Advance JavaScript

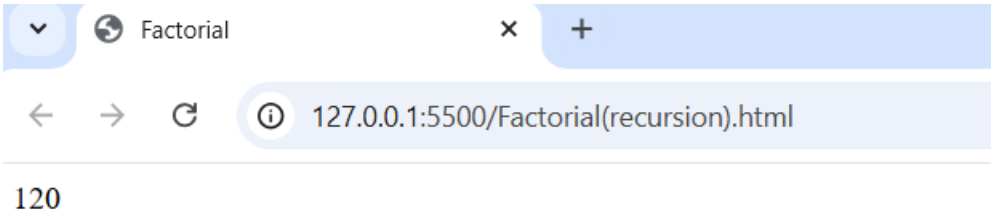
1.Recursion and stack

Task 1:

Code:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Factorial</title>
    </head>
    <body>
        <script>
            let num = 5;
            let fact = (num)=>{
                if(num == 0 || num == 1)
            {
                return 1;
            }
            else{
                return num *fact(num -1);
            }
            }
            document.write(fact(num));
        </script>
    </body>
</html>
```

Output:

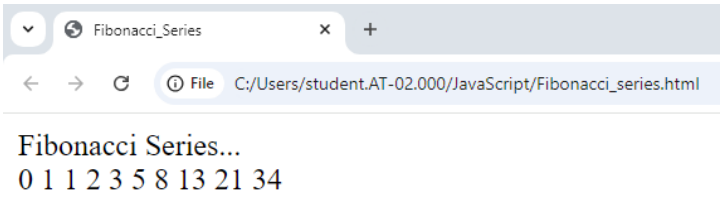


Task 2:Code:

```
<html>
    <head>
```

```
        <title>Fibonacci_Series</title>
    </head>
    <body>
        <script>
            let fib =(n,prev1,prev2)=> {
                if(n < 3)
                return;

                var crnt = prev1 + prev2;
                document.writeln(crnt +" ");
                prev2 = prev1;
                prev1 = crnt;
                fib(n-1,prev1,prev2);
            }
            let printfib= (n)=>{
                var prev2 = 0;
                var prev1 = 1;
                if(n < 1)
                document.writeln("Invalid input!");
                else if(n == 1)
                document.writeln(prev2 +" ");
                else if(n >= 2){
                document.writeln(prev2+" "+prev1+" ");
                fib(n,prev1,prev2);
            } }
            var n = 10;
             document.writeln("Fibonacci Series...<br>");
            printfib(n);
        </script>
    </body>
</html>
```

Output:

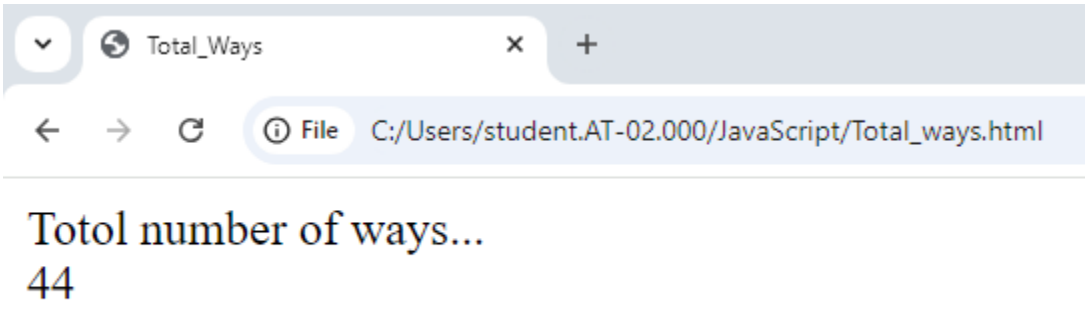


Task 3:

Code:

```
<html>
    <head>
        <title>
            Total_Ways
        </title>
    </head>
    <body>
        <script>
            let countways = (n)=>{
                if(n == 0)
                return 1;
                if(n < 0)
                return 0;
                return countways(n-1) + countways(n-2) + countways(n-3);
            }
            var n = 7;
           document.writeln("Totol number of ways...<br>");

      document.writeln(countways(n));

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

Output:



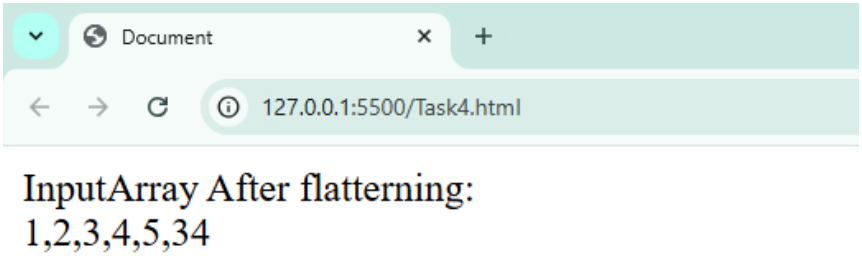
Task 4:

Code:

```
<!DOCTYPE html>
<html lang="en">

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

```
    <title>Document</title>
</head>

<body>
    <script>
        function flattenArray(ipArray) {
            let outputArray = [];
            recursion(0, ipArray, outputArray);
            return outputArray;
        }
        function recursion(index, ipArray, outputArray) {
            if (index >= ipArray.length)
                return;

            if (Array.isArray(ipArray[index]))
                recursion(0, ipArray[index], outputArray);
            else
                outputArray.push(ipArray[index]);

            recursion(index + 1, ipArray, outputArray);
        }
        var inputArray = [1, 2, [3, [4]], 5, 34];
        document.writeln("InputArray After flatterning:<br>");
        var flatArray = flattenArray(inputArray);
        document.writeln(flatArray);
        // document.writeln(flatArray[3]);
    </script>
</body>

</html>
```

Output:

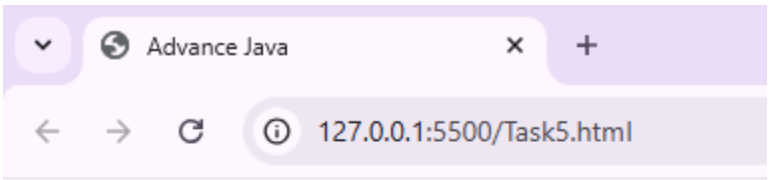


Task 5:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Advance Java </title>
</head>
<body>
    <script>
        function towerOfHanoi(n, from_rod, to_rod, aux_rod) {
            if (n == 0) {
                return;
            }
            towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
            document.write("Move disk " + n + " from rod " + from_rod + " to rod
" + to_rod + "<br/>");
            towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
        }
        var N = 3;
        towerOfHanoi(N, 'A', 'C', 'B');
    </script>
</body>
</html>
```

Output:



Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C

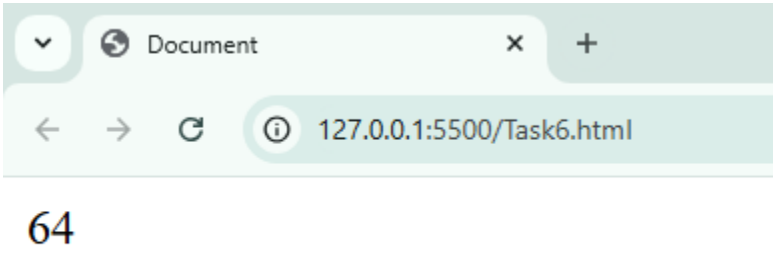
2. JSON and variable length arguments/spread syntax:

Task 1:

Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <script>
        function add(...arr) {
            return arr.reduce((sum, cur) => sum + cur, 0);
        }
        document.writeln(add(4, 43, 2, 7, 8));
    </script>

</body>
</html>
```

Output:



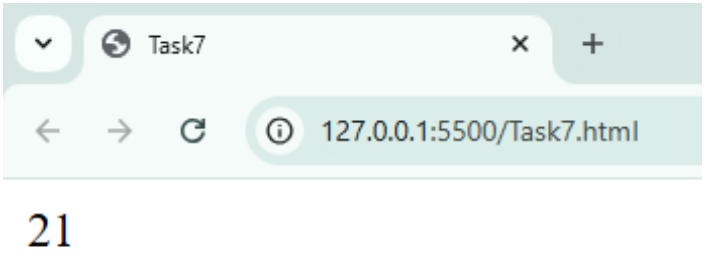
Task 2:

Code:

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Task7</title>
</head>

<body>
    <script>
        function add(...arr) {
            return arr.reduce((sum, cur) => sum + cur, 0);
        }
        function num(nums) {
            return add(...nums);
        }
```

```
        document.writeln(num([1, 2, 3, 4, 5, 6]));
    </script>
</body>

</html>
```

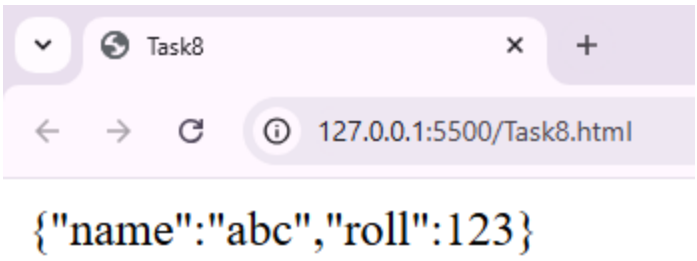
Output:



Task 3:
Code:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Task8</title>
</head>
<body>
    <script>
        const std = {
            name: "abc",
            roll: 123
        };
        const p = { ...std };
        document.writeln(JSON.stringify(p));
    </script>
</body>
</html>
```

Output:

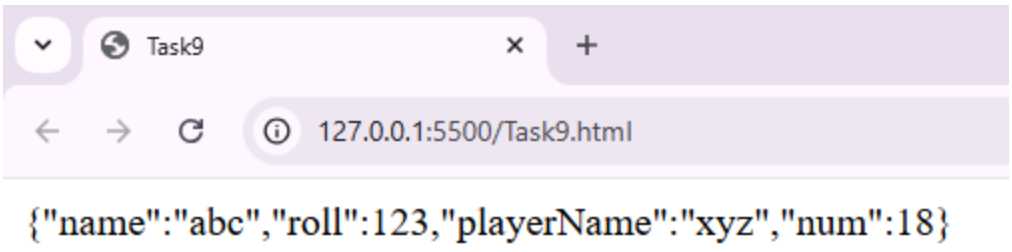


Task 4:

Code:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Task9</title>
</head>

<body>
    <script>
        const std = {
            name: "abc",
            roll: 123
        };

        const player = {
            name: "xyz",
            num: 18
        };

        const p = { ...std, playerName: player.name, num: player.num };

        document.writeln(JSON.stringify(p));
    </script>
</body>

</html>
```

Output:

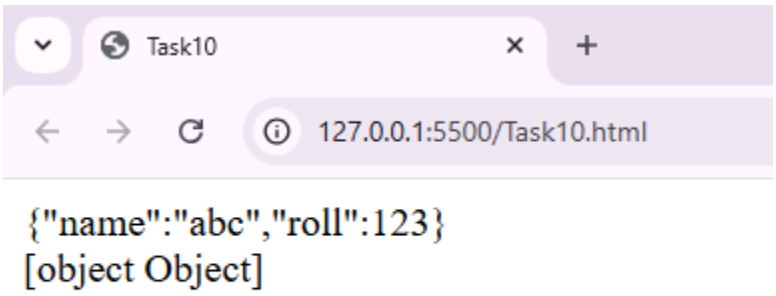


Task 5:

Code:

```
<html>
    <head><title>Task10</title></head>
   <body>
    <script>
        const std={
            name:"abc",
            roll:123
        };
        const p=JSON.stringify({...std});
        document.writeln(p+"<br>");
        const x=JSON.parse(p);
        document.writeln(x);
    </script>
   </body>
</html>
```

Output:



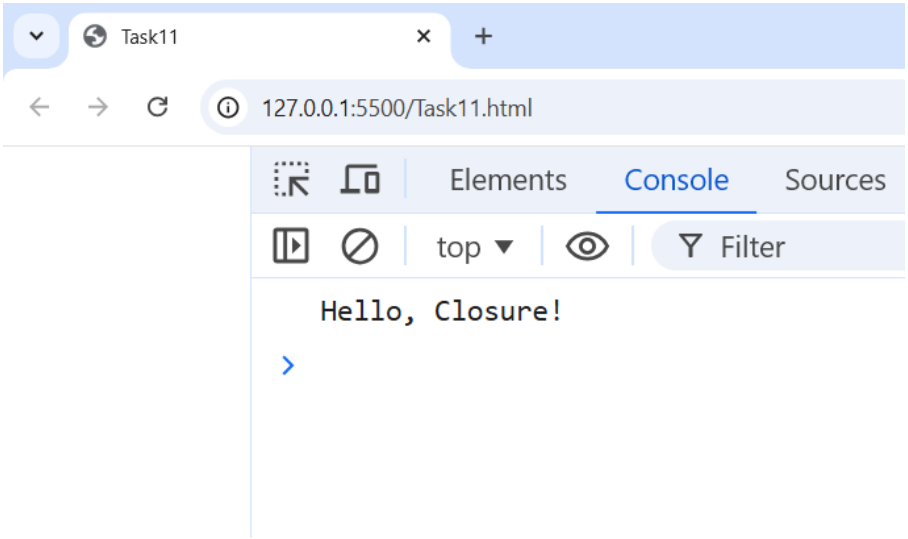
3. Closure:

Task1:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task11</title>
</head>
<body>
    <script>
        function outerFunction() {
    let localVariable = "Hello, Closure!";
    return function innerFunction() {
        return localVariable;
    };
}
const func = outerFunction();
console.log(func());
    </script>
</body>
</html>
```

Output:



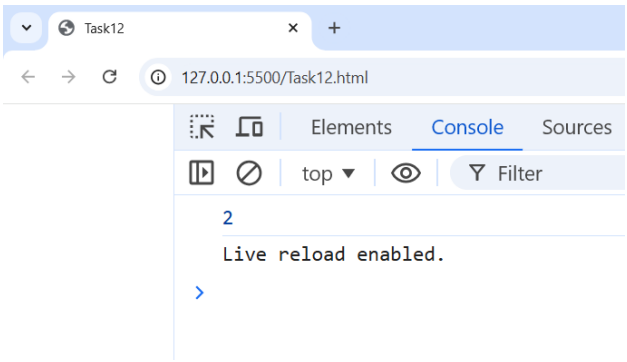
Task2:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task12</title>
</head>
<body>
    <script>
        function counter() {
    let count = 0;
    return {
        increment: function () {
            count++;
        },
        getCount: function () {
            return count;
        }
    };
}
const myCounter = counter();
myCounter.increment();
myCounter.increment();
console.log(myCounter.getCount());

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

Output:



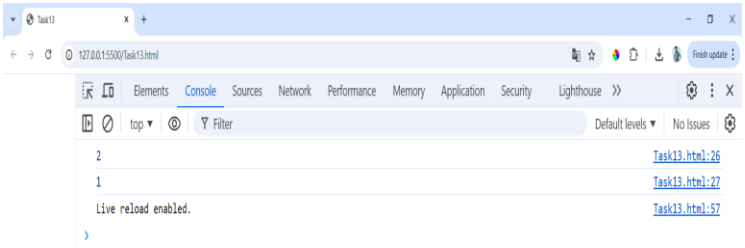
Task 3:

Code:

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

```
    <title>Task13</title>
</head>
<body>
    <script>
    function createCounter() {
    let count = 0;
    return {
        increment: function () {
            count++;
        },
        getCount: function () {
            return count;
        }
    };
}
const counter1 = createCounter();
const counter2 = createCounter();
counter1.increment();
counter1.increment();
counter2.increment();
console.log(counter1.getCount());
console.log(counter2.getCount());
    </script>
</body>
</html>
```

Output:



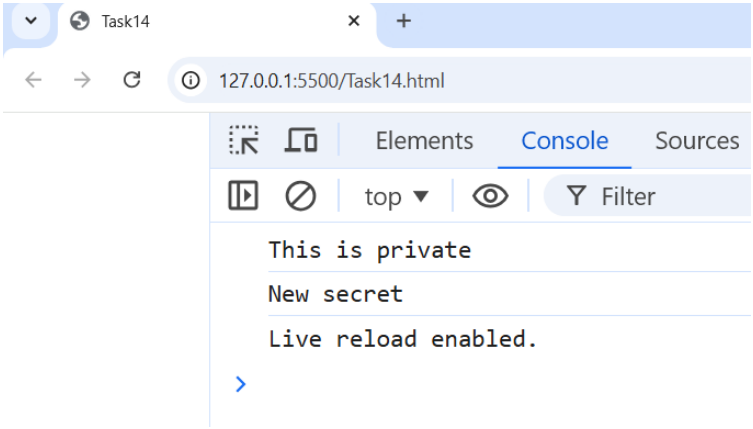
Task 4:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task14</title>
</head>
```

```
<body>
    <script>
        function privateVariable() {
    let secret = "This is private";
    return {
        getSecret: function () {
            return secret;
        },
        setSecret: function (newSecret) {
            secret = newSecret;
        }
    };
}

const secretHandler = privateVariable();
console.log(secretHandler.getSecret());
secretHandler.setSecret("New secret");
console.log(secretHandler.getSecret());

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

Output:



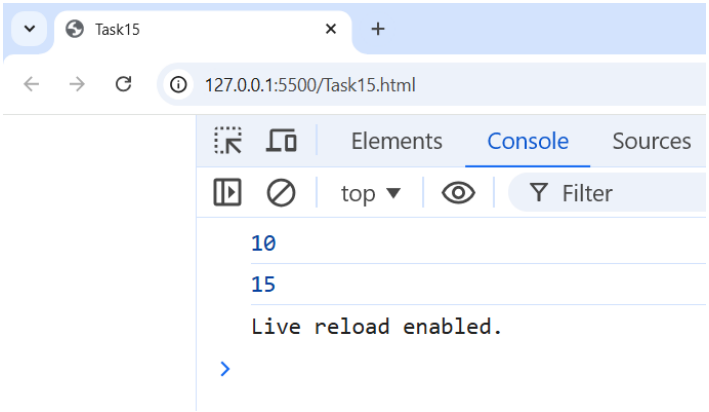
Task 5:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task15</title>
</head>
<body>
    <script>
        function multiplier(factor) {
    return function (number) {
        return number * factor;
    };
}
const double = multiplier(2);
const triple = multiplier(3);
console.log(double(5));
console.log(triple(5))

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

Output:



4. Promise, Promises chaining:

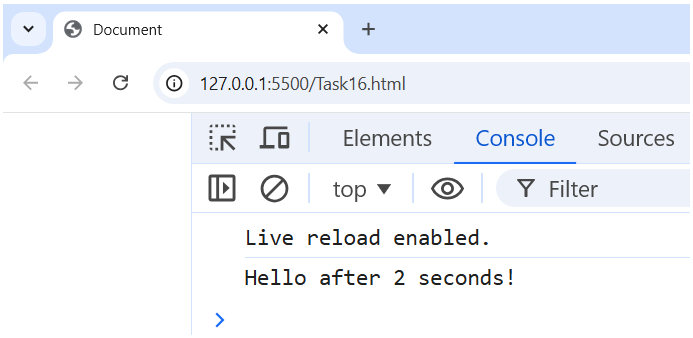
Task1:

Code:

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

```
<body>
    <script>
        function delayedGreeting(seconds) {
    return new Promise((resolve) => {
        setTimeout(() => resolve("Hello after " + seconds + " seconds!"), seconds
* 1000);
    });
}
delayedGreeting(2).then(console.log);
    </script>
</body>
</html>
```

Output:



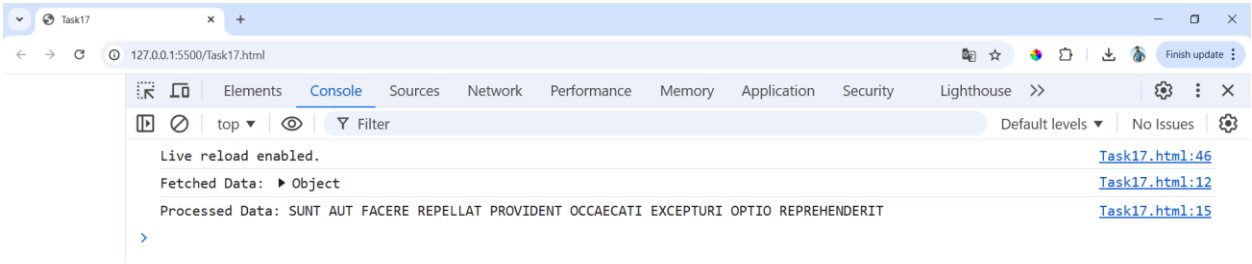
Task 2:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task17</title>
</head>
<body>
    <script>
        fetch('https://jsonplaceholder.typicode.com/posts/1').then(response =>
response.json())
    .then(data => {
        console.log("Fetched Data:", data);
        return data.title.toUpperCase();
    })
    .then(uppercaseTitle => console.log("Processed Data:", uppercaseTitle))
    .catch(error => console.error("Error:", error));
    </script>
</body>
```

```
</html>
```

Output:

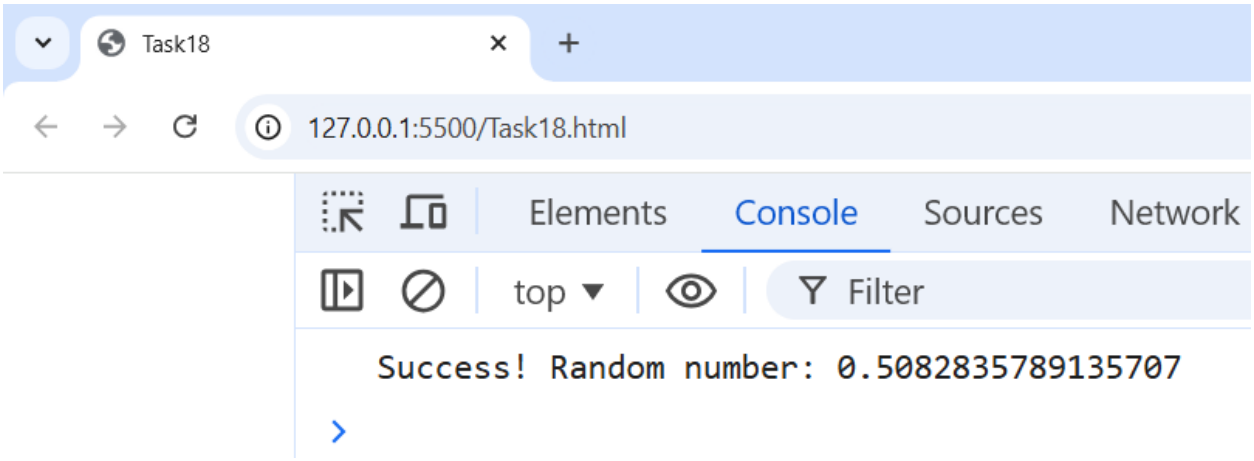


Task 3:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task18</title>
</head>
<body>
    <script>
        function randomPromise() {
    return new Promise((resolve, reject) => {
        const random = Math.random();
        if (random > 0.5) {
            resolve("Success! Random number: " + random);
        } else {
            reject("Failure! Random number: " + random);
        }
    });
}
randomPromise()
    .then(console.log)
    .catch(console.error);

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

Output:

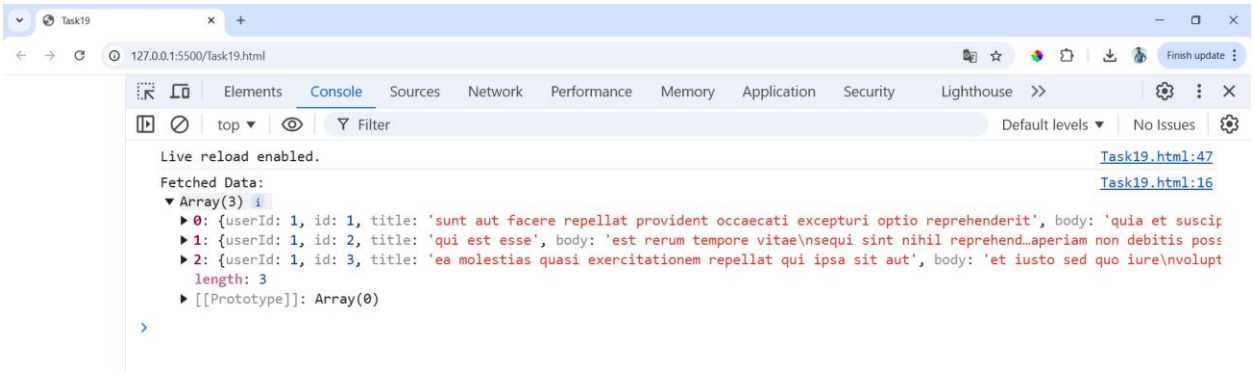


Task4:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task19</title>
</head>
<body>
    <script>
        const urls = [
    'https://jsonplaceholder.typicode.com/posts/1',
    'https://jsonplaceholder.typicode.com/posts/2',
    'https://jsonplaceholder.typicode.com/posts/3'
];
Promise.all(urls.map(url => fetch(url).then(res => res.json())))
    .then(data => console.log("Fetched Data:", data))
    .catch(error => console.error("Error:", error));
    </script>
</body>
</html>
```

Output:

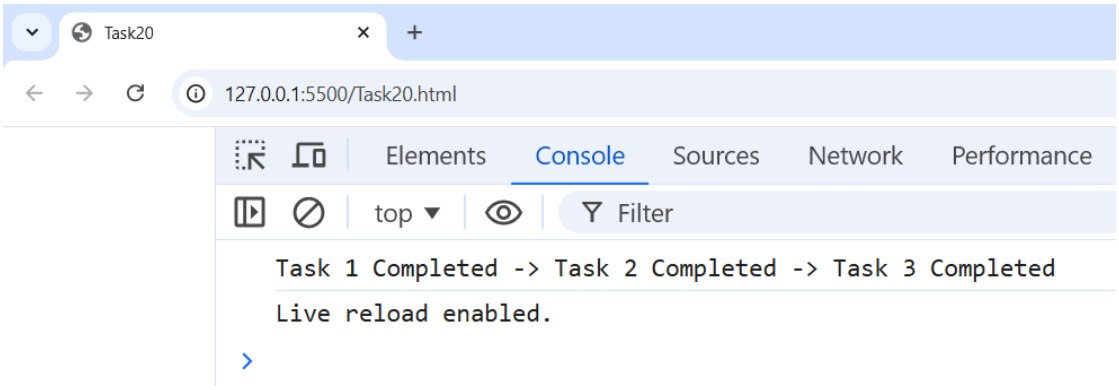


Task5:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task20</title>
</head>
<body>
    <script>
        function task1() {
    return Promise.resolve("Task 1 Completed");
}

function task2(previousResult) {
    return Promise.resolve(previousResult + " -> Task 2 Completed");
}

function task3(previousResult) {
    return Promise.resolve(previousResult + " -> Task 3 Completed");
}
task1()
    .then(result => task2(result))
    .then(result => task3(result))
    .then(finalResult => console.log(finalResult))
    .catch(error => console.error("Error:", error));
    </script>
</body>
</html>
```

Output:



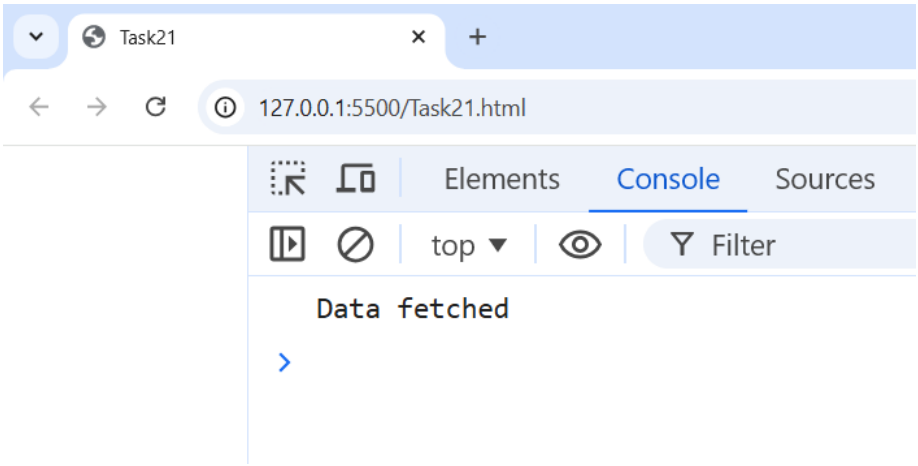
5. Async/await:

Task1:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task21</title>
</head>
<body>
    <script>
function fetchData() {
    return new Promise((resolve, reject) => {
        setTimeout(() => resolve("Data fetched"), 1000);
    });
}
async function fetchDataAsync() {
    const data = await fetchData();
    console.log(data);
}
fetchDataAsync();
    </script>
</body>
</html>
```

Output:

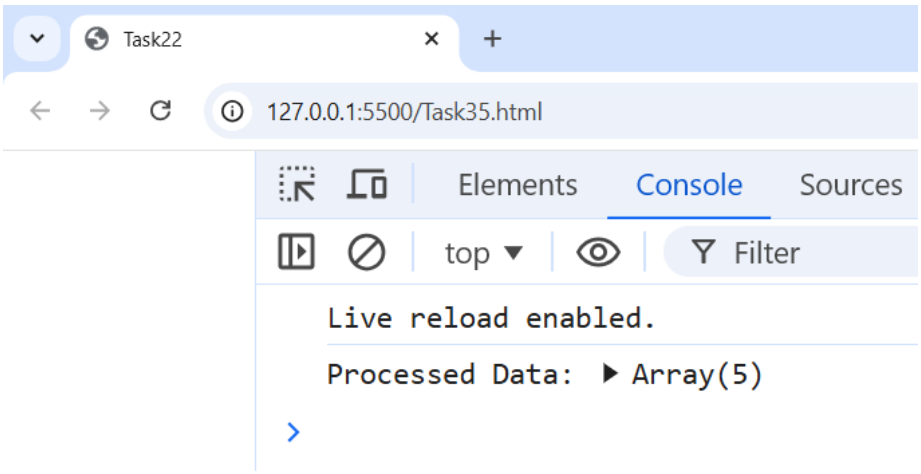


Task 2:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task22</title>
</head>
<body>
    <script>
        async function fetchAndProcessData() {
    const response = await fetch('https://jsonplaceholder.typicode.com/posts');
    const data = await response.json();
    console.log('Processed Data:', data.slice(0, 5));
}
fetchAndProcessData();

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

Output:

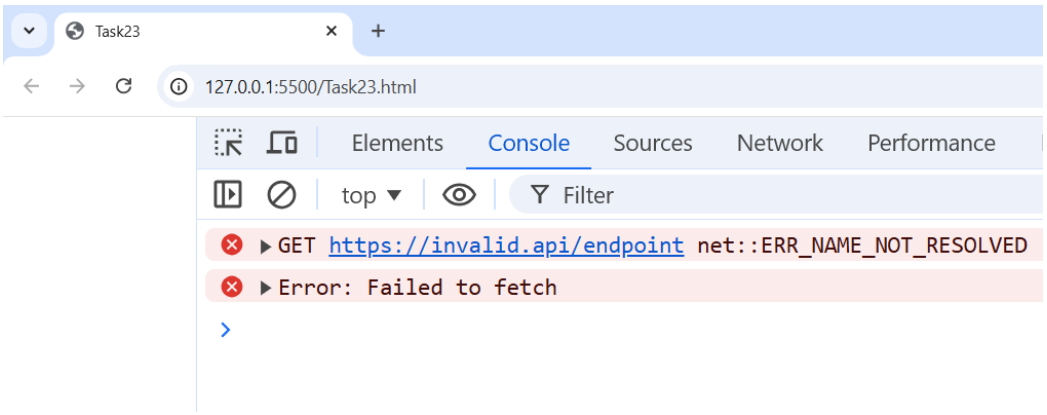


Task 3:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task23</title>
</head>
<body>
    <script>
        async function fetchWithErrorHandling() {
    try {
        const response = await fetch('https://invalid.api/endpoint');
        if (!response.ok) throw new Error('API responded with an error');
        const data = await response.json();
        console.log(data);
    } catch (error) {
        console.error('Error:', error.message);
    }
}
fetchWithErrorHandling();

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

Output:

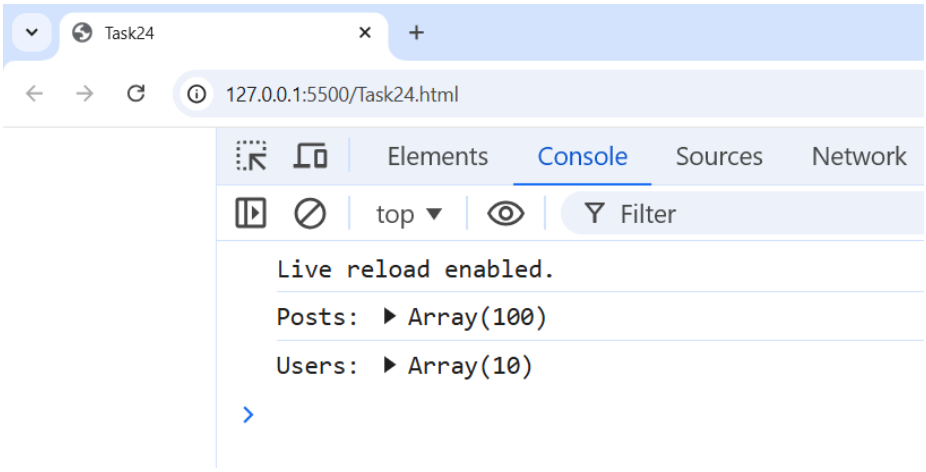


Task4:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task24</title>
</head>
<body>
    <script>
        async function fetchMultipleData() {
    const urls = [
        'https://jsonplaceholder.typicode.com/posts',
        'https://jsonplaceholder.typicode.com/users'
    ];
    const promises = urls.map(url => fetch(url).then(res => res.json()));
    const [posts, users] = await Promise.all(promises);
    console.log('Posts:', posts);
    console.log('Users:', users);
}
fetchMultipleData();

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

Output:

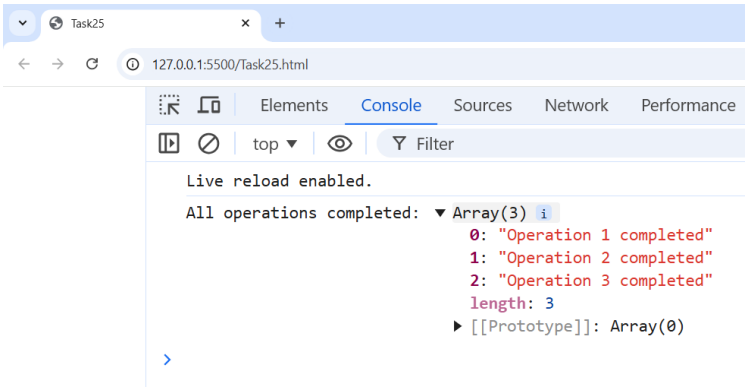


Task5:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task25</title>
</head>
<body>
    <script>
        async function completeOperations() {
    const op1 = new Promise(resolve => setTimeout(() => resolve('Operation 1
completed'), 1000));
    const op2 = new Promise(resolve => setTimeout(() => resolve('Operation 2
completed'), 2000));
    const op3 = new Promise(resolve => setTimeout(() => resolve('Operation 3
completed'), 1500));

    const results = await Promise.all([op1, op2, op3]);
    console.log('All operations completed:', results);
}
completeOperations();
    </script>
</body>
</html>
```

Output:



6. Modules introduction, Export and Import:

Task1:

Code:

Module.js

```
export const myVar = 42;

export function myFunction() {
    return 'Hello from myFunction!';
}

export class MyClass {
    greet() {
        return 'Hello from MyClass!';
    }
}
```

Output:

No output. Because this file only have export functions,variable and class!!!

Task 2:

Code:

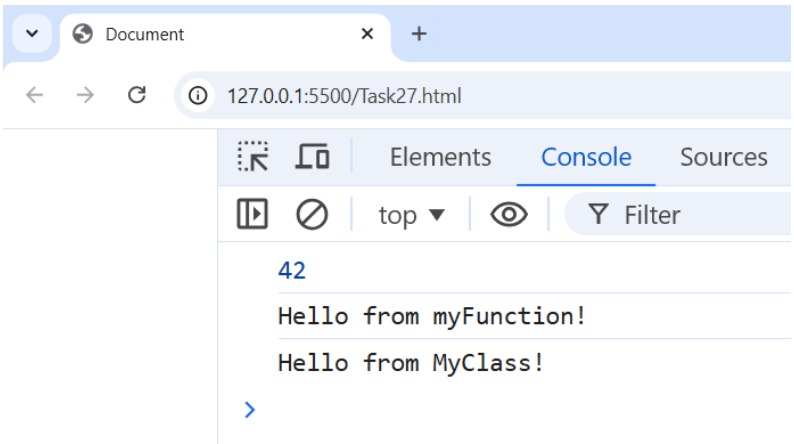
Module.js:

```
export const myVar = 42;

export function myFunction() {
    return 'Hello from myFunction!';
}
```

```
export class MyClass {
    greet() {
        return 'Hello from MyClass!';
    }
}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script type="module">
import { myVar, myFunction, MyClass } from './module.js';
console.log(myVar);
console.log(myFunction());
const myClassInstance = new MyClass();
console.log(myClassInstance.greet());
    </script>
</body>
</html>
```

Output:



Task 3:

Code:

```
export function namedExport(){
    return "This is the javascript file,It contains Named export are here!!!";
```

```
}
export function namedExport2(){
    return "This is the another javascript file for named export!!!";
}
```

Output:

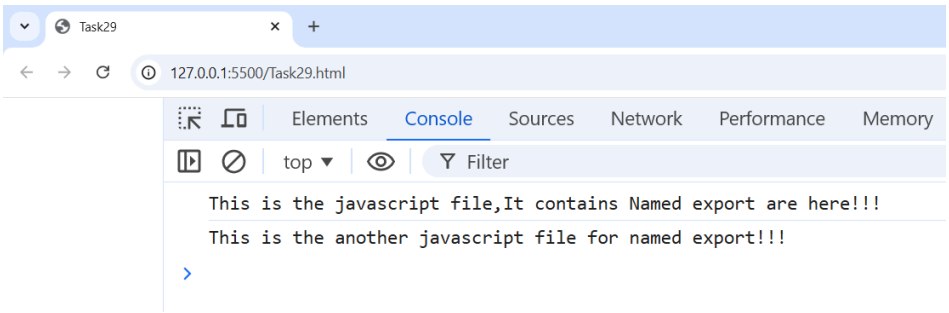
No output.Because this file only have export functions!!!

Task4:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task29</title>
</head>
<body>
    <script type ="module">
        import {namedExport,namedExport2} from './Task28.js';
        console.log(namedExport());
        console.log(namedExport2());
    </script>
</body>
</html>
```

Output:



Task5:

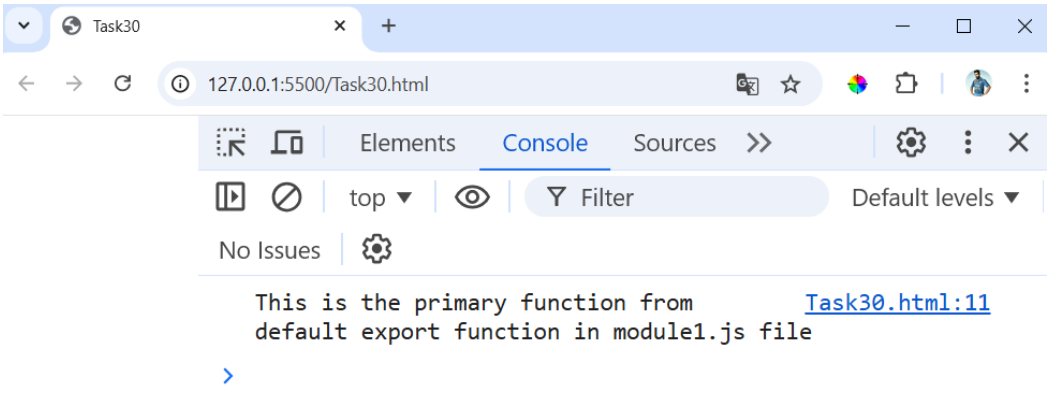
Code:

Module1.js

```
function Primary()
```

```
{
    return "This is the primary function from default export function in
module1.js file";
}
export default Primary;
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task30</title>
</head>
<body>
    <script type="module">
        import primary from './module1.js';
        console.log(primary());
    </script>
</body>
</html>
```

Output:



7. Browser: DOM Basics:

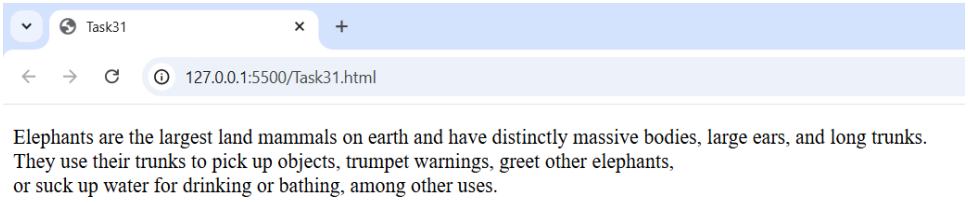
Task1:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task31</title>
```

```
</head>
<body>
    <p id="para"></p>
    <script>
        var p = document.getElementById("para");
        const content ="Elephants are the largest land mammals on earth and have
distinctly massive bodies, large ears, and long trunks.<br> They use their trunks
to pick up objects, trumpet warnings, greet other elephants, <br>or suck up water
for drinking or bathing, among other uses.";
        p.innerHTML = content;
  </script>
</body>
</html>
```

Output:



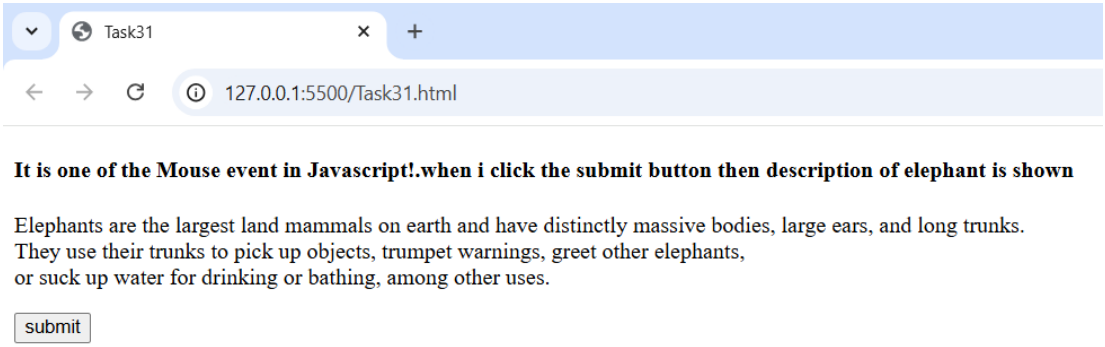
Task 2:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task31</title>
</head>
<body>
    <h4>It is one of the Mouse event in Javascript!.when i click the submit
button then description of elephant is shown</h4>
    <p id="para"></p>
    <input type ="button" value ="submit" onclick="MyElephant()">
    <script>
        function MyElephant()
        {
        var p = document.getElementById("para");
        const content ="Elephants are the largest land mammals on earth and have
distinctly massive bodies, large ears, and long trunks.<br> They use their trunks
```

