JSON VARIABLE LENGTH ARGUMENTS/SPREAD SYNTAX

```
TASK-1
<html>
<head>
</head>
<body>
<script>
function sum(...arguments)
{
let total = 0;
for(let num of arguments)
{
total+=num;
}
return total;
}
console.log(sum(9,8,6,5));
</script>
</body>
</html>
Output:
```

```
← → C ① File C:/Users/Student.MAT-48.000/Desktop/201... ☆
                                                 ( it's you
               Elements Console Sources
                                         Network >>
               28
                                                      tasks.html:16
Task 2:
<html>
<body>
<script>
function sum(...arguments)
{
let total = 0;
for(let num of arguments)
{
total+=num;
}
return total;
}
const arr = [5,6,4,7]
console.log(sum(...arr));
</script>
</body>
</html>
```

```
① File C:/Users/Student.MAT-48.000/Deskto... ☆
                                                     (6) Verify it's you
    K [0
              Elements
                      Console
                               Sources
                                        Network >>
    Default levels ▼
                                                        1 Issue: 📮 1
       22
                                                         tasks.html:17
TASK-3
<html>
<body>
<script>
function deepclone(obj)
{
return JSON.parse(JSON.stringify(obj));
}
const originalobj = {name:"Ram",details:{age: "24",Dept : "IT"}};
const cloneobj = deepclone(originalobj);
console.log(cloneobj);
</script>
</body>
</html>
OUTPUT:
              ① File C:/Users/Student.MAT-48.000/Deskto... ☆
                  Elements Console Sources Network >>
                  D O top ▼ O Y Filter
                                                 Default levels ▼ 1 Issue: ■ 1 😵
                     ▼ {name: 'Ram', details: {...}} 📵
                                                                tasks.html:10
                      ▶ details: {age: '24', Dept: 'IT'
                       name: "Ram"
                      ▶ [[Prototype]]: Object
```

```
<!DOCTYPE html>
<html>
<body>
<script>
let student =
{
name:"Sam",
age:32,
dept:"CSE"
};
let Employee = {
name:"Sam",
age:45,
salary:76000
};
let obj = {...student,...Employee};
console.log(JSON.stringify(obj));
</script>
</body>
</html>
OUTPUT:
```

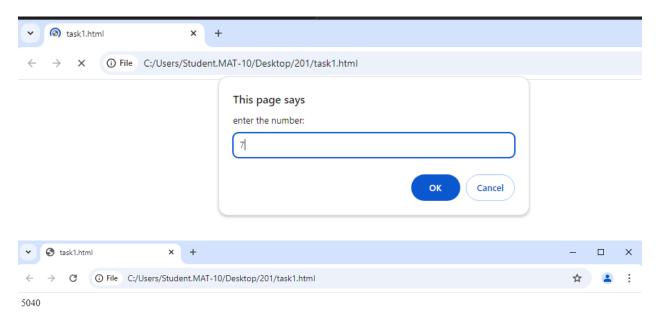
```
\leftarrow \rightarrow G
              ① File C:/Users/Student.MAT-48.000/Deskto... ☆
                                                               (6) Verify it's you
                           Elements
                                   Console
                                           Sources
                                                   Network >>
                  Default levels ▼ 1 Issue: 🗖 1
                     {"name": "Sam", "age": 45, "dept": "CSE", "salary": 76000}
                                                                   tasks.html:16
TASK-5
<!DOCTYPE html>
<html>
<body>
<script>
const obj = {name:"John",Salary:65000,Dept:"ECE",Age:25};
let jsonString = JSON.stringify(obj);
console.log(jsonString);
let parseobj = JSON.parse(jsonString);
console.log(parseobj);
</script>
</body>
</html>
```

```
① File C:/Users/Student.MAT-48.000/Deskto... ☆
                                                             (it's you)
G
          K [0
                                               Network >>
                   Elements
                            Console
                                      Sources
          Default levels ▼
                                                                  1 Issue: 🗏 1
            {"name": "John", "Salary": 65000, "Dept": "ECE", "Age": 25}
                                                                  tasks.html:6
             ▼ {name: 'John', Salary: 65000, Dept: 'ECE', Age: 25} ( tasks.html:8
                Age: 25
                Dept: "ECE"
                Salary: 65000
                name: "John"
               ▶ [[Prototype]]: Object
```

