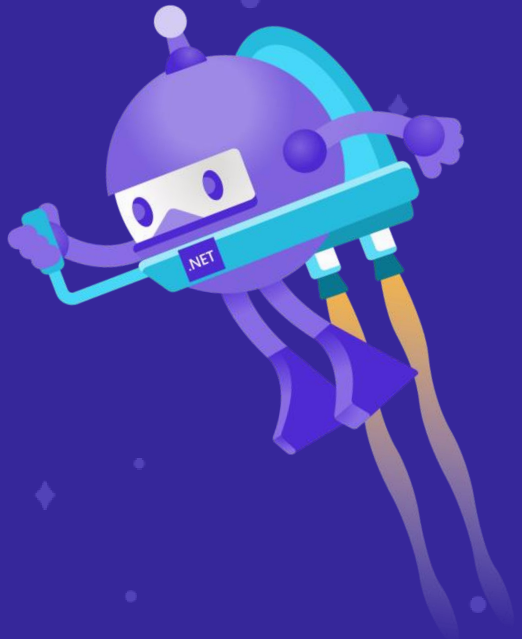
What’s new in F# 5 & 6

Don Syme |> @dsymetweets



Agenda

- One pandemic, two language versions
- Changes in how we talk about F#
- Walk through F# 5 and F# 6

An update on how we talk about F#

A marketing exercise we did

- Don Syme, Phillip Carter, Niklas Gustafsson also Reed Copsey, Isaac Abraham and others
- Target markets:
  - **Python programmers** hitting the limits w.r.t typing and performance
  - **C# programmers** wanting succinct, data-oriented programming without leaving the .NET ecosystem
  - Plus Scala, Julia, Go, Rust and everything else ☺

A marketing exercise

Methodology

- **Write down all the technical features/qualities of F#**
- **Mark them for relevance to the target market**
- **Choose the top three**

A marketing exercise, for example:

Quality	Features	Python devs hitting limits	C# devs wanting low ceremony
Succinctness	Type inference Expression-oriented Tuples Active patterns ….	Table stakes ✓	Appeals! ✓ ✓ ✓
Robustness	Types No-nulls Functional abstraction …	Appeals! ✓ ✓ ✓	Table stakes ✓
Performance	Typed code Generics without boxing ….	Appeals! ✓ ✓ ✓	Table stakes ✓
Expressiveness	…	…	…
Interoperable	…	…	…

