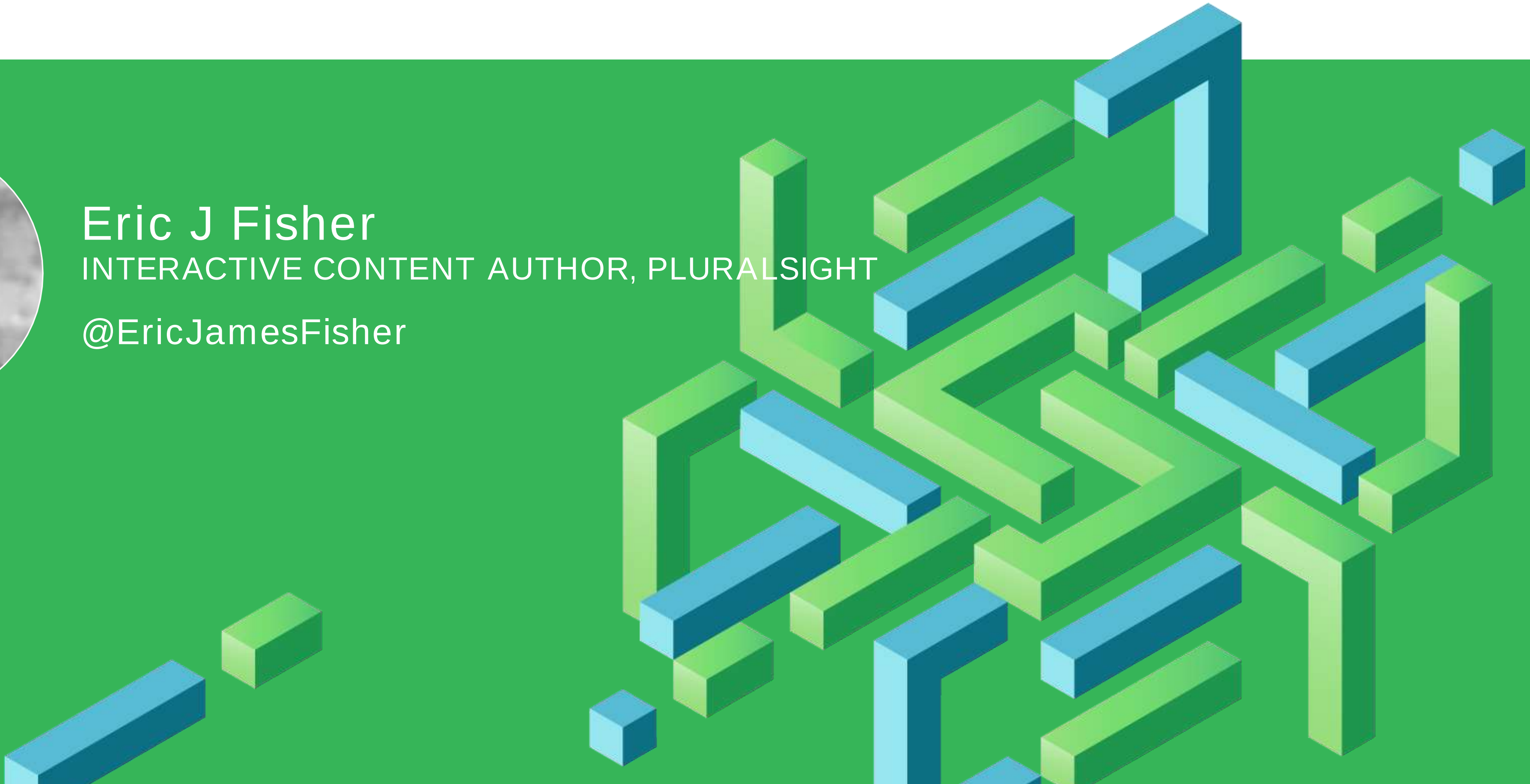
C#: Using Async and Await to Run Code Asynchronously

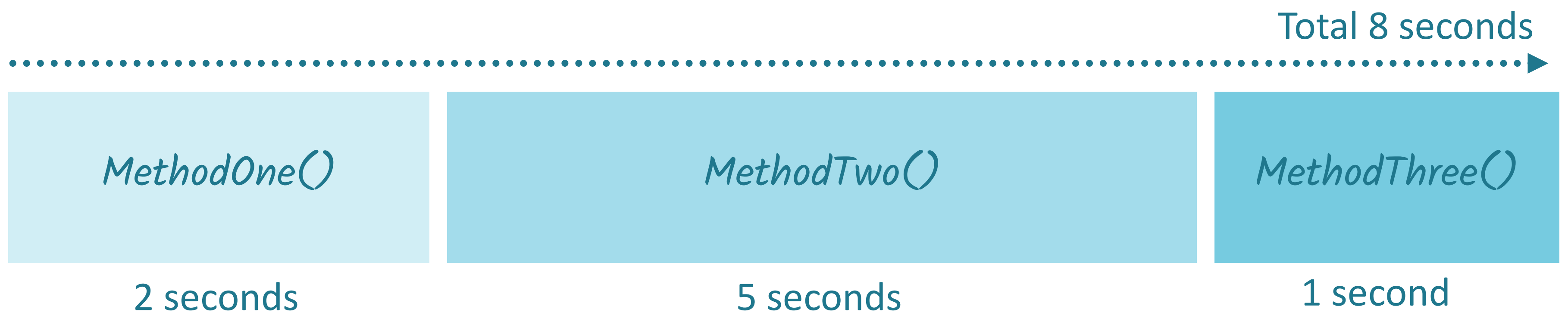
Eric J Fisher
INTERACTIVE CONTENT AUTHOR, PLURALSIGHT

@EricJamesFisher

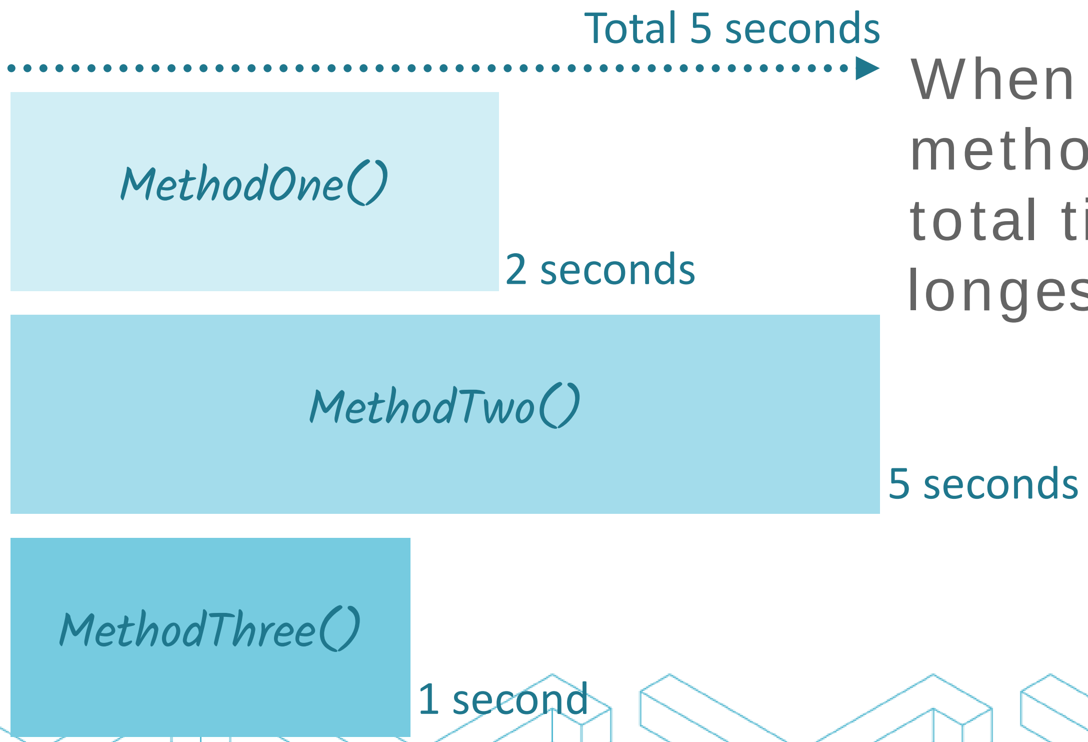


Synchronous Code

When running code synchronously, the total execution time is the sum of the time it takes to run all methods.



Asynchronous Code



When running code asynchronously, multiple methods can start running at the same time, but the total time spent running is only as long as the longest running method.

Synchronous Method

We'll be converting this synchronous method to be asynchronous

Example.cs

```
public int Addition()
{
    var a = SlowMethodOne();
    var b = SlowMethodTwo();
    return a + b;
}
```

Before we step through how to convert this, I'm going to show you the asynchronous version of this method to show how they're very similar, but with a few differences.

Asynchronous Version of Method

The asynchronous version is very similar to the synchronous version

Example.cs

```
public async Task<int> AdditionAsync()
{
    var a = SlowMethodOneAsync();
    var b = SlowMethodTwoAsync();
    return await a + await b;
}
```

Over the next few slides we'll go back to the synchronous version and convert it to be asynchronous one piece at a time.

Adding the Async Modifier

async specifies a method is asynchronous

Example.cs

```
public async int AdditionAsync()
{
    var a = SlowMethodOne();
    var b = SlowMethodTwo();
    return a + b;
}
```

Standard naming convention is to end asynchronous method names with async

The async modifier is used in the method declaration

⚠

If we run this code we'll get a compile error. This is because async methods have their own special return types.

Return Types used for Async

Task

Task represents an asynchronous operation.

Used where you'd normally use void in synchronous methods

Task<T>

Task<T> returns a Task containing a returned value.

Used where you'd normally return a value in synchronous methods

Void

Void returns nothing.

Generally you only use void with event handlers in async as it has side effects such as creating error handling complexities.

Update the Return Type

We will need to change the return type to use Task<T>

Example.cs

```
public async Task<int> AdditionAsync()
{
    var a = SlowMethodOne();
    var b = SlowMethodTwo();
    return a + b;
}
```

When a method is returning something we'll use Task<T> return type

Adding Await Keywords

await tells the method it must wait until the awaited call finishes running

Example.cs

```
public async Task<int> AdditionAsync()
{
    var a = SlowMethodOneAsync();
    var b = SlowMethodTwoAsync();
    return await a + await b;
}
```

We'll switch to calling the async versions of these methods

We won't be able to get the values for a and b until we've awaited them

The Method is Now Asynchronous

async and await is all that is needed to call async methods asynchronously

Example.cs

```
public async Task<int> AdditionAsync()
{
    var a = SlowMethodOneAsync();
    var b = SlowMethodTwoAsync();
    return await a + await b;
}
```

Note: async in C# does NOT create new threads by default! This means async works well for UIs and IO bound methods. (For CPU bound methods async isn't typically effective without multithreading)

Calling Async Methods Synchronously

Result allows you to get the results of an async method synchronously

Example.cs

```
public int Addition()
{
    var a = SlowMethodOneAsync().Result;
    var b = SlowMethodTwoAsync().Result;
    return a + b;
}
```

Result will get the value of an async method synchronously

You should try to avoid synchronously calling async methods when possible as it can lead to deadlocks and greatly complicate error handling.

Summary

Use return type Task when nothing is returned

Use return type Task<T> when returning something

Avoid return type void. (except with event handlers)

async modifier is used to make a method able to run asynchronously

await keyword tells a method to wait until the async Task completes

Result will get results of an async method synchronously (but has side effects)