```
to pick up objects, trumpet warnings, greet other elephants, <br>or suck up water
for drinking or bathing, among other uses.";
        p.innerHTML = content;
        }
  </script>
</body>
</html>
```

Output:



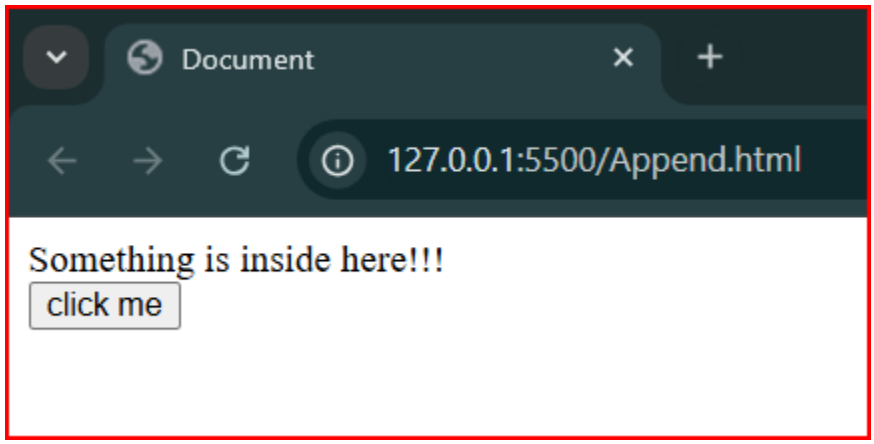
Task 3:

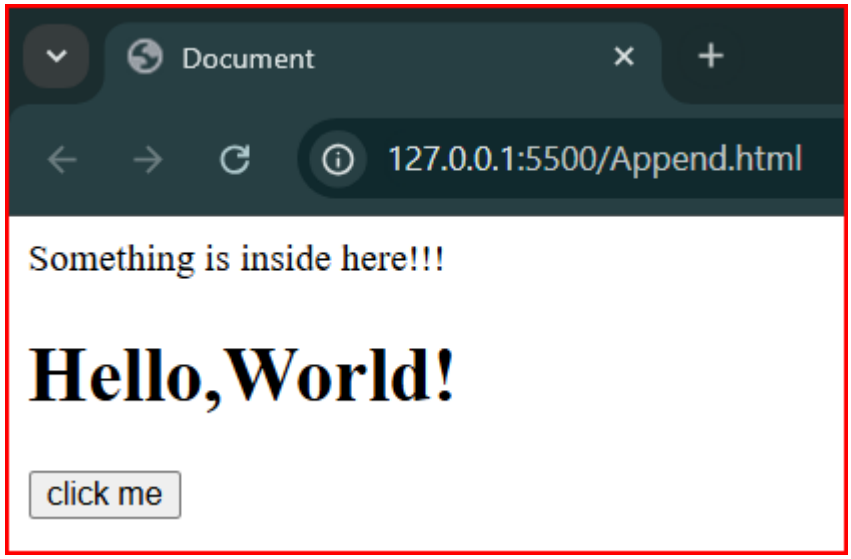
Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body>
    <div id ="di">
        Something is inside here!!!
    </div>
    <input type="button" value="click me" id ="bd" onclick="app()"/>
    <script>
        var div = document.getElementById("di");
        function app(){
        var ele = document.createElement('h1');
        ele.innerHTML ="Hello,World!"
```

```
        div.appendChild(ele);
        }
    </script>
</body>
</html>
```

Output:





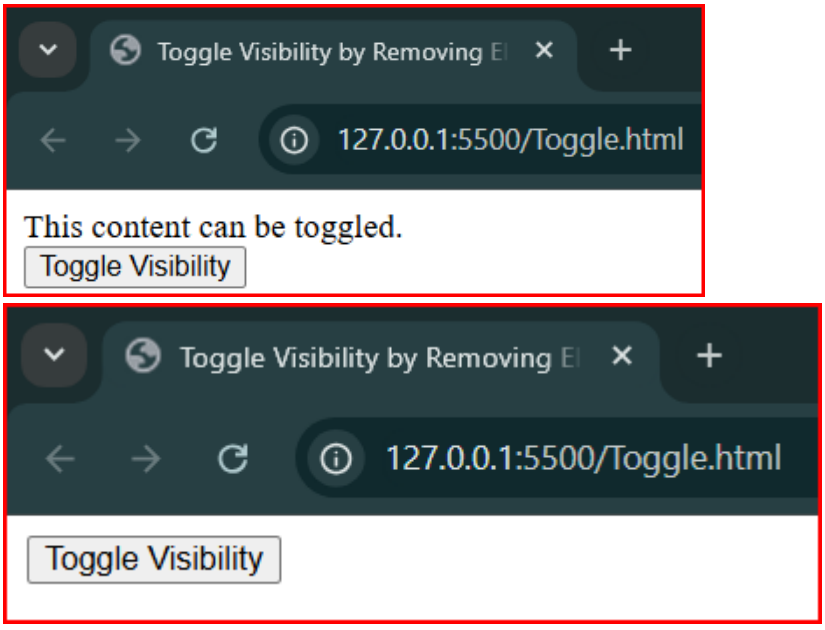
Task4:

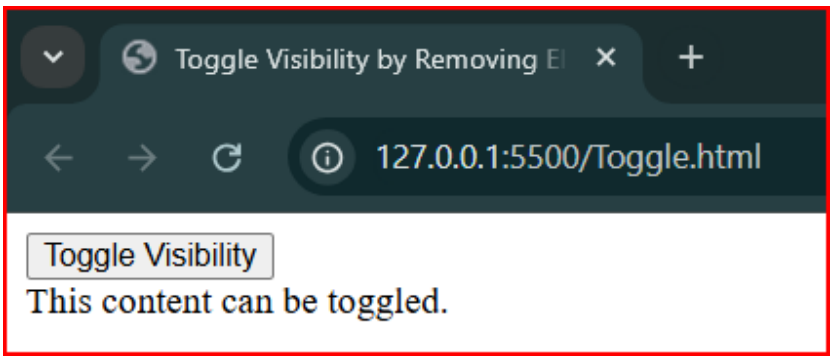
Code:

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

```
    <title>Toggle Visibility by Removing Element</title>
</head>
<body>
    <div id="content">
        This content can be toggled.
    </div>
    <button onclick="toggleVisibility()">Toggle Visibility</button>
    <script>
        var content = document.getElementById("content");
        var parent = content.parentNode;

        function toggleVisibility() {
            if (content.parentNode) {
                parent.removeChild(content);
            } else {
                parent.appendChild(content);
            } }
    </script>
</body>
</html>
```

Output:





Task5:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Retrieve and Modify Attributes</title>
</head>
<body>
    <img id="image"
src="https://static.vecteezy.com/system/resources/thumbnails/026/525/162/s
mall/lion-animal-isolated-photo.jpg" alt="Example Image" height="150px"
width="200px"><br>
    <button onclick="modifyAttributes()">Modify Attributes</button>

    <script>
        function modifyAttributes() {

            var img = document.getElementById("image");

            var currentSrc = img.getAttribute("src");
            var currentAlt = img.getAttribute("alt");

            console.log("Current src:", currentSrc);
            console.log("Current alt:", currentAlt);
```

```
        img.setAttribute("src",
"https://www.healthyeating.org/images/default-source/home-0.0/nutrition-topics-2.0/general-nutrition-wellness/2-2-2-3foodgroups_fruits_detailfeature.jpg?sfvrsn=64942d53_4"); // Change the image source
        img.setAttribute("alt", "Updated Example Image");
        var updatedSrc = img.getAttribute("src");
        var updatedAlt = img.getAttribute("alt");

        console.log("Updated src:", updatedSrc);
        console.log("Updated alt:", updatedAlt);
    }
  </script>
</body>
</html>
```

Output: