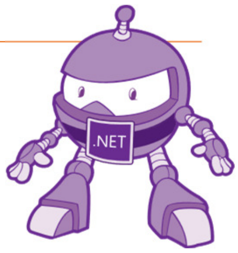
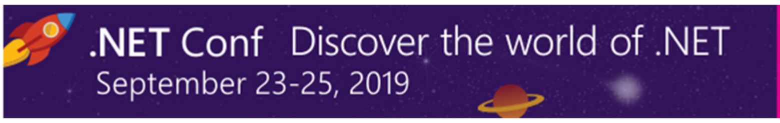
What’s new in C# 8.0

MANOJ KUMAR SHARMA
Platform & Developer Evangelist
mailme@manojkumarsharma.com
LinkedIn http://www.linkedin.com/in/contact4manoj





.NET Conf 2019
Local Event
Bengaluru
19-OCT-2019

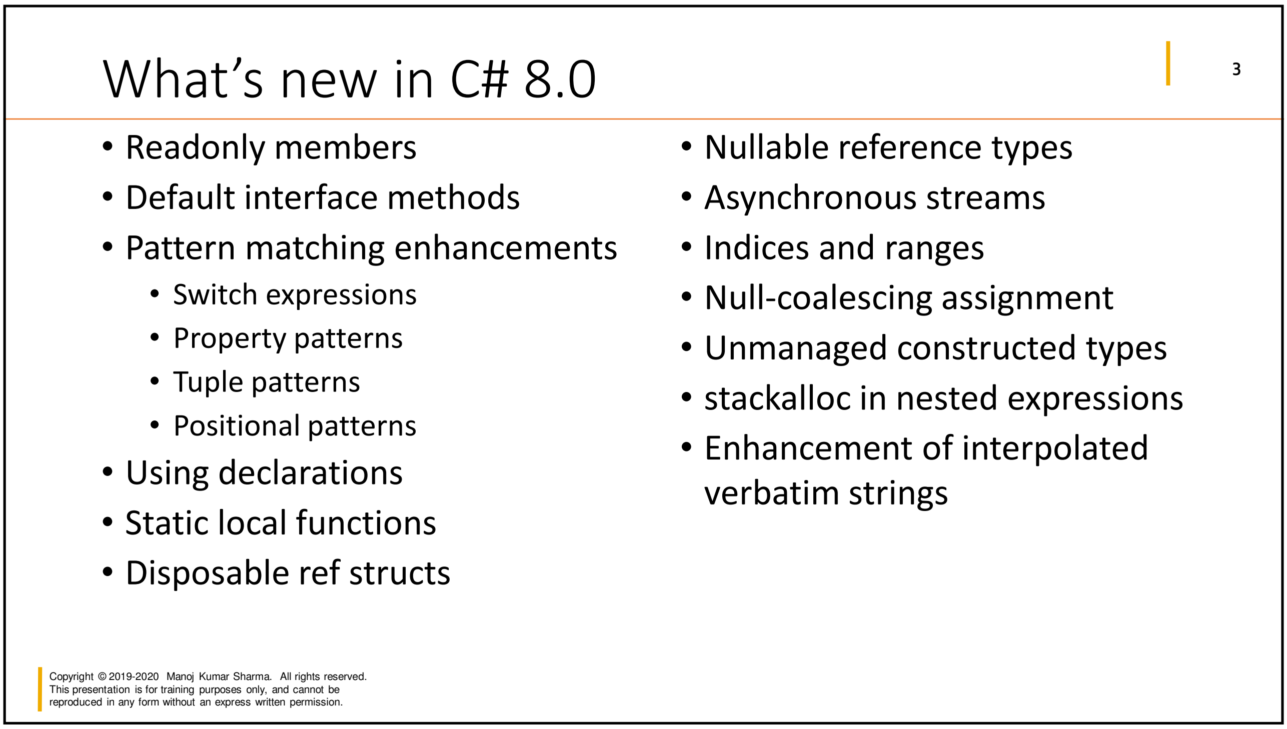
bdotnet.in
People, Technology & Passion

1

Introduction

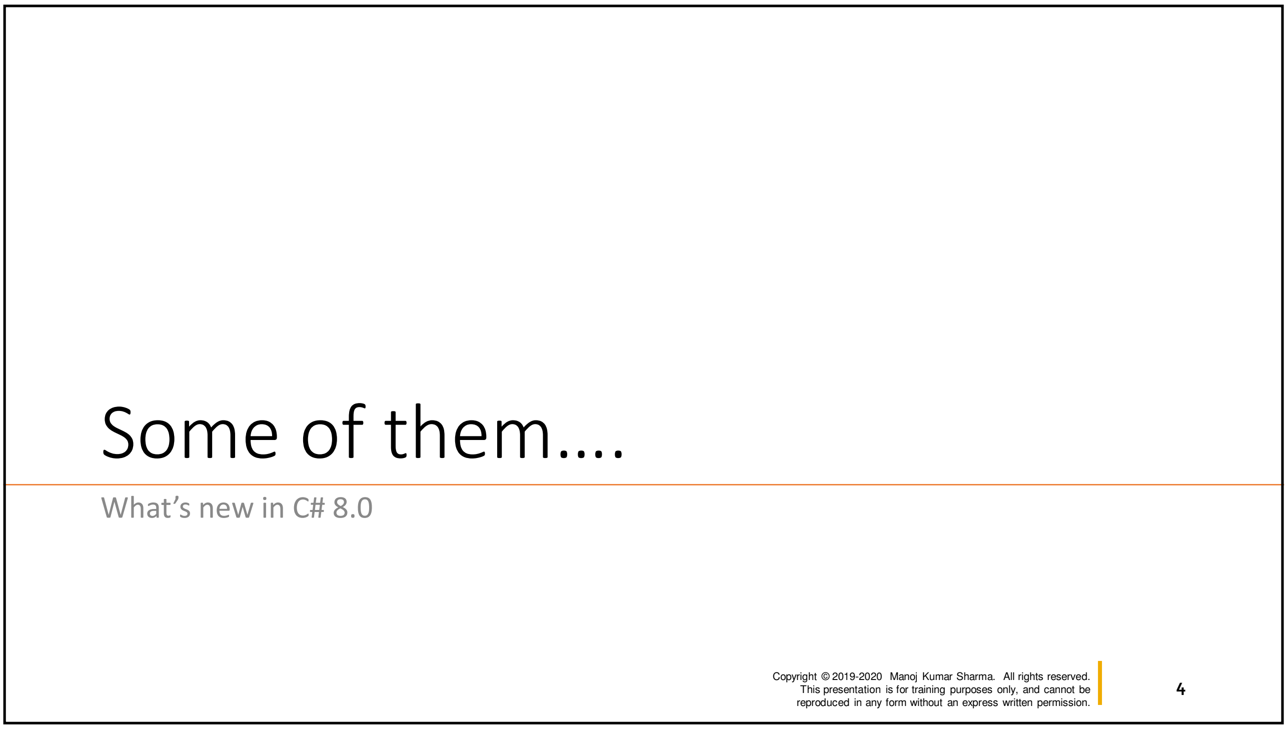
What’s new in C# 8.0

2



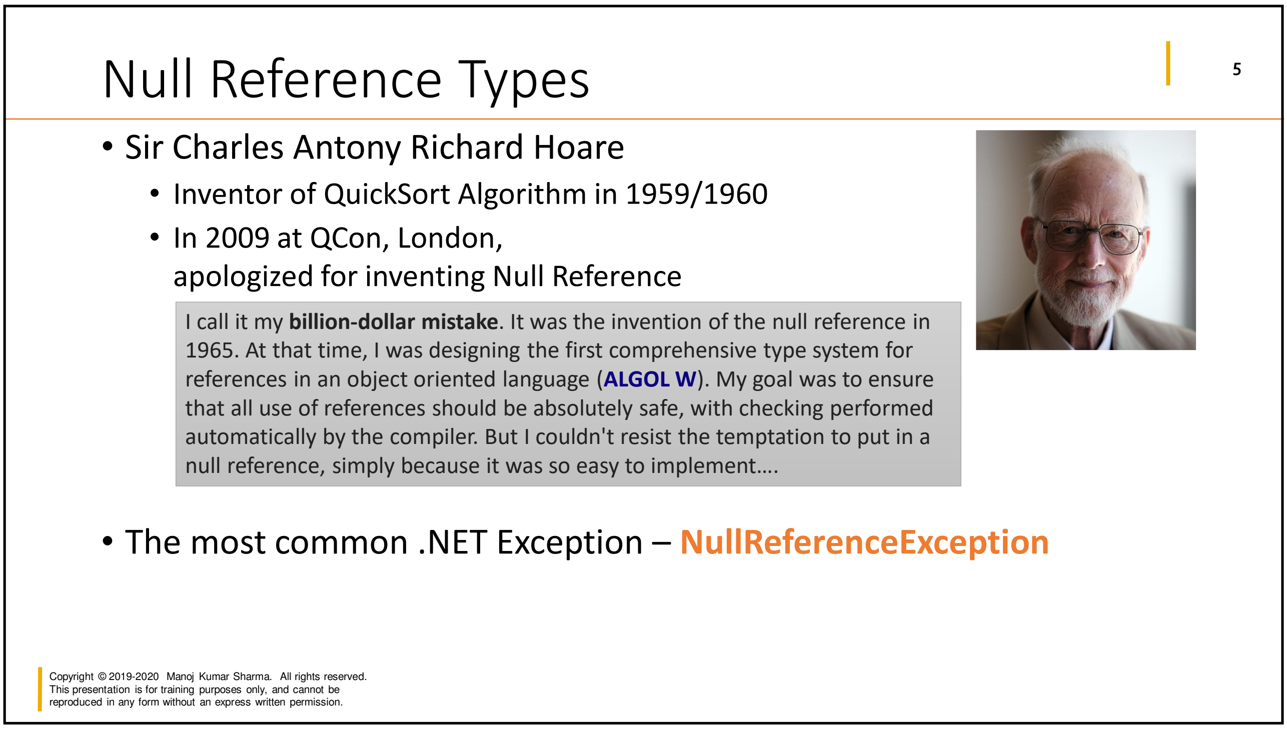
What’s new in C# 8.0

- Readonly members
- Default interface methods
- Pattern matching enhancements
  - Switch expressions
  - Property patterns
  - Tuple patterns
  - Positional patterns
- Using declarations
- Static local functions
- Disposable ref structs
- Nullable reference types
- Asynchronous streams
- Indices and ranges
- Null-coalescing assignment
- Unmanaged constructed types
- stackalloc in nested expressions
- Enhancement of interpolated verbatim strings



Some of them….

What’s new in C# 8.0

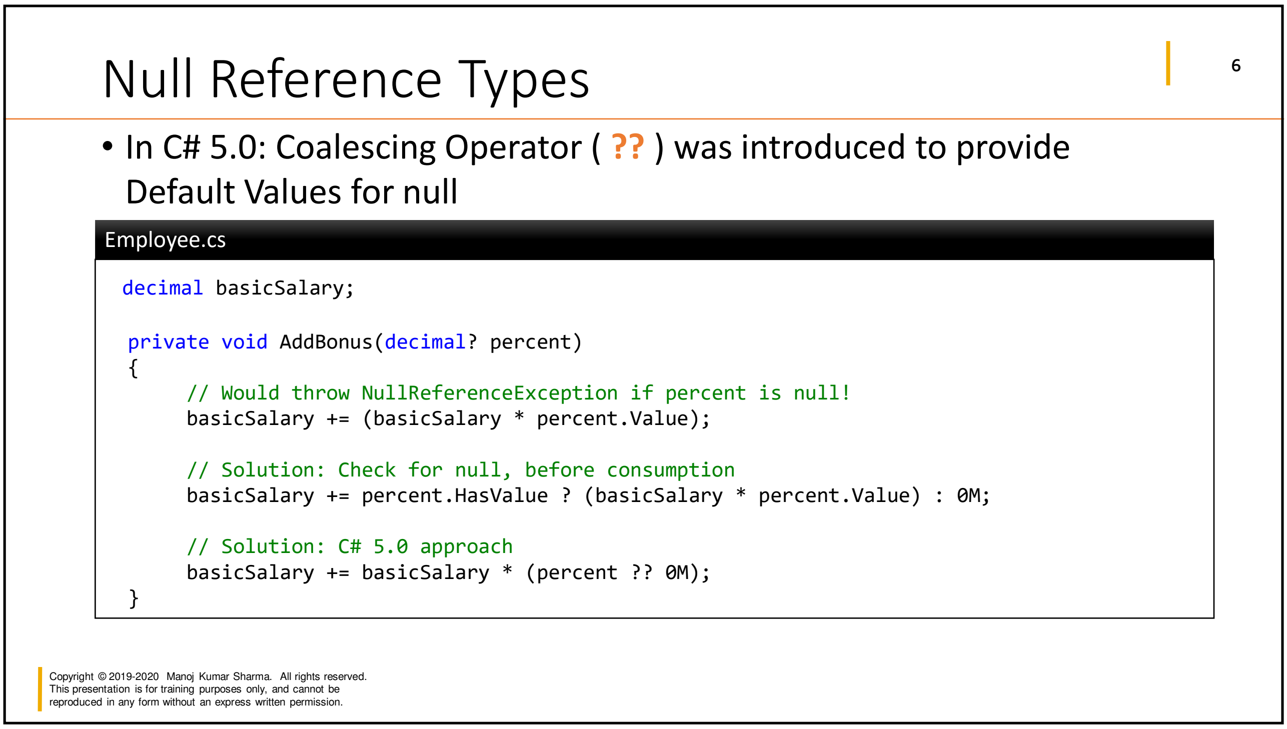


Null Reference Types

- Sir Charles Antony Richard Hoare
  - Inventor of QuickSort Algorithm in 1959/1960
  - In 2009 at QCon, London,
    apologized for inventing Null Reference

> I call it my **billion-dollar mistake**. It was the invention of the null reference in 1965. At that time, I was designing the first comprehensive type system for references in an object oriented language (**ALGOL W**). My goal was to ensure that all use of references should be absolutely safe, with checking performed automatically by the compiler. But I couldn't resist the temptation to put in a null reference, simply because it was so easy to implement….

- The most common .NET Exception – **NullReferenceException**

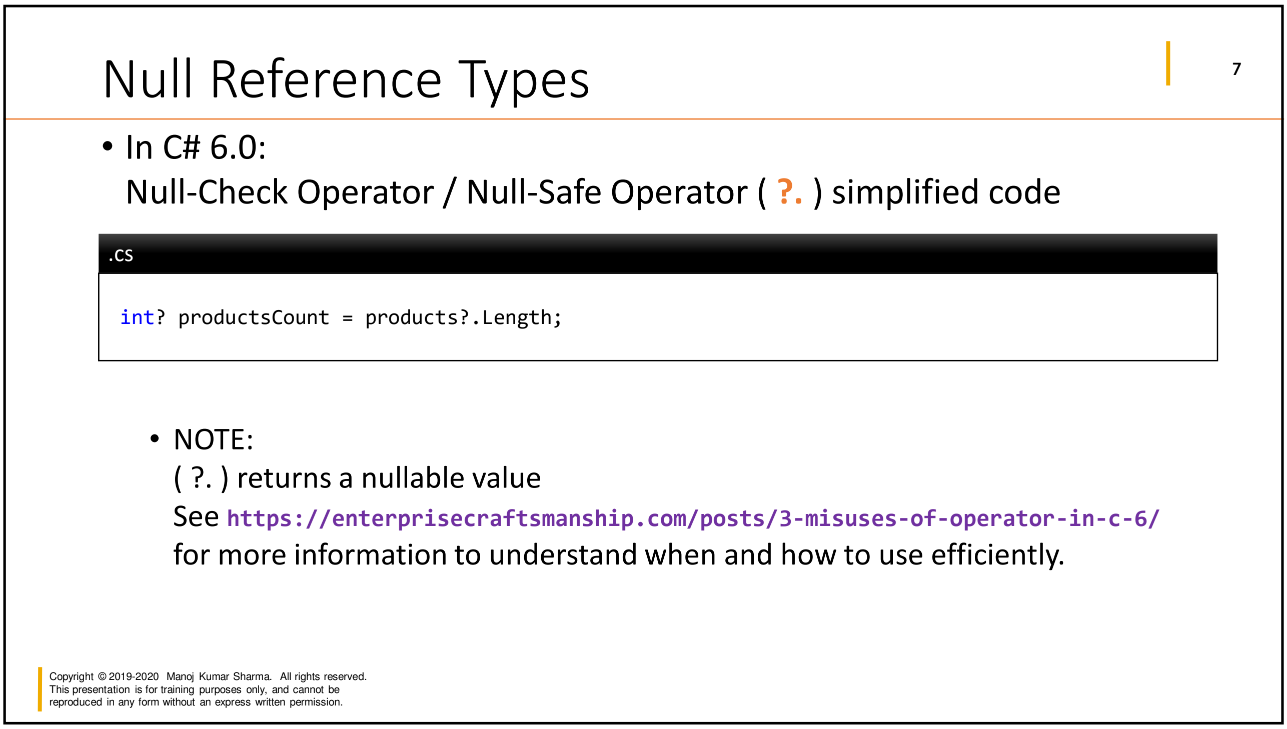


Null Reference Types

- In C# 5.0: Coalescing Operator (??) was introduced to provide Default Values for null

