Async / Await

.NET CORE

Asynchronous programming enables code that reads like a sequence of statements but executes in a much more complicated order based on external resource allocation and when tasks complete.

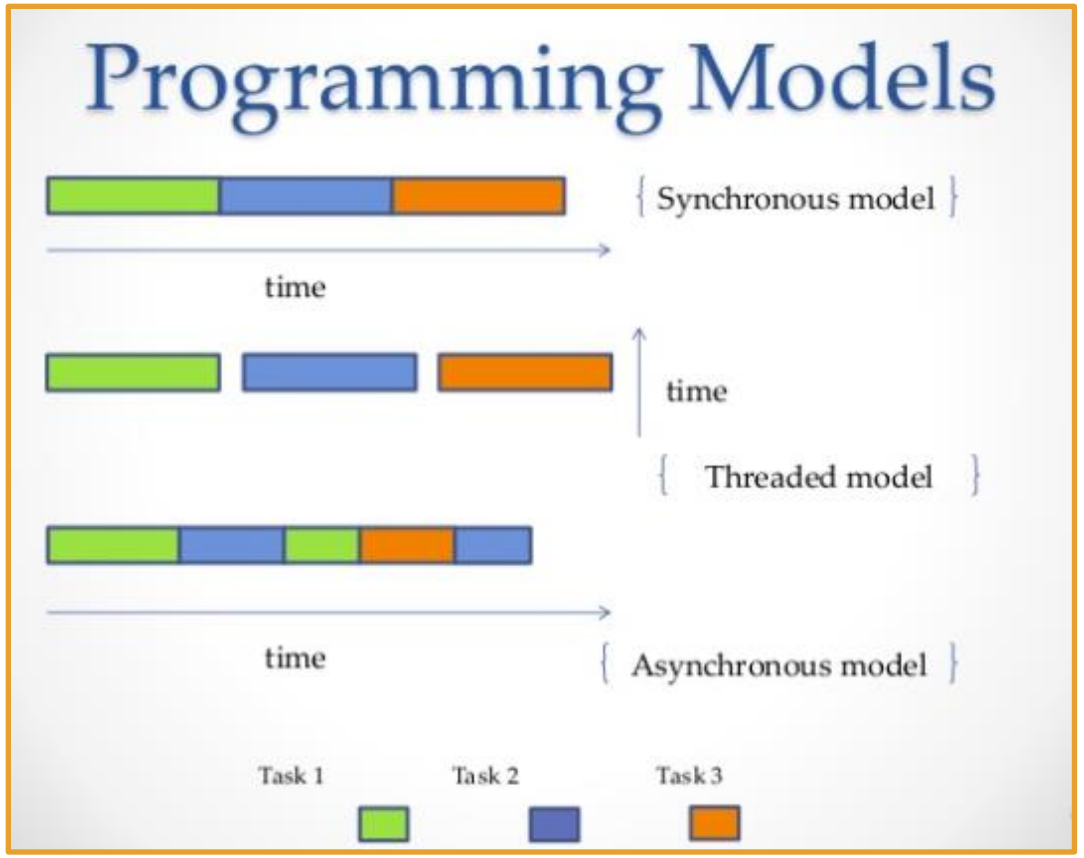
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/

Task<> asynchronous programming model

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/task-asynchronous-programming-model
https://medium.com/velotio-perspectives/an-introduction-to-asynchronous-programming-in-python-af0189a88bbb

Asynchronous programming is used to avoid performance bottlenecks and enhance the overall responsiveness of an application. C# 5 introduced ***async*** programming. The compiler does most of the syntactical work so that the code can retain a synchronous-like structure.

Asynchrony is essential for potentially blocking code. If an activity is blocked in a synchronous process, the entire application must wait. In an asynchronous process, the application can continue with other work until the blocking task finishes.



Modifiers – Async

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/async
https://www.codeproject.com/Articles/1054993/async-await-What-You-Should-Know-Updated

---

- Use the ***async*** modifier to specify that a method is asynchronous.
- An ***async*** method uses the ***await*** operator to continue doing work without blocking the caller's thread.
- An ***async*** method runs synchronously until it reaches its first ***await*** expression, at which point it is suspended until the ***awaited*** task is complete. In the meantime, control returns to the caller of the async method.
- The ***async*** keyword is contextual in that it's a keyword only when it modifies a method. In all other contexts, it's interpreted as an identifier.
- The ***async*** method can't declare any ***in***, ***ref*** or ***out*** parameters, nor can it have a reference return value, but it can call methods that have such parameters.

Async/Await

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/task-asynchronous-programming-model#BKMK_HowtoWriteanAsyncMethod

By using the keywords ***async*** and ***await***, you can use resources in .NET Framework and .NET Core to create an asynchronous method.

- The method signature includes the ***async*** modifier.
- The return type is Task<int> or Task.
- Conventionally, the method name ends in ...Async.
- GetStringAsync returns a Task<string>.
- When you await the ***task,*** you'll get a string (urlContents).
- Before ***await***ing the task, you can do work that doesn't rely on the string from GetStringAsync.

```
async Task<int> AccessTheWebAsync()
{
    // You need to add a reference to System.Net.Http to declare client.
    var client = new HttpClient();

    // GetStringAsync returns a Task<string>. That means that when you await the
    // task you'll get a string (urlContents).
    Task<string> getStringTask = client.GetStringAsync("https://docs.microsoft.com/dotnet");

    // You can do work here that doesn't rely on the string from GetStringAsync.
    DoIndependentWork();

    // The await operator suspends AccessTheWebAsync.
    //  - AccessTheWebAsync can't continue until getStringTask is complete.
    //  - Meanwhile, control returns to the caller of AccessTheWebAsync.
    //  - Control resumes here when getStringTask is complete.
    //  - The await operator then retrieves the string result from getStringTask.
    string urlContents = await getStringTask;

    // The return statement specifies an integer result.
    // Any methods that are awaiting AccessTheWebAsync retrieve the length value.
    return urlContents.Length;
}
```

Modifiers – Async

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

- An ***async*** method can have ***only*** the return types Task or Task<TResult>
- An ***async*** method can't declare any ***in***, ***ref*** or ***out*** parameters, nor can it return a reference value, but it can call methods that have such parameters.
- Only ***async*** methods can call other ***async*** methods.
- By convention, ***async*** methods have and await statement in them.

```
public static async Task<string> ShowTodaysInfo()
{
    string ret = $"Today is {DateTime.Today:D}\n" +
                 "Today's hours of leisure: " +
                 $"{await GetLeisureHours()}";
    return ret;
}

static async Task<int> GetLeisureHours()
{
    // Task.FromResult is a placeholder for actual work that returns a string.
    var today = await Task.FromResult<string>(DateTime.Now.DayOfWeek.ToString());

    // The method then can process the result in some way.
    int leisureHours;
    if (today.First() == 'S')
        leisureHours = 16;
    else
        leisureHours = 5;

    return leisureHours;
}
// The example displays output like the following:
//    Today is Wednesday, May 24, 2017
//    Today's hours of leisure: 5
```