Chosen Themes

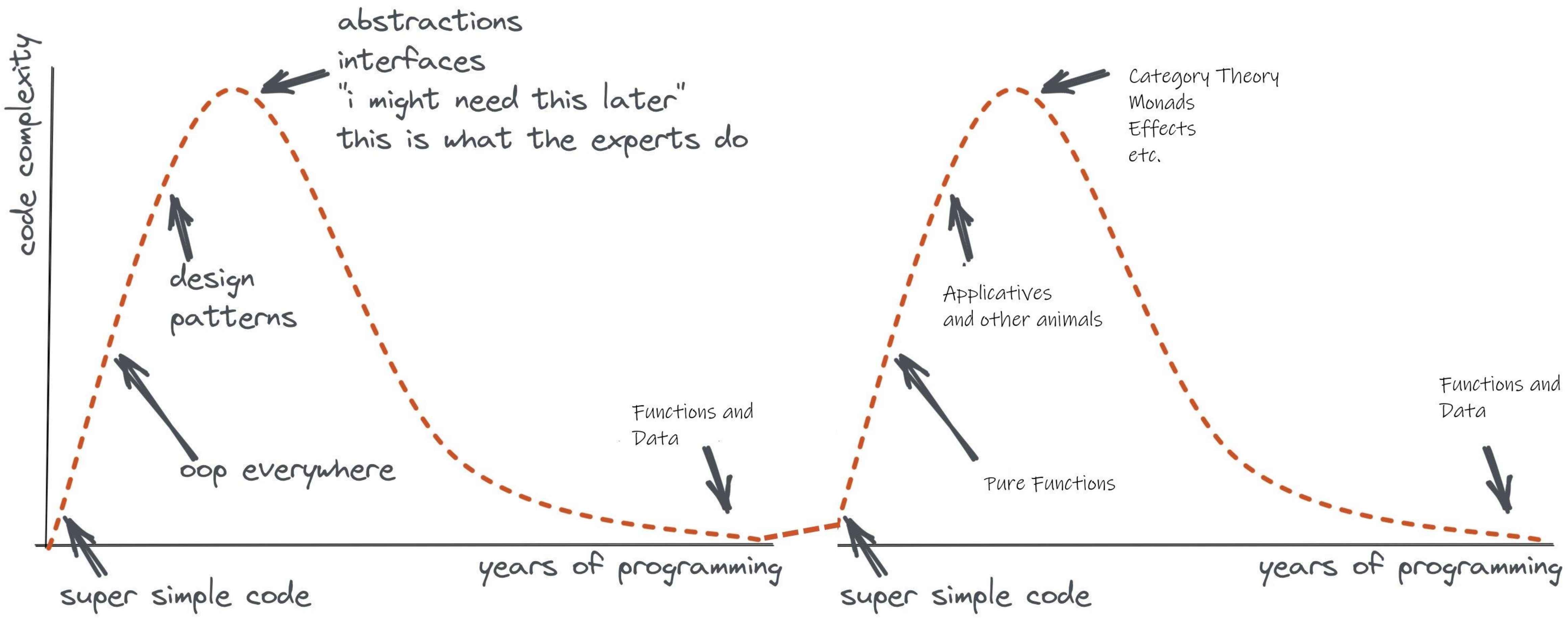
Succinct

Robust

Performant

Remember the eternal sweet spot of programming!

Functions + Data + Types



What is F# good at?

dot.net/fsharp

F# is good for **programming**

dot.net/fsharp

F# is good for **web programming**

dot.net/fsharp

F# is good for **cloud programming**

dot.net/fsharp

F# is good for **AI/ML/data-science programming**

dot.net/fsharp

F# is good for **succinct programming**

dot.net/fsharp

F# is good for **correct programming**

dot.net/fsharp

F# is good for **performant programming**

dot.net/fsharp

F# is good for **programming**

dot.net/fsharp

F# get started

```
dotnet new -lang F#

dotnet build
```

F# tools are part of the .NET SDK, available everywhere

.NET

www.dot.net

F# for the web backend

```
dotnet new -i "giraffe-template::*"

dotnet giraffe
```

High perf, functional server-side programming

GIRAFFE

A functional ASP.NET Core micro web framework for building rich web applications.

github.com/giraffe-fsharp/Giraffe

F# for the web frontend (JavaScript)

```
dotnet new -i "Fable.Template::*"

dotnet new fable
npm install
npm start
```

You can use F# as a JavaScript language

websharper.com

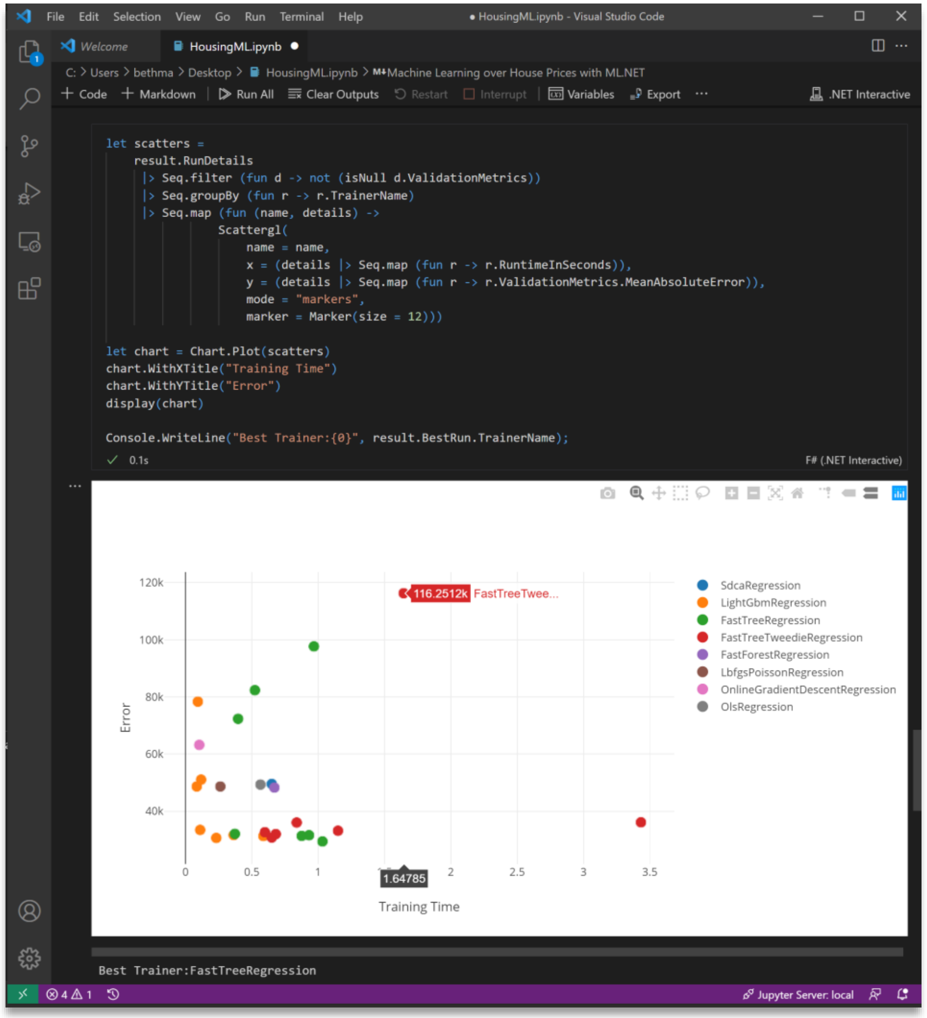
fable.io

F# for the full stack with SAFE

```
dotnet new -i SAFE.Template

dotnet new SAFE
dotnet tool restore
dotnet run
```

