

**Asynchronous
Programming in**

JS

Callbacks

Callbacks are functions passed as arguments to another function and executed once the operation is complete.

```
JS main.js

1  function fetchData(callback) {
2      setTimeout(() => {
3          const data = 'This is some data fetched
4  asynchronously with a callback';
5          callback(data);
6      }, 2000);
7  }
8
9  // Using the callback function
10 fetchData(function(data) {
11     console.log(data);
12 });
```

Promises

Promises represent a value that may be available now, in the future, or never. They provide a cleaner alternative to callback-based approaches.

```
JS main.js

1  function fetchData() {
2      return new Promise((resolve, reject) => {
3          setTimeout(() => {
4              const data = 'This is some data fetched
5 asynchronously with a Promise';
6              resolve(data);
7          }, 2000);
8      });
9  }
10 // Using the Promise
11 fetchData()
12   .then(data => {
13     console.log(data);
14   })
15   .catch(error => {
16     console.error(error);
17   });
```

Async/Await

Async/await provides syntactic sugar on top of Promises, making asynchronous code more readable and easier to understand.

```
JS main.js

1  async function fetchData() {
2    return new Promise(resolve => {
3      setTimeout(() => {
4        const data = 'This is some data fetched
5 asynchronously with async/await';
6        resolve(data);
7      }, 2000);
8    });
9  }
10 // Using async/await
11 async function getData() {
12   try {
13     const data = await fetchData();
14     console.log(data);
15   } catch (error) {
16     console.error(error);
17   }
18 }
19 getData();
```