What's new in C# 7.1

C# 7.1 is the first point release to the C# language. It marks an increased release cadence for the language. You can use the new features sooner, ideally when each new feature is ready. C# 7.1 adds the ability to configure the compiler to match a specified version of the language. That enables you to separate the decision to upgrade tools from the decision to upgrade language versions.

C# 7.1 adds the language version selection configuration element, three new language features and new compiler behavior.

The new language features in this release are:

- `async` `Main` method
  - The entry point for an application can have the `async` modifier.
- `default` literal expressions
  - You can use default literal expressions in default value expressions when the target type can be inferred.
- Inferred tuple element names
  - The names of tuple elements can be inferred from tuple initialization in many cases.

Finally, the compiler has two options `/refout` and `/refonly` that control reference assembly generation.

To use the latest features in a point release, you need to configure the compiler language version and select the version.

Async main

An *async main* method enables you to use `await` in your `Main` method. Previously you would need to write:

```
static int Main()
{
    return DoAsyncWork().GetAwaiter().GetResult();
}
```

You can now write:

```
static async Task<int> Main()
{
    // This could also be replaced with the body
    // DoAsyncWork, including its await expressions:
    return await DoAsyncWork();
}
```

If your program doesn't return an exit code, you can declare a `Main` method that returns a Task:

```
static async Task Main()
{
    await SomeAsyncMethod();
}
```

You can read more about the details in the async main topic in the programming guide.

Default literal expressions

Default literal expressions are an enhancement to default value expressions. These expressions initialize a variable to the default value. Where you previously would write:

```
Func<string, bool> whereClause = default(Func<string, bool>);
```

You can now omit the type on the right-hand side of the initialization:

```
Func<string, bool> whereClause = default;
```

You can learn more about this enhancement in the C# Programming Guide topic on default value expressions.

This enhancement also changes some of the parsing rules for the default keyword.

Inferred tuple element names

This feature is a small enhancement to the tuples feature introduced in C# 7.0. Many times when you initialize a tuple, the variables used for the right side of the assignment are the same as the names you'd like for the tuple elements:

```
int count = 5;
string label = "Colors used in the map";
var pair = (count: count, label: label);
```

The names of tuple elements can be inferred from the variables used to initialize the tuple in C# 7.1:

```
int count = 5;
string label = "Colors used in the map";
var pair = (count, label); // element names are "count" and "label"
```

You can learn more about this feature in the Tuples topic.

Reference assembly generation

There are two new compiler options that generate *reference-only assemblies*: /refout and /refonly. The linked topics explain these options and reference assemblies in more detail.