerOfHanoi(3, 'A', 'C', 'B');
   </script>
</body>
</html>
```

Output:



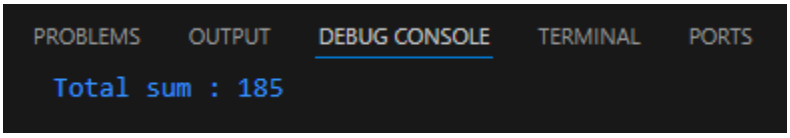
2.JSON and variable length arguments/spread syntax:

Task 1:

```
<html>
<head>
   <title>Arbitary sum</title>
</head>
<body>
   <script>
      function sum(...args){
         var total = 0;
         for(let n of args){
            total += n;
         }
```

```
        return total;
    }
    console.log("Total sum : " + sum(56,4,3,56,7,5,6,4,7,3,5,6,4,6,4,6,3));
  </script>
</body>
</html>
```

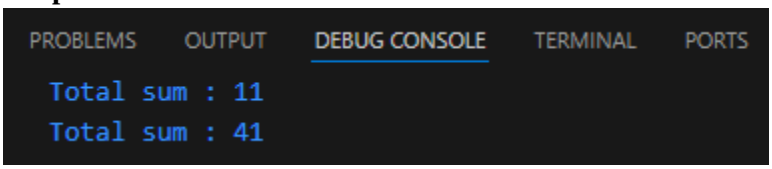
Output:



Task 2:

```
<html>
<head>
  <title>Arbitary array sum</title>
</head>
<body>
  <script>
    function sum(...args){
      var total = 0;
      for(let n of args){
        for(let m of n){
          total += m;
        }
      }
      return total;
    }
    console.log("Total sum : " + sum([2,4,5]));
    console.log("Total sum : " + sum([2,4,5],[6,2,3],[6,4,9]));
  </script>
</body>
</html>
```

Output:

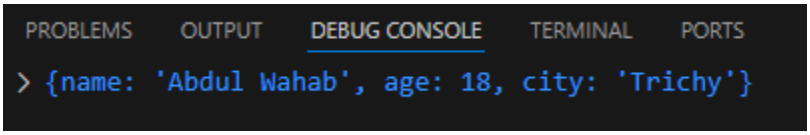


Task 3:

```
<html>
<head>
  <title>Arbitary array sum</title>
</head>
<body>
  <script>
    let student = {
      name : "Abdul Wahab",
      age : 18,
      city : "Trichy"
    };
    console.log(JSON.parse(JSON.stringify(student)));
```

```
    </script>
</body>
</html>
```

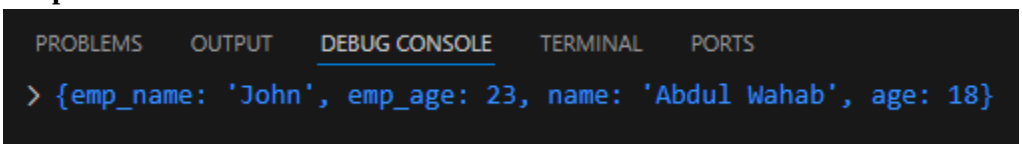
Output:



Task 4:

```
<html>
<head>
    <title>Merge object</title>
</head>
<body>
    <script>
        function merge(employee,student){
            return {...employee,...student};
        }
        let employee = {
            emp_name : "John",
            emp_age : 23
        }
        let student = {
            name : "Abdul Wahab",
            age : 18
        };
        console.log(merge(employee,student));
    </script>
</body>
</html>
```

Output:

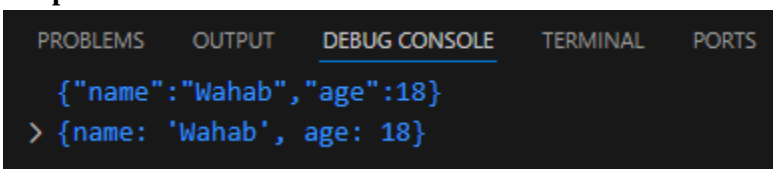


Task 5:

