3.CLOSURE
TASK-1

```
<!DOCTYPE html>
<html>
 <head>
  <body>
   <script>
    function outerFunction()
    {
     let outervariable = "It is the Outer Function";
     return function innerFunction()
     {
      console.log(outervariable);
     };
    }
    let myFunction = outerFunction();
    myFunction();
   </script>
  </body>
 </head>
</html>
```

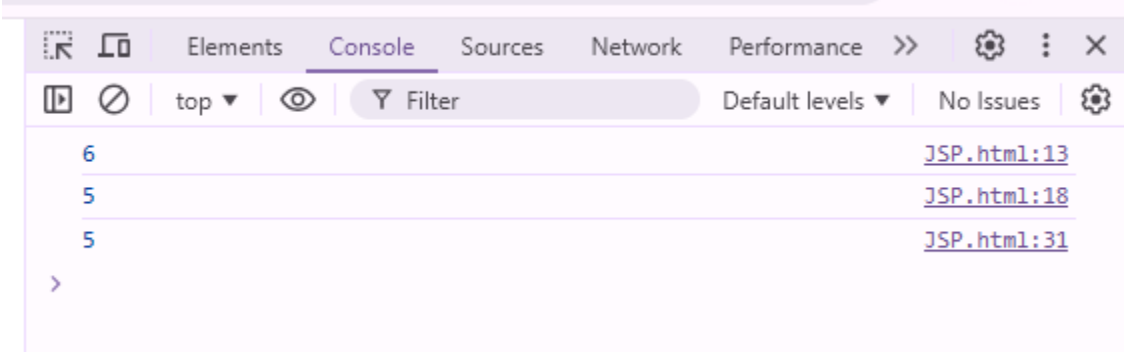
OUTPUT:

TASK-2

```
<!DOCTYPE html>
<html>
 <head>
  <body>
   <script>
    function createCounter()
    {
     let count = 5;
     return {
     increment:function()
     {
      count++;
      console.log(count);
     },
```

```
        decrement : function()
        {
          count--;
    console.log(count);
        },
        getCount : function()
        {
          return count;
        }
       };
       }
       let myCounter = createCounter();
       myCounter.increment();
       myCounter.decrement();
       console.log(myCounter.getCount());
     </script>
   </body>
  </head>
</html>
```

OUTPUT:

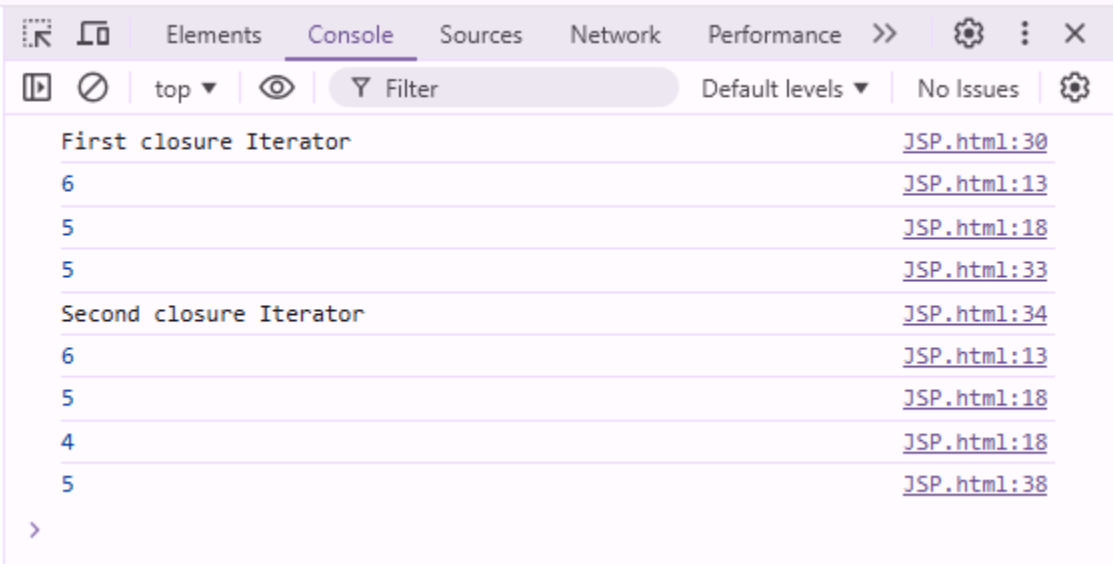


TASK-3

```
<!DOCTYPE html>
<html>
  <head>
    <body>
      <script>
        function createCounter()
        {
          let count = 5;
          return {
          increment:function()
          {
            count++;
            console.log(count);
          },
```

```
      decrement : function()
      {
       count--;
       console.log(count);
      },
      getCount : function()
      {
       return count;
      }
     };
     }
     let myCounter = createCounter();
     let myCounter1 = createCounter();
    console.log("First closure Iterator");
     myCounter.increment();
     myCounter.decrement();
     console.log(myCounter.getCount());
   console.log("Second closure Iterator");
     myCounter1.increment();
     myCounter1.decrement();
     myCounter1.decrement();
     console.log(myCounter.getCount());
   </script>
  </body>
 </head>
</html>
```

OUTPUT:

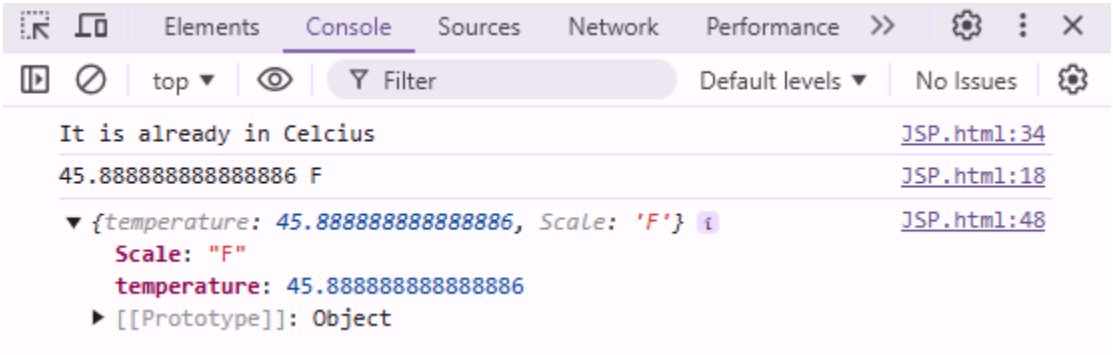


TASK-4

```
<!DOCTYPE html>
<html>
 <head>
  <body>
   <script>
    function temperatureConverter(initialtemp , scale)
    {
     let temperature = initialtemp;
     let Scale = scale;

     return{
      convertToFahrenheit : function()
      {
       if(Scale === 'C')
      {
       temperature = (temperature * 5/9) + 32;
       Scale = 'F';
       console.log(`${temperature} F`);
      }
      else{
       console.log("It is already in Fahrenheit");
      }
     },

      convertToCelcius : function(){
       if(Scale === 'F')
      {
       temperature = (temperature - 32) * 9/5;
       Scale = 'C';
       console.log(`${temperature} C`);
      }
      else
      {
       console.log("It is already in Celcius");
      }
      },
      gettemperatureConverter : function(){
       return {temperature,Scale};
      }

      };
      }


     let temperature = temperatureConverter(25,'C');
```

```
        temperature.convertToCelcius();
        temperature.convertToFahrenheit();
        console.log(temperature.gettemperatureConverter());
    </script>
  </body>
 </head>
</html>
```

OUTPUT:

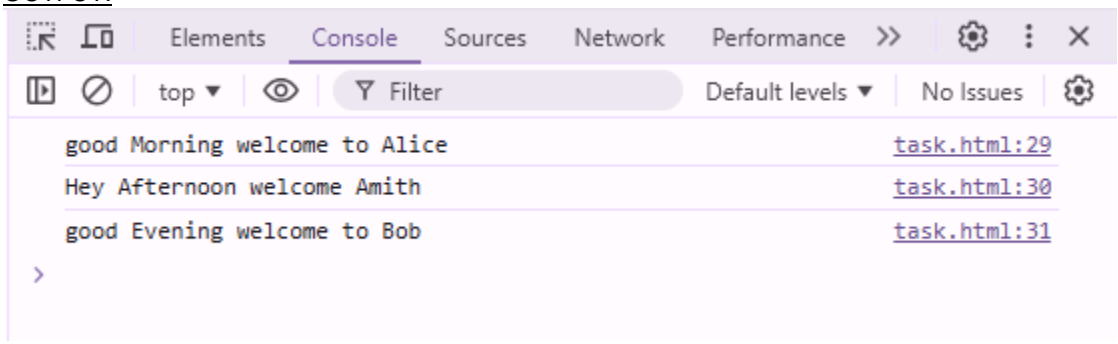


TASK-5

```
<!DOCTYPE html>
<html>
  <head>
    <body>
      <script>
        function createGreeting(greetingType,timeOfDay,name)
        {
          let greeting='';
          return function(){
            if(greetingType === 'formal')
          {
            greeting = (`good ${timeOfDay} welcome to ${name}`);
          }
          else if (greetingType === 'informal')
          {
            greeting = (`Hey ${timeOfDay} welcome ${name}`);
          }
          else{
            greeting = (`good ${timeOfDay} welcome to ${name}`);
          }
          return greeting;
          };
        }

        let morningGreet = createGreeting('formal','Morning','Alice');
        let afternoonGreet = createGreeting('informal','Afternoon','Amith');
```

```
            let eveningGreet = createGreeting('Normal','Evening','Bob');

            console.log(morningGreet());
            console.log(afternoonGreet());
            console.log(eveningGreet());
          </script>
        </body>
    </head>
</html>
```

OUTPUT:

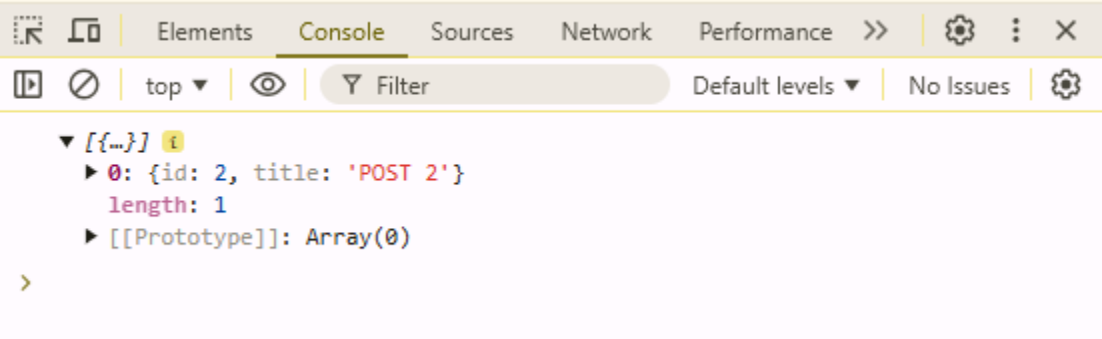


5.ASYNC/WAIT
TASK-2

```
<!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>
```

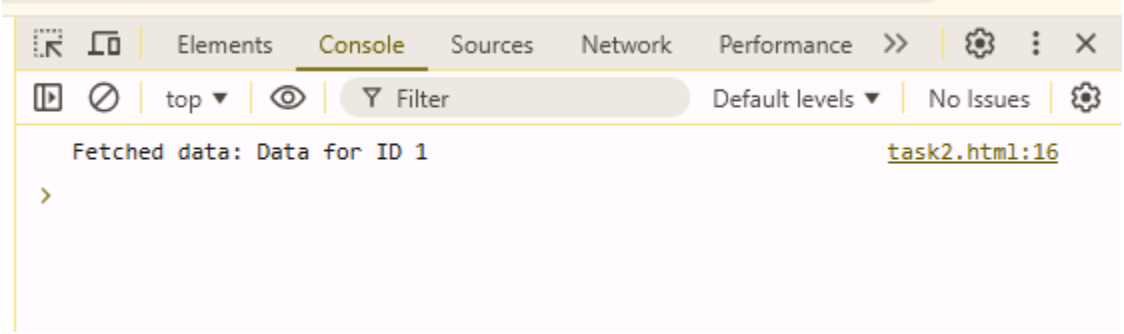
OUTPUT:



TASK-3:

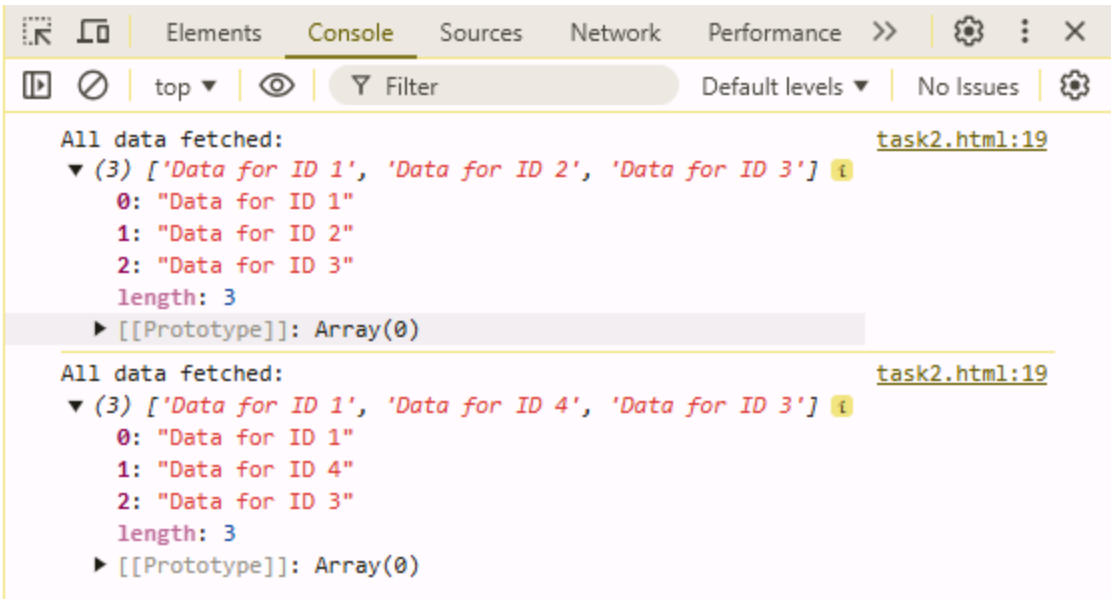
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width,initial-scale=1.0" />
<title>Task</title>
<script>
   async function fetchData(id) {
  try {
   const mockFetch = (id) =>
     new Promise((resolve, reject) => {
       setTimeout(() => (id > 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:



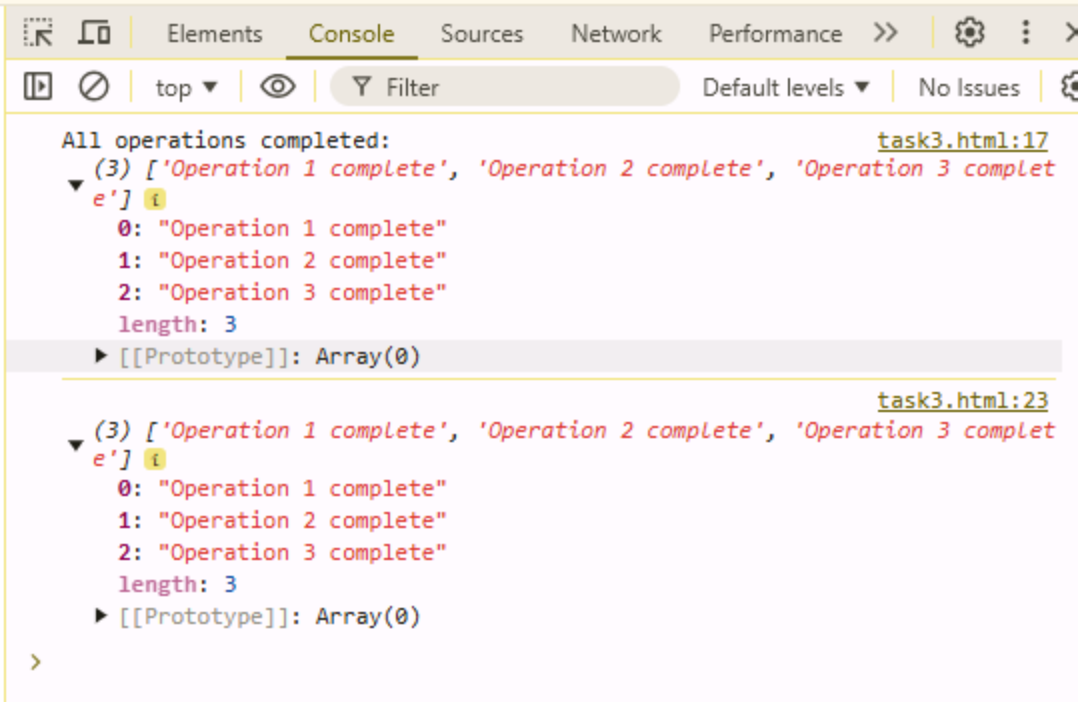
TASK-4

```
<!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>
```



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>
```



6.MODULES,INTRODUCTION IMPORT AND EXPORT
TASK-1

```
<!DOCTYPE html>
<html>
  <head>
    <title>Module Example</title>
  </head>
  <body>
    <script type="module" src = "App.js">
    </script>
  </body>
</html>
```

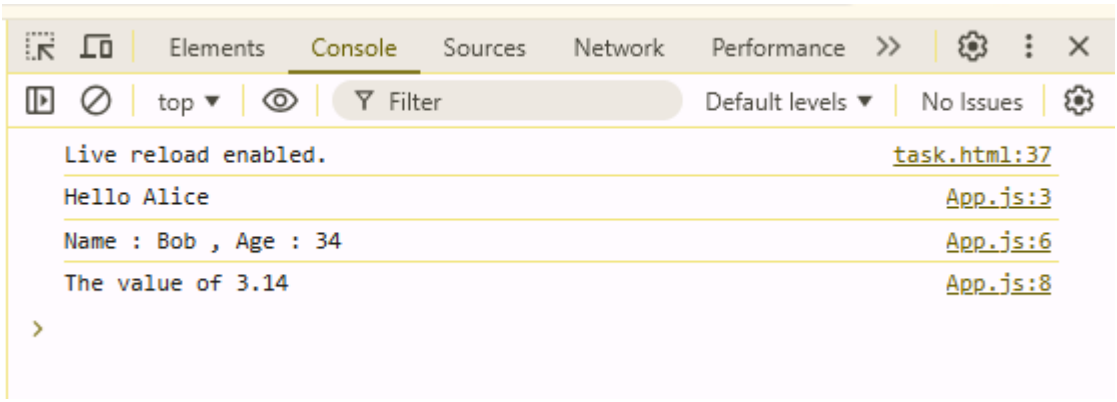
```
export function greet(name)
{
  return `Hello ${name}`;
}
export class Person {
  constructor(name,age){
    this.name = name;
    this.age = age;
  }
  introduce()
  {
    return `Name : ${this.name} , Age : ${this.age}`;
  }
```

```
}
export const pi = 3.14;
```

TASK-2
IMPORT FUNCTION:

```
import {greet,Person,pi} from "./myModule.js";

console.log(greet("Alice"));

const person1 = new Person("Bob","34");
console.log(person1.introduce());

console.log(`The value of ${pi}`);
```

OUTPUT:



TASK-3

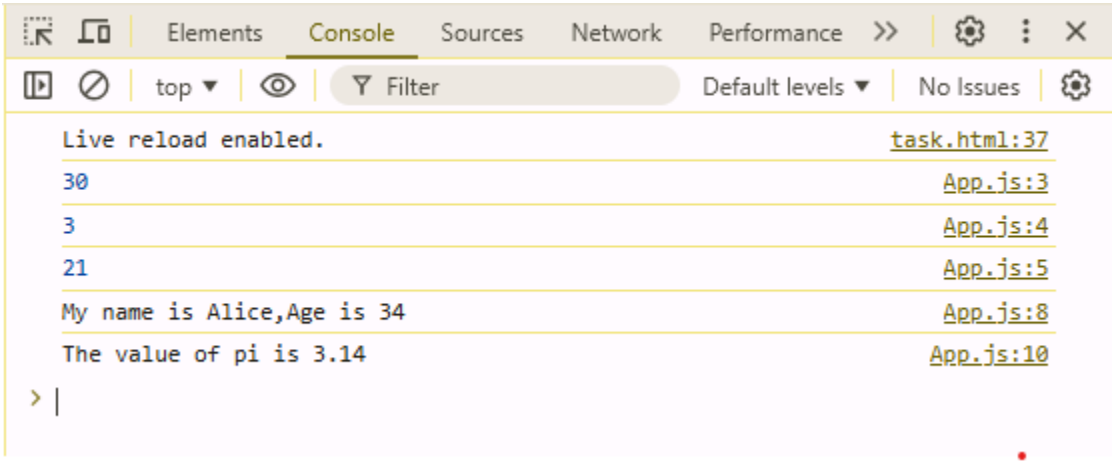
```
<!DOCTYPE html>
<html>
  <head>
    <title>Module Example</title>
  </head>
  <body>
    <script type="module" src = "App.js">
    </script>
  </body>
</html>


export function multiply(a,b)
{
  return a * b;
}
```

```
export function subtract(a,b)
{
  return a - b;
}
export function add(a,b,c)
{
  return a+b+c;
}
export class Person{
  constructor(name,age){
    this.name=name;
    this.age=age;
  }
  introduce()
  {
    return `My name is ${this.name},Age is ${this.age}`;
  }
}
export const pi = 3.14;
```

TASK-4
IMPORT FUNCTION:

```
import {multiply,subtract,add,Person,pi} from "./myModule.js";

console.log(multiply(5,6));
console.log(subtract(7,4));
console.log(add(4,8,9));

const person1 = new Person("Alice",34);
console.log(person1.introduce());

console.log(`The value of pi is ${pi}`);
```

OUTPUT:



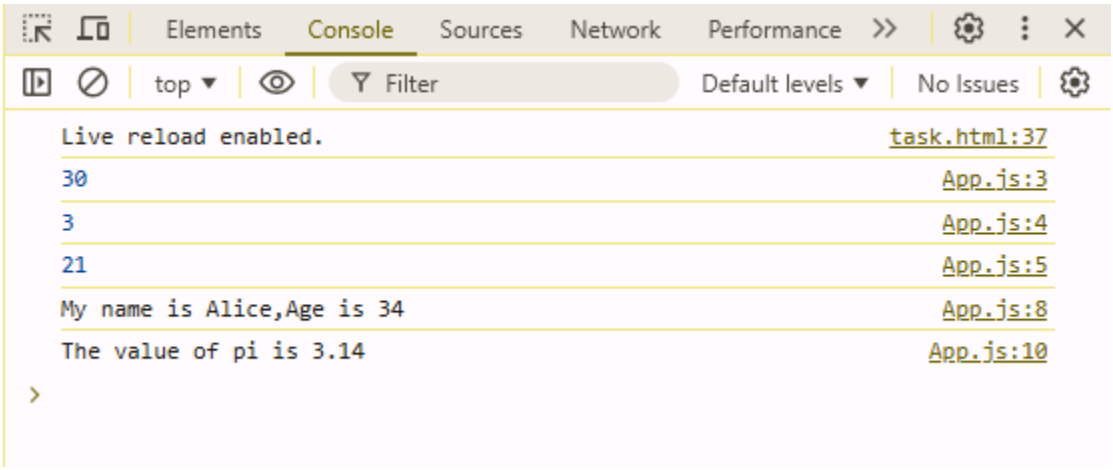
TASK-5

```
<!DOCTYPE html>
<html>
  <head>
    <title>Module Example</title>
  </head>
  <body>
    <script type="module" src = "App.js">
    </script>
  </body>
</html>
```

```
export default function multiply(a,b)
{
  return a * b;
}
export  function subtract(a,b)
{
  return a - b;
}
export function add(a,b,c)
{
  return a+b+c;
}
export class Person{
  constructor(name,age){
    this.name=name;
    this.age=age;
  }
  introduce()
  {
    return `My name is ${this.name},Age is ${this.age}`;
  }
}
export const pi = 3.14;
```

```
import  multiply, {subtract ,add,Person,pi} from "./myModule.js";

console.log(multiply(5,6));
console.log(subtract(7,4));
console.log(add(4,8,9));

const person1 = new Person("Alice",34);
console.log(person1.introduce());

console.log(`The value of pi is ${pi}`);
```

OUTPUT:


Elements
Console
Sources
Network
Performance
top
Filter
Default levels
No Issues
Live reload enabled.
task.html:37
30
App.js:3
3
App.js:4
21
App.js:5
My name is Alice,Age is 34
App.js:8
The value of pi is 3.14
App.js:10