JavaScript Async Programming Summary(Lecture 4)

1. Synchronous vs Asynchronous

Synchronous functions run line by line. Each line must finish executing before the next starts.

Asynchronous functions can run without blocking the rest of the code. Examples of async operations: setTimeout, setInterval, HTTP requests, events.

2. Callbacks and Callback Hell

Callbacks are functions passed into other functions to be called later. They are commonly used in async operations, but nesting too many callbacks leads to 'callback hell' which makes code hard to read and maintain.

3. Promises

A Promise is an object representing the eventual completion or failure of an asynchronous operation. It has three states: pending, fulfilled, and rejected.
You handle promises with .then() for success and .catch() for errors.

4. Fetch API

The Fetch API allows you to make HTTP requests. It returns a Promise.
Example:

```
fetch('url')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.log(error));
```

5. JSON

JSON (JavaScript Object Notation) is a format for structuring data. Use JSON.stringify(obj) to convert an object to JSON string.
Use JSON.parse(jsonString) to convert JSON string back to object.

6. Async / Await

Async/await is a modern way to write async code. It makes the code look synchronous.
Use 'await' inside an 'async' function to wait for a Promise to resolve.
Example:

```
async function getData() {
  try {
    const response = await fetch('url');
    const data = await response.json();
    console.log(data);
  } catch (error) {
    console.error(error);
  }
}
```