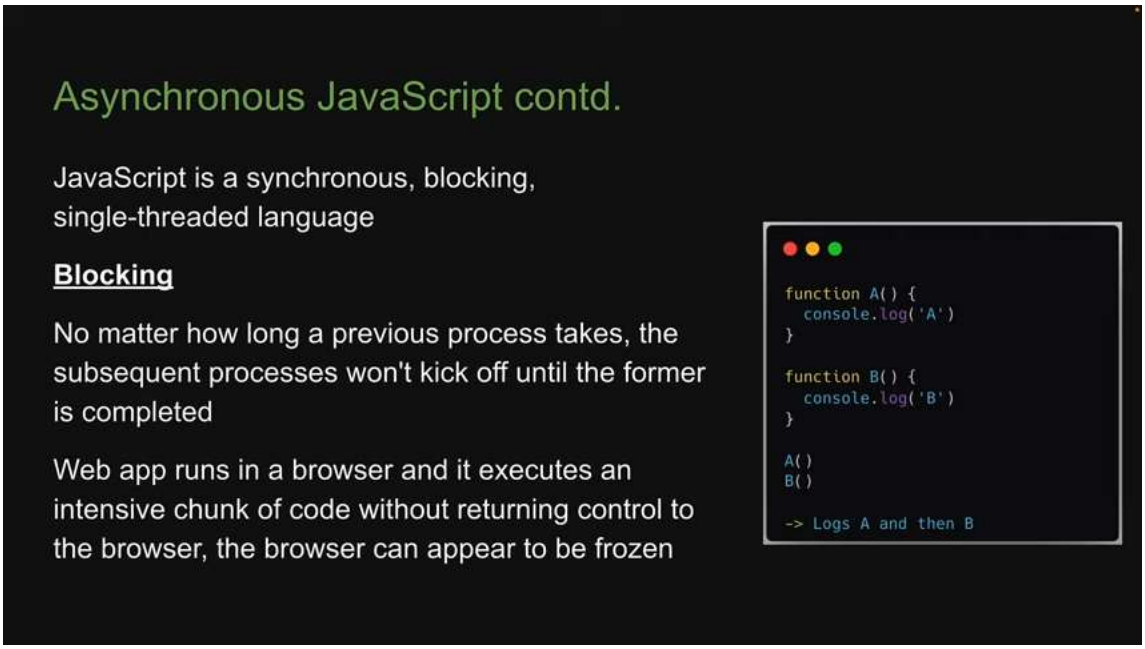
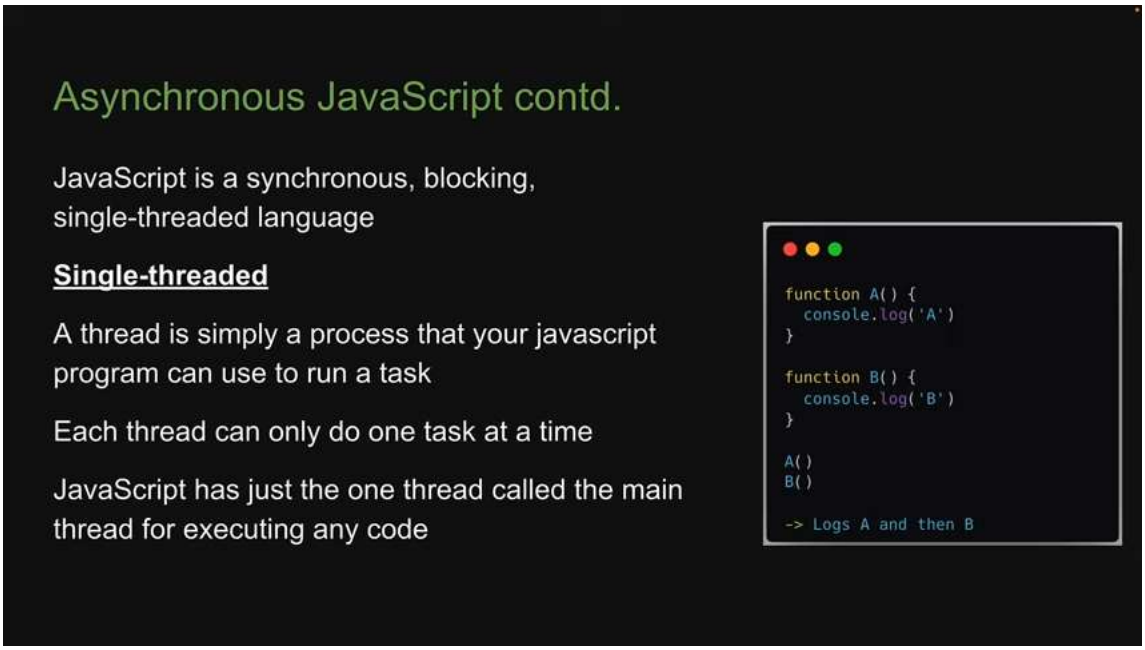
# Asynchronous JS

08 April 2024 15:49



## Asynchronous JavaScript contd.

Problem with synchronous, blocking, single-threaded model of JavaScript

```
let response = fetchDataFromDB('endpoint')
displayDataFromDB(response)
```

fetchDataFromDB('endpoint') could take 1 second or even more

During that time, we can't run any further code

JavaScript, if it simply proceeds to the next line without waiting, we have an error because data is not what we expect it to be

## Asynchronous JavaScript contd.

Just JavaScript is not enough

We need new pieces which are outside of JavaScript to help us write asynchronous code

For front-end, this is where web browsers come into play. For back-end, this is where Node.js comes into play

Web browsers and Node.js define functions and APIs that allow us to register functions that should not be executed synchronously, and should instead be invoked asynchronously when some kind of event occurs

For example, that could be the passage of time ( setTimeout or setInterval), the user's interaction with the mouse (addEventListener), data being read from a file system or the arrival of data over the network (callbacks, Promises, async-await)

You can let your code do several things at the same time without stopping or blocking your main thread

## Asynchronous JavaScript Summary

JavaScript is a synchronous, blocking, single-threaded language

This nature however is not beneficial for writing apps

We want non-blocking asynchronous behaviour which is made possible by a browser for FE and Node.js for backend

This style of programming where we don't block the main JavaScript thread is fundamental to Node.js and is at the heart of how some of the built-in module code is written