Employee.cs

```
decimal basicSalary;

private void AddBonus(decimal? percent)
{
    // Would throw NullReferenceException if percent is null!
    basicSalary += (basicSalary * percent.Value);

    // Solution: Check for null, before consumption
    basicSalary += percent.HasValue ? (basicSalary * percent.Value) : 0M;

    // Solution: C# 5.0 approach
    basicSalary += basicSalary * (percent ?? 0M);
}
```



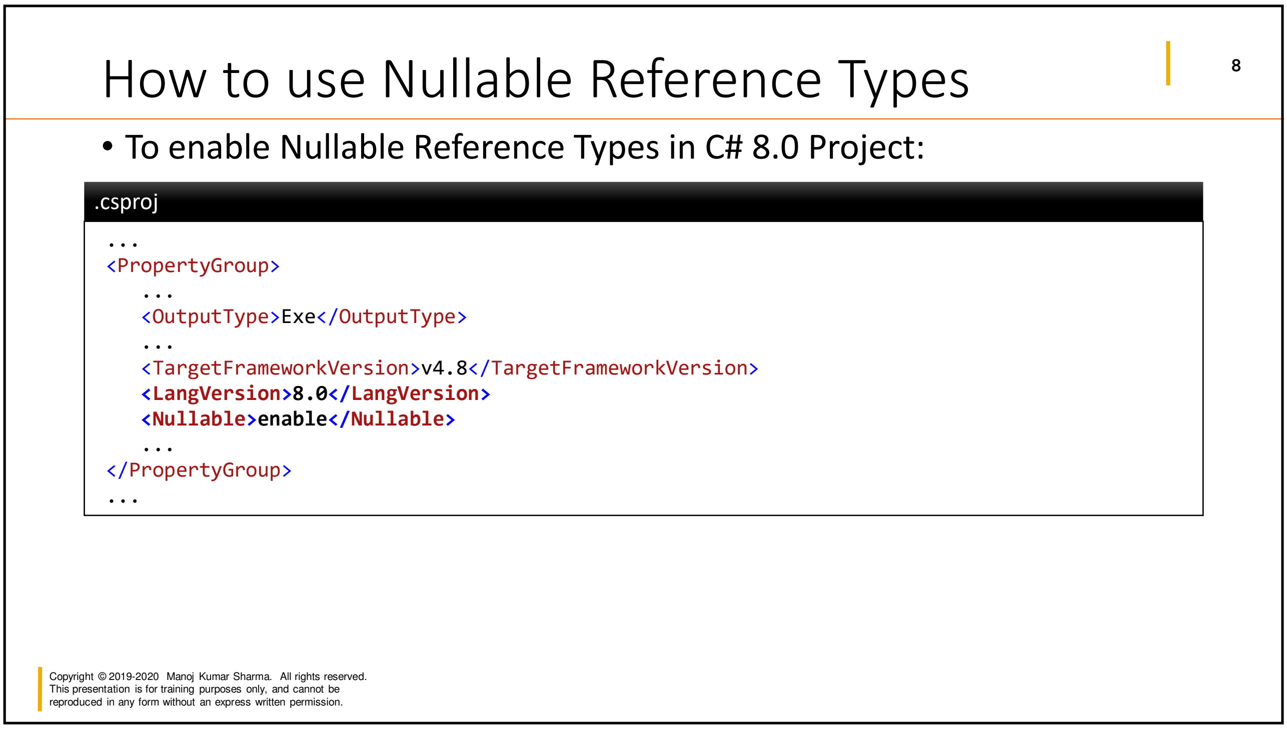
Null Reference Types

- In C# 6.0:
  Null-Check Operator / Null-Safe Operator (**?.**) simplified code

.cs

```
int? productsCount = products?.Length;
```

- NOTE:
  (?.) returns a nullable value
  See `https://enterprisecraftsmanship.com/posts/3-misuses-of-operator-in-c-6/`
  for more information to understand when and how to use efficiently.

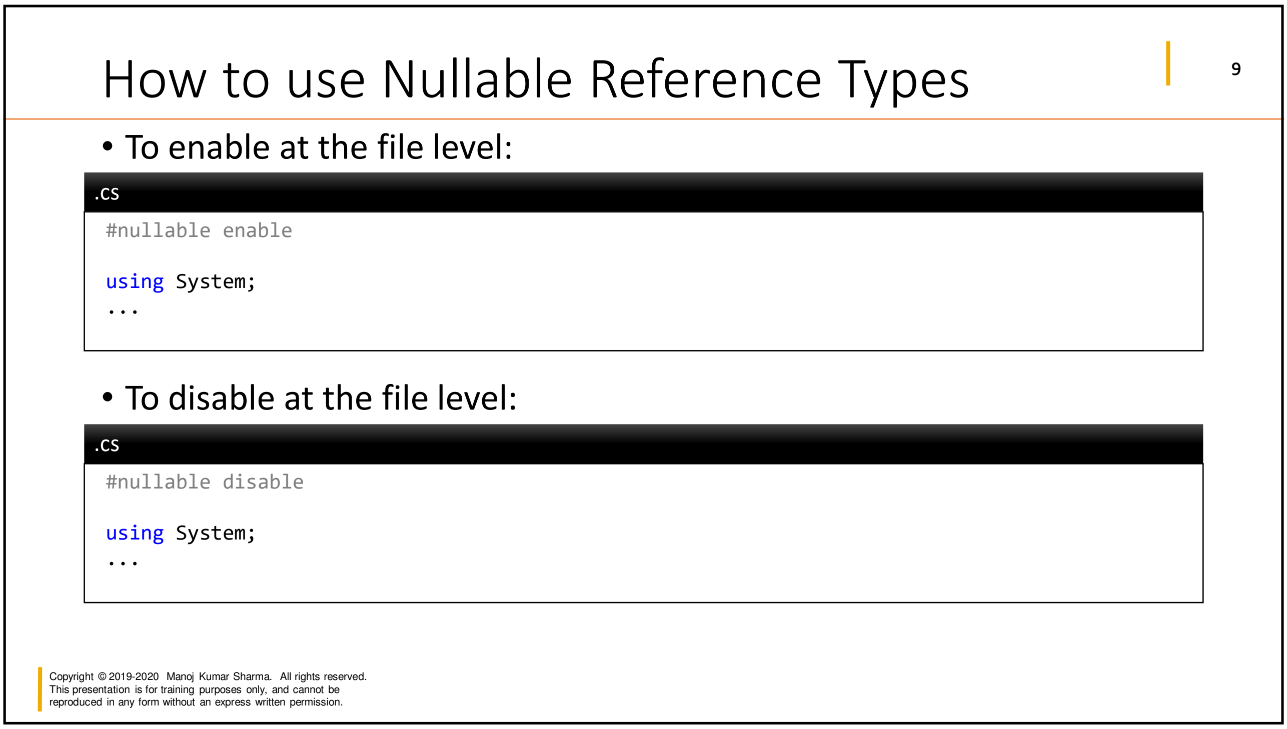


How to use Nullable Reference Types

- To enable Nullable Reference Types in C# 8.0 Project:

.csproj

```
...
<PropertyGroup>
   ...
   <OutputType>Exe</OutputType>
   ...
   <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
   <LangVersion>8.0</LangVersion>
   <Nullable>enable</Nullable>
   ...
</PropertyGroup>
...
```



How to use Nullable Reference Types

- To enable at the file level:

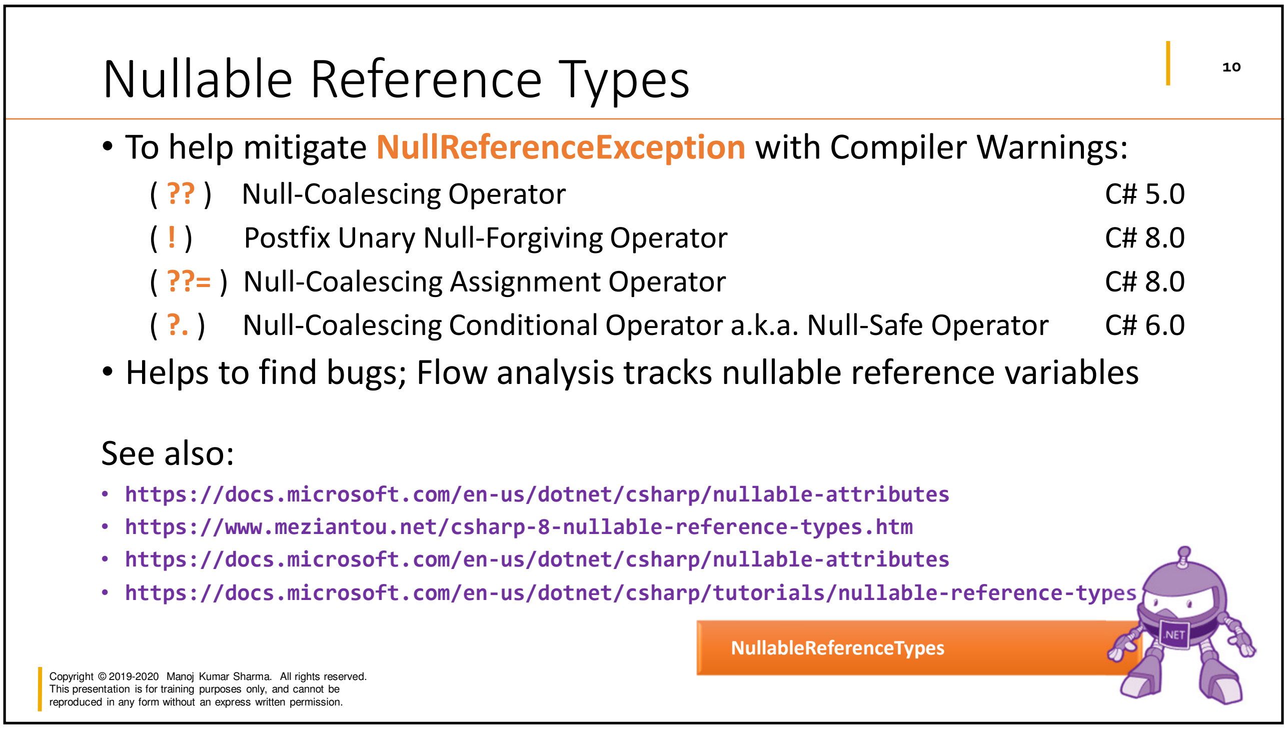
.cs

```
#nullable enable

using System;
...
```

- To disable at the file level:

.cs

```
#nullable disable

using System;
...
```



Nullable Reference Types

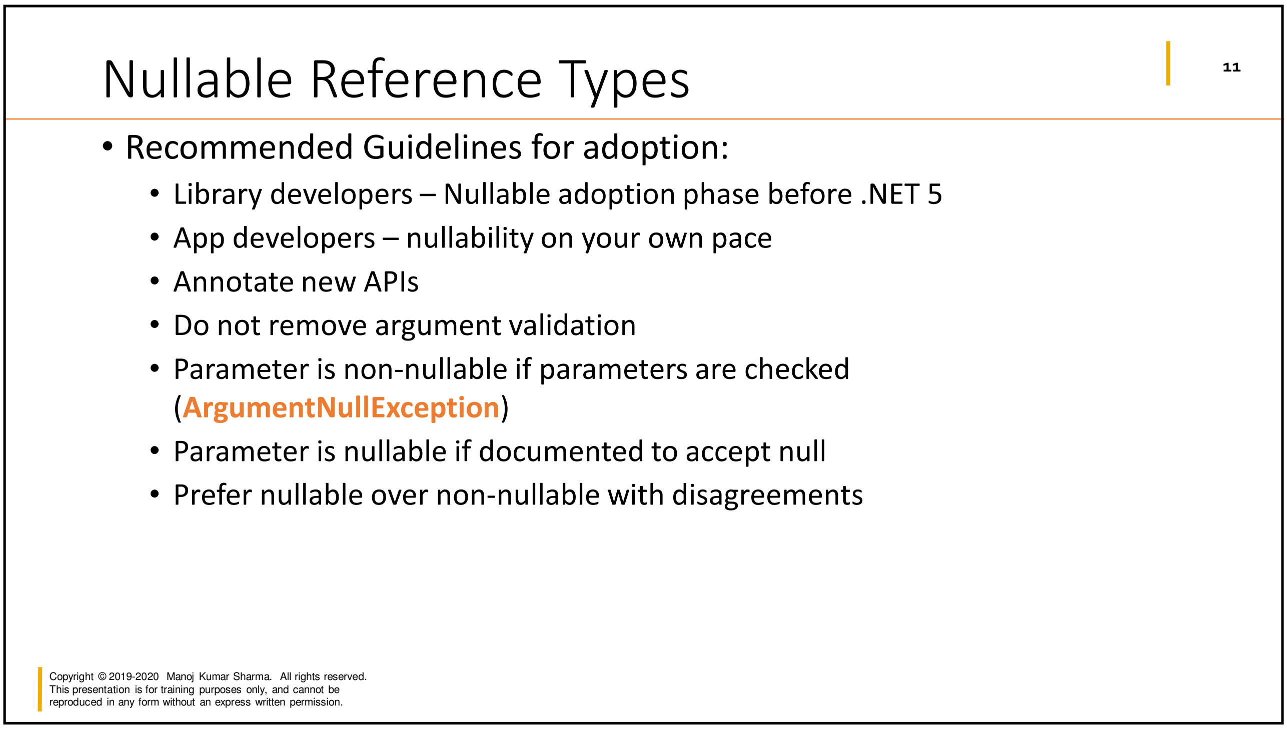
- To help mitigate **NullReferenceException** with Compiler Warnings:

(??)	Null-Coalescing Operator	C# 5.0
(!)	Postfix Unary Null-Forgiving Operator	C# 8.0
(??=)	Null-Coalescing Assignment Operator	C# 8.0
(?.)	Null-Coalescing Conditional Operator a.k.a. Null-Safe Operator	C# 6.0

- Helps to find bugs; Flow analysis tracks nullable reference variables

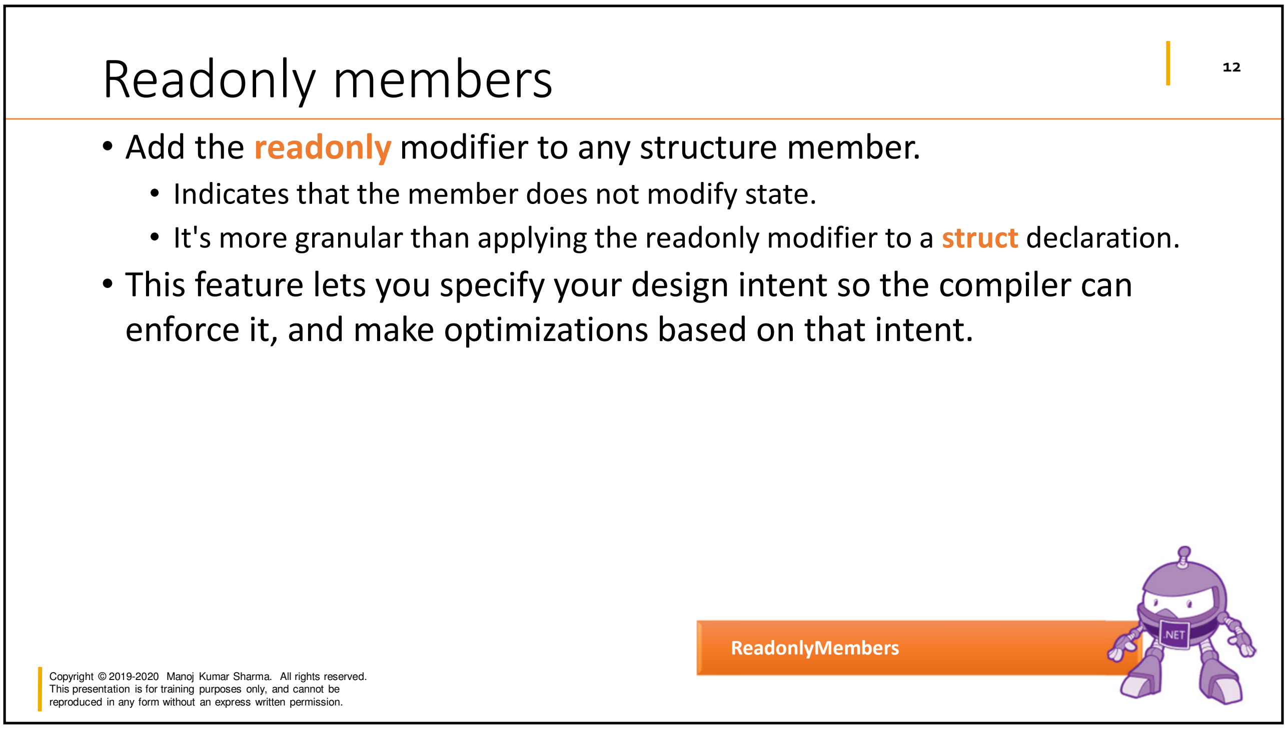
See also:

- https://docs.microsoft.com/en-us/dotnet/csharp/nullable-attributes
- https://www.meziantou.net/csharp-8-nullable-reference-types.htm
- https://docs.microsoft.com/en-us/dotnet/csharp/nullable-attributes
- https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/nullable-reference-types



Nullable Reference Types

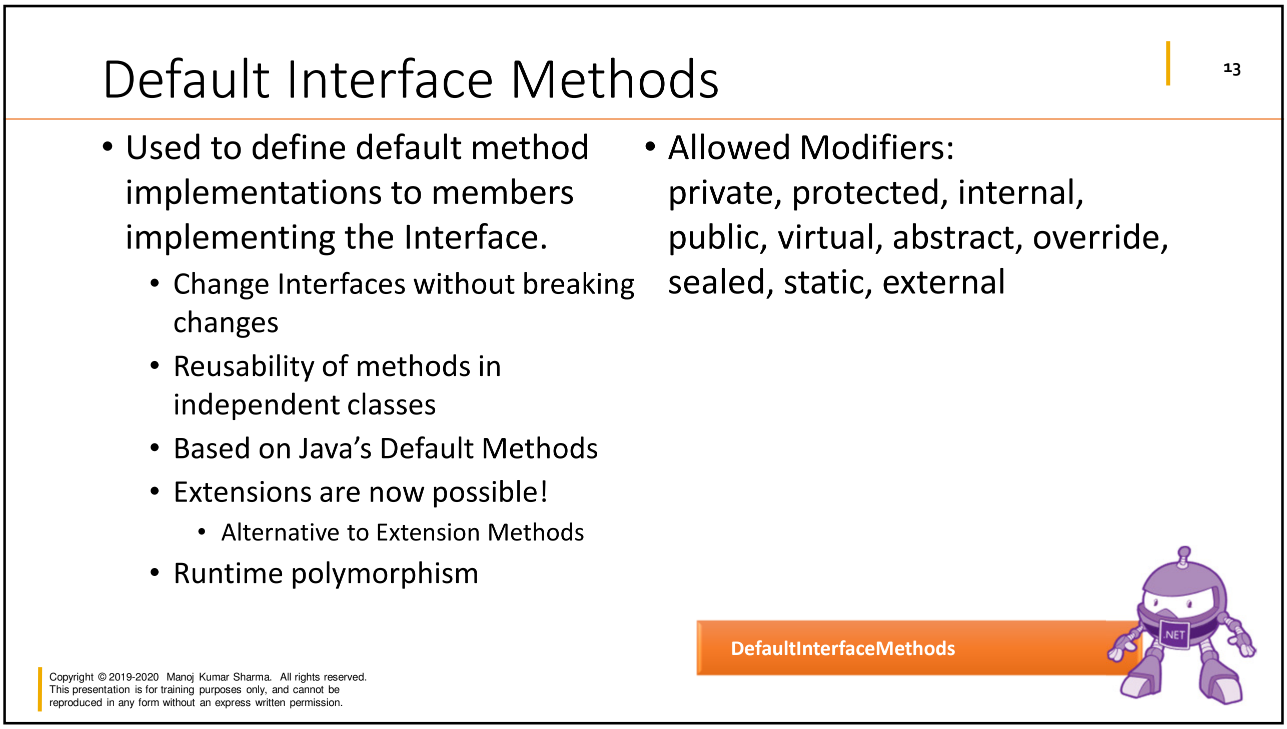
- Recommended Guidelines for adoption:
  - Library developers – Nullable adoption phase before .NET 5
  - App developers – nullability on your own pace
  - Annotate new APIs
  - Do not remove argument validation
  - Parameter is non-nullable if parameters are checked (**ArgumentNullException**)
  - Parameter is nullable if documented to accept null
  - Prefer nullable over non-nullable with disagreements



Readonly members

- Add the **readonly** modifier to any structure member.
  - Indicates that the member does not modify state.
  - It's more granular than applying the readonly modifier to a **struct** declaration.
- This feature lets you specify your design intent so the compiler can enforce it, and make optimizations based on that intent.

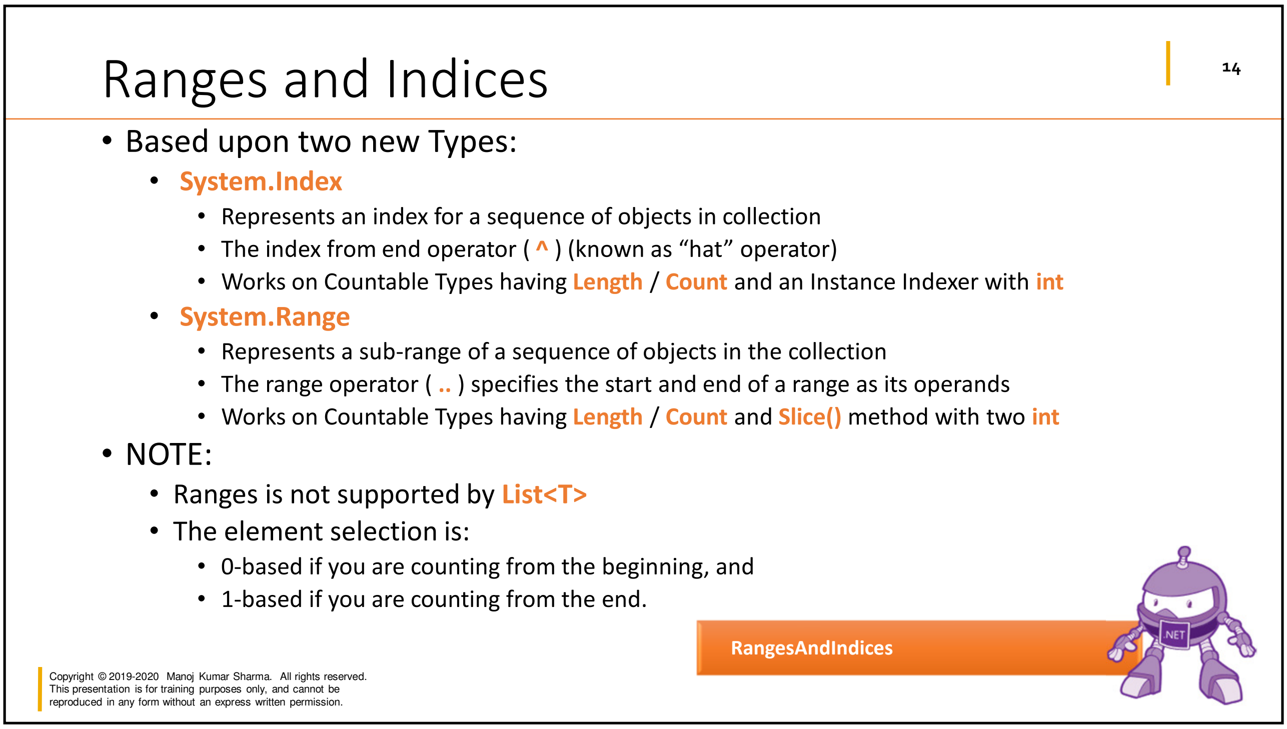
ReadonlyMembers



Default Interface Methods

- Used to define default method implementations to members implementing the Interface.
  - Change Interfaces without breaking changes
  - Reusability of methods in independent classes
  - Based on Java’s Default Methods
  - Extensions are now possible!
    - Alternative to Extension Methods
  - Runtime polymorphism
- Allowed Modifiers: private, protected, internal, public, virtual, abstract, override, sealed, static, external

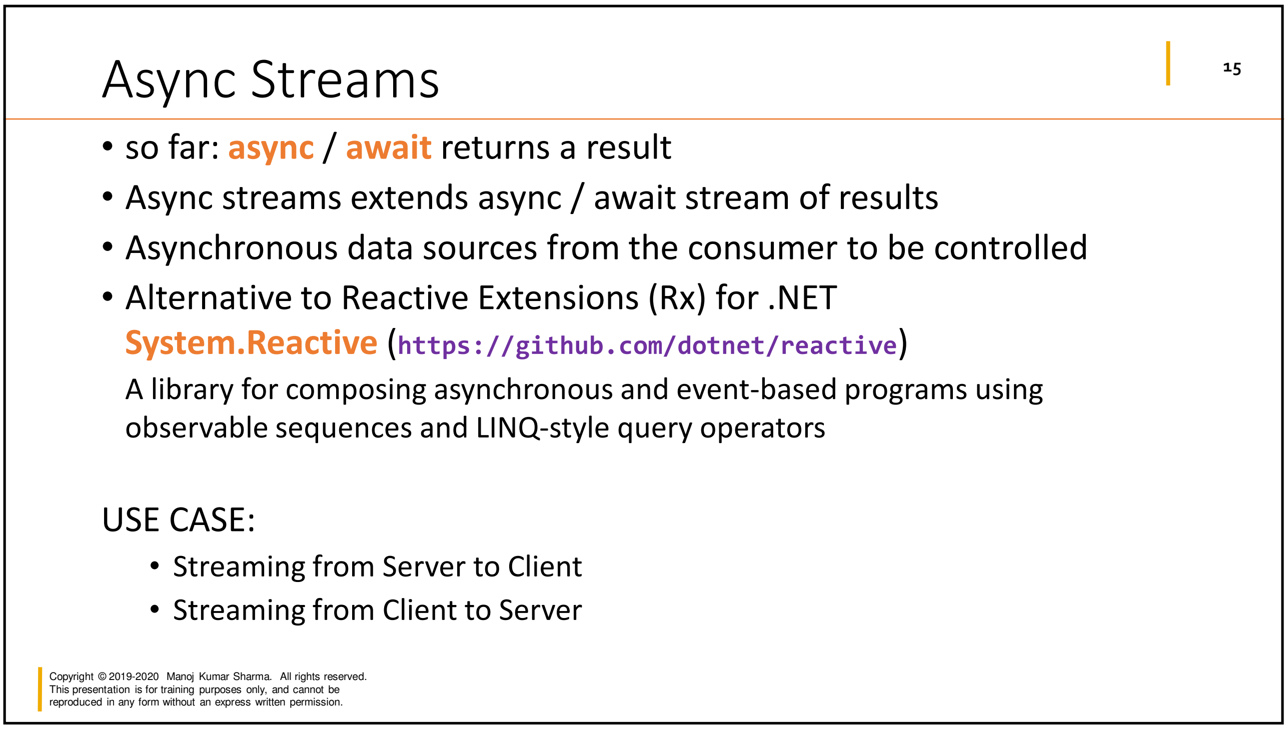
DefaultInterfaceMethods



Ranges and Indices

- Based upon two new Types:
  - **System.Index**
    - Represents an index for a sequence of objects in collection
    - The index from end operator (**^**) (known as “hat” operator)
    - Works on Countable Types having **Length** / **Count** and an Instance Indexer with **int**
  - **System.Range**
    - Represents a sub-range of a sequence of objects in the collection
    - The range operator (**..**) specifies the start and end of a range as its operands
    - Works on Countable Types having **Length** / **Count** and **Slice()** method with two **int**
- NOTE:
  - Ranges is not supported by **List<T>**
  - The element selection is:
    - 0-based if you are counting from the beginning, and
    - 1-based if you are counting from the end.

RangesAndIndices



Async Streams

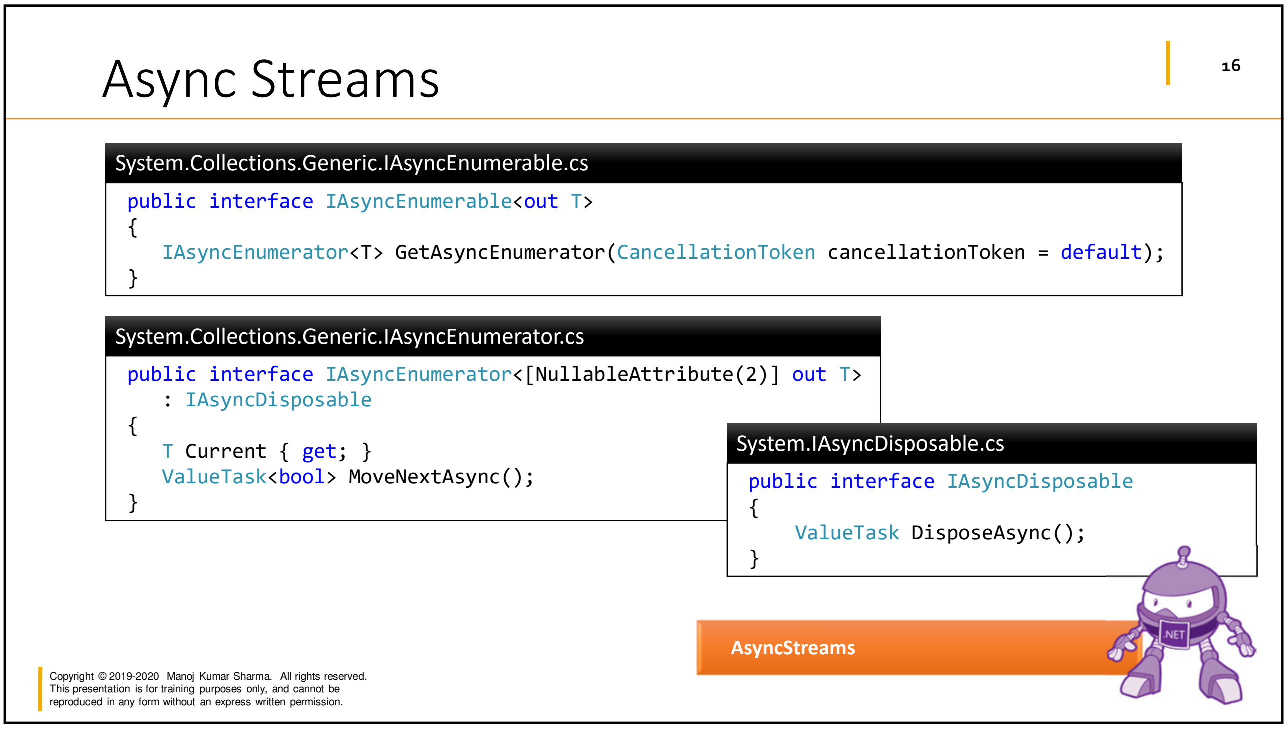
- so far: **async** / **await** returns a result
- Async streams extends async / await stream of results
- Asynchronous data sources from the consumer to be controlled
- Alternative to Reactive Extensions (Rx) for .NET

  System.Reactive (`https://github.com/dotnet/reactive`)

  A library for composing asynchronous and event-based programs using observable sequences and LINQ-style query operators

USE CASE:

- Streaming from Server to Client
- Streaming from Client to Server



Async Streams

System.Collections.Generic.IAsyncEnumerable.cs

```
public interface IAsyncEnumerable<out T>
{
    IAsyncEnumerator<T> GetAsyncEnumerator(CancellationToken cancellationToken = default);
}
```

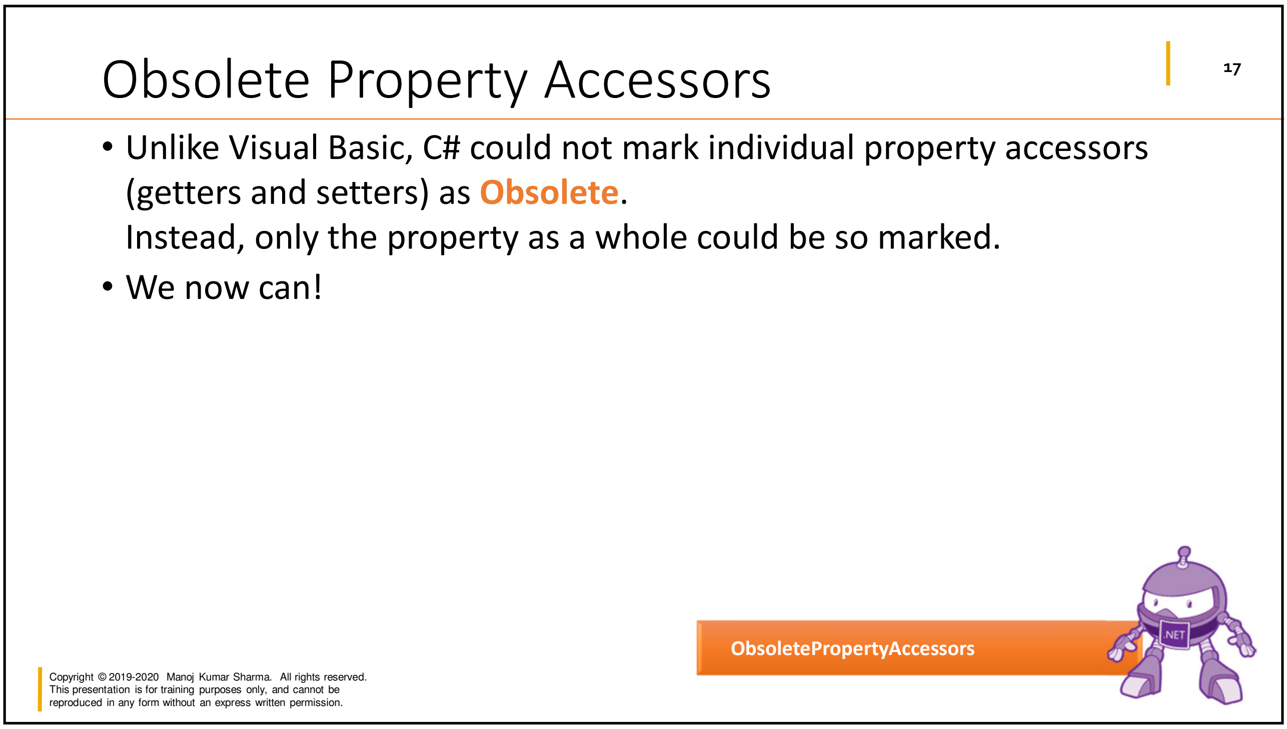
System.Collections.Generic.IAsyncEnumerator.cs

```
public interface IAsyncEnumerator<[NullableAttribute(2)] out T>
    : IAsyncDisposable
{
    T Current { get; }
    ValueTask<bool> MoveNextAsync();
}
```

System.IAsyncDisposable.cs

```
public interface IAsyncDisposable
{
    ValueTask DisposeAsync();
}
```

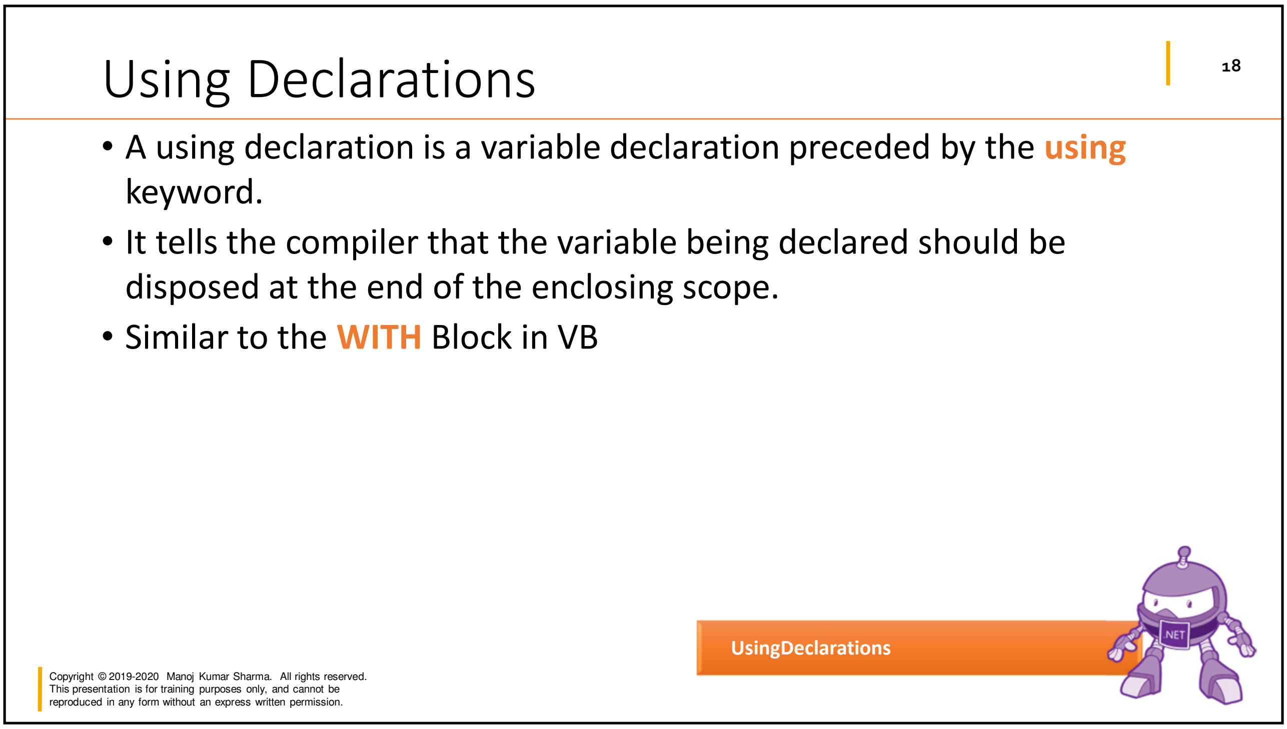
AsyncStreams



Obsolete Property Accessors

- Unlike Visual Basic, C# could not mark individual property accessors (getters and setters) as **Obsolete**.
  Instead, only the property as a whole could be so marked.
- We now can!

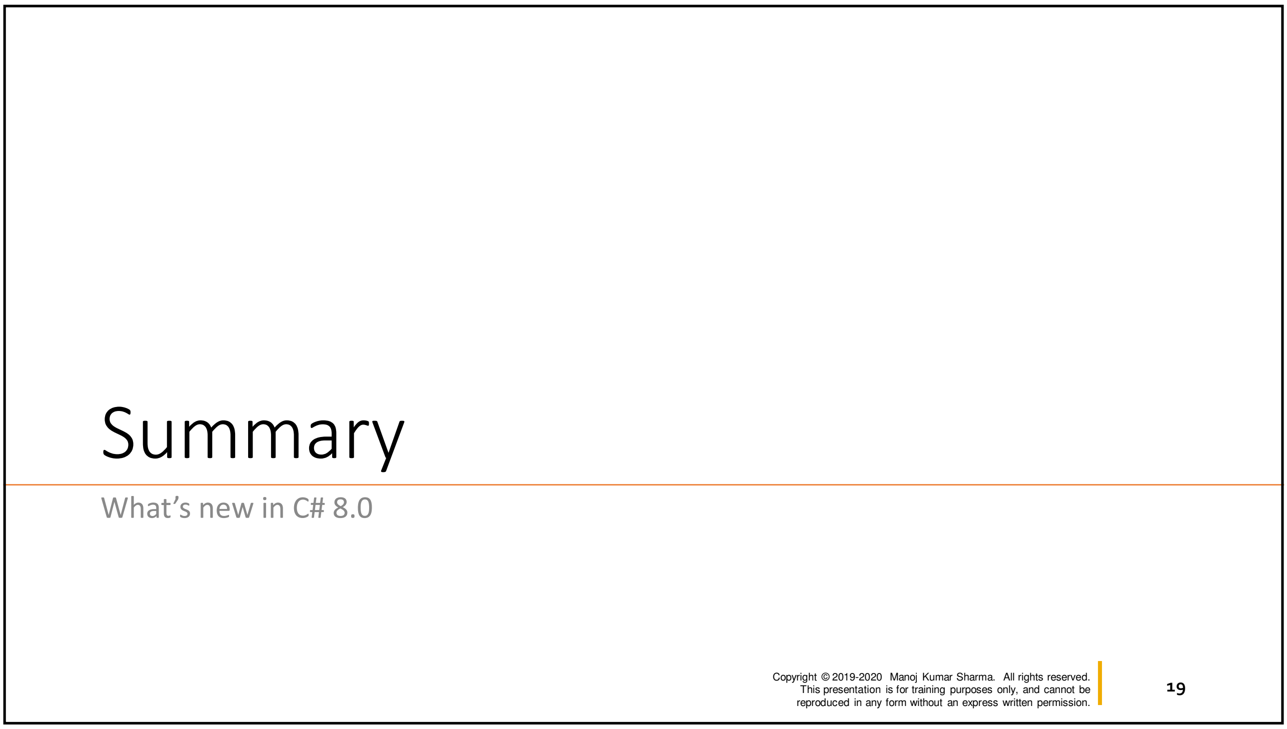
ObsoletePropertyAccessors



Using Declarations

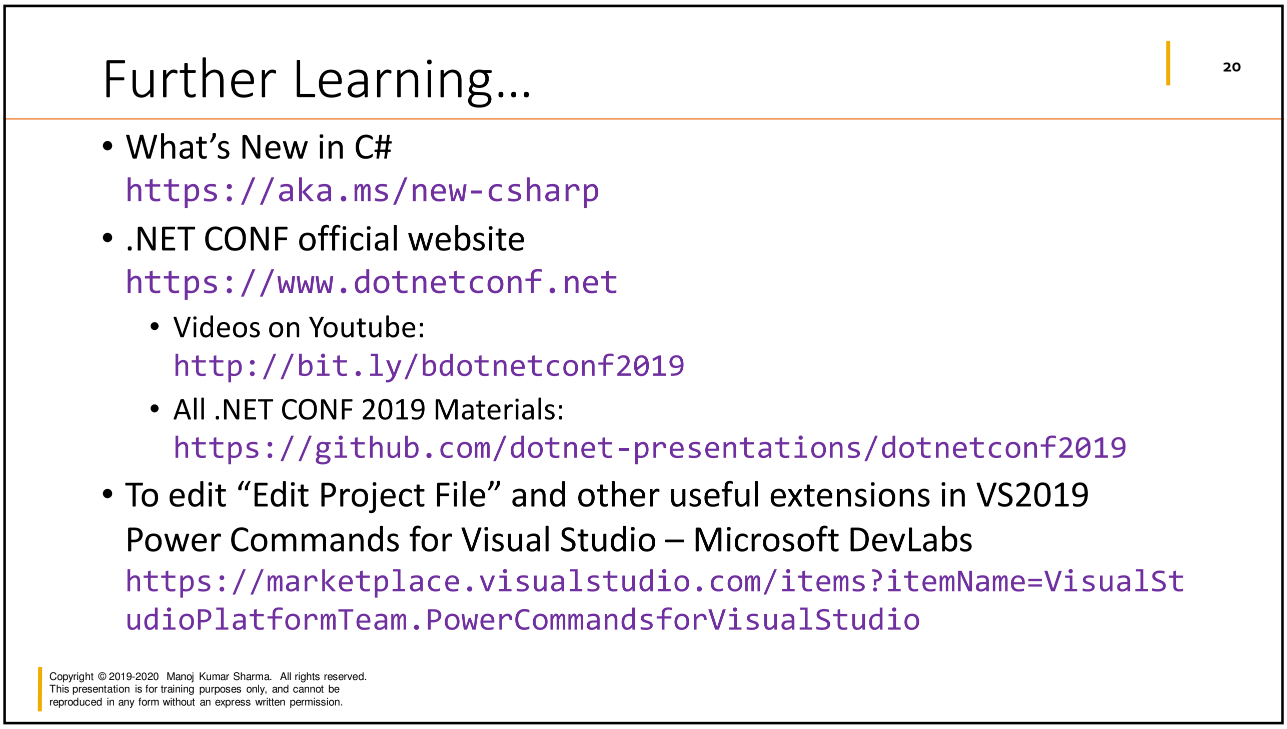
- A using declaration is a variable declaration preceded by the **using** keyword.
- It tells the compiler that the variable being declared should be disposed at the end of the enclosing scope.
- Similar to the **WITH** Block in VB

UsingDeclarations



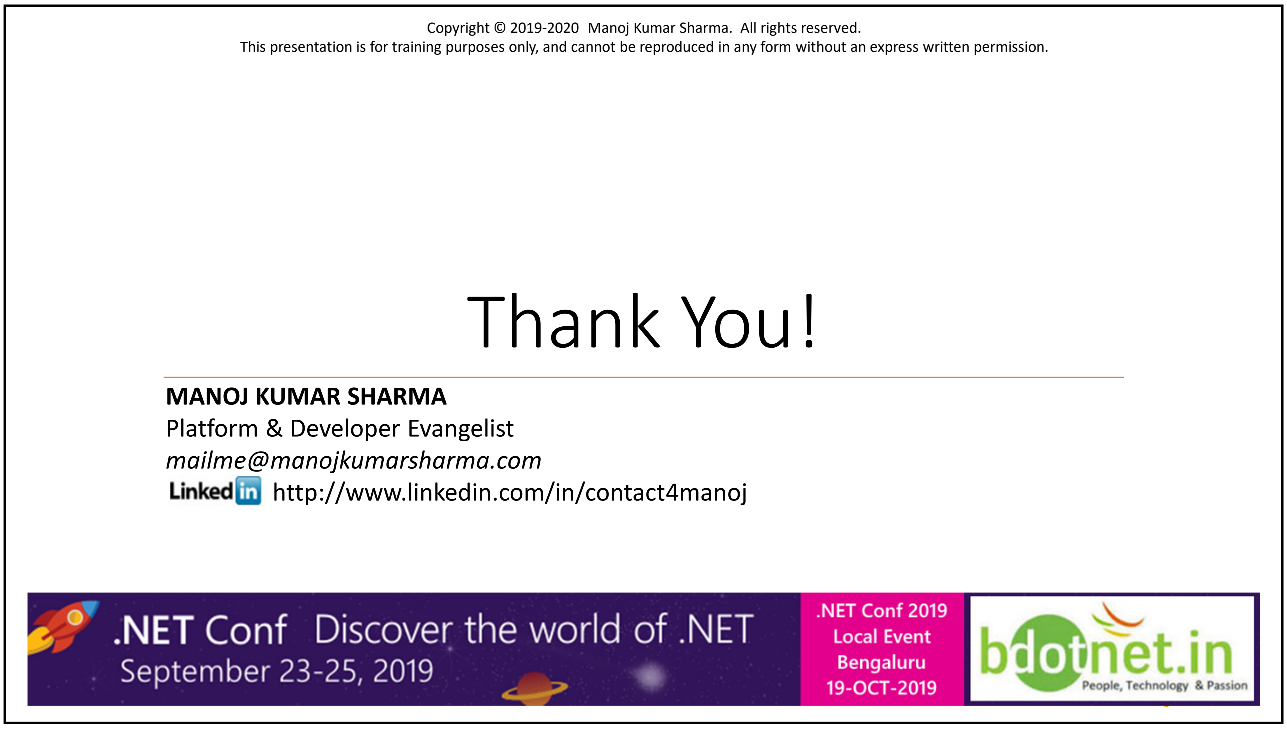
Summary

What’s new in C# 8.0



Further Learning…

- What’s New in C#
  `https://aka.ms/new-csharp`
- .NET CONF official website
  `https://www.dotnetconf.net`
  - Videos on Youtube:
    `http://bit.ly/bdotnetconf2019`
  - All .NET CONF 2019 Materials:
    `https://github.com/dotnet-presentations/dotnetconf2019`
- To edit “Edit Project File” and other useful extensions in VS2019
  Power Commands for Visual Studio – Microsoft DevLabs
  `https://marketplace.visualstudio.com/items?itemName=VisualStudioPlatformTeam.PowerCommandsforVisualStudio`



Thank You!

MANOJ KUMAR SHARMA
Platform & Developer Evangelist
mailme@manojkumarsharma.com
LinkedIn http://www.linkedin.com/in/contact4manoj

.NET Conf Discover the world of .NET
September 23-25, 2019

.NET Conf 2019
Local Event
Bengaluru
19-OCT-2019

bdotnet.in
People, Technology & Passion