Lesson Weekend

Intermediate JavaScript (/intermediate-javascript) / Asynchrony and APIs (/intermediate-javascript/asynchrony-and-apis) / Asynchrony in JavaScript

Text

One of JavaScript's biggest strengths is the fact that code can be either **synchronous** or **asynchronous**. These terms can be confusing for new developers, but the good news is that you've already been working extensively with both synchronous and asynchronous code.

Let's start by simplifying the difference between sync and async code.

- **Synchronous** means the code runs *right now*.
- **Asynchronous** means the code will run *later*.

In a nutshell, that's really the only difference. Logic problems like Roman Numerals, Shape Tracker and Pig Dice all rely heavily on synchronous code. For instance, if you look at the file that contains the logic for one of those projects, that is synchronous code. The JavaScript engine goes through each line of your business logic one at a time.

What about asynchronous code? When have we used that? A prime example is when our application needs **user input** to perform a desired behavior. When our application loads, we don't know when a user will click a button or fill out fields, but it will certainly happen later, not right now. In this situation, we use event listeners to wait and listen for an event that will happen later.

Why is JavaScript asynchronous? What would happen if it weren't?

JavaScript is **single-threaded** and **non-blocking**:

- **Single-threaded** means JavaScript can only do one thing at a time.
- **Non-blocking** means the JavaScript engine won't wait until something is finished (if it's async code) before moving on to the next line of code.

Imagine you throw a ball in the air, but you can only throw one at a time. This is the equivalent of **single-threaded.**

However, once you throw a ball in the air, you can throw a second ball before the first lands. That's the equivalent of **non-blocking.** If you didn't have this non-blocking capacity, however, you wouldn't be able to do anything else while a ball is in the air. You wouldn't be able to throw another ball or move around or do anything at all. In other words, you'd freeze up.

Ouch. Nobody likes when the browser freezes up. It's one of the most frustrating UI experiences a user can have.

It may appear that a JavaScript application can do many things at the same time, but that's just because JavaScript does things very quickly. When an application does need to engage in long-running processes, it will do so (or at least should do so) through **concurrency**. Concurrency is the process of interweaving many tasks, doing a little of one and then a little of another, engaging in this process so quickly that it appears as if these tasks are occurring simultaneously. We won't be covering concurrency in depth, but at the very least, you should know what it means. If you would like to learn more about concurrency, you can do so on MDN's reference page on the event loop (https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop).

Asynchrony is a cornerstone of JavaScript development. Sometimes we *need* to implement it in our applications, such as when we make API calls. (We'll learn more about API calls in the next lesson.) In other situations we'll *choose* to implement it. If a synchronous function takes a long time to process, for example, it will freeze up our application, so it would be better to make it async and concurrent. While there are many ways to use asynchrony in our code, we'll focus on learning about asynchronous code by making API calls.

Previous (/intermediate-javascript/asynchrony-and-apis/asynchrony-and-apis-in-javascript-objectives)
Next (/intermediate-javascript/asynchrony-and-apis/introduction-to-apis)

Lesson 2 of 33
Last updated more than 3 months ago.

disable dark mode

(http://www.epicodus.com)