

Other Language Improvements



Matt Honeycutt

HEROIC CONSULTING

@matthoneycutt www.trycatchfail.com



Overview



Default interface members

Using declarations

Async streams

Static local functions

Disposable ref structs

Read-only members

Unmanaged constructed types

Null-coalescing assignment

Interpolated-verbatim strings





Animal Widgets

v1.0



Animal Widgets

```
public interface IAnimalWidget
{
    string Name { get; }

    int Happiness { get; set; }
}
```



Animal Widgets

```
public interface IAnimalWidget
{
    string Name { get; }

    int Happiness { get; set; }

    void Feed();
}
```



Demo



Default interface implementation



Demo



Using declarations



Demo



Async streams



Demo



Static local functions




There is still more!



Disposable *ref* Structs

```
public unsafe ref struct UnmanagedHandle
{
    private int* handle;
    public UnmanagedHandle(int* handle) =>
        this.handle = handle;

    public int* GetHandle()
    {
        return handle;
    }
}
```



Disposable *ref* Structs

```
public unsafe ref struct UnmanagedHandle
{
    private int* handle;

    public UnmanagedHandle(int* handle) =>
        this.handle = handle;
}
```

ref structs cannot implement
interfaces!



Disposable *ref* Structs

```
public unsafe ref struct UnmanagedHandle
{
    private int* handle;

    public UnmanagedHandle(int* handle) =>
        this.handle = handle;
}
```

We don't need to implement an interface in C# 8!



Disposable *ref* Structs

```
public unsafe ref struct UnmanagedHandle
{
    private int* handle;

    public UnmanagedHandle(int* handle) =>
        this.handle = handle;

    public int* GetHandle()
    {
        return handle;
    }

    public void Dispose()
    {
        ResourceManager.Free(this.handle);
    }
}
```



Disposable *ref* Structs

With a using statement:

```
using (var handle = GetHandle())  
{  
    DoWork(handle);  
}
```

With a using declaration:

```
using var handle = GetHandle();  
DoWork(handle);
```



Readonly *struct* Members

```
public struct Vertex
{
    public double X { get; private set; }
    public double Y { get; private set; }
    public double Z { get; private set; }

    public void MoveTo(double x, double y, double z) =>
        (X, Y, Z) = (x, y, z);

    public (double X, double Y, double Z) AsTuple() =>
        (X, Y, Z);

    public override string ToString() =>
        $"Vertex{AsTuple()}";
}
```



Readonly *struct* Members

```
public struct Vertex
{
    public double X { get; private set; }
    public double Y { get; private set; }
    public double Z { get; private set; }

    public void MoveTo(double x, double y, double z) =>
        (X, Y, Z) = (x, y, z);
    ↓
    public readonly (double X, double Y, double Z) AsTuple() =>
        (X, Y, Z);
    ↓
    public readonly override string ToString() =>
        $"Vertex{AsTuple()}";
}
```




Readonly *struct* Members

```
public struct Vertex
{
    public double X { get; private set; }
    public double Y { get; private set; }
    public double Z { get; private set; }

    public void MoveTo(double x, double y, double z) =>
        (X, Y, Z) = (x, y, z);

    public readonly (double X, double Y, double Z) AsTuple() =>
        (X, Y, Z);

    public readonly override string ToString() =>
        $"Vertex{AsTuple()}";
}
```



Unmanaged Constructed Types

Unmanaged Types

byte

int

char

float

bool

... and more

NOT Unmanaged

Constructed Types

*(and everything
else)*



Unmanaged Constructed Types

C# 7

NOT Unmanaged

```
public struct Pair<T>
{
    public T Left;
    public T Right;

    public Pair(T left, T right) =>
        (Left, Right) = (left, right);
}
```

Pair<int>

Pair<object>



Unmanaged Constructed Types

C# 8

Unmanaged

```
public struct Pair<T>
```

```
{
```

```
    public T Left;
```

```
    public T Right;
```

```
    public Pair(T left, T right) =>  
        (Left, Right) = (left, right);
```

```
}
```

Pair<int>

NOT Unmanaged

Pair<object>



Null-Coalescing Assignment

```
var entity = GetEntityIfDefined(id: 1);  
entity = entity ?? BuildNewEntity();
```



Null-Coalescing Assignment

```
var entity = GetEntityIfDefined(id: 1);  
entity ??= BuildNewEntity();
```



Interpolated-Verbatim Strings



```
Console.WriteLine($"@"  
This is valid in C#{7}  
");
```


Interpolated-Verbatim Strings

This is backwards!




```
Console.WriteLine(@"$"  
This is NOT valid in C#{7}  
");
```



Interpolated-Verbatim Strings

This is fine!



```
Console.WriteLine(@"$"  
This is valid in C#{8}  
");
```

Summary



Default interface members

Using declarations

Async streams

Static local functions

Disposable ref structs

Readonly members

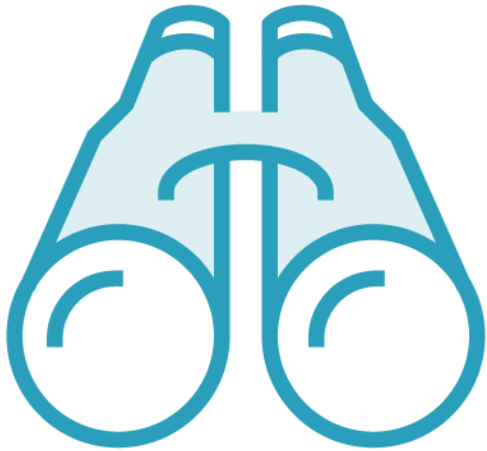
Unmanaged constructed types

Null-coalescing assignment

Interpolated-verbatim string definition



Up Next



More .NET core platform improvements

