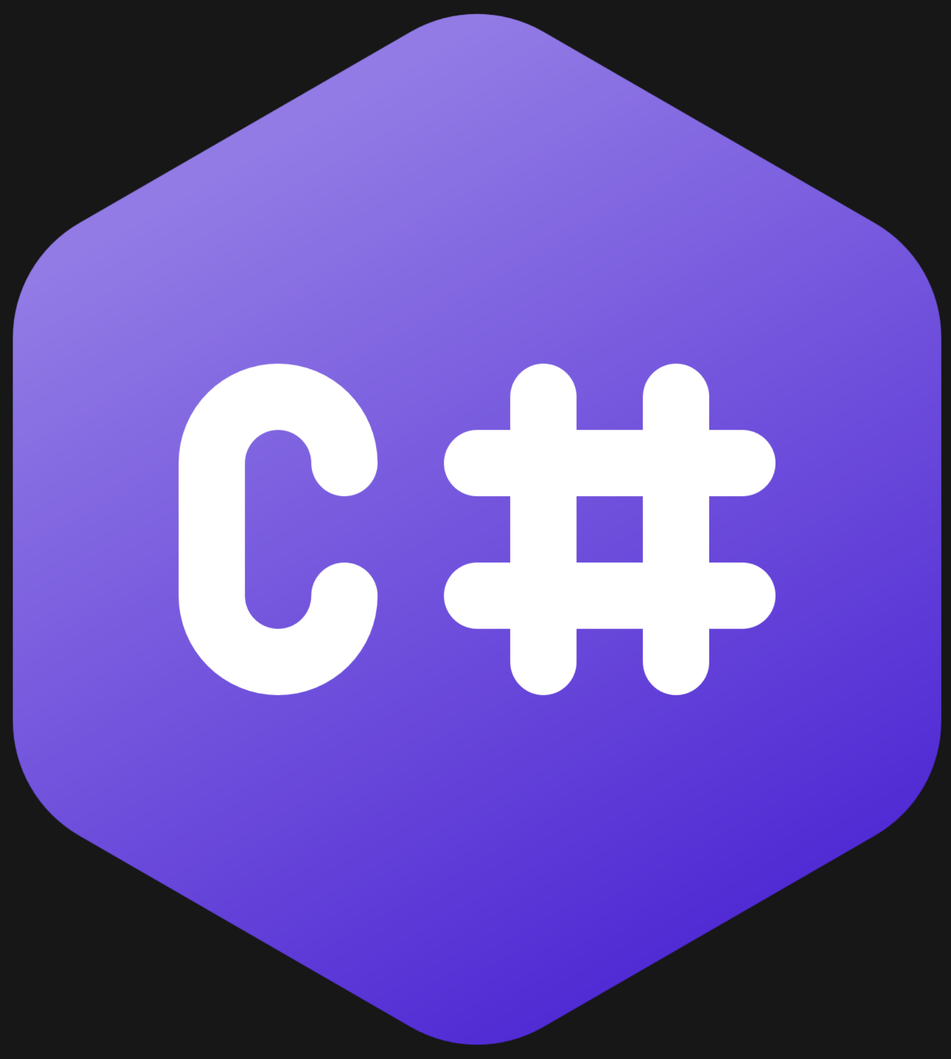
Key Features review



9

10

11

12

Nabi Karampoor
@thisisnabi

1 Top Level Statement

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Hello, world!");
    }
}
```

Allow you to write C# code without explicitly defining a class or a Main method

```
using System;

Console.WriteLine("Hello, world!");
```

Statement is at the highest level

Nabi Karampoor
@thisisnabi

2 Global Using Directive

You can consolidate all commonly used namespaces across the project in a dedicated file

Usings.cs

```
global using System;
global using System.Collections.Generic;
```

Access types from these namespaces without explicitly including the using directives

FileA.cs

```
class FileA
{
    public List<int> MyList { get; set; }
}
```

FileB.cs

```
class FileB
{
    public List<int> MyList { get; set; }
}
```

Nabi Karampoor
@thisisnabi

3 Const Interpolated

```
const string Name = "Benz";
const string Designation = $"Benz - E250";
```

Interpolated strings refers string literals that contain expressions or variables enclosed in curly braces within the string.

```
const string Name = "Benz";
const string Designation = $"{Name} - E250";
```

Nabi Karampoor
@thisisnabi

4 Extended Property Pattern

Before C# 10

```
if (obj is Person { Address: { City: "Seattle" } })
    Console.WriteLine("Seattle");
```

```
object obj = new Person
{
    FirstName = "Kathleen",
    LastName = "Dollard",
    Address = new Address { City = "Seattle" }
};
```

```
if (obj is Person { Address.City: "Seattle" })
    Console.WriteLine("Seattle");
```

Easier to access nested property values

Nabi Karampoor
@thisisnabi

5 Caller Expression attribute

```
void CheckExpression(bool condition,
    [CallerArgumentExpression("condition")] string? message = null )
{
    Console.WriteLine($"Condition: {message}");
}
```

```
CheckExpression(true);
CheckExpression(b);
CheckExpression(a > 5);
```

Output

```
// Condition: true
// Condition: b
// Condition: a > 5
```

Nabi Karampoor
@thisisnabi

Convert your condition into the string and pass to method as an optional param

6 Natural type for lambdas

Before C# 10

```
Func<string, int> parse = (string s) => int.Parse(s);
```

```
var parse = (string s) => int.Parse(s);
```

do not need a Target Type

Nabi Karampoor
@thisisnabi

7 Attributes on lambdas

```
Func<string, int> parse = [Example(1)] (s) => int.Parse(s);
```

Just like local functions applied AttributeTargets.Method

Nabi Karampoor
@thisisnabi

8 File-Scoped namespace

```
namespace SomeNamespace
{
    class SomeClass
    {

    }
}
```

```
namespace SomeNamespace;

class SomeClass
{

}
```

This removes a level of indentation, making your code simpler and easier to understand.

Nabi Karampoor
@thisisnabi