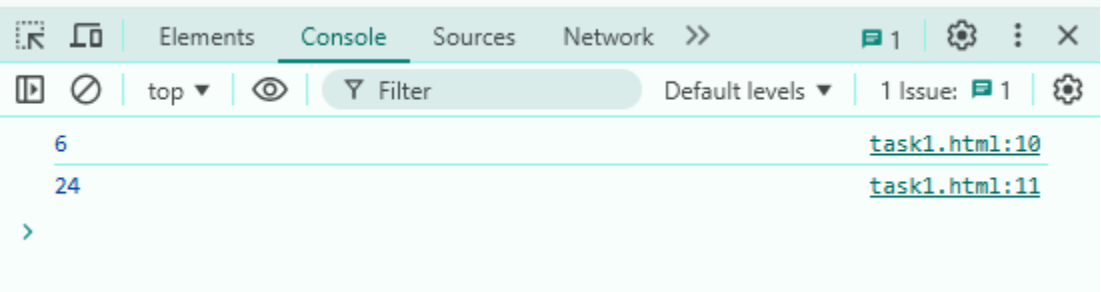
Recursion and stack

TASK1

```
<body>
<script>
function factorial(n){
if(n==0)
{
return 1;
}
return n*factorial (n-1);
}
console.log(factorial(3));
console.log(factorial(4));
</script>
```

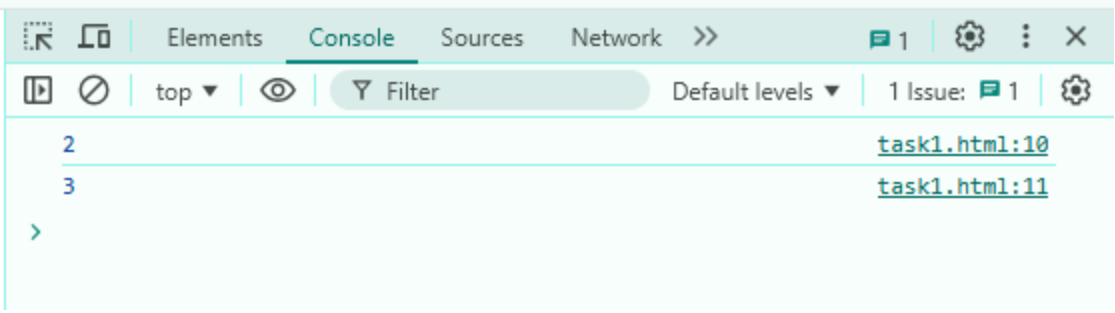
OUTPUT



TASK 2

```
<body>
  <script>
  function fibbonacci(n){
  if(n<=1)
  {
  return n;
  }
  return fibbonacci (n-1)+fibbonacci(n-2);
  }
  console.log(fibbonacci(3));
  console.log(fibbonacci(4));
</script>
</body>
```

OUTPUT



JSON and variable length arguments/spread

TASK 1

```
<body>
  <script>
    function sum(...arguments){
      let total =0;
      for(let num of arguments){
        total+=num;
      }
      return total;
    }
    console.log(sum(1,4,5,6,));
    </script>
</body>
```

OUTPUT

TASK 2

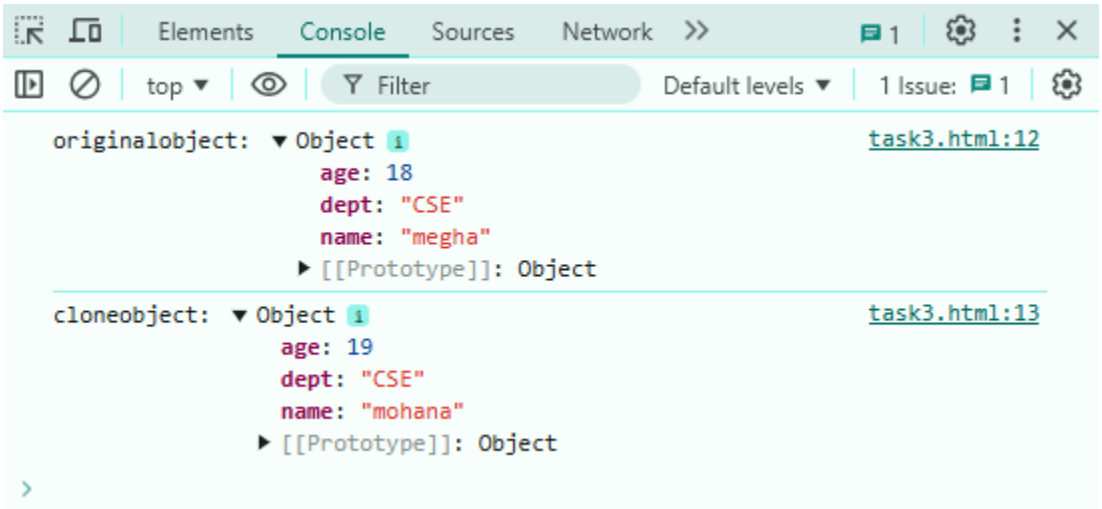
```
<body>
  <script>
    function sum(...arguments){
      let total =0;
      for(let num of arguments){
        total+=num;
      }
      return total;
    }
    let arr=[9,4,5,6];
    console.log(sum(...arr));
    </script>
</body>
```