```
<html>
<head>
    <title>parse and stringify method in JSON</title>
</head>
<body>
    <script>
        let object = {
            name : "Wahab",
            age : 18
        };
        let a = JSON.stringify(object);
        console.log(a);
        console.log(JSON.parse(a));
    </script>
```

```
</body>
</html>
```

Output:

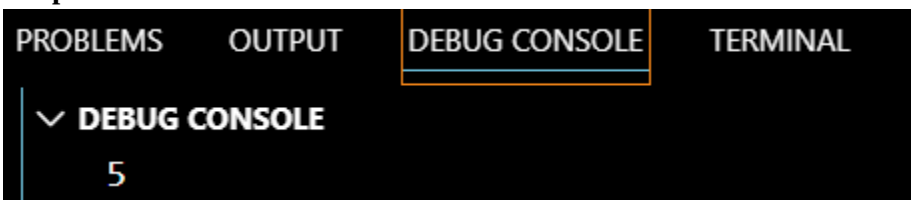


3.Closure:

Task 1:

```
<html>
<head>
   <title>Returning a function</title>
</head>
<body>
  <script>
    function fun(){
       let n = 5;
       function num(){
          console.log(n);
       }
       num();
    }
    fun();
  </script>
</body>
</html>
```

Output:

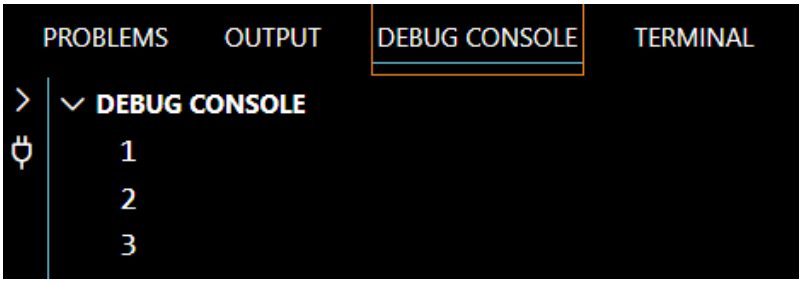


Task 2:

```
<html>
<head>
   <title>Closure</title>
</head>
<body>
  <script>
    function counter(){
       let count = 0;
       return {
          increment : function(){
             count++;
          },
          getCount : function(){
             return count;
          }
       };
```

```
    }
    let a = counter();
    a.increment();
    console.log(a.getCount());
    a.increment();
    console.log(a.getCount());
    a.increment();
    console.log(a.getCount());
  </script>
</body>
</html>
```

Output:

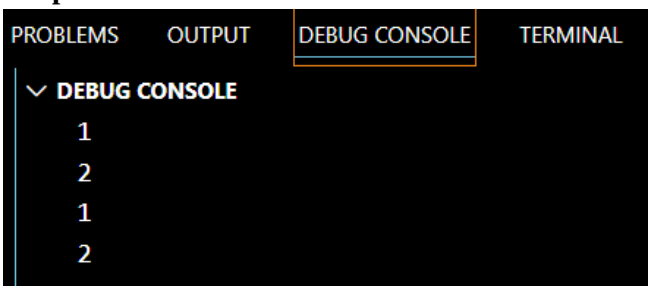


Task 3:

```
<html>
<head>
    <title>Closure</title>
</head>
<body>
  <script>
    function counter(){
        let count = 0;
        return {
            increment : function(){
                count++;
            },
            getCount : function(){
                return count;
            }
        };
    }
    let a = counter();
    let b = counter();
    a.increment();
    console.log(a.getCount());
    a.increment();
    console.log(a.getCount());
    b.increment();
    console.log(b.getCount());
    b.increment();
    console.log(b.getCount());
  </script>
</body>
```

```
</html>
```

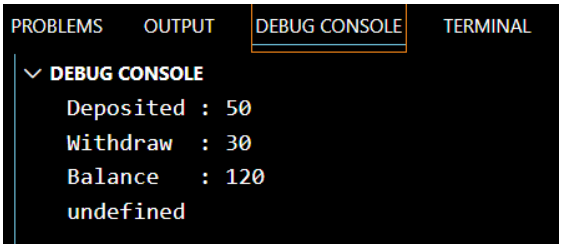
Output:



Task 4:

```
<html>
<head>
   <title>Private variable in Closure</title>
</head>
<body>
  <script>
    function createAccount(currBalance){
        let balance = currBalance;
        return {
            deposit : function(amount){
                balance += amount;
                console.log("Deposited : "+amount);
            },
            withdraw : function(amount){
                if(amount < balance){
                    balance -= amount;
                    console.log("Withdraw  : "+amount);
                }
            },
            getBalance : function(){
                console.log("Balance   : "+balance);
            }
        }
    }
    let myAccount = createAccount(100);
    myAccount.deposit(50);
    myAccount.withdraw(30);
    myAccount.getBalance();
    console.log(myAccount.balance);
  </script>
</body>
</html>
```

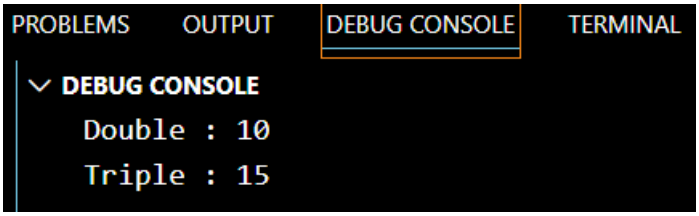
Output:



Task 5:

```
<html>
<head>
   <title>Private variable in Closure</title>
</head>
<body>
  <script>
    function multiplyFactor(factor){
       return function(number){
          return factor * number;
       }
    }
    let double = multiplyFactor(2);
    let triple = multiplyFactor(3);
    console.log("Double : "+double(5));
    console.log("Triple : "+triple(5));
  </script>
</body>
</html>
```

Output:



4. Promise, Promises chaining:

Task 1:

```
<html>
<head>
   <title>Promise in JS</title>
</head>
<body>
  <script>
    function greet(){
       return new Promise((resolve,reject) => {
          setTimeout(() => {
             const value = true;
             if(value){
                resolve("Success");
             }
             else{
                reject("Try Again");
             }
          },1000);
       });
    }
    greet().then(() => console.log("Good Morning!"));
  </script>
</body>
```

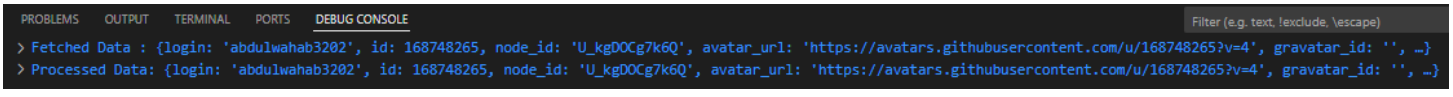
```
</html>
```

Output:

Task 2:

```
<html>
<head>
   <title>Promise in JS</title>
</head>
<body>
  <script>
    function fetchData() {
       return fetch("https://jsonplaceholder.typicode.com/posts")
          .then((response) => {
             if (!response.ok) {
                throw new Error("Failed to fetch data.");
             }
             return response.json(); // Parse JSON data
          });
    }
    function processData(data) {
       return new Promise((resolve) => {
          const processedData = data.slice(0, 5);
          resolve(processedData);
       });
    }
    fetchData()
       .then((data) => {
          console.log("Fetched Data (First 5):", data.slice(0, 5));
          return processData(data);
       })
       .then((processedData) => {
          console.log("Processed Data:", processedData);
       })
       });
  </script>
</body>
</html>
```

Output:

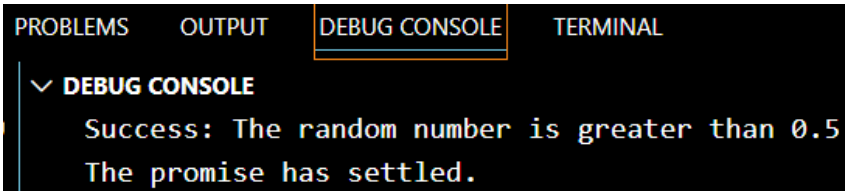


Task 3:

```
<html>
<head>
   <title>Promise in JS</title>
</head>
```

```
<body>
  <script>
    function randomPromise() {
    return new Promise((resolve, reject) => {
        const randomNumber = Math.random();
        if (randomNumber > 0.5) {
            resolve("Success: The random number is greater than 0.5");
        } else {
            reject("Error: The random number is less than or equal to 0.5");
        }
    });
}
    randomPromise()
    .then((message) => {
        console.log(message);
    })
    .catch((error) => {
        console.error(error);
    })
    .finally(() => {
        console.log("The promise has settled.");
    });
  </script>
</body>
</html>
```

Output:

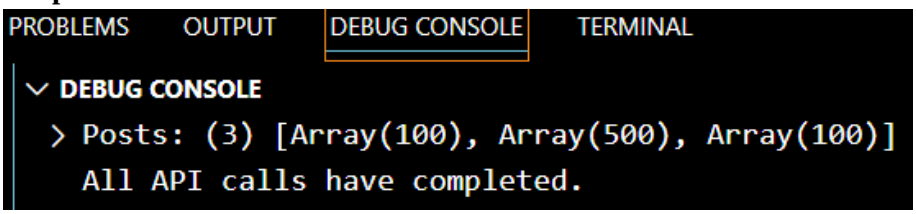


Task 4:

```
<html>
<head>
   <title>Promise in JS</title>
</head>
<body>
  <script>
function fetchData(url) {
    return fetch(url)
        .then((response) => {
            if (!response.ok) {
                throw new Error(`Failed to fetch from ${url}`);
            }
            return response.json(); // Parse and return JSON data
        });
}
const url1 = "https://jsonplaceholder.typicode.com/posts";
const url2 = "https://jsonplaceholder.typicode.com/comments";
```

```
const url3 = "https://jsonplaceholder.typicode.com/albums";
Promise.all([fetchData(url1), fetchData(url2), fetchData(url3)])
    .then((results) => {
        const posts = results;
        console.log("Posts:", posts);
    })
    .catch((error) => {
        console.error("Error:", error);
    })
    .finally(() => {
        console.log("All API calls have completed.");
    });
  </script>
</body>
</html>
```

Output:

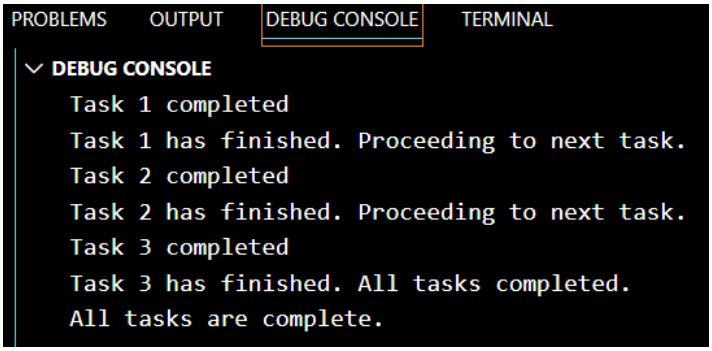


Task 5:

```
<html>
<head>
    <title>Promise in JS</title>
</head>
<body>
  <script>
function asyncTask(taskName, delay) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            console.log(`${taskName} completed`);
            resolve(taskName)
        }, delay);
    });
}
asyncTask("Task 1", 1000)
    .then((result) => {
        console.log(`${result} has finished. Proceeding to next task.`);
        return asyncTask("Task 2", 2000);
    })
    .then((result) => {
        console.log(`${result} has finished. Proceeding to next task.`);
        return asyncTask("Task 3", 1500);
    })
    .then((result) => {
        console.log(`${result} has finished. All tasks completed.`);
    })
    .catch((error) => {
```

```
      console.error("An error occurred:", error);
    })
    .finally(() => {
      console.log("All tasks are complete.");
    });
  </script>
</body>
</html>
```

Output:



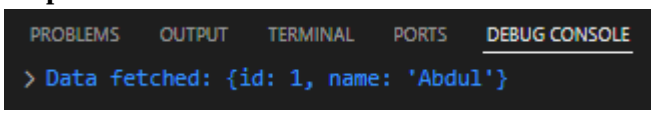
5.Async/await:

Task 1:

```
<html>
<head>
   <title>async/await</title>
</head>
<body>
   <script>
      async function fetchData() {
         return new Promise((resolve, reject) => {
         setTimeout(() => {
           const data = { id: 1, name: 'Abdul' };
           if (data) {
             resolve(data);
           } else {
             reject(new Error('No data found'));
           }
         }, 1000);
         });
   }
   async function main() {
     try {
      const result = await fetchData();
      console.log('Data fetched:', result);
     } catch (error) {
      console.error('Error fetching data:', error);
     }
   }
   main();
   </script>
</body>
```

```
</html>
```

Output:

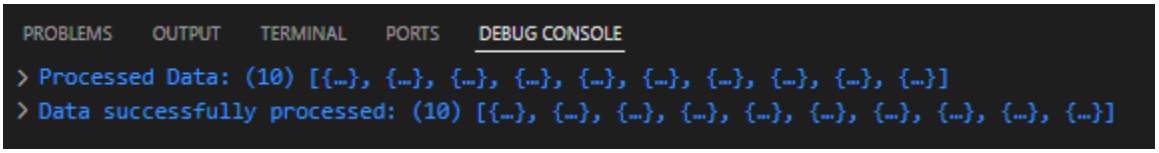


Task 2:

```
<html>
<head>
   <title>async/await</title>
</head>
<body>
   <script>
      async function fetchAndProcessData(apiUrl) {
  try {
   const response = await fetch(apiUrl);
   if (!response.ok) {
    throw new Error(`HTTP error! Status: ${response.status}`);
   }
   const data = await response.json();
   const processedData = data.map(item => ({
    id: item.id,
    name: item.name.toUpperCase(),
   }));
   console.log('Processed Data:', processedData);
   return processedData;

  } catch (error) {
   console.error('Error fetching or processing data:', error);
   throw error;
  }
}
const apiUrl = 'https://jsonplaceholder.typicode.com/users';
fetchAndProcessData(apiUrl)
  .then(data => console.log('Data successfully processed:', data))
  .catch(error => console.error('Error:', error));
   </script>
</body></html>
```

Output:



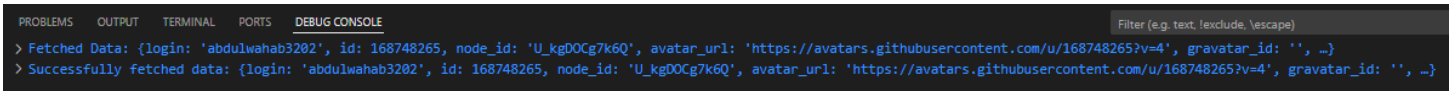
Task 3:

```
<html>
<head>
   <title>Promise in JS</title>
</head>
<body>
  <script>
   async function fetchDataWithErrorHandling(apiUrl) {
```

```
  try {
    const response = await fetch(apiUrl);
    if (!response.ok) {
      throw new Error(`HTTP Error: ${response.status} - ${response.statusText}`);
    }
    const data = await response.json();
    console.log('Fetched Data:', data);
    return data;
  } catch (error) {
    console.error('Error occurred:', error.message);
    return { error: error.message };
  }
}
const apiUrl = 'https://api.github.com/users/abdulwahab3202';

fetchDataWithErrorHandling(apiUrl)
  .then(result => {
    if (result.error) {
      console.error('Failed to fetch data:', result.error);
    } else {
      console.log('Successfully fetched data:', result);
    }
  })
  .catch(err => console.error('Unhandled error:', err));
  </script>
</body>
</html>
```

Output:



Task 4:

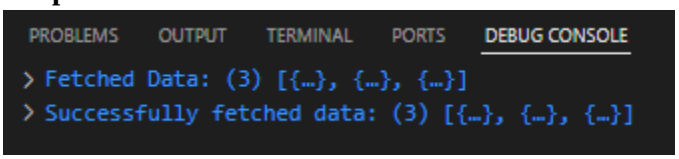
```
<html>
<head>
   <title>async/await in JS</title>
</head>
<body>
  <script>
    async function fetchMultipleApis(apiUrls) {
  try {
    const fetchPromises = apiUrls.map(url => fetch(url));
    const responses = await Promise.all(fetchPromises);
    responses.forEach(response => {
      if (!response.ok) {
        throw new Error(`HTTP error for ${response.url}: ${response.status}`);
      }
    });
    const data = await Promise.all(responses.map(response => response.json()));

    console.log('Fetched Data:', data);
```

```
    return data;

  } catch (error) {
    console.error('Error occurred:', error.message);
    throw error;
  }
}
const apiUrls = [
  'https://api.github.com/users/abdulwahab3202',
  'https://api.github.com/users/abdulwahab3202',
  'https://api.github.com/users/abdulwahab3202',
];
fetchMultipleApis(apiUrls)
  .then(data => console.log('Successfully fetched data:', data))
  .catch(error => console.error('Error during API fetch:', error));
  </script>
</body>
</html>
```

Output:

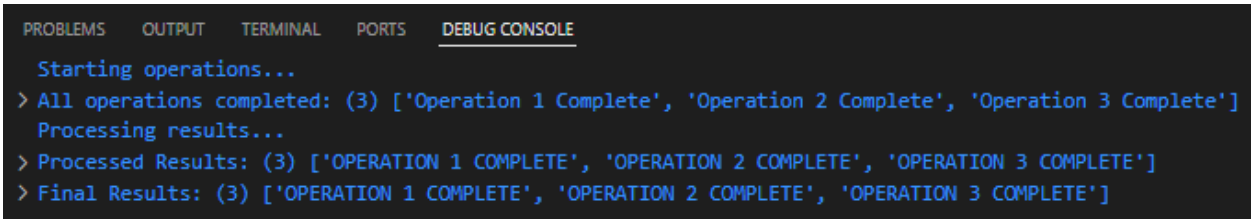


Task 5:

```
<html>
<head>
    <title>async/await in JS</title>
</head>
<body>
  <script>
    async function performMultipleOperations() {
  try {
    const operation1 = new Promise(resolve => setTimeout(() => resolve('Operation 1 Complete'), 2000));
    const operation2 = new Promise(resolve => setTimeout(() => resolve('Operation 2 Complete'), 1000));
    const operation3 = new Promise(resolve => setTimeout(() => resolve('Operation 3 Complete'), 1500));
    console.log('Starting operations...');
    const results = await Promise.all([operation1, operation2, operation3]);
    console.log('All operations completed:', results);
    console.log('Processing results...');
    const processedResults = results.map(result => result.toUpperCase());
    console.log('Processed Results:', processedResults);
    return processedResults;
  } catch (error) {
    console.error('Error occurred during operations:', error.message);
    throw error;
  }
}
performMultipleOperations()
  .then(processedResults => console.log('Final Results:', processedResults))
```

```
  .catch(error => console.error('Error during execution:', error));
  </script>
</body>
</html>
```

Output:

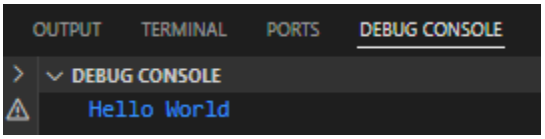


6. Module Introduction, Export and Import:

Task 1:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Promise Chain Example</title>
</head>
<body>
 <script>
 export let name = ()=>{console.log("Hell!js..")};
 export const val = 2015;
 export class User {
 constructor(name) {
 this.name = name;
 }
 sayhi = ()=>{console.log("Hai");}
 }
 </script>
</body>
</html>
```

