async Function:

The async keyword is used to declare an asynchronous function, which always returns a promise.

```
async function asyncFunction() {
  return 'Async function completed';
}

asyncFunction().then((result) => {
  console.log(result);
});
```

The **await** keyword can only be used inside an **async** function. It pauses the execution of the async function until the promise is settled (either resolved or rejected), and then resumes the execution with the resolved value.

```
async function fetchData () {
    try {
        const response = await fetch ('https://api.example.com/data')
        const data = await response.json()
        return data
    }
    catch (error) {
        console.error('Error fetching data:', error);
        throw error;
    }
}

fetchData().then(result => console.log(result)).catch(error => console.error(error))
```

Sequential and Parallel Execution:

You can use async/await to perform asynchronous operations sequentially or in parallel. For sequential execution, you can use multiple await statements. For parallel execution, you can use Promise.all() with await.

```
async function fetchData() {
  const response1 = await fetch('https://api.example.com/data1');
  const response2 = await fetch('https://api.example.com/data2');

  const data1 = await response1.json();
  const data2 = await response2.json();

  return [data1, data2];
}

const fetchData = async () => {
   try {
        const [data1, data2] = await Promise.all([
            fetch("https://api.example.com/data1").then(res => res.json()),
            fetch("https://api.example.com/data2").then(res => res.json())
        ])
```

```
        return [data1, data2]
    }
    catch (err) {
        console.log("Error", err)
        throw err
    }
}
```

```
const fetchData = () => {
    return new Promise ((resolve, reject) => {
    setTimeout(() => {
        const data = {name: "John", age: 23}
        resolve(data)},
        2000)
})
}

const fetchDataAsync = async () => {
    try{
        const data = await fetchData()
        console.log(data)
    }
    catch (err) {
        console.error("Error:", err)
    }
}

fetchDataAsync();
```