Introduction to Asynchronous Programming in JavaScript

Learn strategies for implementing asynchronous code. Learn why it's necessary.

It's time to introduce asynchronous code. So far, with the exception of a couple of lessons, everything we've seen and done has been synchronous - one thing happens right after the other. A program starts and runs immediately to completion.

I normally try to use runnable code in my lessons, but explaining the common use cases of asynchronous code is not plausible in this limited environment. In this section, we'll have to work with hypothetical situations.

The Need for Async

Loading Websites

The need for asynchronous code arises as soon as requests need to be made. A web request takes measurable, palpable time, on the order of tenths of a second or even seconds. Many pages make several requests.

If code were synchronous, these several requests would have to be made in series. One request would launch and no other code would run until that request completed, either 100ms or up to a few seconds later. We wouldn't be able to scroll or move our mouse during this as the engine would be waiting for a response, frozen until it receives one.

Imagine a dozen of those requests all loading one-by-one to create one website. It would be ages before the page allowed us to do anything.

With asynchronous code, we can launch all of these requests at once and deal with the data as it comes in. In the meantime, we can listen to user actions such as scrolling and button clicks.

We attach callbacks, event listeners, to these requests. Once data arrives, we trigger our callbacks and do something with the data, such as adding it to the DOM.

In the next lesson, we’ll learn some common techniques for implementing asynchronous code into our projects.