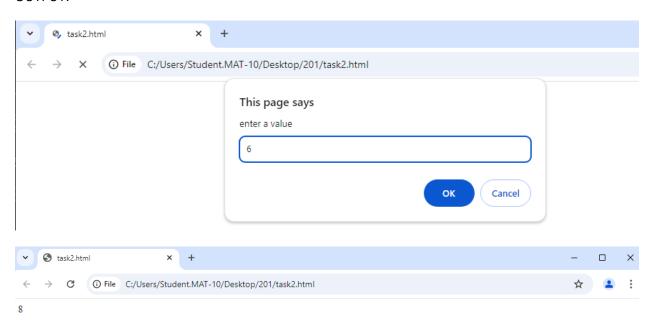
RECURSION

TASK 1:

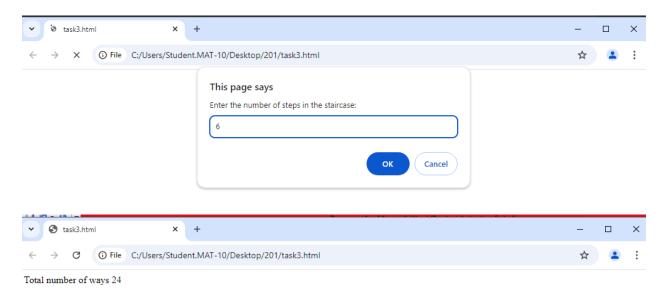
OUTPUT:



TASK 2:



TASK 3:



TASK 4:

TASK 5:

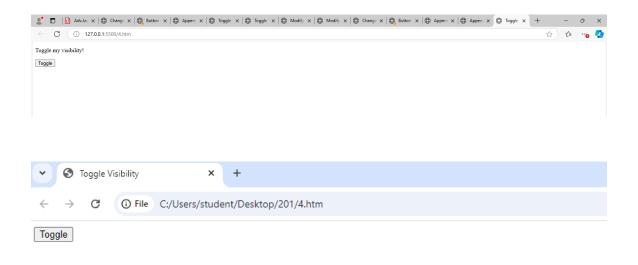
```
∨ 3 task5.html
                                                                                                               ← → C ① File C:/Users/Student.MAT-10/Desktop/201/task5.html
                                                        \square \square Elements Console Sources Network \gg
                                                                                                          □1 🕸 : ×
                                                        Move disk 1 from A to C
                                                                                                           task5.html:6
                                                          Move disk 2 from A to B
                                                                                                          task5.html:10
                                                          Move disk 1 from C to B
                                                                                                           task5.html:6
                                                          Move disk 3 from A to C
                                                                                                          task5.html:10
                                                          Move disk 1 from B to A
                                                                                                           task5.html:6
                                                          Move disk 2 from B to C
                                                                                                          task5.html:10
                                                          Move disk 1 from A to C
                                                                                                           task5.html:6
```

```
TASK1:
<!DOCTYPE html>
<html>
<head>
 <title>Change Content</title>
</head>
<body>
 Hello, World!
 <script>
   document.getElementById("my").innerHTML = "Hello, JavaScript!";
 </script>
</body>
</html>
OUTPUT:
G (i) 127.0.0.1:5500/1.htm
Hello, JavaScript!
TASK2:
<!DOCTYPE html>
<html>
<head>
 <title>Button Click Event</title>
</head>
<body>
 <button id="myButton">Click Me!</button>
```

```
<script>
     document.getElementById("myButton").addEventListener("click", function() {
       alert("Button was clicked!");
     });
  </script>
</body>
</html>
🙎 🗂 🗎 Adv.Java.pdf x | 🖨 Change Corr x | 📵 Button Click x | 🖨 Append Ber x | 🖨 Toggle Visib x | 🖨 Toggle Visib x | 🖨 Modify Attr x | 🖨 Modify Attr x | 🖨 Modify Attr x | 🖨 Change Corr x | 🐯 Button Click x +
    C (1) 127.0.0.1:5500/2.htm
Click Mel
                                           127.0.0.1:5500 says
TASK3:
<!DOCTYPE html>
<html>
<head>
  <title>Append Element</title>
</head>
<body>
  <div id="container"></div>
  <script>
     var newElement = document.createElement("p");
     newElement.innerHTML = "This is a new paragraph.";
     document.getElementById("container").appendChild(newElement);
  </script>
</body>
</html>
OUTPUT:
```

```
💒 🛅 🗎 Adv.lev: X | 🖨 Change: X | 📵 Button C X | 🖨 Append: X | ④ Toggle V X | ⊕ Toggle V X | ⊕ Modify = X | ⊕ Modify = X | ⊕ Change: X | ⊕ Eutton C X | ⊕ Append: X |
            C (1) 127.0.0.1:5500/3.htm
  This is a new paragraph.
TASK4:
<!DOCTYPE html>
<html>
<head>
         <title>Toggle Visibility</title>
</head>
<body>
         Toggle my visibility!
          <button id="toggleButton">Toggle/button>
          <script>
                   document.getElementById("toggleButton").addEventListener("click", function() {
                            var paragraph = document.getElementById("toggleParagraph");
                            if (paragraph.style.display === "none") {
                                      paragraph.style.display = "block";
                            } else {
                                      paragraph.style.display = "none";
                            }
                  });
         </script>
</body>
```

</html>



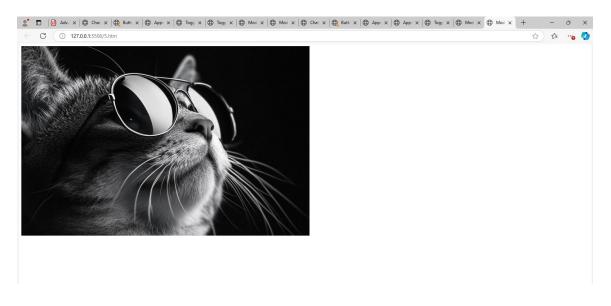
```
TASK5:

<!DOCTYPE html>
<html>
<head>
    <title>Modify Attributes</title>
</head>
<body>
    <img id="mylmage" src="image.jpg" alt="Initial Image">
         <script>
            var image = document.getElementByld("mylmage");
            console.log(image.src);
            image.src = "newlmage.jpg";
            console.log(image.alt);
            image.alt = "Updated Image";

</script>
```

</body>

</html>



```
ASYNC/AWAIT
TASK 1:
<html>
  <body>
    <script>
      async function getUserData(userId) {
  const users = {
   1: { name: 'Alice', age: 25 },
   2: { name: 'Bob', age: 30 },
  };
  const getUser = new Promise((resolve, reject) => {
   setTimeout(() => {
    if (users[userId]) {
     resolve(users[userId]);
    } else {
     reject('User not found');
    }
   }, 1000);
  });
  try {
   const user = await getUser;
   console.log('User data:', user);
```

} catch (error) {

```
console.error('Error:', error);
 }
}
getUserData(1);
    </script>
  </body>
</html>
OUTPUT:
   K [0
              Elements
                                              Network >>
                         Console
                                    Sources
   ▼ Filter
                                                                      1 Issue: 🗷 1 🛛 🕸
                                                      Default levels ▼
                                                                      task1.html:22
      User data: ▼ Object 1
                      age: 25
                      name: "Alice"
                    ▶ [[Prototype]]: Object
TASK 2:
<html>
  <body>
    <script>
async function fetchAndProcessData() {
const apiUrl = 'https://jsonplaceholder.typicode.com/users';
try {
  const response = await fetch(apiUrl);
 if (!response.ok) {
  throw new Error(`HTTP error! Status: ${response.status}`);
```

```
}
const data = await response.json();
console.log('Processing fetched data...');
data.forEach(user => {
  console.log(`User Name: ${user.name}`);
});
}
catch (error) {
  console.error('Error fetching or processing data:', error);
}

fetchAndProcessData();

  </script>
  </body>

</html>
OUTPUT:
```

```
Elements
                                                                    □1 🛞 🗄 ×
                                             Network >>
                        Console
                                   Sources
  I top ▼    ▼ Filter
                                                                   1 Issue: 🗷 1 🔞
                                                     Default levels ▼
     Processing fetched data...
                                                                    task2.html:13
     User Name: Leanne Graham
                                                                    task2.html:15
     User Name: Ervin Howell
                                                                    task2.html:15
     User Name: Clementine Bauch
                                                                    task2.html:15
     User Name: Patricia Lebsack
                                                                    task2.html:15
     User Name: Chelsey Dietrich
                                                                    task2.html:15
     User Name: Mrs. Dennis Schulist
                                                                    task2.html:15
     User Name: Kurtis Weissnat
                                                                    task2.html:15
     User Name: Nicholas Runolfsdottir V
                                                                    task2.html:15
     User Name: Glenna Reichert
                                                                    task2.html:15
     User Name: Clementina DuBuque
                                                                    task2.html:15
TASK 3:
<html>
 <body>
    <script>
   async function getUserInfo(userId) {
try {
 const response = await fetch(`https://jsonplaceholder.typicode.com/users/1`);
 if (!response.ok) {
  throw new Error(`HTTP error! Status: ${response.status}`);
  }
 const user = await response.json();
 if (!user | | !user.name | | !user.email) {
```

throw new Error('Invalid user data received');

}

```
return user;
} catch (error) {
 console.error('Error getting user info:', error.message);
  return null;
}
   }
getUserInfo(1)
 .then(user => {
 if (user) {
   console.log('User Info:', user);
  } else {
   console.log('Failed to get user info');
 }
})
 .catch(error => {
 console.log('Unexpected error:', error);
});
    </script>
  </body>
</html>
OUTPUT:
```

```
K [0
             Elements
                         Console
                                              Network >>
                                   Sources
  Default levels ▼
                                                                       1 Issue: 🖃 1
            top ▼
                         ▼ Filter
     User Info:
                                                                       task3.html:24
     ▼ Object 🗊
       ▶ address: {street: 'Kulas Light', suite: 'Apt. 556', city: 'Gwenborough'
        company: {name: 'Romaguera-Crona', catchPhrase: 'Multi-layered client-s
          email: "Sincere@april.biz"
          name: "Leanne Graham"
          phone: "1-770-736-8031 x56442"
         username: "Bret"
         website: "hildegard.org"
         [[Prototype]]: Object
TASK 4:
<html>
  <body>
    <script>
async function fetchUser(userId) {
const response = await fetch(`https://jsonplaceholder.typicode.com/users/${userId}`);
if (!response.ok) throw new Error(`User with ID ${userId} not found`);
return response.json();
}
async function fetchUserPosts(userId) {
const response = await fetch(`https://jsonplaceholder.typicode.com/posts?userId=${userId}`);
if (!response.ok) throw new Error(`Posts for user with ID ${userId} not found`);
return response.json();
}
async function fetchUserAndPosts(userId) {
try {
  const [user, posts] = await Promise.all([
   fetchUser(userId),
   fetchUserPosts(userId),
```

```
]);
console.log('User Info:', user);
console.log('User Posts:', posts);
} catch (error) {
  console.error('Error:', error.message);
}
}
fetchUserAndPosts(1);

  </script>
  </body>
</html>
OUTPUT:
```

```
K [0
             Elements
                        Console
                                  Sources
                                            Network
                                                      >>
            top ▼
                    0
                          ▼ Filter
                                                    Default levels ▼
                                                                     1 Issue: = 1
     User Info:
                                                                     task4.html:21
     ▼ Object 1
       ▶ address: {street: 'Kulas Light', suite: 'Apt. 556', city: 'Gwenborough',
       ▶ company: {name: 'Romaguera-Crona', catchPhrase: 'Multi-layered client-se
         email: "Sincere@april.biz"
         id: 1
         name: "Leanne Graham"
         phone: "1-770-736-8031 x56442"
         username: "Bret"
         website: "hildegard.org"
       ▶ [[Prototype]]: Object
     User Posts:
                                                                    task4.html:22
     ▼ Array(10) 1
       ▶ 0: {userId: 1, id: 1, title: 'sunt aut facere repellat provident occaec
       ▶ 1: {userId: 1, id: 2, title: 'qui est esse', body: 'est rerum tempore v
       ▶ 2: {userId: 1, id: 3, title: 'ea molestias quasi exercitationem repella
       ▶ 3: {userId: 1, id: 4, title: 'eum et est occaecati', body: 'ullam et sa
       ▶ 4: {userId: 1, id: 5, title: 'nesciunt quas odio', body: 'repudiandae v
       ▶ 5: {userId: 1, id: 6, title: 'dolorem eum magni eos aperiam quia', body
       ▶ 6: {userId: 1, id: 7, title: 'magnam facilis autem', body: 'dolore plac
       ▶ 7: {userId: 1, id: 8, title: 'dolorem dolore est ipsam', body: 'digniss
       ▶ 8: {userId: 1, id: 9, title: 'nesciunt iure omnis dolorem tempora et ac
       ▶ 9: {userId: 1, id: 10, title: 'optio molestias id quia eum', body: 'quo
         length: 10
       ► [[Prototype]]: Array(0)
TASK 5:
<html>
  <body>
    <script>
async function fetchUser(userId) {
const response = await new Promise((resolve) =>
 setTimeout(() => resolve({ id: userId, name: 'John Doe' }), 2000)
);
return response;
}
async function fetchUserPosts(userId) {
```

```
const response = await new Promise((resolve) =>
  setTimeout(() => resolve([{ postId: 1, title: 'First Post' }, { postId: 2, title: 'Second Post' }]), 1500)
 );
 return response;
}
async function fetchUserComments(postId) {
 const response = await new Promise((resolve) =>
  setTimeout(() => resolve([{ commentId: 1, body: 'Great post!' }]), 1000)
 );
 return response;
}
async function fetchData(userId) {
 try {
  const [user, posts, comments] = await Promise.all([
   fetchUser(userId),
   fetchUserPosts(userId),
   fetchUserComments(1),
  ]);
  console.log('User Data:', user);
  console.log('User Posts:', posts);
  console.log('User Comments:', comments);
 } catch (error) {
  console.error('Error fetching data:', error.message);
 }
}
fetchData(1);
```

```
</script>
</body>
</html>
OUTPUT:
```

```
□1 🕸 : ×
κ []
        Elements
                   Console
                            Sources
                                     Network >>
Default levels ▼ 1 Issue: 📮 1 🛞
  User Data: ▼ Object 🗓
                                                          task5.html:30
                name: "John Doe"
              ▶ [[Prototype]]: Object
                                                          task5.html:31
  User Posts: ▼ Array(2) 1
               ▶ 0: {postId: 1, title: 'First Post'}
               ▶ 1: {postId: 2, title: 'Second Post'}
                length: 2
               ► [[Prototype]]: Array(0)
  User Comments: ▼ Array(1) 1
                                                          task5.html:32
                  ▶ 0: {commentId: 1, body: 'Great post!'}
                   length: 1
                  ▶ [[Prototype]]: Array(0)
>
```

```
TASK 1:
<html>
<script>
const mypromise=new Promise((resolve)=>{
setTimeout(()=>
resolve("welcome");
},1000)
});
mypromise.then((value)=>{
document.write(value);
})
</script>
</html>
OUTPUT:
                                                                                             ✓ ⑤ 1.html
                  C:/Users/Student.MAT-09/Desktop/201/1.html

    Google Lens 
    ☆

 welcome
TASK2:
<html>
<script>
// Fetch data from an API (e.g., JSONPlaceholder)
function fetchData(url) {
```

```
return fetch(url)
.then(response => {
if (!response.ok) {
throw new Error('Network response was not ok');
}
return response.json(); // Parse the JSON data
});
}
// Process the fetched data (e.g., extract the name of the first user)
function processData(data) {
return new Promise((resolve, reject) => {
if (data && data.length > 0) {
// For example, return the name of the first user
resolve(`First user's name is: ${data[0].name}`);
reject('No data found to process.');
}
});
```

```
// Use the fetchData function to get data, then chain the processData function
const apiUrl = 'https://jsonplaceholder.typicode.com/users';
fetchData(apiUrl)
.then((data) => {
// Chain another promise to process the data
return processData(data);
})
.then((processedData) => {
console.log(processedData); // Output the processed data
})
.catch((error) => {
console.error('Error:', error); // Handle any errors
});
</script>
</html>
OUTPUT:
                                                                                   html:33.2
TASK3:
<html>
<script>
function random(){
return new Promise((resolve,reject)=>{
let random1=Math.random();
if(random1>0.5){
resolve("success");
}
else{
reject("failure");
}
});
random()
.then((value)=>{
document.write(value);
})
.catch((error)=>{
document.write(error);
})
</script>
</html>
OUTPUT:
```



```
TASK4:
<html>
<script>
// URLs to fetch data from
const url1 = 'https://jsonplaceholder.typicode.com/users';
const url2 = 'https://jsonplaceholder.typicode.com/posts';
const url3 = 'https://jsonplaceholder.typicode.com/albums';
// Create an array of fetch promises
const fetchPromises = [
fetch(url1),
fetch(url2),
fetch(url3)
];
// Use Promise.all to fetch all resources in parallel
Promise.all(fetchPromises)
.then(responses => {
// All fetch requests are resolved, now process the responses
return Promise.all(responses.map(response => response.json())); // Convert all responses to JSON
})
.then(data => {
// Process the data from all the responses
const users = data[0];
const posts = data[1];
const albums = data[2];
console.log('Users:', users); // Output the users
console.log('Posts:', posts); // Output the posts
console.log('Albums:', albums); // Output the albums
})
.catch(error => {
// Handle any error from the fetch requests or data processing
console.error('Error:', error);
});
</script>
</html>
OUTPUT:
```

```
        PROBLEMS
        OUTPUT
        DEBUG CONSOLE
        TERMINAL
        PORTS
        Filter (e.g. text. lexclude, ...)
        Open 4.html
        ✓ 蓋 ^ X

        > Users: (100) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {.
```

```
TASK5:
<html>
<script>
// Simulate an asynchronous task with a delay
function asyncTask(message, delay) {
return new Promise((resolve, reject) => {
setTimeout(() => {
console.log(message);
resolve(message);
}, delay);
});
}
// Chain promises to perform tasks in sequence
asyncTask("Task 1: Starting", 1000)
.then((result1) => {
// Task 1 is done, now run Task 2
return asyncTask("Task 2: Starting", 1500);
})
.then((result2) => {
// Task 2 is done, now run Task 3
return asyncTask("Task 3: Starting", 2000);
.then((result3) => {
// Task 3 is done, now run Task 4
return asyncTask("Task 4: Finished", 1000);
})
.catch((error) => {
console.error("Error:", error);
});
</script>
</html>
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter (e.g. text, lexclude, ... Open 5.html V X ^ X

Task 1: Starting
Task 2: Starting
Task 3: Starting
Task 4: Finished

Task 4: Finished
```

```
CLOSURE:
TASK 1:
<html>
  <body>
    <script>
function outer() {
  let message = "Hello from the outer function!";
  return function inner() {
    console.log(message);
}
const closureFunction = outer();
closureFunction();
</script>
</body>
</html>
OUTPUT:
Elements
                Console
                         Sources
                                 Network >>
Default levels ▼ 1 Issue: ■ 1
  Hello from the outer function!
                                                    task1.html:8
  Live reload enabled.
                                                    task1.html:42
TASK 2:
<html>
  <body>
```

```
<script>
    function createCounter() {
    let count = 0;
    return {
    increment: function() {
      count++;
    },
    getCount: function() {
      console.log(count);
   }
  };
}
const counter = createCounter();
counter.increment();
counter.increment();
counter.getCount();
  </script>
</body>
</html>
OUTPUT:
K [0
                                                     >>
            Elements
                       Console
                                  Sources
                                            Network
Default levels ▼
                                                                   1 Issue: ■ 1 🛞
    2
                                                                     TASK2.HTML:11
    Live reload enabled.
                                                                     TASK2.HTML:48
```

```
TASK 3:
<html>
  <body>
  <script>
    function createCounter() {
  let count = 0;
  return {
    increment: function() {
      count++;
    },
    getCount: function() {
      console.log(count);
    }
 };
}
const counter1 = createCounter();
const counter2 = createCounter();
counter1.increment();
counter1.getCount();
counter2.increment();
counter2.getCount();
counter1.increment();
counter1.getCount();
  </script>
  </body>
</html>
OUTPUT:
```

```
(€) : ×
K [0
                                      Network >>
         Elements
                   Console
                            Sources
top ▼ 🔘
                   ▼ Filter
                                            Default levels ▼
                                                          1 Issue: 🗏 1 🔞
                                                           TASK3.html:11
  2
                                                           TASK3.html:11
  Live reload enabled.
                                                           TASK3.html:52
```

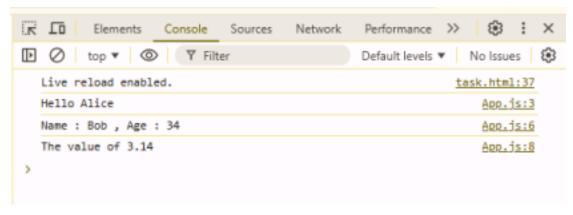
```
TASK 4:
<html>
  <body>
    <script>
    function createPerson(name, age) {
    let _name = name;
    let _age = age;
    return {
     getName: function() {
      return _name;
     },
     getAge: function() {
      return _age;
     },
     setAge: function(newAge) {
      if (newAge > _age) {
        _age = newAge;
      }
     }
    };
   }
```

```
const person = createPerson('Alice', 30);
console.log(person.getName());
console.log(person.getAge());
person.setAge(31);
console.log(person.getAge());
    </script>
  </body>
</html>
OUTPUT:
K [0
            Elements
                        Console
                                             Network
                                                      >>
                                  Sources
1 Issue: 🗷 1 🔞
           top ▼ 🔘
                        ▼ Filter
                                                     Default levels ▼
    Alice
                                                                      task4.html:22
    30
                                                                      task4.html:23
    31
    Live reload enabled.
                                                                      task4.html:55
TASK 5:
<html>
  <body>
    <script>
    function functionFactory(type) {
    if (type === 'greet') {
     return function(name) {
      console.log(`Hello, ${name}!`);
     };
    } else if (type === 'farewell') {
     return function(name) {
      console.log(`Goodbye, ${name}!`);
    };
```

```
} else {
     return function() {
      console.log('Unknown type');
     };
    }
   }
const greetFunction = functionFactory('greet');
greetFunction('Alice');
const farewellFunction = functionFactory('farewell');
farewellFunction('Bob');
const unknownFunction = functionFactory('unknown');
unknownFunction();
    </script>
  </body>
</html>
OUTPUT:
 K [0
             Elements
                         Console
                                              Network
                                   Sources
                                                                       1 Issue: 🗷 1 🏻 🕸
                         ▼ Filter
            top ▼ 🔘
                                                      Default levels ▼
    Hello, Alice!
                                                                      task 5.html:7
    Goodbye, Bob!
                                                                     task 5.html:11
    Unknown type
                                                                     task 5.html:15
    Live reload enabled.
                                                                     task 5.html:54
  > |
```

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;

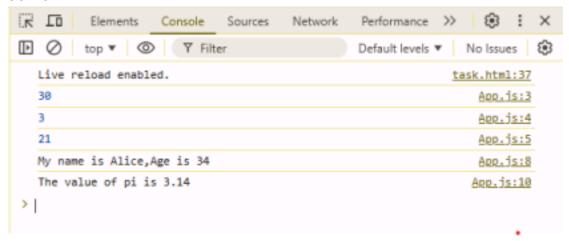
TASK-4
IMPORT FUNCTION:

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}`);
```



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:
  K [0
              Elements
                          Console
                                    Sources
                                               Network
                                                          Performance
           top ▼ ◎ ▼ Filter
                                                          Default levels ▼
     Live reload enabled.
```

30

3

21

My name is Alice, Age is 34

The value of pi is 3.14

No Issues 🔞

App.is:3

App.js:4

App.js:5

App.js:8

App. js: 10

task.html:37