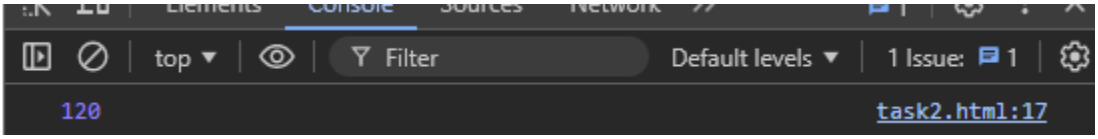
Task 1:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
       let factorial = function(num)
       {
        if(num==0 || num==1){
            return 1;
        }
        else{
            return num*factorial(num-1);
        }
       }
       console.log(factorial(5));
        </script>
    </body>
</html>
```

Output:



Task 2:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
       let  fibonacci  = function(num){
        if(num==0){
            return 0;
        }
        if(num==1){
            return 1;
        }
```

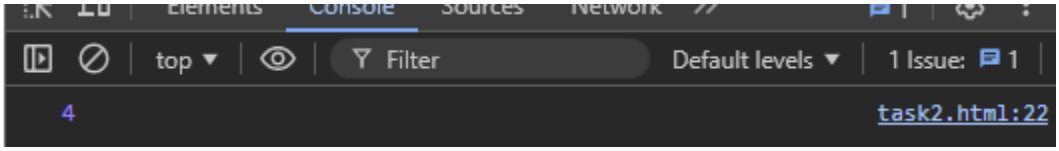
```
        else{
            return  fibonacci(num-1) + fibonacci(num-2);
        }
       }
       console.log( fibonacci (5));
        </script>
    </body>
</html>
```

Output:

Task 3:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
      let countways = function(num){
        if(num==0){
            return 1;
        }
        if(num==1){
            return 1;
        }
        if(num==2){
            return 2;
        }
        else{
            return countways(num-1)+countways(num-2)+countways(num-3);
        }
    }
    console.log(countways(3));
        </script>
    </body>
</html>
```

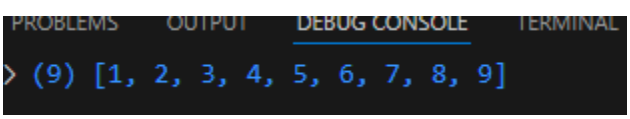
Output:



Task 4:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
        function flattenArray(arr) {
    let result = [];
    for (let i = 0; i < arr.length; i++) {
        if (Array.isArray(arr[i])) {
            result = result.concat(flattenArray(arr[i]));
        } else {
            result.push(arr[i]);
        }
    }
    return result;
}
let nestedArray = [1, [2, 3], [4, [5, 6]], 7, [8, [9]]];
let flattenedArray = flattenArray(nestedArray);

console.log(flattenedArray);


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

Output:



Task 5:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
        function towerOfHanoi(n, from, to, aux) {
    if (n === 1) {
        console.log(`Move disk 1 from ${from} to ${to}`);
        return;
    }
    towerOfHanoi(n - 1, from, aux, to);
    console.log(`Move disk ${n} from ${from} to ${to}`);
    towerOfHanoi(n - 1, aux, to, from);
}
let n = 3;
towerOfHanoi(n, 'A', 'C', 'B');
        </script>
    </body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
  Move disk 1 from A to C
  Move disk 2 from A to B
  Move disk 1 from C to B
  Move disk 3 from A to C
  Move disk 1 from B to A
  Move disk 2 from B to C
  Move disk 1 from A to C
```

Task 6:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
```

```
        <script>
      function sum() {
    let total = 0;
    for (let i = 0; i < arguments.length; i++) {
        total += arguments[i];
    }
    return total;
}
console.log(sum(1, 2, 3));
console.log(sum(2,2));
console.log(sum(5, 5));
</script>
</body>
</html>
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

6
4
10
```

Task 7:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
      function sum(...numbers) {
    let total = 0;
    for (let num of numbers) {
        total += num;
    }
    return total;
}
let arr = [1, 2, 3, 4, 5];
console.log(sum(...arr));
</script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
  15
```

Task 8:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
     function deepClone(obj) {
  return JSON.parse(JSON.stringify(obj));
}

const original = {
  name: 'Bob',
  age: 30,
  address: {
    street: '123 Main St',
    city: 'Smalltown'
  },
  hobbies: ['reading', 'coding']
};

const cloned = deepClone(original);
cloned.address.city = 'Bigcity';
cloned.hobbies.push('photography');
console.log('Original object:', original);
console.log('Cloned object:', cloned);

