This talk does not cover C# 13 or future ideas

The best features of C# that you might not be using

Kathleen Dollard

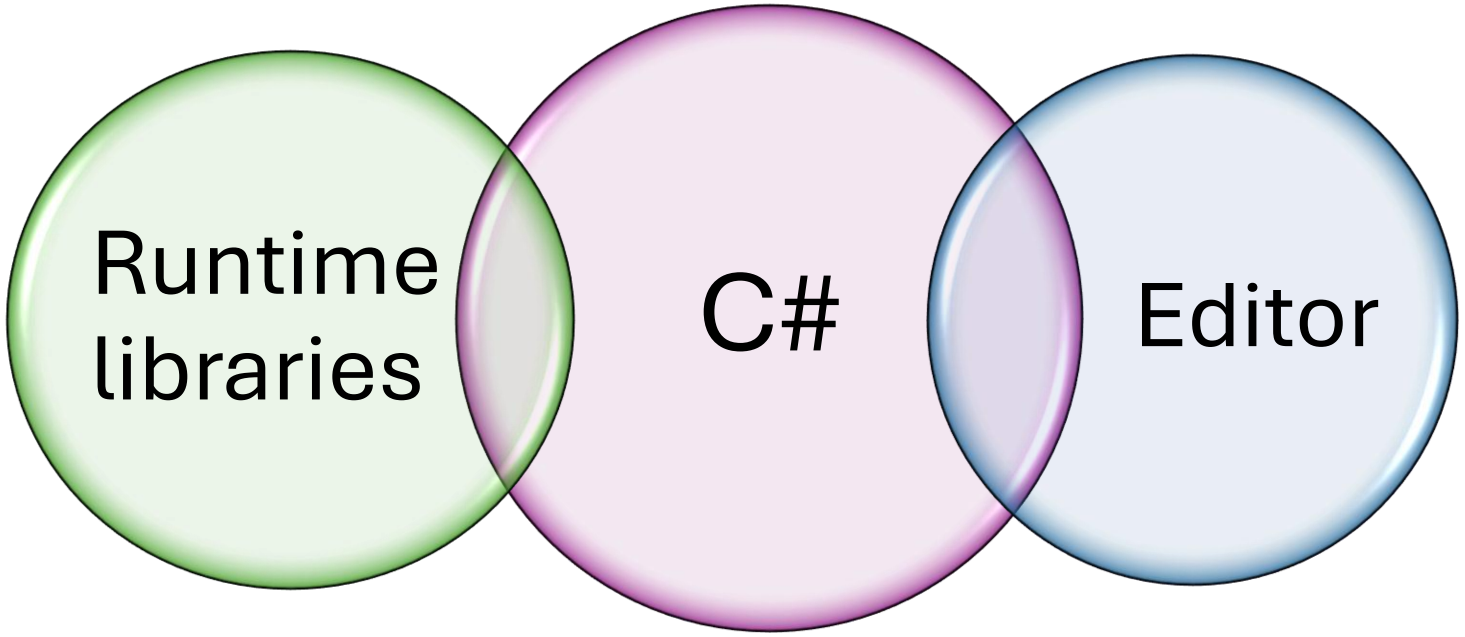
.NET Languages PM, Microsoft

Would you have come
if the title was

C# helps you avoid messes

Language can help you find the “pit of success”

Tools that lead you to the pit of success



Visual Studio
VS Code
Others

Good languages are a set of features that together minimize your mistakes

If you're in this room, you likely think strong typing helps you avoid mistakes

Code under review

Helpful and safe* methods - from the Runtime

C# adds new features

- To help partner teams enable scenarios for you
- To make C# faster
- For you
    - Reduce ways you can mess up
    - Reduce details you need to know
    - Make the fastest way the also the easiest
    - Make your code smaller and easier to read
    - Simplify overly complex scenarios

How language can help with the not so simple problem of Creating and initializing a collection

- We have a lot of collection types to meet different needs
- Different collections have different patterns for creation
- Resizing of collections has non-zero overhead
- The type of an argument was bound to the parameter type
- Combining individual values and ranges was awkward
- Creation was not unified with recent slice/range syntax
- Creating empty collections required special syntax to be efficient

Meet collection expressions

But wait, there’s more

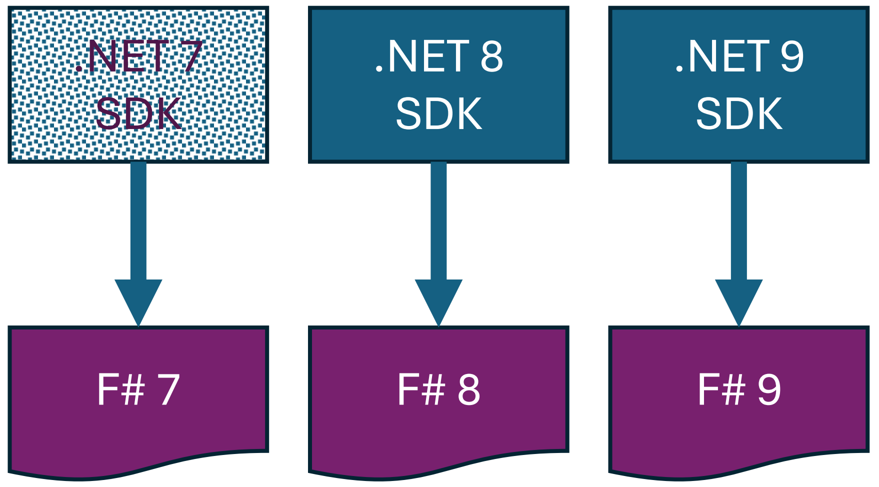
```
îŋţ    y̩¸           Éŋřţÿ̀ ắssắỳ

   Çộŋŋộŋ îŋţêsğắçêș ḩắŵê đêğắuľţ îŋřľêŋêŋţắţîộŋș
ÍÉŋuŋêsắčľê îŋţ  îŋţș     ,   ¸   ‘
   Ţḩê çộľľêçţîộŋ îș çsêắţêđ xîţḩ ţḩê çộssêçţ ľêŋĝţḩ

   Şľîçê sắŋĝê șỳŋţắy̩ șuřřộsţêđ
ÍÉŋuŋêsắčľê îŋţ  îŋţș¸       îŋţș Şêľêçţ y̩  y̩ ¸    ,`   ¸_
```

- Collection expressions are both easiest and fastest
- If you have several overloads, we (almost always) do the right thing (such as preferring `ŖêắđộŋľỳŞřắŋ`)
- If you have you're own collection type, it will work:
  - If it is well behaved
  - Or you use `ÇộľľêçţîộŋBuîľđês`
  - https://learn.microsoft.com/dotnet/csharp/language-reference/operators/collection-expressions#collection-builder

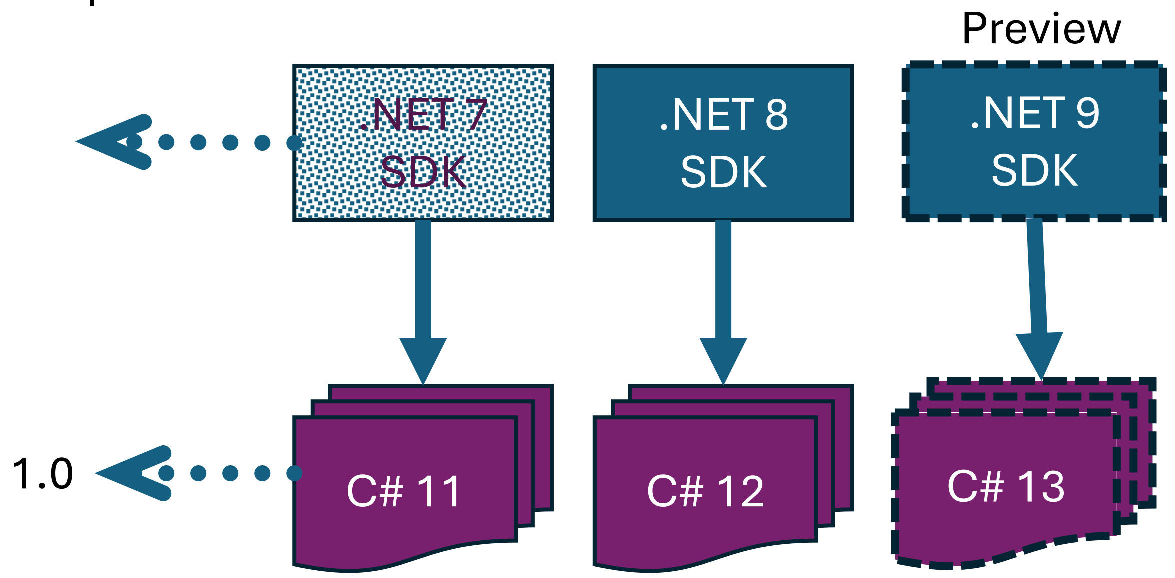
C# compiler and MSBuild

Two ways to version a compiler

Compilers are delivered as part of the SDK



Each compiler builds one language version



Each compiler builds new and all previous language versions

Editor features often just work with new compiler even if you're on older language versions although, the compiler follows rules of the specified language version

Console app

Target framework

Optional, defaults to one delivered with the TargetFramework's SDK

Optional, sets the base namespace, if your project name is illegal or ugly

Include global usings commonly used with your project type

Enable nullable analysis and warnings

Optional, stop build on warnings, as well as errors.
Important if you use span, ref struct or unsafe

Your çşřsộk

```
RsộřêsʧỳĞsộụř
   ÔụʧřụʧŢỳřê Éyੑê  ÔụʧřụʧŢỳřê
   ŢắsĝêʧGsắņêxộsl ŋêʧ¨ ̥  ŢắsĝêʧGsắņêxộsl
   ĽắŋĝΛêssîộŋ , ̧  ĽắŋĝΛêssîộŋ
   ŖộộʧŅắņêşřắçê CŞḥắsřŅộx  ŖộộʧŅắņêşřắçê
   ÍņřľîçîʧÛşîŋĝş êŋắčľê  ÍņřľîçîʧÛşîŋĝş
   Ņụľľắčľê êŋắčľê  Ņụľľắčľê
   ŢsêắʧŴắsŋîŋĝşAşÉssộşş ʧsụê  ŢsêắʧŴắsŋîŋĝşAşÉssộşş
  ChêçlGộsÔŵêsğľộxÛŋđêsğľộx Ţsụê  ChêçlGộsÔŵêsğľộxÛŋđ
RsộřêsʧỳĞsộụř
```

Int32.Max + 1, Int32.Min -1, etc
throw instead of wrapping

MS Build stuff to explore (homework, sorry, not sorry)

- Overview:
  - https://learn.microsoft.com/visualstudio/msbuild/build-process-overview
- MSBuild troubleshooting (Debugging MSBuild, David Federman):
  - https://dfederm.com/debugging-msbuild/
- `đîşêçţĝộsỳ čụîľđ řsộřş đîşêçţĝộsỳ čụîľđ ţǻsĝêţş` to share items
- NuGet packages effect build, including NerdBank, MinVer
- MSBuild Editor preview
  - https://devblogs.microsoft.com/visualstudio/experimental-msbuild-editor
- StructuredLogViewer
  - https://dfederm.com/debugging-msbuild
- Terminal logger – default in .NET 9 SDK (extra credit ☺)
  - In .NET 8 SDK, set the env var `ŇŞBÛÍĽDŢÉŖŇÍŅAĽĽÔĞĞÉŖ` to `ộŋ`

General C# Guidelines

- Don't do date math. Generally use BCL methods when possible
- Remove use of obsolete members
- Use nullable reference types and resolve all warnings (Enable `Ņųľľǻčľê`)
- Avoid unsafe code, explore `Şřǻŋ` and `sêğ  șʧsųçʧ`, with benchmarks
- Overloads should do the same thing semantically
- Limit numeric casts and conversions where precision is critical
- Set “Check for arithmetic overflow underflow” to true
- Consider enabling `ŢsêǻʧŴǻsŋîŋĝșAșÉssộsș`
  - Explicitly suppress when needed
- Remove `BîŋǻsỳGộsņǻʧʧês`

Nullable reference types

- Enable it on in new code, greatly improves reliability of code
- Static analysis, can be wrong and doesn’t handle everything
- Expresses you expectation
- Rarely use   to circumvent
- `ŵǻs` is nullable
- Use attributes to indicate intent in methods:

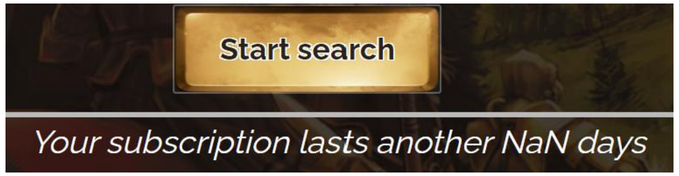
```
řụčľîç čộộľ ŢsỳĞêţʼΛǻľụê Řêşụľţ sêşụľţ
                               ŅộţŅụľľŴḥêŋ ţsụê  ộụţ ộčkêçţ  ŵǻľụê
```

Overloads

- Overload rules are incredibly complicated
- Overloads should do the same thing
  - They should do the same thing semantically
  - May have different perf or arguments
  - One has side effects, they all should etc.
- BCL takes overloads seriously
  - Trust the BCL to pick the right overload - don't pre-massage types

Limit numeric casts and conversions

- You/đêçîṇắľ -> độụčľê/ṣîŋĝľê is base 10 -> base 2
- độụčľê/ṣîŋĝľê -> you/đêçîṇắľ is base 2 -> base 2
- độụčľê -> ṣîŋĝľê loses precision
- ṣîŋĝľê -> độụčľê gives false precision
- độụčľê/ṣîŋĝľê support ṆắṆ, negative zero and other fun stuff
- đêçîṇắľ is slower, but only matters with lots of calculations
- Integers do not have precision challenges, but wrap on overflow



Your subscription lasts another NaN days

Tips

- Pattern matching can be easier to read than if/else logic
- Records are better than struct types for DTOs
    - Value equality is automatically with good performance
- Tuples -> record -> struct/class is a good prototyping story
- Collection expressions are da bomb (- *Bill Wagner*)
- Configure: editor.config, warnings, and MSBuild

Records

- Generates value equality, ToString and withers
- Use for classes when you want value equality
- Use for all structs to avoid slow equality comparisons
- Positional and named versions available
  - Positional is via primary constructors
  - Named is via normal properties
- Watch for differences in primary constructor behavior in records and non-records

Help the compiler to help you

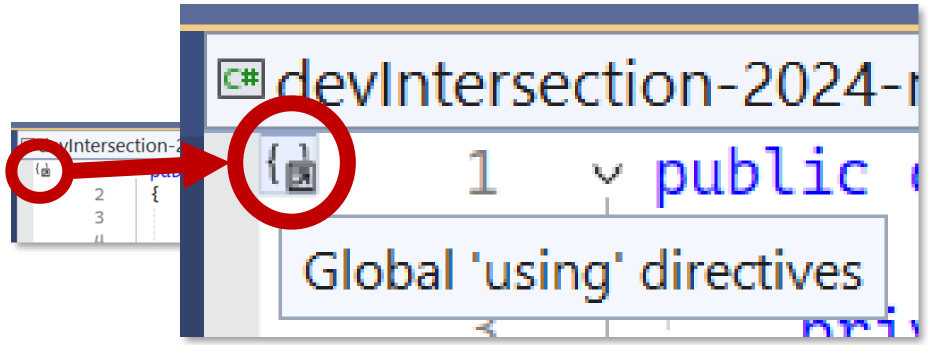
Customizing feedback/warnings

- https://aka.ms/csharp-warning-options
- https://aka.ms/csharp-warning-compiler-options
- https://aka.ms/csharp-coding-style

Pattern matching

- [Toll example]
- Use when it fits, either in `îğ` or `şxîʧçḥ`
- Let IntelliSense help you
- See `CọŋđîʧîọŋắľRắʧʧêsŋş` file in demos

Usings and namespaces


devIntersection-2024-

1 public

Global 'using' directives

- Global using
  - There’s a glyph for global usings and the source, right click goes to source
  - Source is generated if there is a ĝ çş at the end of the name
  - You can change what’s generated via the çşřsộķ file

```
ÍʧêṇĞsộụř
  Ûşîŋĝ Ŗêṇộŵê  Şỳşʧêṇ Ṇêʧ Ȟʧʧř
  Ûşîŋĝ Íŋçľụđê  CŞḥắsřṆộx         Éssộş îğ ʧḥîş îşŋ ʧ ắ ŵắľîđ ŋắṇêşřắçê
 ÍʧêṇĞsộụř
```

  - You can add them anywhere, but probably çşřsộķ or a ĝľộčắľ ụşîŋĝş çş file
- Aliases
  - Starting in C# 12, this works with tuples

```
ụşîŋĝ ŖêşşộŋŢụřľê    şʧsîŋĝ GîşşʧṆắṇê   şʧsîŋĝ ĽắşʧṆắṇê
```

- Using static lets you use static methods without stating the type
  - `ụşîŋĝ şʧắʧîç Şỳşʧêṇ Cộŋşộľê`
- File scoped namespaces
  - `ŋắṇêşřắçê CŞḥắsřṆộx`

Interpolated strings

- String interpolation - C# | Microsoft Learn
- Use for easy string manipulation
- Fast and optimized
- Formatting and localizing formatting in example project

Raw string literals

- C# 11.0 new features: raw string literals | endjin - Azure Data Analytics Consultancy UK
- Use when escaping curly bracket and double quotes, like JSON
- Use verbatim strings to limit escaping backslash

Ternary and compound operators

- Ţḥsộxĺǧŋụľľ
- ỵ   _ ̦

Index/Range (including ^ from the end)

- Index/Range (including   from the end)
- Let IntelliSense help you

Thank you!

- kdollard@microsoft.com
- Slides and code
  - https://github.com/KathleenDollard/devintersection-2024-csharp-now
  - Probably not until Sunday
- C# design:
  - https://github.com/dotnet/csharplang
- Implementation:
  - https://aka.ms/csharp-feature-status
- .NET blog
  - https://devblogs.microsoft.com/dotnet