Welcome to C# 7.1

Mads

October 31st, 2017

The first point release

C# 7.1 is therefore a small release with just a few (but well-chosen) new language features; useful we think, but definitely in the minor range. It's supported by Visual Studio 2017, starting with Update 15.3.

Let's look at each of the new C# 7.1 features used in here. For a full overview of the features in C# 7.1, check out the docs.

Async Main

The `Main` entry point method can now return a `Task` or a `Task<int>`. When it does, execution will wait for the returned task to complete, before shutting down the program.

Of course, the common use of this will be to make the `Main` method `async`, which you can also do:

```
    static async Task Main(string[] args)
```

This lets you `await` directly in the `Main` method, something you couldn't do before.

```
    WriteLine($"Fib {tuple.input} = {await tuple.task}");
```

What you had to do previously was quite unappetizing: first you'd create an async helper method, `MainAsync`, say, with all the logic in. Then you'd write this cryptic `Main` method:

```
    static void Main(string[] args) => MainAsync().GetAwaiter().GetResult();
```

Now you can just make your `Main` method async, and the compiler will rewrite it for you.

Inferred tuple element names

In this lambda expression inside the query:

```
    input => (input, task: FibonacciAsync(input))
```

You notice that we create a tuple, but only give a name, `task`, for the second element. Yet a few lines later we are able to say

```
    WriteLine($"Fib {tuple.input} = {await tuple.task}");
```

Accessing the first element by the name `tuple.input`. That's because when you create a tuple with an expression that "has" a name, like `input` in the lambda expression above, we'll now automatically give the corresponding tuple element that name.

Default literals

If there's an expected type for a `default` expression, you can now omit the mention of the type, as we do for the `CancellationToken` in in the signature of the `FibonacciAsync` method:

```
    private static Task<int> FibonacciAsync(int n, CancellationToken token = default)
```

This avoids tedious repetition of type names, or typing out long ones when they are already given by context.