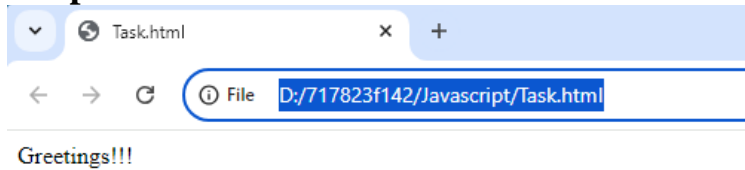


Task-16:**Code:**

```
<!DOCTYPE html>
<head>
  <title>Task</title>
</head>
<body>
  <script>
    let mypromise= new Promise(function(resolve,reject){
      setTimeout(()=>{resolve("Greetings!!!")},3000)
    })
    mypromise.then(
      function(value){
        document.write(value)
      }
    );
  </script>
</body>
</html>
```

Output:**Task-17:****Code:**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>GitHub User Info</title>
</head>
<body>
  <script>
    fetch('https://api.github.com/users/iliakan')
      .then(response => {
        if (!response.ok) return Promise.reject('Failed to fetch data');
        return response.json();
      })
      .then(data => {
        const processedData = {
          username: data.login,
          name: data.name,
          publicRepos: data.public_repos
        }
      })
  </script>
</body>
</html>
```

```
    };
    console.log(processedData);
  })
  .catch(error => console.error('Error:', error));
</script>
</body>
</html>
```

Output:



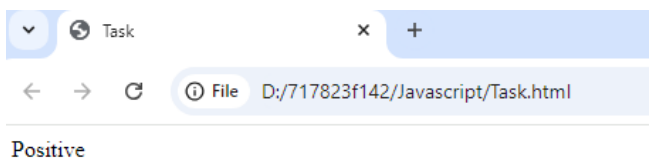
```
> {username: 'iliakan', name: 'Ilya Kantor', publicRepos: 149}
```

Task-18:

Code:

```
<!DOCTYPE html>
<head>
  <title>Task</title>
</head>
<body>
  <script>
    let mypromise= new Promise(function(resolve,reject){
      let x=Number(prompt("enter a number"));
      if(x>=0){
        resolve("Positive")
      }
      else{
        reject("negative")
      }
    }
  );
  mypromise.then(
    function(value){
      document.write(value)
    },
    function(e){
      document.write(e)
    }
  );
</script>
</body>
</html>
```

Output:

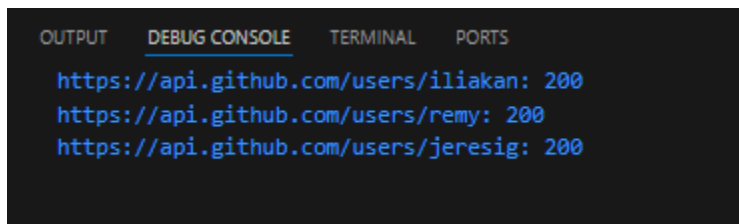


Task-19:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JavaScript</title>
</head>
<body>
  <script>
    let urls = [
      'https://api.github.com/users/iliakan',
      'https://api.github.com/users/remy',
      'https://api.github.com/users/jeresig'
    ];
    let requests = urls.map(url => fetch(url));
    Promise.all(requests)
      .then(responses => responses.forEach(
        response => console.log(` ${response.url}: ${response.status}`)
      ));
  </script>
</body>
</html>
```

Output:



```
OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
https://api.github.com/users/iliakan: 200
https://api.github.com/users/remy: 200
https://api.github.com/users/jeresig: 200
```

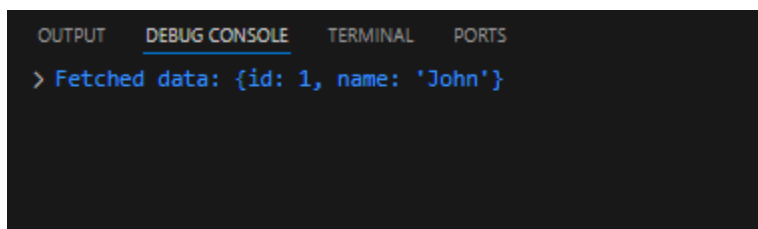
Task-20:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Promise Chain Example</title>
</head>
<body>
  <script>
    function fetchData() {
      return new Promise((resolve) => {
        setTimeout(() => {
```

```
        const data = { id: 1, name: "John" };
        console.log('Fetched data:', data);
        resolve(data);
      }, 1000);
    });
  }
  function processData(data) {
    return new Promise((resolve) => {
      setTimeout(() => {
        data.name = data.name.toUpperCase();
        console.log('Processed data:', data);
        resolve(data);
      }, 1000);
    });
  }
  function logResult(data) {
    return new Promise((resolve) => {
      setTimeout(() => {
        console.log('Final result:', data);
        resolve('Process complete');
      }, 1000);
    });
  }
  fetchData()
    .then(data => processData(data))
    .then(processedData => logResult(processedData))
    .then(result => console.log(result))
    .catch(error => console.error('Error:', error));
</script>
</body>
</html>
```

Output:



Task-21:

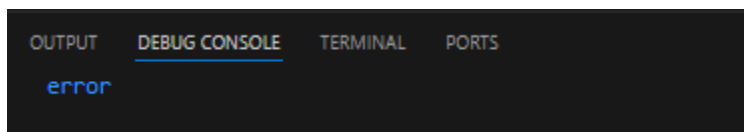
Code:

```
<!DOCTYPE html>
<head>
  <title>TASK</title>
</head>
<body>
  <script>
    var s
    async function fun(a){
```

```

    var promise= new Promise((resolve, reject) => {
        if(a==0){
            setTimeout(() => {
                resolve('Success!');
            }, 1000);}
        else{
            reject("error")
        }
    });try{
        var s= await promise
        document.write(s)
    }catch(error){
        console.log(error);
    }
}
fun(10)
</script>
</body>
</html>

```



Task-22:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JavaScript</title>
</head>
<body>
  <script>
    async function fetchData(url) {
      try {
        const response = await fetch(url);
        if (!response.ok) {
          throw new Error('Network response was not ok');
        }
        const data = await response.json();
        return data;
      } catch (error) {
        console.error('There was a problem with the fetch operation:', error);
      }
    }
    function processData(data) {
      if (data && typeof data === 'object') {
        for (const [key, value] of Object.entries(data)) {
          console.log(`${key}: ${value}`);
        }
      }
    }
  </script>

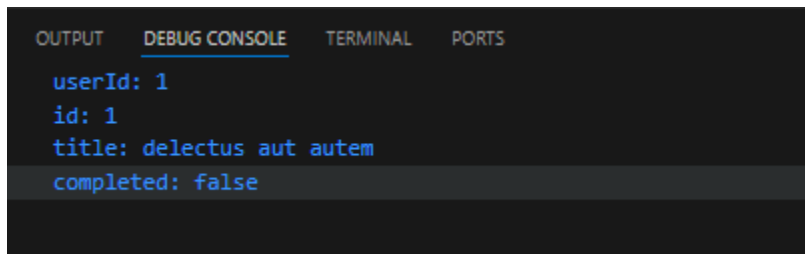
```

```
    }
  } else {
    console.log('Received non-object data:', data);
  }
}

async function main() {
  const url = 'https://jsonplaceholder.typicode.com/todos/1';
  const data = await fetchData(url);
  if (data) {
    processData(data);
  }
}
main();

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

Output:



Task-23:

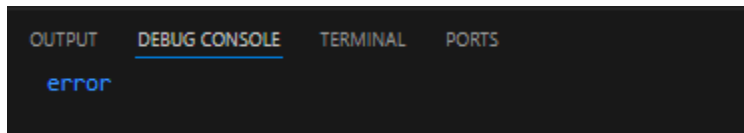
Code:

```
<!DOCTYPE html>
<head>
  <title>TASK</title>
</head>
<body>
  <script>
    var s
    async function fun(a){
      var promise= new Promise((resolve, reject) => {
        if(a==0){
          setTimeout(() => {
            resolve('Success!');
          }, 1000);}
        else{
          reject("error")
        }
      });try{
        var s= await promise
        document.write(s)
      }catch(error){
        console.log(error);
      }
    }
  </script>

```

```
}  
fun(10)  
</script>  
</body>  
</html>
```

Output:



Task-24

Code:

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>JavaScript</title>  
</head>  
<body>  
  <script>  
    async function fetchData(url) {  
      const response = await fetch(url);  
      if (!response.ok) {  
        throw new Error('Network response was not ok');  
      }  
      return response.json();  
    }  
  
    async function main() {  
      const urls = [  
        'https://jsonplaceholder.typicode.com/todos/1',  
        'https://jsonplaceholder.typicode.com/todos/2',  
        'https://jsonplaceholder.typicode.com/todos/3'  
      ];  
  
      try {  
        const results = await Promise.all(urls.map(url => fetchData(url)));  
        results.forEach((data, index) => {  
          console.log(`Data from URL ${urls[index]}:`);  
          console.log(data);  
        });  
      } catch (error) {  
        console.error('There was an error:', error);  
      }  
    }  
  
    main();  
  
  </script>  
</body>  
</html>
```

Output:

```

OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
Data from URL https://jsonplaceholder.typicode.com/todos/1:
> {userId: 1, id: 1, title: 'delectus aut autem', completed: false}
Data from URL https://jsonplaceholder.typicode.com/todos/2:
> {userId: 1, id: 2, title: 'quis ut nam facilis et officia qui', completed: false}
Data from URL https://jsonplaceholder.typicode.com/todos/3:
> {userId: 1, id: 3, title: 'fugiat veniam minus', completed: false}
  
```

Task-25:

Code:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Current Date and Time</title>
</head>
<body>
  <script>
    placeorder=(food)=>{
      return new Promise((resolve)=>{
        setTimeout(()=>{
          document.write(`${food} order placed`+"<BR>");
          resolve(food);
        },1000)
      })
    }
    preparefood=(food)=>{
      return new Promise((resolve)=>{
        setTimeout(()=>{
          document.write(`${food} Food prepared`+"<BR>");
          resolve(food);
        },10000);
      })
    }
    deliver=(food)=>{
      return new Promise((resove)=>{
        setTimeout(()=>{
          document.write(`${food} deleiverd`+"<BR>");
          resove(food);
        },3000)
      })
    }
    async function Orderfood(fooditem) {
      const order=await placeorder(fooditem);
      const preparedfood=await preparefood(order);
      const delivered=await deliver(preparedfood)
    }
    var food_item=prompt("Enter the Food");
  
```



```
    Orderfood(food_item);  
</script>  
</body>  
</html>
```

Output:

