

Forum:[Hack Forums](#)

Thread: [C# Switch Expressions](#)

MavenGradle - December 15, 2019, 4:48 pm

Introduction:

This is going to be a short but informative tutorial regarding a new feature that was introduced in C# 8.0 called a switch expression. If you've been keeping up to date about C#, you should already know about this. Chances are, you'll be using it already. This tutorial is mostly meant for those among us who have no clue what this is all about or how switch expressions work in the latest C# language specifications.

Requirements:[Visual Studio 2019](#) / [Jetbrains Rider](#) (latest versions)

[Knowledge about switch statements](#)

Basic understanding of C# programming

Switch statements:

Let's start of by explaining what a switch statement is. I'm going to be brief about this, since this is one of the most basic features that C# has to offer. If you do not know what switch statements are and how they operate, I would recommend that you take a look at the requirements above. There's a link there that contains a lot of information that you can use to learn and enhance your programming skills. Regardless, Microsoft has perfectly laid this information out for us in a single sentence:

Quote:The switch is a selection statement that chooses a single switch section to execute from a list of candidates based on a pattern match with the match expression.

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/switch>

An example of a switch statement:

```
Code int caseSwitch = 1;
switch (caseSwitch)
{
    case 1:
        Console.WriteLine("Case 1");
        break;
    case 2:
        Console.WriteLine("Case 2");
        break;
    case 3:
        Console.WriteLine("Case 3");
        break;
    case 4:
        Console.WriteLine("Case 4");
        break;
    default:
        Console.WriteLine("Default case");
}
```

```
        break;
    }
}
```

Now in this particular example, the output of the console will be "Case 1", seeing how the variable that is being evaluated (caseSwitch) is equal to one and matches the first case. The default case is the one that will be reached when all other cases do not match the evaluation.

These kind of switch statements can greatly improve general code readability and maintainability when compared to if statements. Especially if your if statements are extremely large. In some cases switch statements can perform better than if statements as well, but the opposite can be true as well, so take this with a grain of salt and always select the best method of action.

If we go back to our code and convert the example above which contains a switch statement to an if statement, it would look something like this:

```
Code int caseSwitch = 1;
if (caseSwitch == 1) {
    Console.WriteLine("Case 1");
} else if (caseSwitch == 2) {
    Console.WriteLine("Case 2");
} else if (caseSwitch == 3) {
    Console.WriteLine("Case 3");
} else if (caseSwitch == 4) {
    Console.WriteLine("Case 4");
} else {
    Console.WriteLine("Default case");
}
```

As you can see, it becomes harder and harder to maintain as more options are added to the list of cases that can be handled.

Switch expressions:

Now imagine you're using a switch statement to evaluate an input and to return some kind of value, depending on the case, in a method:

```
Code private string GetResult(int input) {
    switch (input)
    {
        case 1:
            return "a";
        case 2:
            return "b";
        case 3:
            return "c";
        case 4:
            return "d";
        case 5:
            return "e";
        case 6:
            return "f";
        case 7:
            return "g";
        default:
            throw new ArgumentOutOfRangeException(nameof(input));
    }
}
```

With the new switch expressions, the example above can be simplified to the following:

```
Code private string GetResult(int input) =>
    input switch
    {
        1 => "a",
        2 => "b",
        3 => "c",
        4 => "d",
        5 => "e",
        6 => "f",
        7 => "g",
        _ => throw new ArgumentOutOfRangeException(nameof(input)),
    };
```

The default case has been replaced with the "_" character, simplifying and reducing the amount of text you have to type. In addition to that, arrow functions are used to remove the braces of the function and to indicate a return value, this once again reduces the amount of text you, as a programmer, have to type and increases readability. All of this effectively turn the body to an expression and as such, it can no longer be considered a statement.

The variable that is being evaluated is also placed before the switch keyword. It has been implemented this way to make it easier to distinguish switch expressions and switch statements.

Switch expressions can also be used to assign (or yield) a value to a variable depending on the case, like so:

```
Code var result = input switch
{
    1 => true,
    2 => true,
    _ => false
};
```

Some thoughts:

All in all, these changes are made to increase the readability and maintainability of your code. It is also useful to know that C# 8.0 features are available in Visual Studio 2019 and of course in the wonderful JetBrains Rider. These features have to be explicitly turned on before you can use them or you will face errors and warnings. Do note that you should also be using the latest version of your IDE, because these features, for example, will not work in Visual Studio 2010.

In order to enable C# 8.0 you will need to edit your .csproj file and use the following attribute for your configurations:

```
Code <LangVersion>8.0</LangVersion>
```

I hope you learned something.

Pawnhub - December 15, 2019, 10:14 pm

Thanks for posting maven, I didn't know about this yet.

Seems great for clean code unlike if and normal switches.

Narwin - December 16, 2019, 5:40 am

Note: C# version 8.0 doesn't officially support .NET Framework but .NET Core 3.x and .NET Standard 2.1+, using this version for .NET Framework (4.7.2 and above) requires [Microsoft.Net.Compilers](https://docs.microsoft.com/en-us/dotnet/core/install/linux-rhel).

Good Article

MavenGradle - December 16, 2019, 1:08 pm

(Dec 16, 2019 06:40 AM)Narwin Wrote: [\[CITE\]](#) Note: C# version 8.0 doesn't officially support .NET Framework but .NET Core 3.x and .NET Standard 2.1+, using this version for .NET Framework (4.7.2 and above) requires [Microsoft.Net.Compilers](#).

Good Article

Good note, but to add to this:

Visual Studio 2019 supports C# 8.0 out of the box (latest version that is) without the need of installing any nuget packages, even for the .NET Framework. However, there's more to it than that, like you are saying. Someone on stackoverflow made an extensive comment about all of this: <https://stackoverflow.com/a/57020770>

asmx86 - December 16, 2019, 10:55 pm

Thanks for the information MavenGradle, it's good to know!

riu41776 - December 18, 2019, 1:59 am

How in depth can you go with a simple concept such as switch statement...?

MavenGradle - December 18, 2019, 4:06 am

(Dec 18, 2019 02:59 AM)riu41776 Wrote: [\[CITE\]](#) How in depth can you go with a simple concept such as switch statement...?

As much as needed. The more details the better. Besides, this is about switch expressions and not statements.

In fact adding two more paragraphs would be easy since there's much to tell about them, no matter how 'simple' they might seem.

SIM-3867235 - December 18, 2019, 11:09 am

This is great info! Might use this in my programming class! Amazing contribution!

MavenGradle - December 20, 2019, 2:03 am

(Dec 18, 2019 12:09 PM)waytoosplashy Wrote: [\[CITE\]](#) This is great info! Might use this in my programming class! Amazing contribution!

Be careful, you might make the teacher look like a fool for not keeping up to date! They don't like it when you do that, at least in my experience.

DNA - December 29, 2019, 5:18 pm

A very clean tutorial.

Def's gonna +5 you when I'm a Vendor. You're doing the community a great service by guiding the novices with tutorials like these.

Shisho - December 29, 2019, 6:30 pm

I disagree with the notion that if-else statements can perform better than switch statements. Switch statements only need to check for the value of the expression to select the appropriate case. If-else statements need to evaluate and then check the value of the expression and if not correct, require to move on to the next if-else and consequently get worse as the if-else nest gets longer and more complex.

Switch statements are faster and more easily maintainable. There are just cases where if-else statements are easier to implement.

Washi - December 30, 2019, 6:47 am

(Dec 29, 2019 07:30 PM)Shisho Wrote: [\[CITE\]](#) I disagree with the notion that if-else statements can perform better than switch statements. Switch statements only need to check for the value of the expression to select the appropriate case. If-else statements need to evaluate and then check the value of the expression and if not correct, require to move on to the next if-else and consequently get worse as the if-else nest gets longer and more complex.

Switch statements are faster and more easily maintainable. There are just cases where if-else statements are easier to implement.

This assumes that you are using a switch statement involving integral types only, as the integer is used as an index inside a jump table. For switches with strings or pattern matching, the switch is actually compiled as if it were a series of if-else-if statements.

Suspect. - *December 31, 2019, 5:21 am*

I like to use switch statements mostly on buttons, wasn't aware of this thanks for the share boom boom