- Task
  - Every Async method will return the type of a Task or Task<T>
- await
  - Pauses the current method until Task is complete
  - Does not block the current thread
- async
  - This is the method signature that you give an Async method
  - Tells the compiler to treat the "await" keyword
- .Result
  - Should only be used inside of a continuation because it will cause a block and possibly a deadlock

- CancellationToken
  - This is used to cancel an Async operation
- Dispatcher
  - Used to update the UI in a Continuation

HOW DOES ASYNCHRONOUS PROGRAMMING HELP ME AND WHY SHOULD I CARE?

- It allows you to have multiple process that don't rely on each other to run at the same time
- If you don't want to lock up your UI anytime that a request is made but instead have it run in the background
- If you are trying to pull data from an API or save data to a data base
- It helps with running methods in Parallel (threading)

THINGS TO KEEP IN MIND WHEN USING ASYNC

- Once you start using async you must use it throughout the rest of the method calls

- You must use the await keyword or else you can run into issues where methods that shouldn't run until after a certain step run anyway

- When writing Async methods always add the suffix to the method name of "Async" this way the person using the method understands that they are about to be using an Asynchronous method

THINGS TO BE CAREFUL ABOUT

- Async void method
  - Should only be used for events
- Code running out of order
  - This could be due to a missing await

ADVANCED TOPICS

- Asynchronous Stream - Running items as they come in
  - await before the foreach
  - have a yield return
- Asynchronous Disposable – Dispose after use
  - await using var service = new Service();

HELPFUL RESOURCES

- http://jeremybytes.com/

- https://docs.microsoft.com/en-us/dotnet/csharp/async

- https://github.com/jeremybytes/async-workshop-2022

- https://app.pluralsight.com/library/courses/applying-asynchronous-programming-c-sharp/table-of-contents