OUTPUT

TASK 3

```
<body>
  <script>
    let originalobject={
      name:"megha",
      age:18,
      dept:"CSE"

    };
    let cloneobject=JSON.parse(JSON.stringify(originalobject));
    cloneobject.name="mohana";
    cloneobject.age=19;
    console.log("originalobject:",originalobject);
    console.log("cloneobject:",cloneobject);
  </script>
</body>
```

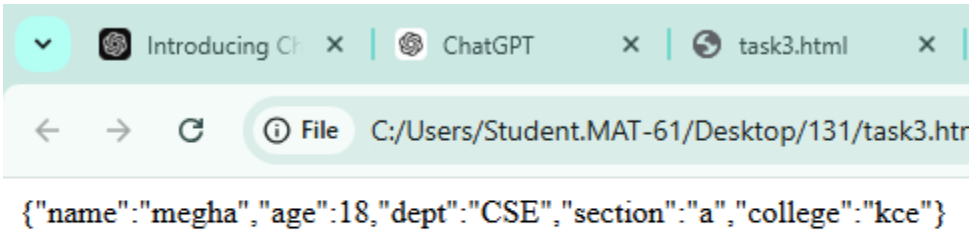
OUTPUT



TASK4

```
<body>
  <script>
    let originalobject={
      name:"megha",
      age:18,
      dept:"CSE"

    };
    let cloneobject={
       section:"a",
       college:"kce"

    };
    function result(originalobject,cloneobject) {
       return{...originalobject,...cloneobject};
    }
  let result1 = result(originalobject,cloneobject);
    document.writeln(JSON.stringify(result1));
  </script>
</body>
```

OUTPUT

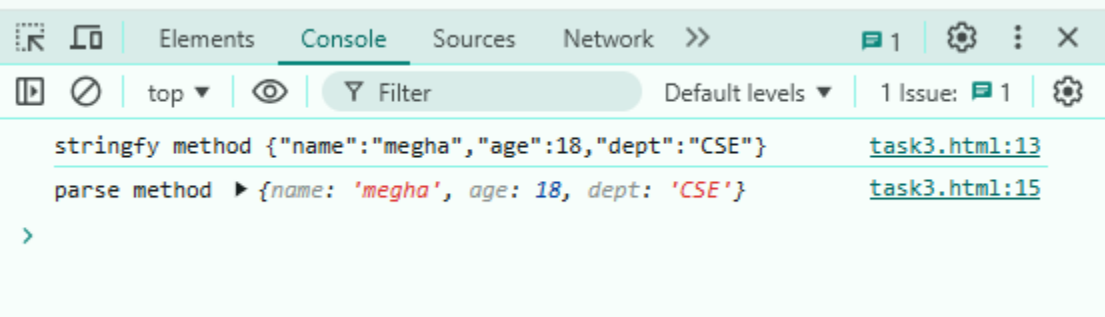


TASK 5

```
<body>
  <script>
    let originalobject={
      name:"megha",
      age:18,
      dept:"CSE"

    };


  let result1 =(JSON.stringify( originalobject));
 console.log("stringfy method", result1);
```

```
  let result2=(JSON.parse(result1));
  console.log("parse method", result2);
  </script>
</body>
```

OUTPUT

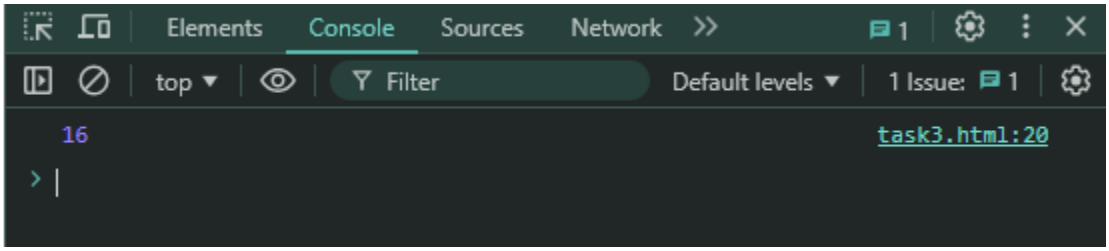


Recursion and stack:

