Lesson 42 - Asynchronous Programming

What is asynchronous programming?

It takes time for a function to fetch data. Asynchronous programming was devised to accommodate for the lag between when a function is called to when the value of that function is returned. Without asynchronous programming, apps would spend a long time on loading screens. A loading screen might appear:

- When a user signs in, waiting for all their user data to be returned from the database.
- During the user experience, waiting for the data to load at each new screen.

Asynchronous programming allows a user to go about his business in an application, while processes run in the background, thus enhancing the user experience.

> Data may take a long time to submit to a database. With asynchronous programming, the user can move to another screen while the function continues to execute. This frees the user from having to wait on the same screen for the data to load, and the user can continue with other tasks.

Baking a cake synchronously

Synchronous programming follows a strict set of sequences. When the code runs in a synchronous program, it will follow each step of an algorithm. It does so in order and will wait for the present operation to finish before continuing on to the next.

Synchronous programming follows a “Bake a cake” algorithm.

- Measure the ingredients.
- Mix flour, eggs, and sugar.
- Heat oven and bake.
- Eat.

Each step must happen in order. The ingredients must be measured, mix must be mixed, before the mix is baked. And, to taste like a cake, it should be baked before it is eaten. Because only one person is doing all the work, you must complete one task fully before starting the next.

Baking a cake asynchronously

Asynchronous cake baking, by contrast, allows multiple people to be working on the task at once. One person can gather and measure the ingredients while another person begins mixing the ingredients together. Asynchronous programming allows multiple processes to be started, lets the processes do their work, and when their job is finished, it gets the result and puts it through the steps.

If the oven finishes heating before the cake mix is fully prepared, asynchronous programming says that is okay. Synchronous programming would never have started the oven without the mix having been prepared. When the mix is completed, it sends an update to the algorithm to come back and pick up the result of the mix and push it through the process. Now, when the cake mix is prepared, it can be passed into a heated oven that is already heated to the right temperature, ready to bake the cake.

Eating the cake

Unfortunately, asynchronous programming won’t help you eat your cake, but it will help get the cake down the line faster. The baking must still happen before you can eat it. (And, if the eater is called to eat before the cake is ready, like how the oven was heating before the cake mix was ready, the one eating can pace the kitchen hungrily.)

Overview of the asynchronous model

The core of async programming is the `Task` and `Task<T>` objects, which model asynchronous operations. They are supported by the `async` and `await` keywords. The model is fairly simple in most cases:

- For I/O-bound code, you await an operation that returns a `Task` or `Task<T>` inside of an async method.
- For CPU-bound code, you await an operation that is started on a background thread with the `Task.Run()` method.

The `await` keyword is where the magic happens. It yields control to the caller of the method that performed `await`, and it ultimately allows a UI to be responsive or a service to be elastic.

Key Points

- Async code can be used for both I/O-bound and CPU-bound code, but differently for each scenario.
- Async code uses `Task<T>` and `Task`, which are constructs used to model work being done in the background.
- The `async` keyword turns a method into an async method, which allows you to use the `await` keyword in its body.
- When the `await` keyword is applied, it suspends the calling method and yields control back to its caller until the awaited task is complete.
- `await` can only be used inside an async method.