**await (C# Reference)**

The await operator is applied to a task in an asynchronous method to insert a suspension point in the execution of the method until the awaited task completes. The task represents ongoing work.

await can only be used in an asynchronous method modified by the [async](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/async) keyword. Such a method, defined by using the async modifier and usually containing one or more awaitexpressions, is referred to as an async method.

Code:   
public async Task MyMethodAsync()

{

Task<int> longRunningTask = LongRunningOperationAsync();

// independent work which doesn't need the result of LongRunningOperationAsync can be done here

//and now we call await on the task

int result = await longRunningTask;

//use the result

Console.WriteLine(result);

}

public async Task<int> LongRunningOperationAsync() // assume we return an int from this long running operation

{

await Task.Delay(1000); //1 seconds delay

return 1;

}

OK, so what happens here:

1. Task<int> longRunningTask = LongRunningOperationAsync(); starts executing LongRunningOperation
2. Independent work is done on let's assume the Main Thread (Thread ID = 1) then await longRunningTask is reached.

Now, if the longRunningTask hasn't finished and it is still running, MyMethodAsync() will return to its calling method, thus the main thread doesn't get blocked. When the longRunningTask is done then a thread from the ThreadPool (can be any thread) will return to MyMethodAsync() in its previous context and continue execution (in this case printing the result to the console).

A second case would be that the longRunningTask has already finished its execution and the result is available. When reaching the await longRunningTask we already have the result so the code will continue executing on the very same thread. (in this case printing result to console). Of course this is not the case for the above example, where there's a Task.Delay(1000) involved.

Ref:  
<https://richnewman.wordpress.com/2012/12/03/tutorial-asynchronous-programming-async-and-await-for-beginners/>  
<https://weblogs.asp.net/dixin/understanding-c-sharp-async-await-3-runtime-context>  
http://michd.me/blog/using-async-in-c-sharp-part-1/  
<http://www.dotnetcurry.com/csharp/1307/async-await-asynchronous-programming-examples>  
https://stephenhaunts.com/2014/10/10/simple-async-await-example-for-asynchronous-programming/  
<http://www.c-sharpcorner.com/UploadFile/dacca2/asynchronous-programming-in-C-Sharp-5-0-part-1-understand-async/>