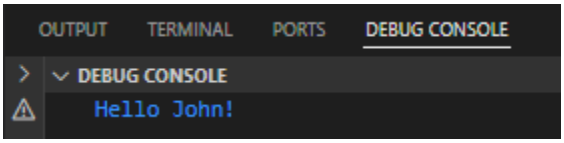
Output:



Task 2:

```
<html>
<head>
 <title>Promise Chain Example</title>
</head>
<body>
 <script type="module">
 import { sayHi } from './task.js';
 sayHi('John');
 </script>
</body>
</html>
```

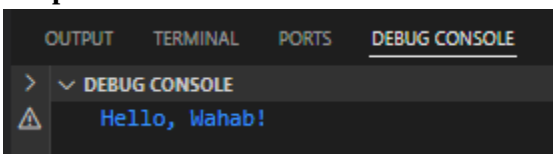
Output:



Task 3:

```
<html>
<head>
 <title>Promise Chain Example</title>
</head>
<body>
 <script type="module">
 export function sayHi(user) {
 console.log(`Hello, ${user}!`);
 }
 export function sayBye(user) {
 console.log(`Bye, ${user}!`);
 }
 sayHi("Wahab")
 </script>
</body>
</html>
```

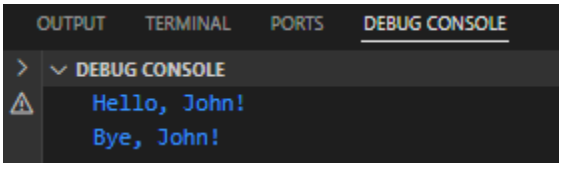
Output:



Task 4:

```
<html>
<head>
 <title>Promise Chain Example</title>
</head>
<body>
 <script type="module">
 import { sayHi,sayBye } from './task.js';
 sayHi('John');
 sayBye('john')
 </script>
</body>
</html>
```

Output:

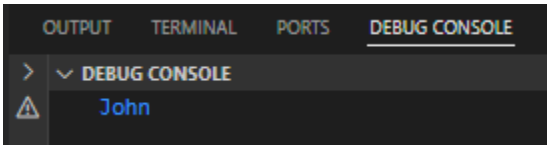


Task 5:

```
<html>
<head>
 <title>Promise Chain Example</title>
```

```
</head>
<body>
 <script type="module">
 import User from './task.js';
 new User('John');
 </script>
</body>
</html>
```

Output:



7. Browser: DOM Basics:

Task 1:

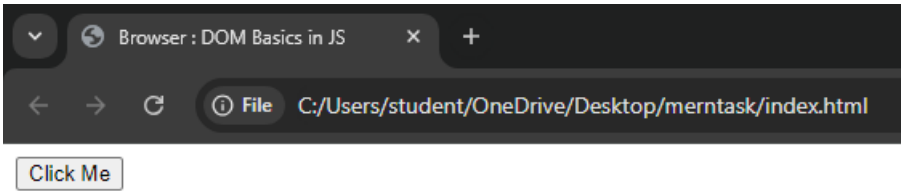
```
<html>
<head>
   <title>Browser : DOM Basics in JS</title>
</head>
<body>
   <h1 id="demo">Hello User</h1>
   <script>
      var demo = document.getElementById('demo');
      demo.textContent = 'Welcome to Coimbatore!';
   </script>
</body>
</html>
```

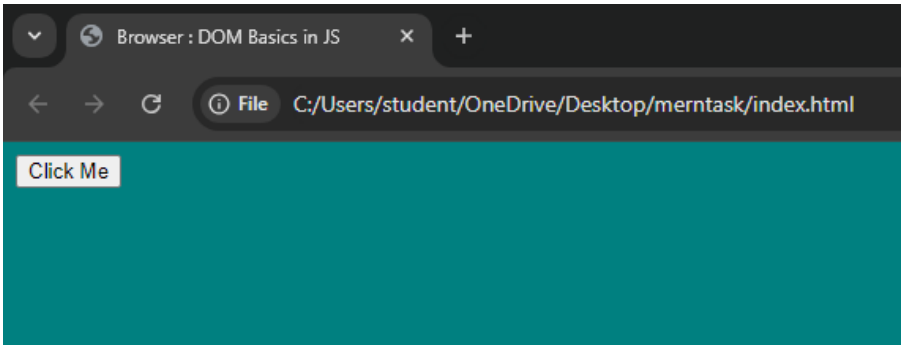
Output:

Task 2:

```
<html>
<head>
   <title>Browser : DOM Basics in JS</title>
</head>
<body>
   <button id="btn">Click Me</button>
   <script>
      var button = document.getElementById('btn');
      button.addEventListener('click',() => {
         document.body.style.backgroundColor = 'teal';
      })
   </script>
</body>
</html>
```

Output:

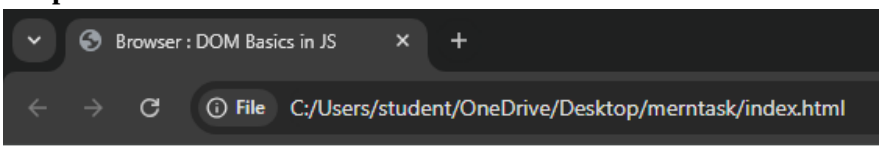




Task 3:

```
<html>
<head>
   <title>Browser : DOM Basics in JS</title>
</head>
<body>
   <h1>Hello World</h1>
   <script>
      var newElement = document.createElement('h2');
      newElement.textContent = 'Appended the child element';
      document.body.appendChild(newElement);
   </script>
</body>
</html>
```

Output:

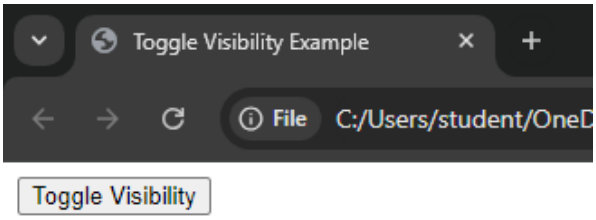


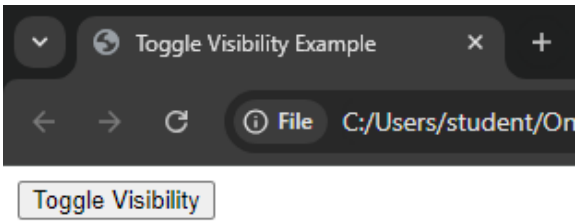
Task 4:

```
<html>
<head>
   <title>Toggle Visibility Example</title>
</head>
<body>
   <button onclick="toggleVisibility()">Toggle Visibility</button>
   <div id="myElement" style="display: none; margin-top: 20px; padding: 20px; border: 2px solid black;margin-right:
1300px;">
```

```
    This is a toggleable element.
  </div>
  <script>
    function toggleVisibility() {
      const element = document.getElementById('myElement');
      if (element.style.display === 'none' || element.style.display === '') {
        element.style.display = 'block';
      } else {
        element.style.display = 'none';
      }
    }
  </script>
</body>
</html>
```

Output:





This is a toggleable element.

Task 5:

```
<html>
<head>
  <title>Modify Element Attributes</title>
</head>
<body>
  <h1 id="myHeading" class="header" style="color: blue;" data-custom="someValue">Hello World!</h1>
  <button onclick="modifyAttributes()">Modify Attributes</button>
  <script>
    function modifyAttributes() {
      const heading = document.getElementById('myHeading');
      const currentClass = heading.getAttribute('class');
      console.log('Current class:', currentClass);
      heading.setAttribute('style', 'color: red; font-size: 36px;');
      heading.setAttribute('data-custom', 'newValue');
      heading.setAttribute('id', 'newHeading');
      heading.removeAttribute('class');
```

```
            console.log('Updated Element:', heading);
        }
    </script>
</body>
</html>
```

Output: