This talk is about

C# 13 and Beyond

Tomorrow's talk is how to effectively use C# today (C# 12 and below)

Kathleen Dollard

.NET Languages, Microsoft

Changes in communicating new features

- Push to docs over blogs for how to use new features
- Docs pushes how to use new features to permanent location
- What’s new is a reference to new features
- Blogs advertise new features
- Feature status is a more reliable source of truth

- Similar changes happening for communicating new .NET features

C# 13 features

Escape character

Method group natural type improvements

Lock object

Implicit indexer access in object initializers

Params-collections

Ref/unsafe in iterators/async

allows ref struct constraint

Overload Resolution Priority (for library authors and advanced scenarios)

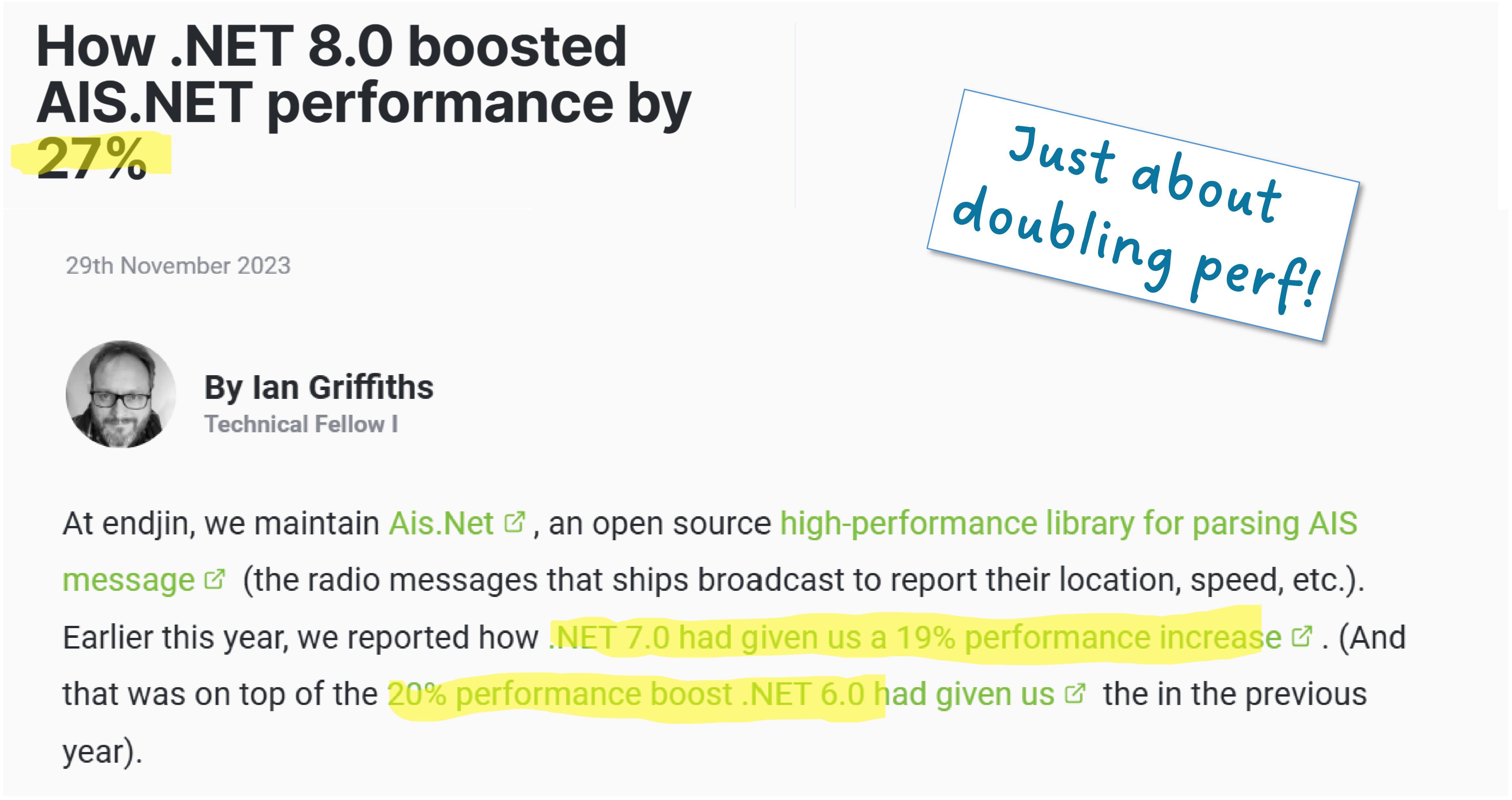
Partial properties

Ref Struct Interfaces

sêǧ ṣʧsuçʧ

- You are very happy the .NET Libraries (BCL) use them
- Your app probably does not need them
- They require an investment in understanding, and a commitment to resolving warnings
- Mistakes can crash your app/make it unstable
- *But it’s awesome if you are using it to replace* ụŋṣắǧê *code*
- Şřắŋ Ţ is often a safer/better choice

- We keep investing in it because this and similar work has contributed significantly to our ongoing performance improvements



How .NET 8.0 boosted AIS.NET performance by 27%

Just about doubling perf!

29th November 2023

By Ian Griffiths
Technical Fellow I

At endjin, we maintain Ais.Net, an open source high-performance library for parsing AIS message (the radio messages that ships broadcast to report their location, speed, etc.). Earlier this year, we reported how .NET 7.0 had given us a 19% performance increase. (And that was on top of the 20% performance boost .NET 6.0 had given us the in the previous year).

- https://endjin.com/blog/2023/11/how-dotnet-8-boosted-ais-dotnet-performance-by-27-percent-for-free

C# 13 action items
(current list, may be updated)

- Search for `ǧîêľđ` (case sensitive, whole word)
  - Consider renaming
- Search for `ľộçl`
  - Replace any lock on `ộčkêçʧ` with locks on the `Şỳșʧêņ Ţḥsêẳđîŋĝ Ľộçl`
- Search for `ŢộAssẳỳ` being used to support params
  - Consider updating your API
- Search for `ỵ̥̥̦č` (with varying numbers of 0)
  - Replace with `ê`
- Explore replacing unsafe blocks with safer alternatives

And beyond…

Field access in properties

```
řųčľîç șțșîŋĝ Ňǻņê   ĝêț  șêț     ğîêľđ   ŵǻľųê Ţsîņ
```

- You can access the backing field of an auto-property
- Can be multi-line
- Useful for DI
- Breaking change (next slide)

```
řųčľîç șțșîŋĝ Ŷỳć

    ĝêț
    șêț

        îğ  ğîêľđ    ŵǻľųê  sêțųsŋ
        ğîêľđ   ŵǻľųê
        ÔŋŶỳćCḥǻŋĝêđ
```

- We anticipate releasing field access as a preview feature available within C# 13/.NET 8
- Delay is to get the breaking change strategy correct

Proposal: `field` keyword in properties · Issue #140 · dotnet/csharplang (github.com)

Field access is a breaking change

- Breaking change because within the code of a property, field means the backing field, even if there is another `ǧîêľđ` in scope
  - Access the class field with `ʧḥîș`   or
- We will introduce a breaking change system along with this feature
- We anticipate you will get extra warning when you upgrade your SDK if you use `ǧîêľđ`
  - The breaking change mechanism is to let you silence those warnings

Dictionary expressions

Like type expressions, but for dictionaries

```
řụčľîç ŵộîđ Éyẳņřľê

    ŵẳș ğộľľș Dîçţîộŋẳșỳ șţșîŋĝ  îŋţ   ŋẳņêŢộAĝê

            Ķẳţĥľêêŋ    ¸,
            Dụșţîŋ    ¸¸
            Ňẳđș    ¸‘

    Côŋșộľê ẄsîţêĽîŋê șţșîŋĝ Ķộîŋ Éŋŵîșộŋŋêŋţ ŅêxĽîŋê
          ŋẳņêŢộAĝê
            Şêľêçţ y       y Ķêỳ  îș  y Λẳľụê
```

[Proposal]: Dictionary expressions · Issue #7822 · dotnet/csharplang (github.com)

Extensions types

Like extension methods, they enhance the API

- Implicit extension types
  - Do more with extensions, like extension methods, but more
  - Properties, static methods, and static properties
  - Perhaps, indexers and operators
  - Maybe, interfaces and inheritance for extension reuse
- Explicit extension types – *aka Roles*
  - Disambiguation (resolve naming collisions)
  - Roles – only when requested to include the extensions
    - Key point of the demo is that not all options are ComparableSymbols, but if they are declared to be, they get extra methods and properties

[Proposal]: Extensions · Issue #5497 · dotnet/csharplang (github.com)

Discriminated Unions

Yes, we are working on designs, but it will take a while

- Four types
  - Standard or class union (F# like)
  - Special or struct union
  - Ad hoc or anonymous (TypeScript like)
  - Custom unions
- Requirements
  - It has to *feel* like C# (not feel bolted on)
  - We want it to be performant with good memory usage
  - Anonymous unions are challenging

csharplang/proposals/field-keyword.md at main · dotnet/csharplang (github.com)

Standard or class union

- Declaration

```
ụŋîộŋ Û

    A îŋʧ ỵ  ṣʧsîŋĝ ỳ
    B îŋʧ ć
    C
```

- Construction

```
Û ụ   ŋêx A , ˳   ʧêŋ
```

- Deconstruction

```
îğ  ụ îṣ A ắ
îğ  ụ îṣ A ŵắs ỵ  ŵắs ỳ

îğ  ụ îṣ A   ỳ  ŵắs ỳ
```

Special or struct union

- Declared and used much like a class union
- Declaration would include șʧsụçʧ
- Memory footprint may be sum of members
- Type test use special features
- Boxing creates a boxed union, which must be unboxed
- sêǧ unions may be allowed

Ad hoc or anonymous unions

- Declaration

```
 A ộs B ộs C
ĝľộčăľ ụṣîŋĝ Û    A ộs B ộs C
```

- Construction

```
sêçộsđ A îŋʧ ỵ  ṣʧsîŋĝ ỳ
sêçộsđ B îŋʧ ċ
sêçộsđ C
řụčľîç ṣʧắʧîç C Ṣîŋĝľêʧộŋ
      ŋêx C

 A ộs B ộs C  ụ   ŋêx A ,。   ʧêŋ
```

- Deconstruction

```
îğ  ụ îṣ A ắ

   Ôs ŋắỳ ľộộl kụṣʧ ľîlê ṣʧắŋđắsđ
îğ  ụ îṣ A ŵắs ỵ  ŵắs ỳ
```

- Equivalence

```
 A ộs B  ỵ   ŋêx A ,。   ʧêŋ
 B ộs A  ỳ   ỵ
```

- And possibly, assignability

```
 A ộs B  ỵ   ŋêx A ,。   ʧêŋ
 A ộs B ộs C  ỳ   ỵ
```

Custom unions

- Declaration (note these are not nested, they could probably be records)

```
 Cľộșêđ
řụčľîç çľẳșș Û
řụčľîç çľẳșș A îŋʧ ỵ  șʧsîŋĝ ỳ     Û
řụčľîç çľẳșș B îŋʧ ć     Û
```

- What scope would you want: file or assembly?

Questions on Discriminated Unions

Email me, or even better discuss at https://github.com/dotnet/csharplang/

- **What are your scenarios?**
  - Feel free to send many or explain how they align with the four types
- Do allocations matter more than usability in unions?
- Does memory footprint matter more than usability in unions?
- Does the deconstruction in `ŢộľľCắľçụľắʧộs` make sense?
- If you add a new member to your union, do you expect using code to break?
  - Example: Uncommenting DeliveryTruck would break TollCalculator if exhaustiveness was enforced
- Are there other things – maybe enums – that you wish were exhaustive?

Thank you!

- kdollard@microsoft.com
- https://github.com/KathleenDollard/devintersection-2024-csharp13
- Docs:
  - https://learn.microsoft.com/dotnet/csharp/whats-new/csharp-13
  - Docs philosophy is to add new features in permanent location
- C# design:
  - https://github.com/dotnet/csharplang
- Implementation:
  - https://aka.ms/csharp-feature-status
- .NET blog
  - https://devblogs.microsoft.com/dotnet