safe = GIRAFFE + FABLE + Microsoft Azure

F# for data science & ML

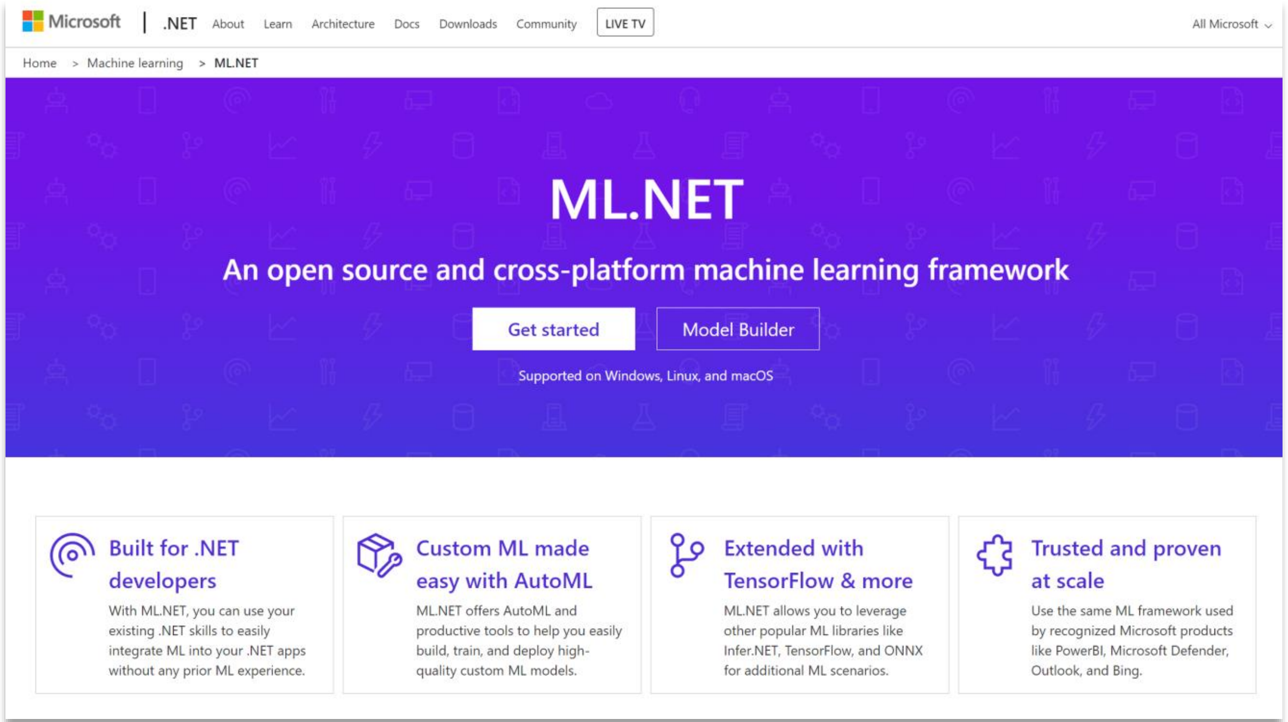


```
let scatters =
    result.RunDetails
      |> Seq.filter (fun d -> not (isNull d.ValidationMetrics))
      |> Seq.groupBy (fun r -> r.TrainerName)
      |> Seq.map (fun (name, details) ->
                Scattergl(
                    name = name,
                    x = (details |> Seq.map (fun r -> r.RuntimeInSeconds)),
                    y = (details |> Seq.map (fun r -> r.ValidationMetrics.MeanAbsoluteError)),
                    mode = "markers",
                    marker = Marker(size = 12)))

let chart = Chart.Plot(scatters)
chart.WithXTitle("Training Time")
chart.WithYTitle("Error")
display(chart)

Console.WriteLine("Best Trainer:{0}", result.BestRun.TrainerName);
```

```
Best Trainer:FastTreeRegression
```

File Edit Selection View Go Run Terminal Help
• HousingML.ipynb - Visual Studio Code
Welcome
HousingML.ipynb
C: > Users > bethma > Desktop > HousingML.ipynb > M↓Machine Learning over House Prices with ML.NET
+ Code + Markdown Run All Clear Outputs Restart Interrupt Variables Export .NET Interactive
✓ 0.1s
F# (.NET Interactive)
116.2512k FastTreeTwee...
120k 100k 80k 60k 40k
Error
0 0.5 1 1.64785 2 2.5 3 3.5
Training Time
SdcaRegression
LightGbmRegression
FastTreeRegression
FastTreeTweedieRegression
FastForestRegression
LbfgsPoissonRegression
OnlineGradientDescentRegression
OlsRegression
⊗4 ⚠1
Jupyter Server: local

```
#r "nuget:Microsoft.ML,1.4.0"
#r "nuget:Microsoft.ML.AutoML,0.16.0"
```



Microsoft | .NET About Learn Architecture Docs Downloads Community LIVE TV All Microsoft
Home > Machine learning > ML.NET
Get started
Model Builder

ML.NET

An open source and cross-platform machine learning framework

Supported on Windows, Linux, and macOS

Built for .NET developers

With ML.NET, you can use your existing .NET skills to easily integrate ML into your .NET apps without any prior ML experience.

Custom ML made easy with AutoML

ML.NET offers AutoML and productive tools to help you easily build, train, and deploy high-quality custom ML models.

Extended with TensorFlow & more

ML.NET allows you to leverage other popular ML libraries like Infer.NET, TensorFlow, and ONNX for additional ML scenarios.

Trusted and proven at scale

Use the same ML framework used by recognized Microsoft products like PowerBI, Microsoft Defender, Outlook, and Bing.

github.com/dotnet/interactive

dot.net/ml

F# 5!   Release Nov 2020

F# 5 - Laundry list

- FSharp.Core targets netstandard2.0
- **#r "nuget: ...“**
- .NET Interactive
- **String interpolation!**
- nameof
- **“open type”**
- Slicing updates
- **Applicatives**
- Stack traces for async
- Multi generic instantiations
- Default interface member consumption
- Interop with nullable value types
- **Improved Map/Set performance**
- Improved Compiler performance
- **Checked XML docs**

F# 6! Release Nov 2021

F# 6 - Laundry list

- **task { … }**
- Resumable code
- **expr[idx]**
- Structs for partial active pattern returns
- Overloaded custom operations in CEs
- “as” patterns
- Indentation syntax revisions
- InlineIfLambda
- **4x perf for list and array expressions**
- Additional implicit conversions
- Immutable collection updates
- More unit annotations
- **! and := soft deprecation**
- Remove legacy features
- **Pipeline debugging**
- Value shadowing
- **Tooling perf, scalability (64-bit, multi-thread)**
- **.NET Core default for VS scripting**
- **Scripting respects global.json**
- General improvements in .NET 6
- General improvements in Visual Studio 2022

String interpolation

- The very first F# RFC FS-1001, from 2014
- Design has been iterated, very happy with the result
- Why?
  - Improved readability
  - Reduced errors
  - More succinct
  - Less to learn
- Aim
  - Align with C#
  - But keep strong typing by unifying with printfn
  - Allow use with printfn + friends
- Demo

“open type”

- RFC FS-1068
- Increase expressivity of F# DSLs and ultra-succinct mode programming
  - Single-word “functions” can be overloaded members accepting named, optional, ParamArray arguments
- Demo
- **Caveat emptor**. Don’t inflict this on yourself or your friends without real thought about ramifications

#r "nuget: ..."

- People rate this as the #1 feature for Python programmers
- Combined with **global.json** in script directory allows you to "lock down" F# scripts as trustworthy

- #r "nuget: Newtonsoft.Json"
- #r "nuget: Newtonsoft.Json, 11.0.1"

- #i "nuget: https://my-remote-package-source/index.json"

Applicatives

- RFC FS-1063
- Directly from the last F# London meetup ☺
- Applicatives are often two-phased computations
  - Separate "compose" from "run"
  - Static "composition", then dynamic "run"
  - Example: graphs of fixed size (pre-allocate buffers, then run)
  - Example: validation (compose validators, then run)
- For F#, this means "**let! … and! …**" in computations
  - Maps to overloaded BindReturn/MergeSources
  - See RFC
- Demo

expr[idx]

- Why?
  - We lose people for no reason
  - “Making F# better by moving it towards it already was”
- Demo
- Caveats

task { .. }

- Why, what
- Demo
- Differences with async
  - A task is a single mutable, awaitable register
  - Hot start
  - No implicit cancellation checks
  - No implicit cancellation propagation
  - No tailcalls (to discuss…)
- Use primarily for interop when heavily consuming or producing tasks
- Debugging can be better
- Builds on “resumable code”, has other applications

!, :=, incr, decr soft deprecation

- RFC FS-1111
- These operators are highly confusing to beginners
- They are now rarely used
- Soft deprecation via informational messages
- Demo

Performance and Scalability

- F# Compiler Service new multi-threaded (no reactor queue!)
  - Makes IDEs MUCH more responsive
- Visual Studio 2022 now 64-bit
- Other major performance improvements, see announcements

Pipeline Debugging!

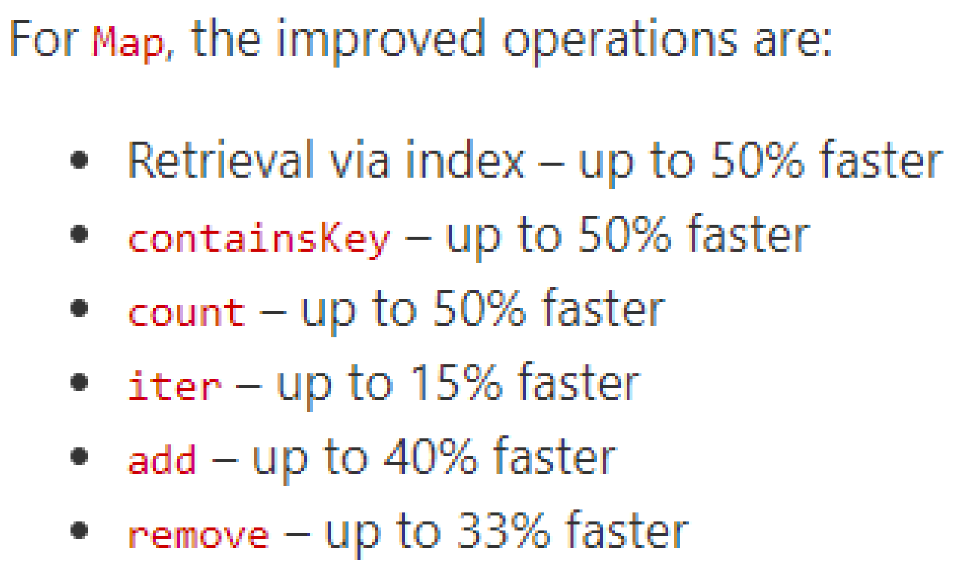
- Demo
- Also shadowing on values shown in debugging

Performance for [..] and [| … |]

- Demo/link

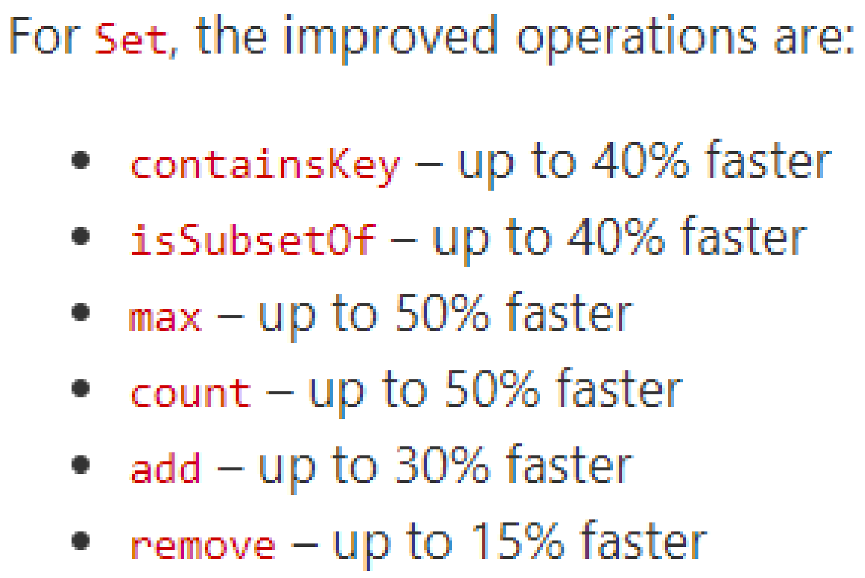
Improved Map/Set performance

- By Victor Baybekov
- In FSharp.Core 5.0.0+



For `Map`, the improved operations are:

- Retrieval via index – up to 50% faster
- `containsKey` – up to 50% faster
- `count` – up to 50% faster
- `iter` – up to 15% faster
- `add` – up to 40% faster
- `remove` – up to 33% faster



For `Set`, the improved operations are:

- `containsKey` – up to 40% faster
- `isSubsetOf` – up to 40% faster
- `max` – up to 50% faster
- `count` – up to 50% faster
- `add` – up to 30% faster
- `remove` – up to 15% faster

Doc improvements

- XML Doc checking!
- FSharp.Core examples! (500)
- docs.microsoft.com/dotnet/fsharp/ revamp!

- Call to action. Stackoverflow is becoming a problem for us. Please update your answers!

Huge thank you to contributors

- https://devblogs.microsoft.com/dotnet/whats-new-in-fsharp-6/#thanks-and-acknowledgments

Thanks!