TASK 3

```
<boby>
  <script>
    function ways(n){
     if(n==0){
     return 1;}
    if(n==1){
       return 1;

    }
    if(n==2){
       return 2;
    }

    if(n==3){
       return 4;
    }
    return ways(n-1)+ways(n-2)+ways(n-3)+ways(n-3);
    }
    let n=5;
    console.log(ways(n));
  </script>
</boby>
```

OUTPUT

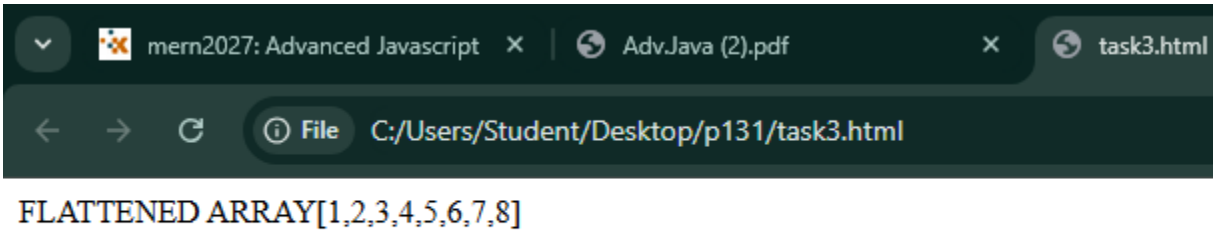


TASK 4

```
<boby>
  <script>
  function flattenarray(arr)
  {
    var result=[];
    for(var i=0;i<arr.length;i++){
      var item = arr[i];
      if(Array.isArray(item)){
        result=result.concat(flattenarray(item));
      }
      else{
        result.push(item);
      }

    }return result;
```

```
    }
    let nestedarray=[1,[2,3,[4,5]],[6,7],8];
    let flatten=flattenarray(nestedarray);
    document.writeln("FLATTENED ARRAY" + JSON.stringify(flatten));
    </script>
</boby>
```

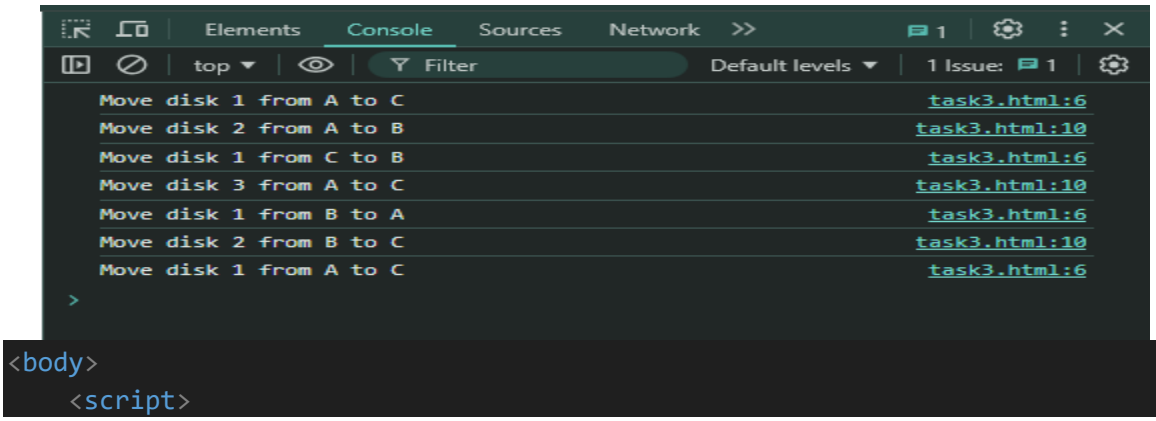
OUTPUT



TASK 5

```
<html>
<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);
}
const n = 3;
towerOfHanoi(n, 'A', 'C', 'B');
</script>
</body>
</html>
```

OUTPUT:

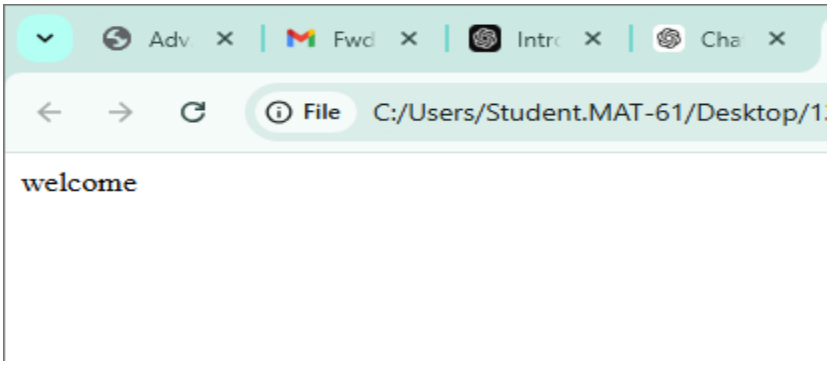


Promise, Promises chaining:

TASK 1

```
<html>
  <script>
    const mypromises = new Promise((resolve) => {
      setTimeout(() => {
        resolve("welcome");
      }, 1000);
    });

    mypromises.then((value) => {
      document.writeln(value);
    });
  </script>
</html>
```

OUTPUT

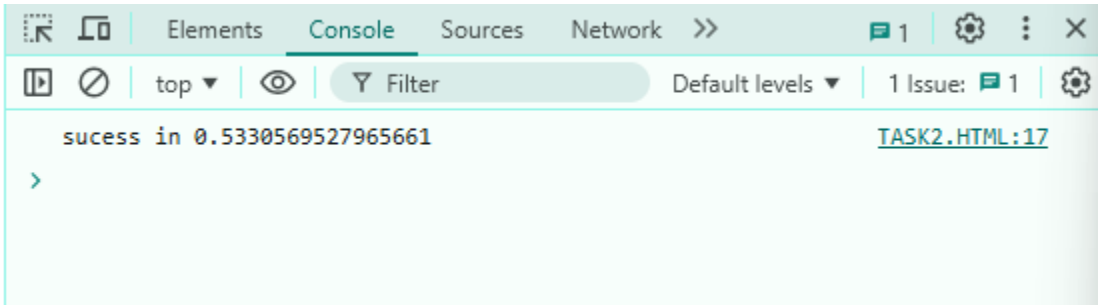


TASK3

```
<html>
  <body>
    <script>
      function randomPromises(){
        return new Promise((resolve,reject)=>{
         const randnum=Math.random();
         if(randnum > 0.5)
        {
         resolve("sucess in "+randnum);
        }
        else{
         reject("fail in"+randnum);
        }
        });
      }
      randomPromises()
      .then((message)=>console.log(message))
      .catch((error)=>console.log(error));
    </script>
```

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

OUTPUT

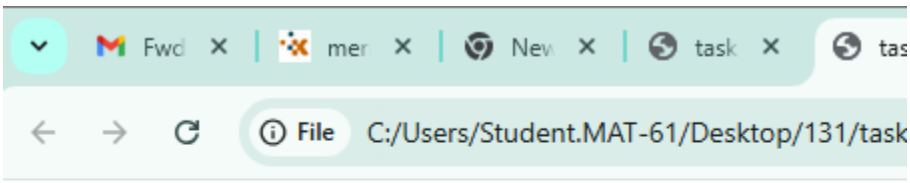


TASK 5

```
<body>
  <script>
    function placeorder(order){
      return new Promise((resolve) => {
        setTimeout(() =>
      {
      document.writeln(`${order} order placed`);
       document.writeln("<br>");
       resolve (order);
      },1000)
    })
    }
    function preparefood(order){
      return new Promise((resolve) => {
        setTimeout(() =>
      {
      document.writeln(`${order} food prepared`);
       document.writeln("<br>");
       resolve (order);
      },1000)
    })
    }
    function deliverfood(order){
      return new Promise((resolve) => {
        setTimeout(() =>
      {
      document.writeln(`${order} order placed`);
       document.writeln("<br>");
       resolve (order);
      },1500)
    })
    }
    async function orderfood(fooditem){
     const order=await placeorder(fooditem);
     const preparedfood=await preparefood(order);
```

```
        const deliver=await deliverfood(order);
        document.writeln();
        document.writeln("process complete");
       }
       orderfood("ButterChicken")

   </script>
</body>
```

OUTPUT



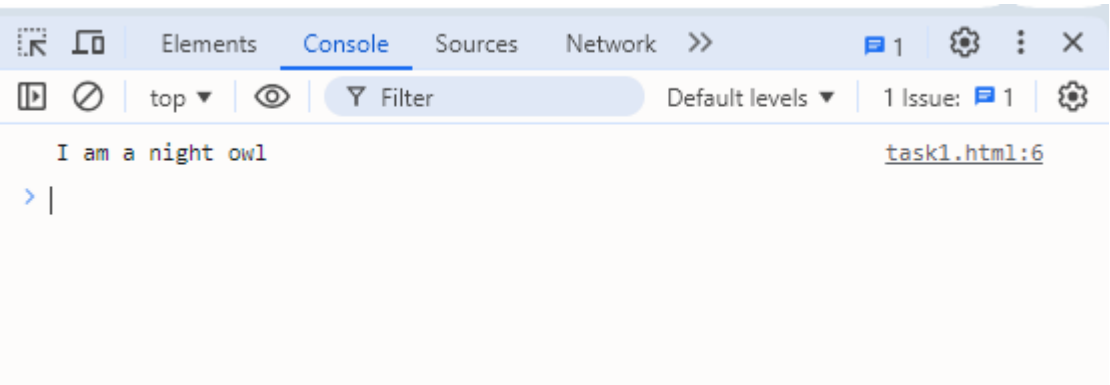
ButterChicken order placed
ButterChicken food prepared
ButterChicken order placed
process complete

CLOSURE

TASK 1

```
<body>
   <script>
   function outer() {
let loc = 'I am a nightowl';
return function inner() {
console.log(loc);
};
}
const closure = outer();
closure();
      </script>
</body>
```

OUTPUT

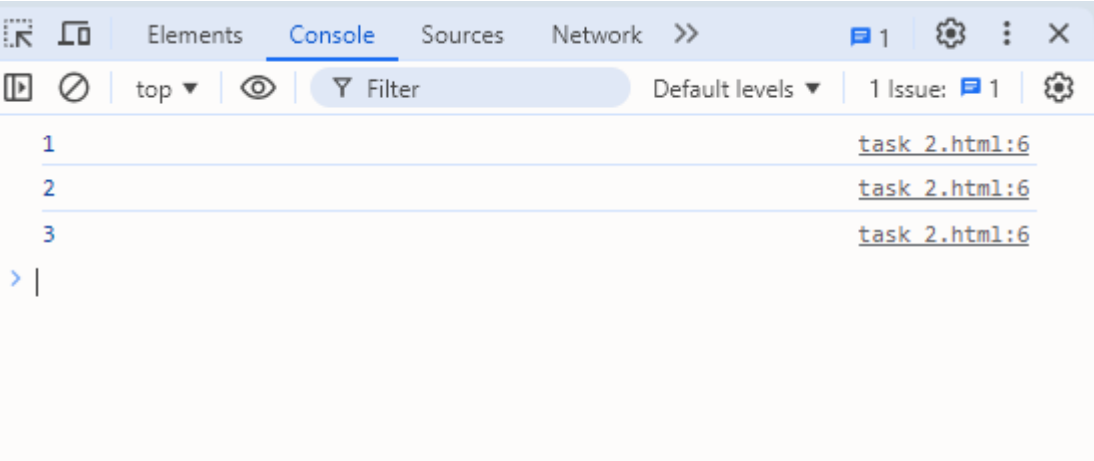


TASK 2

```
<script>
   function counting(){
```

```
let count = 0;
return function(){
count++;
console.log(count);
};
}
const counter = counting()
counter();
counter();
counter();

</script>
```

OUTPUT

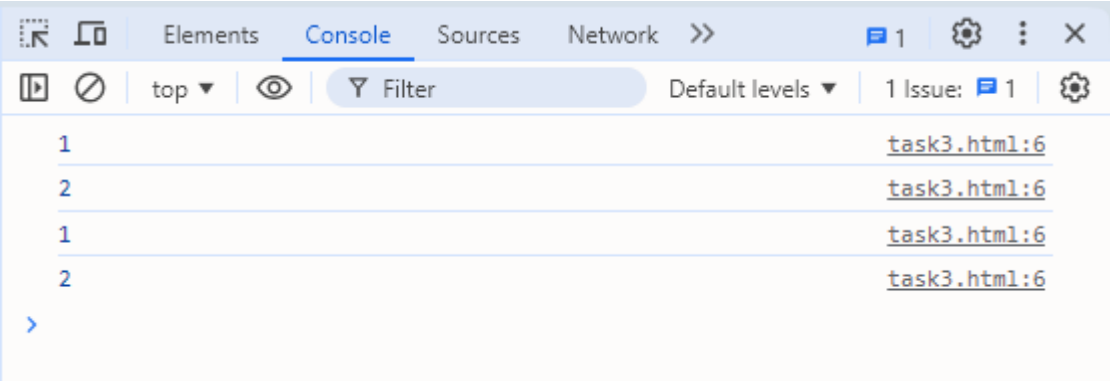


TASK 3

```
<script>
function counting(){
let count = 0;
```

```
return function(){
count++;
console.log(count);
};
}
const counter1 = counting();
const counter2 = counting();
counter1()
counter1()
 counter2()
counter2()
</script>
```

OUTPUT

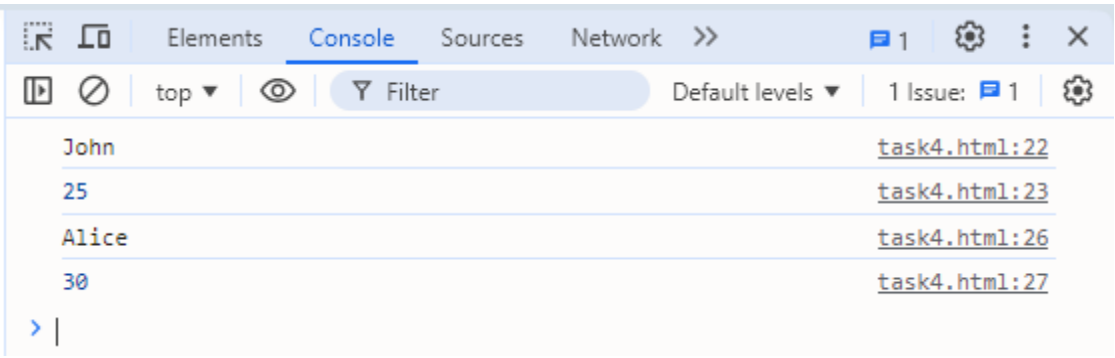


TASK 4

```
<script>
function createPerson(name, age) {
let _name = name;
let _age = age;
```

```
return {
getName: function() {
return _name;
},
getAge: function() {
return _age;

},
setName: function(newName) {
_name = newName;
},
setAge: function(newAge) {
_age = newAge;
}
};
}
const person = createPerson('John', 25);
console.log(person.getName());
console.log(person.getAge());
person.setName('Alice');
person.setAge(30);
console.log(person.getName());
console.log(person.getAge());
</script>
```

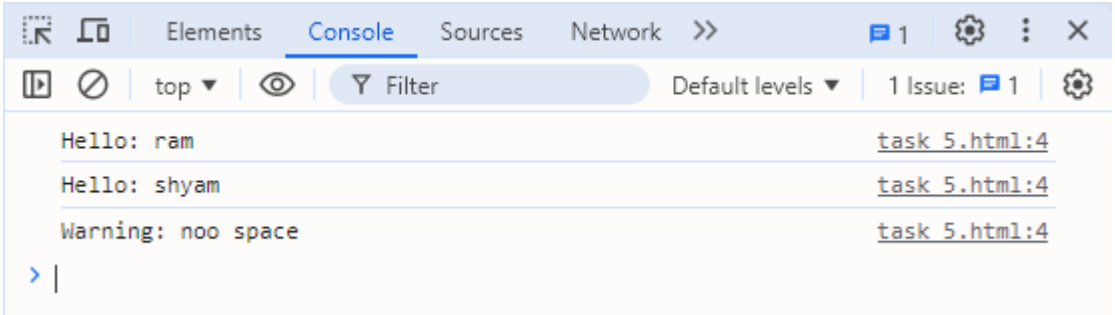
OUTPUT



TASK 5

```
<script>
   function functionFactory(prefix) {
   return function(message) {
   console.log(prefix + ': ' + message);
   };
   }
   const greet = functionFactory('Hello');
   greet('ram');
   greet('shyam');
   const warn = functionFactory('Warning');
   warn('noo space');
   </script>
```

OUTPUT

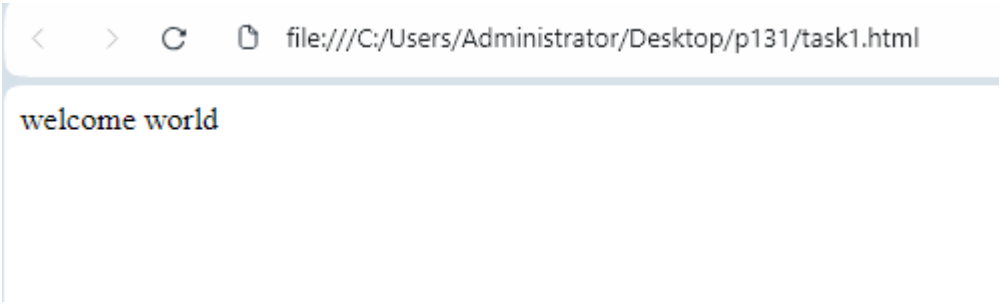


Browser: DOM Basics:

TASK 1:

```
<body>
   <p id="my">HELLO WORLD!</p>
   <script>
      document.getElementById("my").innerHTML="welcome world" ;
      </script>
</body>
```

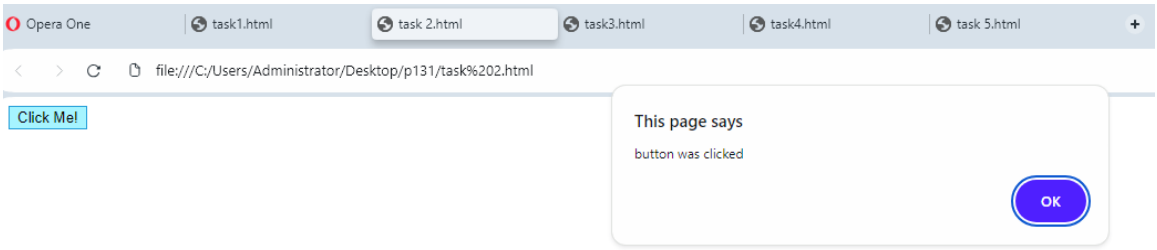
OUTPUT:



TASK 2:

```
<body>
   <button id="mybutton">Click Me!</button>
<script>
   document.getElementById("mybutton").addEventListener("click",function(){
    alert("button was clicked");
   });
    </script>
   </body>
```

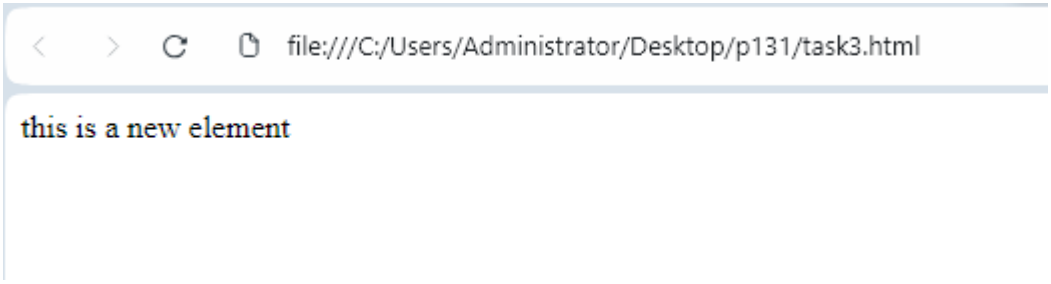
OUTPUT



TASK 3:

```
<body>
   <div id ="hello">
   </div>
<script>
   var newelement=document.createElement("p");
   newelement.innerHTML="this is a new element";
   document.getElementById("hello").appendChild(newelement);
    </script>
</body>
```

OUTPUT

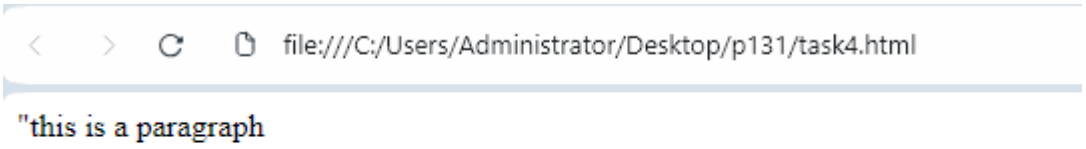


TASK 4:

```
<body>
   <p id="toggle">"this is a paragraph</p>
   <div id="container"></div>
<script>
```

```
document.getElementById("container").addEventListener("click",function(){
    var paragraph=document.getElementById("toggle");
    if(paragraph.style.display===none){
        paragraph.style.display="block";
    }
    else{
        paragraph.style.display="none";
    }
});
</script>
</body>
```

OUTPUT



TASK 5:

```
<body>
    <img id="myimage" src="image.jpg" alt ="inital image" height="200px" width="200px">
    <script>
     var image=document.getElementById("myimage");

     image.src="newimage.jpg";
     console.log(image.alt);
     image.alt="updated image";
    </script>
```

</body>

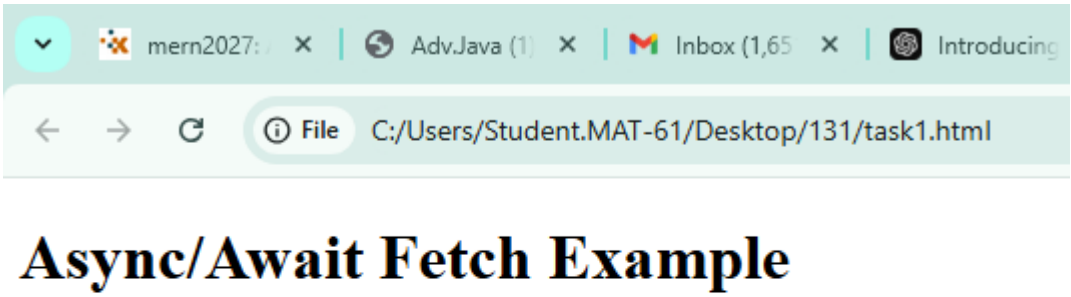
OUTPUT

5.ASYNC/WAIT

TASK 1

```
<body>

  <h1>Async/Await Fetch Example</h1>

  <script>

    async function fetchUserData(userId) {
      try {

        const response = await fetch(`https://jsonplaceholder.typicode.com/users/${userId}`);


        if (!response.ok) {
          throw new Error('Failed to fetch user data');
        }


        const data = await response.json();
        return data;
      } catch (error) {
        console.error('Error:', error);
        throw error;
      }
    }


    async function fetchAndLogUserData() {
      try {

        const user = await fetchUserData(1);
        console.log(user);
      } catch (error) {

        console.log('Failed to retrieve user data:', error);
      }
    }


    fetchAndLogUserData();
  </script>

</body>
```

OUTPUT:



TASK-2

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width,initial-scale=1.0" />
<title>Task</title>
<script>
async function fetchAndProcessMockData() {
const mockFetch = () =>
new Promise((resolve) =>
setTimeout(() => resolve([{ id: 1, title: "Post 1" }, { id: 2, title: "Post 2" }]), 1000)
);
try {
const data = await mockFetch();
```

```
return data
.filter((item) => item.id % 2 === 0)
.map((item) => ({ id: item.id, title: item.title.toUpperCase() }));
} catch (error) {
console.error("Error:", error);
throw error;
}
}
fetchAndProcessMockData().then(console.log).catch(console.error);


</script>
</head>
</html>
```

OUTPUT:

TASK-3:

```
<!DOCTYPE html>
<html>
<head>
```

```
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width,initial-scale=1.0" />
<title>Task</title>
<script>
async function fetchData(id) {
try {
const mockFetch = (id) =>
new Promise((resolve, reject) => {
setTimeout(() => (id > 0 ? resolve(`Data for ID ${id}`) : reject("Invalid ID")), 1000);
});
const data = await mockFetch(id);
console.log("Fetched data:", data);
return data;
} catch (error) {
console.error("Error:", error);
throw error;
}
}
fetchData(1).catch(console.error);
</script>
</head>
</html>
```

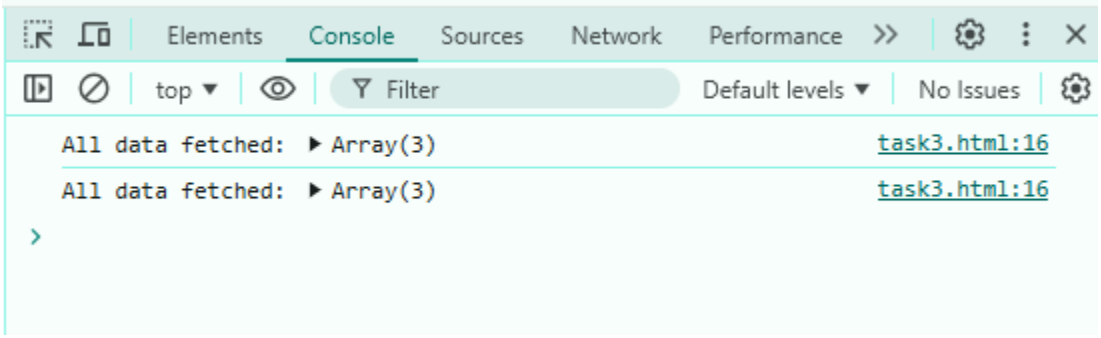
OUTPUT:

TASK-4

```
<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width,initial-scale=1.0" />

<title>Task</title>

<script>

async function fetchData(ids) {

const mockFetch = (id) =>

new Promise((resolve, reject) => {

setTimeout(() => (id > 0 ? resolve(`Data for ID ${id}`) : reject("Invalid ID")), 1000);

});

try {

const fetchPromises = ids.map(id => mockFetch(id));

const results = await Promise.all(fetchPromises);

console.log("All data fetched:", results);

return results;
```

```
} catch (error) {
console.error("Error fetching data:", error);
}
}
fetchData([1, 2, 3]).catch(console.error);
fetchData([1, 4, 3]).catch(console.error)
</script>
</head>
</html>
```

OUTPUT:



TASK-5

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width,initial-scale=1.0" />
<title>Task</title>
<script>
async function processMultipleOperations() {
```

```
const asyncOperation1 = () =>
new Promise((resolve) => setTimeout(() => resolve("Operation 1 complete"), 1000));
const asyncOperation2 = () =>
new Promise((resolve) => setTimeout(() => resolve("Operation 2 complete"), 2000));
const asyncOperation3 = () =>
new Promise((resolve) => setTimeout(() => resolve("Operation 3 complete"), 1500));
try {
const results = await Promise.all([asyncOperation1(), asyncOperation2(), asyncOperation3()]);
console.log("All operations completed:", results);
return results;
} catch (error) {
console.error("Error in operations:", error);
}
}
processMultipleOperations().then((results) => console.log(results));
</script>
</head>
</html>
```

OUTPUT: