Basics of Asynchronous JavaScript

Synchronous Code

Synchronous code is executed sequentially from the top of a file to the end of a file — with each line unable to execute until the previous line has finished executing.

Asynchronous Code

Asynchronous code can be executed in parallel to other code that is already running.

Blocking

Blocking occurs when code prevents a user from interacting with an app due to background code not finishing execution.

Threads

Generally, the amount of tasks a language can execute is limited by the amount of threads the language has access to.

async-javascript-single-threaded

JavaScript is a single-threaded language. However, it can handle asynchronous code using the event loop.

async-javascript-syntaxes

In JavaScript, asynchronous code can be written in a variety of different ways, including:

- Callbacks
- Promises
- async / await

async-javascript-set-timeout

`setTimeout()` accepts two arguments:

- A callback function that is executed asynchronously
- The minimal number of milliseconds a program waits before executing the callback function

async-javascript-set-interval

`setInterval()` accepts two arguments:

- A callback function that is executed asynchronously
- The number of milliseconds for how frequently the callback function executes