QUICK-START C#

Introduction to Extension Methods

What are they?

You can skip this. It's just the opening. Head to Making an Extension Method for the real stuff.

Most of the time in C#, you're using predefined types or classes such as **Vector3** or **string**. These have their own members and methods inside but sometimes not ones you need. A common approach to adding functionality to these types is to create a new method in your own class that adds what's missing for your project. Extension Methods are the same as any other method and adding one is *almost* the same, just with two required keywords, **this** and **static**. ***When it's appropriate to add one*** is still a fairly complex topic though.

Making an Extension Method

Adding an extension method is easy and below is an example. Though depending on access you want, these are the 'general' guidelines:

Global Access
Any class can access the method.

- **Is in a *global* static class**
- **Is a *public* static method**

'*Scoped*' Access
Any class using the namespace the method is in can access the method.

- **Is in a static class in a namespace**
- **Is a *public* static method**

Local Access
Only the class the method is inside can access the method.

- **Is in a static class in the namespace**
- **Is a static method**

The first parameter must always have 'this' prefixed and it's object type will be extended with the extension method.

Method Example

```
public static int FindChar(this string str, char chr)
{
    for (int i = 0; i < str.Length; i++)
    {
        if (str[i] == chr) { return i; } // return our match's index.
    }
    return null; // return null if no match was found.
}
```

Calling Convention

```
string string = "This is a test string";
char character = string.FindChar('e');// Returns 'e'.
```

In the example our extension method is added on to the **string** type and tries to find the index of the character provided.
It will return the *index* of the character *or* null if none is found.

When to use an Extension Method

An extension method is a method that will in essence 'add itself' to the list of methods tied to the type of object you're extending. Before converting a method to an extension method, ***consider if your method meets these points*:**

The method exists to do one set repetitive task.

The variables you pass through to the method contain all that it needs to do it's task.

You need to access the method through various scripts or frequently.

The method doesn't need to access any unrelated classes in order to see if it can run.

There are cases where the first point is invalid, such as performing multiple smaller tasks on say a ***string***, but usually an extension method only does one thing and it does it *really well*.

It's not working, what do I do?

If you get an error, these are the first things to check. The access type is shown for what you could be intending to do.

1. The first parameter is missing the keyword 'this'.	`public static int FindChar(string str, char chr)`	Any Access
2. The method isn't static.	`public int FindChar(this string str, char chr)`	Any Access
3. The method isn't public.	`int FindChar(this string str, char chr)`	Global & 'Scoped'
4. You haven't included the namespace the method (and it's class) is in.		'Scoped' Access

For more detailed coverage, Microsoft Learn provides an in-depth resource:
https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/extension-methods

This document has been written for University Centre Leeds - Alexander Gilbertson