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

Output:

```
Original object:
▼ Object ℹ
  ▶ address: {street: '123 Main St', city: 'Smalltown'}
    age: 30
  ▶ hobbies: (2) ['reading', 'coding']
    name: "Bob"
  ▶ [[Prototype]]: Object
Cloned object:
▼ Object ℹ
  ▶ address: {street: '123 Main St', city: 'Bigcity'}
    age: 30
  ▶ hobbies: (3) ['reading', 'coding', 'photography']
    name: "Bob"
  ▶ [[Prototype]]: Object
```

Task 9:

```html
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
     function mergeObjects(obj1, obj2) {
  return { ...obj1, ...obj2 };
}
const object1 = { name: 'Alice', age: 25 };
const object2 = { age: 30, city: 'New York' };
const mergedObject = mergeObjects(object1, object2);
console.log(mergedObject);
</script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
> {name: 'Alice', age: 30, city: 'New York'}
```

Task 10:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
const person = {
  name: 'Alice',
  age: 25,
  hobbies: ['reading',  'traveling']
};
const jsonString = JSON.stringify(person);
console.log('Serialized string:', jsonString);
const parsedObject = JSON.parse(jsonString);
console.log('Parsed  object:', parsedObject);

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

Output:

```
Serialized string: {"name":"Alice","age":25,"hobbies":
["reading","traveling"]}
Parsed  object: ▼ Object ℹ
                    age: 25
                  ▶ hobbies: (2) ['reading', 'traveling']
                    name: "Alice"
                  ▶ [[Prototype]]: Object
```

Task 11:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
function createCounter() {
  let count = 0;
   return function() {
    count++;
```

```
    return count;
  };
}
const counter = createCounter();
console.log(counter());
console.log(counter());
console.log(counter());
</script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  1
  2
  3
```

Task 12:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
function createCounter() {
  let count = 0;
  return function() {
    count++;
    console.log(count);
  };
}
const counter = createCounter();
counter();
counter();
counter();
</script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE

  1
  2
  3
```

Task 13:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
function createCounter() {
  let count = 0;
  return function() {
    count++;
    console.log(count);
  };
}
const counter1 = createCounter();
const counter2 = createCounter();
counter1();
counter1();
counter2();
counter2();
counter1();
counter2();
</script>
</body>
</html>
```

Output:

```
1                                   task2.html:12
2                                   task2.html:12
1                                   task2.html:12
2                                   task2.html:12
3                                   task2.html:12
3                                   task2.html:12
```

Task 14:

```
<html>
    <head>
        <meta charset ="UTF-8">
        <meta name:"viewport" content="width+device_width,initial-scale=1.0">
    </head>
    <body>
        <script>
function createCounter() {
  let count = 0;
return {
    increment: function() {
      count++;
      console.log(count);
    }
  };
}
const counter = createCounter();
counter.increment();
counter.increment();
counter.increment();
</script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    T

1
2
3
```

Task 15:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function Createcounter(start_value){
    let count=start_value;
    return{
        increment: function(){
            count+=1;},
        getCount: function(){
            return count;} };}
            const counter1=Createcounter(100);
            const  counter2=Createcounter(200)
    counter1.increment();
    counter1.increment();
    counter1.increment();
   counter2.increment();
    counter2.increment();
    counter2.increment();
console.log(counter1.getCount());
console.log(counter2.getCount());
</script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORT
  103
  203
```

Task 16:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function greetAfterDelay(seconds) {
```

```
  return new Promise(resolve => {
    setTimeout(() => {
      resolve('Hello after ' + seconds + ' seconds!');
    }, seconds * 1000);
  });
}
greetAfterDelay(3).then(message => {
  console.log(message);
});
</script>
</body>
</html>
```

Output:

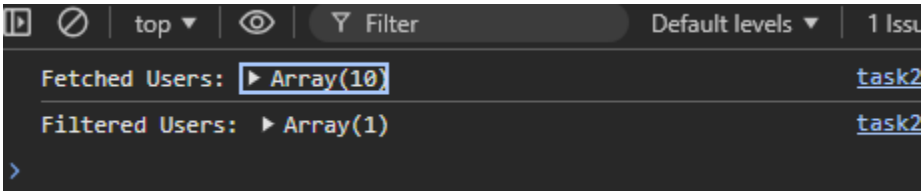
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
  Hello after 3 seconds!
```

Task 17:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function fetchData() {
  return fetch('https://jsonplaceholder.typicode.com/users')
    .then(response => response.json());
}
function processData(users) {
  return users.filter(user => user.name === 'Leanne Graham');
}
fetchData()
  .then(users => {
    console.log('Fetched Users:', users);
    return processData(users);
  })
  .then(filteredUsers => {
    console.log('Filtered Users:', filteredUsers);
  })
  .catch(error => {
    console.error('Error:', error);
  });
```

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

Output:



Task 18:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function randomPromise() {
  return new Promise((resolve, reject) => {
    const randomNumber = Math.random();
    if (randomNumber > 0.5) {
      resolve('Success! The number is greater than 0.5');
    } else {
      reject('Failure! The number is 0.5 or less');
    }
  });
}
randomPromise()
  .then(result => console.log(result))
  .catch(error => console.log(error));
</script>
</body>
</html>
```

Output:

```
top ▾   Filter
Failure! The number is 0.5 or less
```

Task 19:

```html
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function fetchPosts() {
  return fetch('https://jsonplaceholder.typicode.com/posts')
    .then(response => response.json());
}
function fetchUsers() {
  return fetch('https://jsonplaceholder.typicode.com/users')
    .then(response => response.json());
}
Promise.all([fetchPosts(), fetchUsers()])
  .then(results => {
    const [posts, users] = results;
    console.log('Posts:', posts);
    console.log('Users:', users);
  })
  .catch(error => {
    console.log('Error:', error);
  });

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

Output:

```
top ▾   Filter   Default levels ▾   1 Issue:
Posts:  ▸ Array(100)                    task2.html:18
Users:  ▸ Array(10)                     task2.html:19
```

Task 20:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function fetchUserName() {
  return new Promise((resolve) => {
    setTimeout(() => {
      console.log('User name fetched');
      resolve('Alice');
    }, 1000);
  });
}
function fetchUserAge() {
  return new Promise((resolve) => {
    setTimeout(() => {
      console.log('User age fetched');
      resolve(25);
    }, 1000);
  });
}
function fetchUserCity() {
  return new Promise((resolve) => {
    setTimeout(() => {
      console.log('User city fetched');
      resolve('New York');
    }, 1000);
  });
}
fetchUserName()
  .then((name) => {
    console.log('Name:', name);
    return fetchUserAge();
  })
  .then((age) => {
    console.log('Age:', age);
    return fetchUserCity();
  })
  .then((city) => {
    console.log('City:', city);
  })
  .catch((error) => {
```

```
    console.log('Error:', error);
  });

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

Output:

```
User name fetched                                   task2.html:10
Name: Alice                                         task2.html:33
User age fetched                                    task2.html:18
Age: 25                                             task2.html:37
User city fetched                                   task2.html:26
City: New York                                      task2.html:41
```

Task 21:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function fetchUserName() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('Alice');
    }, 1000);
  });
}
function fetchUserAge() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve(25);
    }, 1000);
  });
}
function fetchUserCity() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('New York');
    }, 1000);
  });
}
```

```
async function getUserDetails() {
  const name = await fetchUserName();
  console.log('Name:', name);
  const age = await fetchUserAge();
  console.log('Age:', age);
  const city = await fetchUserCity();
  console.log('City:', city);
}
getUserDetails();
</script>
</body>
</html>
```

Output:

```
Name: Alice
Age: 25
City: New York
```

Task 22:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
async function fetchAndProcessData() {
  try {
    const response = await fetch('https://jsonplaceholder.typicode.com/users');
    if (!response.ok) {
      throw new Error('Network response was not ok');
    }
    const data = await response.json();
    data.forEach(user => {
      console.log(`User: ${user.name}, Email: ${user.email}`);
    });

  } catch (error) {

    console.error('There was an error fetching the data:', error);
  }
```

```
}
fetchAndProcessData();

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

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  User: Leanne Graham, Email: Sincere@april.biz
  User: Ervin Howell, Email: Shanna@melissa.tv
  User: Clementine Bauch, Email: Nathan@yesenia.net
  User: Patricia Lebsack, Email: Julianne.OConner@kory.org
  User: Chelsey Dietrich, Email: Lucio_Hettinger@annie.ca
  User: Mrs. Dennis Schulist, Email: Karley_Dach@jasper.info
  User: Kurtis Weissnat, Email: Telly.Hoeger@billy.biz
  User: Nicholas Runolfsdottir V, Email: Sherwood@rosamond.me
  User: Glenna Reichert, Email: Chaim_McDermott@dana.io
  User: Clementina DuBuque, Email: Rey.Padberg@karina.biz
```

Task 23:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
async function fetchData() {
  try {
    const response = await fetch('https://jsonplaceholder.typicode.com/users');
    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }
    const data = await response.json();
    data.forEach(user => {
      console.log(`User: ${user.name}, Email: ${user.email}`);
    });

  } catch (error) {
    console.error('There was an error fetching the data:', error.message);
  }
```

```
}
fetchData();
</script>
</body>
</html>
```

Output:

```
User: Leanne Graham, Email: Sincere@april.biz                task2.html:15
User: Ervin Howell, Email: Shanna@melissa.tv                 task2.html:15
User: Clementine Bauch, Email: Nathan@yesenia.net            task2.html:15
User: Patricia Lebsack, Email: Julianne.OConner@kory.org     task2.html:15
User: Chelsey Dietrich, Email: Lucio_Hettinger@annie.ca      task2.html:15
User: Mrs. Dennis Schulist, Email: Karley_Dach@jasper.info   task2.html:15
User: Kurtis Weissnat, Email: Telly.Hoeger@billy.biz         task2.html:15
User: Nicholas Runolfsdottir V, Email: Sherwood@rosamond.me  task2.html:15
User: Glenna Reichert, Email: Chaim_McDermott@dana.io        task2.html:15
User: Clementina DuBuque, Email: Rey.Padberg@karina.biz      task2.html:15
```

Task 24:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function fetchUser(id) {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve(`User ${id}`);
    }, 1000);
  });
}
function fetchPost(id) {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve(`Post ${id}`);
    }, 1500);
  });
}
async function fetchData() {
```

```
  try {
    const [user, post] = await Promise.all([
      fetchUser(1),
      fetchPost(1)
    ]);
    console.log(user);
    console.log(post);
  } catch (error) {
    console.error('Error fetching data:', error);
  }
}
fetchData();
</script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
  User 1
  Post 1
```

Task 25:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function asyncTask(name, delay) {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve(`${name} completed after ${delay} ms`);
    }, delay);
  });
}
async function waitForAllTasks() {
  try {
    const results = await Promise.all([
      asyncTask('Task 1', 2000),
      asyncTask('Task 2', 1000),
      asyncTask('Task 3', 1500)
    ]);
```

```
    console.log('All tasks completed:');
    results.forEach((result) => console.log(result));

  } catch (error) {
    console.error('An error occurred:', error);
  }
}
waitForAllTasks();
</script>
</body>
</html>
```

Output:

```
All tasks completed:                                  task2.html:21
Task 1 completed after 2000 ms                        task2.html:22
Task 2 completed after 1000 ms                        task2.html:22
Task 3 completed after 1500 ms                        task2.html:22
```

Task 26:

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>ES Module Example</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
  </style>
</head>
<body>
  <h1>Using JavaScript Modules in the Browser</h1>
  <div id="greeting"></div>
  <div id="introduction"></div>
  <div id="color"></div>
  <script type="module">
    import { greet, Person, favoriteColor } from './myModule.js'
    document.getElementById('greeting').textContent = greet('Alice');
```

```
    const person1 = new Person('Bob', 30);
    document.getElementById('introduction').textContent = person1.introduce();
    document.getElementById('color').textContent = `Favorite color:
${favoriteColor}`;
  </script>

</body>
</html>
```

myModule.js

```
export function greet(name) {
  return `Hello, ${name}!`;
}
export class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
 introduce() {
    return `Hi, I'm ${this.name} and I'm ${this.age} years old.`;
  }
}
export const favoriteColor = 'blue';
```

output:

Using JavaScript Modules in the Browser

Hello, Alice!
Hi, I'm Bob and I'm 30 years old.
Favorite color: blue

Task 27:

Index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JavaScript Modules Example</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    #greeting, #introduction, #color {
      margin: 10px 0;
    }
  </style>
</head>
<body>
  <h1>Using JavaScript Modules with Direct Import</h1>
  <div id="greeting"></div>
  <div id="introduction"></div>
  <div id="color"></div>
  <script type="module">
    import { greet, Person, favoriteColor } from './myModule.js';
    const greetingElement = document.getElementById('greeting');
    greetingElement.textContent = greet('Alice');
    const person1 = new Person('Bob', 30);
    const introductionElement = document.getElementById('introduction');
    introductionElement.textContent = person1.introduce();
    const colorElement = document.getElementById('color');
    colorElement.textContent = `Favorite color: ${favoriteColor}`;
  </script>
</body>
</html>
```

myModule.js

```javascript
export function greet(name) {
  return `Hello, ${name}!`;
}
export class Person {
```

```
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
  introduce() {
    return `Hi, I'm ${this.name} and I'm ${this.age} years old.`;
  }
}
export const favoriteColor = 'blue';
```

output:

Using JavaScript Modules with Direct Import

Hello, Alice!

Hi, I'm Bob and I'm 30 years old.

Favorite color: blue

Task 28:

myModule.js

```
export function greet(name) {
  return `Hello, ${name}!`;
}
export function sum(a, b) {
  return a + b;
}
export function getCurrentYear() {
  return new Date().getFullYear();
}
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JavaScript Modules Example</title>
  <style>
    body {
```

```
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    #greeting, #sum, #year {
      margin: 10px 0;
    }
  </style>
</head>
<body>
  <h1>Using Named Exports in JavaScript Modules</h1>
  <div id="greeting"></div>
  <div id="sum"></div>
  <div id="year"></div>
  <script type="module">
    import { greet, sum, getCurrentYear } from './myModule.js';
    const greetingElement = document.getElementById('greeting');
    greetingElement.textContent = greet('Alice');
    const sumElement = document.getElementById('sum');
    sumElement.textContent = `The sum of 5 and 7 is: ${sum(5, 7)}`;
    const yearElement = document.getElementById('year');
    yearElement.textContent = `The current year is: ${getCurrentYear()}`;
  </script>
</body>
</html>
```

output:

Using Named Exports in JavaScript Modules

Hello, Alice!

The sum of 5 and 7 is: 12

The current year is: 2024

Task 29:

myModule.js

```
export function greet(name) {
  return `Hello, ${name}!`;
}
export function sum(a, b) {
  return a + b;
}
```

```
export function subtract(a, b) {
  return a - b;
}
export function getCurrentYear() {
  return new Date().getFullYear();
}
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Using Named Imports in JavaScript Modules</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    #greeting, #sum, #year {
      margin: 10px 0;
    }
  </style>
</head>
<body>
  <h1>Using Named Imports in JavaScript Modules</h1>
  <div id="greeting"></div>
  <div id="sum"></div>
  <div id="year"></div>
  <script type="module">
    import { greet, sum } from './myModule.js';
    const greetingElement = document.getElementById('greeting');
    greetingElement.textContent = greet('Alice');
    const sumElement = document.getElementById('sum');
    sumElement.textContent = `The sum of 5 and 7 is: ${sum(5, 7)}`;
    const yearElement = document.getElementById('year');
    yearElement.textContent = `The current year is: ${new
Date().getFullYear()}`;
  </script>
</body>
</html>
```

output:

Using Named Imports in JavaScript Modules

Hello, Alice!

The sum of 5 and 7 is: 12

The current year is: 2024

Task 30:

myModule.js

```
function greet(name) {
  return `Hello, ${name}! Welcome to using default exports.`;
}
export default greet;
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Using Default Export and Import</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    #greeting {
      margin: 10px 0;
    }
  </style>
</head>
<body>
  <h1>Using Default Export and Import in JavaScript Modules</h1>
  <div id="greeting"></div>
  <script type="module">
    import greet from './myModule.js';
    const greetingElement = document.getElementById('greeting');
    greetingElement.textContent = greet('Alice');
```

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

Output:

Using Default Export and Import in JavaScript Modules

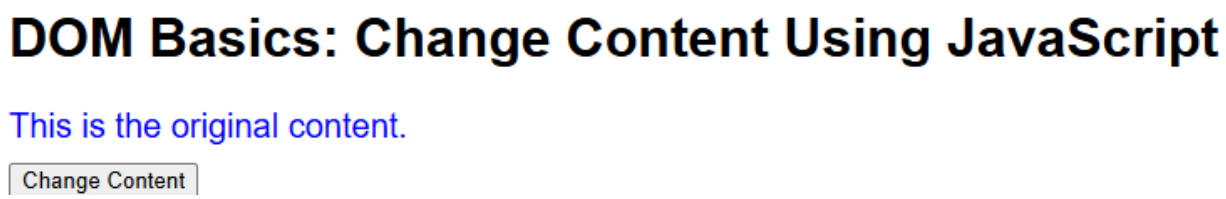
Hello, Alice! Welcome to using default exports.

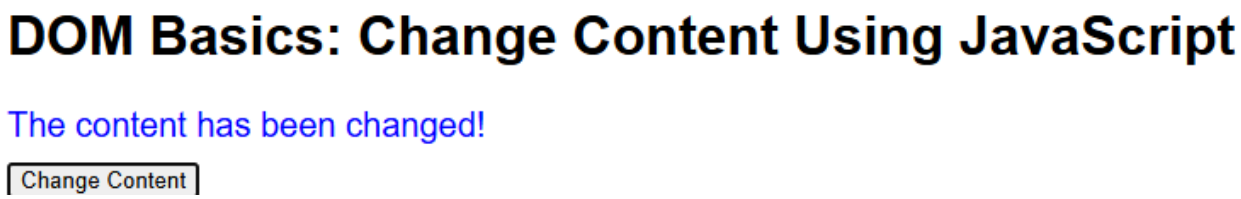
Task 31:

```
 <!DOCTYPE html>

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>DOM Basics: Change Content</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    #message {
      font-size: 20px;
      color: blue;
      margin: 10px 0;
    }
  </style>
</head>
<body>

  <h1>DOM Basics: Change Content Using JavaScript</h1>
  <p id="message">This is the original content.</p>
  <button onclick="changeContent()">Change Content</button>
  <script>
    function changeContent() {
      var element = document.getElementById('message');
      element.textContent = 'The content has been changed!';
    }
  </script>
```

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

Output:





Task 32:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Button Event Listener Example</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    #message {
      font-size: 20px;
      color: green;
      margin: 10px 0;
    }
  </style>
</head>
<body>

  <h1>Attach Event Listener to a Button</h1>
```

```
  <p id="message">Click the button to change this text.</p>
  <button id="changeMessageButton">Change Message</button>

  <script>
    const button = document.getElementById('changeMessageButton');
    const messageElement = document.getElementById('message');
    button.addEventListener('click', function() {
      messageElement.textContent = 'The content has been changed after clicking
the button!';
    });
  </script>

</body>
</html>
```

Output:

Attach Event Listener to a Button

Click the button to change this text.

Change Message

Attach Event Listener to a Button

The content has been changed after clicking the button!

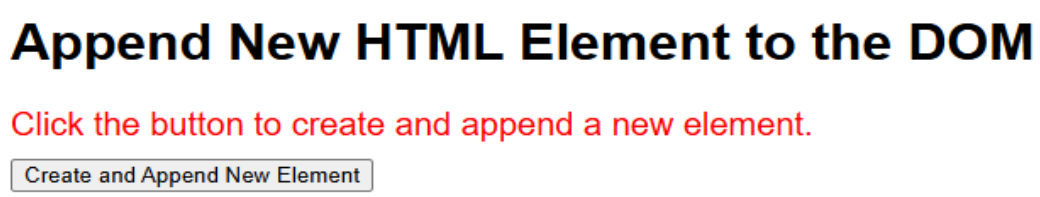
Change Message

Task 33:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Create and Append a New HTML Element</title>
  <style>
    body {
      font-family: Arial, sans-serif;
```

```
      margin: 20px;
    }
    #message {
      font-size: 20px;
      color: red;
      margin: 10px 0;
    }
    #newElementContainer {
      margin-top: 20px;
    }
  </style>
</head>
<body>
  <h1>Append New HTML Element to the DOM</h1>
  <div id="message">Click the button to create and append a new element.</div>
  <button id="createElementButton">Create and Append New Element</button>
  <div id="newElementContainer"></div>
  <script>
    const button = document.getElementById('createElementButton');
    const container = document.getElementById('newElementContainer');
    button.addEventListener('click', function() {
      const newElement = document.createElement('p');
      newElement.textContent = 'This is a newly created element appended to the
DOM!';
      newElement.style.color = 'green';
      newElement.style.fontSize = '18px';
      container.appendChild(newElement);
    });
  </script>
</body>
</html>
```

Output:



Append New HTML Element to the DOM

Click the button to create and append a new element.

Create and Append New Element

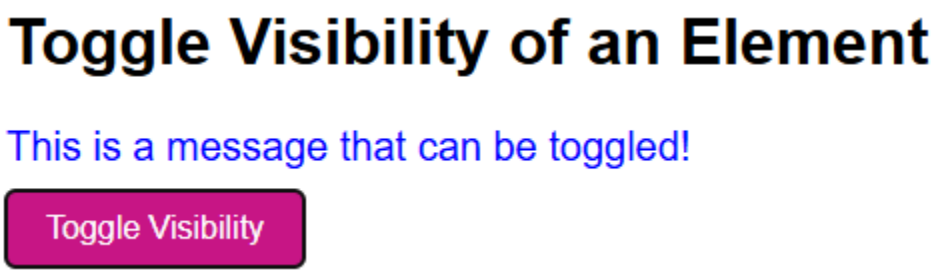
This is a newly created element appended to the DOM!

Task 34:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Toggle Element Visibility</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    #toggleMessage {
      font-size: 20px;
      color: blue;
      margin: 10px 0;
      display: block;
    }
    #toggleButton {
      padding: 10px 20px;
      font-size: 16px;
      cursor: pointer;
      background-color:mediumvioletred;
      color: white;
      border: none;
      border-radius: 5px;
    }
    #toggleButton:hover {
      background-color: pink;
    }
  </style>
</head>
<body>

  <h1>Toggle Visibility of an Element</h1>
  <div id="toggleMessage">This is a message that can be toggled!</div>
  <button id="toggleButton">Toggle Visibility</button>
```

```
  <script>
    const toggleButton = document.getElementById('toggleButton');
    const toggleMessage = document.getElementById('toggleMessage');
    function toggleVisibility() {
      if (toggleMessage.style.display === 'none') {
        toggleMessage.style.display = 'block';
      } else {
        toggleMessage.style.display = 'none';
      }
    }
    toggleButton.addEventListener('click', toggleVisibility);
  </script>

</body>
</html>
```

Output:

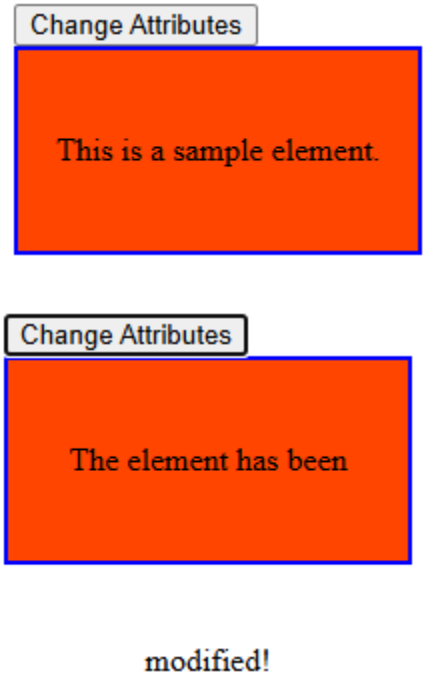


Task 35:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```html
<title>Modify Element Attributes</title>
<style>
#myElement {
width: 200px;
height: 100px;
background-color: orangered;
text-align: center;
line-height: 100px;
border: 2px solid blue;
}
</style>
</head>
<body>
<button onclick="changeAttributes()">Change Attributes</button>
<div id="myElement" class="box" title="Original Title">
This is a sample element.
</div>
<script>
function changeAttributes() {
var element = document.getElementById("myElement");
var currentClass = element.getAttribute("class");
var currentTitle = element.getAttribute("title");
console.log("Current class:", currentClass);

console.log("Current title:", currentTitle);
element.setAttribute("class", "modified-box");
element.setAttribute("title", "Modified Title");
element.textContent = "The element has been modified!";
console.log("New class:", element.getAttribute("class"));
console.log("New title:", element.getAttribute("title"));
}
</script>
</body>
</html>
```

Output:


Change Attributes
This is a sample element.
Change Attributes
The element has been
modified!