Asynchronous JavaScript - Notes (freeCodeCamp)

1. What is Asynchronous JavaScript?

- Synchronous code runs in sequence - each line must finish before the next starts.
- Asynchronous code allows operations to happen in the background.

2. Common Asynchronous Operations

- Fetching data from APIs
- Reading files
- setTimeout / setInterval
- DOM events
- Promises

A. setTimeout() and setInterval()

setTimeout(callback, delay): Executes a function once after a delay.

Example:

```
setTimeout(() => {
  console.log('Runs after 2 seconds');
}, 2000);
```

setInterval(callback, interval): Executes a function repeatedly.

Example:

```
setInterval(() => {
  console.log('Runs every 1 second');
}, 1000);
```

B. Callbacks

A callback is a function passed into another function to run after it completes.

Example:

```
function greet(name, callback) {
  console.log('Hi ' + name);
  callback();
```

```
}

function afterGreet() {
  console.log('Callback executed.');
}

greet('Fuad', afterGreet);
```

Callback Hell

Nested callbacks make code hard to read and debug.

Example:

```
doSomething(() => {
  doNextThing(() => {
    doAnotherThing(() => {
      // messy code
    });
  });
});
```

C. Promises

A Promise represents a value that may be available now, later, or never.

Example:

```
let promise = new Promise((resolve, reject) => {
  let success = true;
  if (success) {
    resolve('Done!');
  } else {
    reject('Failed!');
  }
});

promise
```

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

D. fetch() API

Used to make HTTP requests (returns a Promise).

Example:

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.log('Error:', error));
```

E. async and await

Cleaner syntax to handle asynchronous operations.

Example:

```
async function getData() {
  try {
    let response = await fetch('https://api.example.com/data');
    let data = await response.json();
    console.log(data);
  } catch (error) {
    console.log('Error:', error);
  }
}

getData();
```

Summary of Key Terms

- setTimeout: Run code once after delay
- setInterval: Run code repeatedly
- callback: Function called after another
- Promise: Future value

Asynchronous JavaScript - Notes (freeCodeCamp)

- then(): Runs on resolve
- catch(): Runs on error
- async: Declares async function
- await: Wait for Promise
- fetch(): API request

Real-Life Example: API with async/await

```
async function getUser() {
  try {
    const res = await fetch('https://jsonplaceholder.typicode.com/users/1');
    const user = await res.json();
    console.log(user);
  } catch (err) {
    console.error('Failed to fetch user:', err);
  }
}
```