3BeezTechnologies Pvt. Ltd.

Asynchronous JavaScript:

JavaScript is a single-threaded language, meaning it executes one operation at a time in a synchronous manner. However, there are situations where certain operations, like fetching data from a server or reading a file, may take some time to complete. To avoid blocking the execution of other code, JavaScript uses asynchronous programming.

Asynchronous operations allow the program to continue executing other tasks while waiting for certain operations to finish. Commonly, this is achieved using callbacks, promises, and more recently, async/await.

Callbacks:

Callbacks are functions that are passed as arguments to another function and are executed after the completion of an asynchronous operation.

```
function fetchData(callback) {
    setTimeout(function () {
        console.log("Data fetched!");
        callback();
    }, 2000);
}

function processData() {
```

```
    console.log("Data processed!");
}

// Usage
fetchData(processData);
console.log("Doing something else while waiting...");
```

In this example, fetchData simulates an asynchronous operation (like fetching data from a server) using setTimeout. The processData function is passed as a callback and gets executed after the data is fetched.

Promises:

Promises provide a cleaner way to handle asynchronous operations. They represent a value that may be available now, or in the future, or never.

```
function fetchData() {
    return new Promise(function (resolve, reject) {
        setTimeout(function () {
            console.log("Data fetched!");
            resolve();
        }, 2000);
    });
}
```

```
function processData() {
    console.log("Data processed!");
}


// Usage
fetchData().then(processData);
console.log("Doing something else while waiting...");
```

Here, fetchData returns a promise. The resolve function is called when the asynchronous operation is successful, and reject is called in case of an error. The processData function is then attached using the then method.

Async/Await:

Async/await is a modern way to write asynchronous code in a more synchronous-looking style. It's built on top of promises.

```
function fetchData() {
    return new Promise(function (resolve) {
        setTimeout(function () {
            console.log("Data fetched!");
            resolve();
        }, 2000);
    });
}
```

```
async function processData() {
   console.log("Data processed!");
}


// Usage
async function fetchDataAndProcess() {
   await fetchData();
   processData();
}


fetchDataAndProcess();
console.log("Doing something else while waiting...");
```

In this example, fetchData returns a promise, and processData is an asynchronous function declared using the async keyword. The await keyword is used to pause the execution of fetchDataAndProcess until fetchData completes.