Learning F#

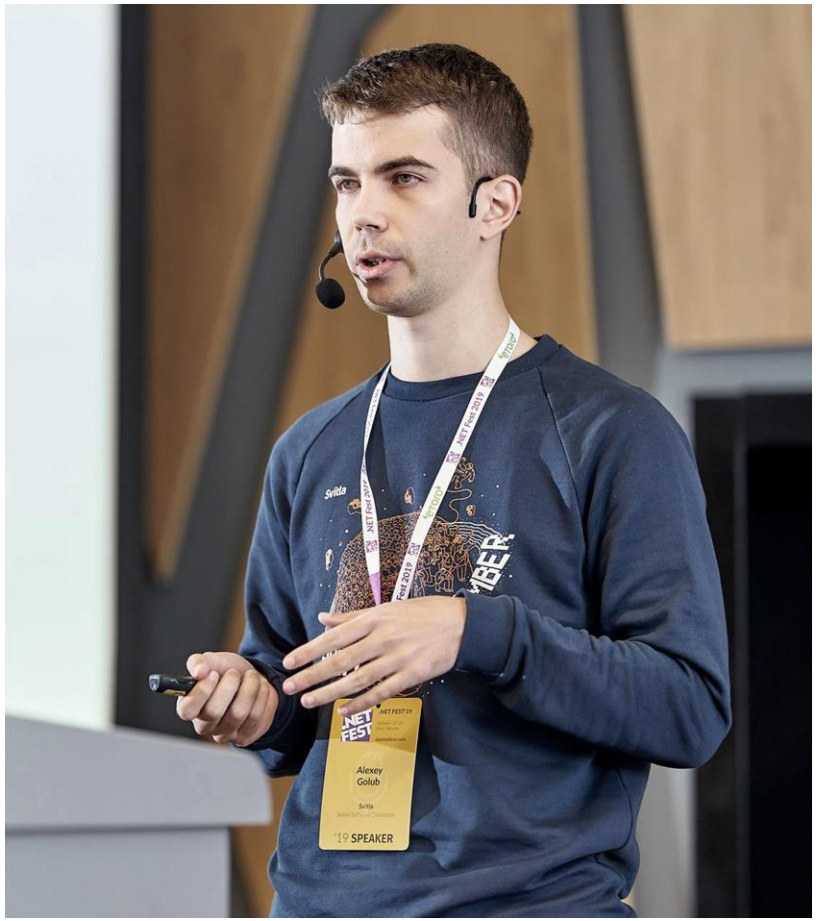
By Designing Your Own Language!

Speaker: Alexey Golub

@Tyrrrz

/whois ${speaker}

- Open-source developer ✨
- Conference speaker & blogger 🌐
- C#, F#, JavaScript 💻
- Cloud & web ☁
- Automation & DevOps ⚙



Speaker: Alexey Golub

@Tyrrrz

What is F#?

- **Functional-first** language for .NET
- **General purpose** & **multi-paradigm**
- **Strongly-typed** with powerful type inference
- **Open source** @ fsharp.org

What can F# be used for?

- **Anything**, really
- **Web** development (Fable, Elmish, Bolero, Giraffe, Suave, Saturn)
- **Desktop** development (Avalonia.FuncUI, Elmish.WPF, WinForms)
- **Mobile** development (Fabulous, Xamarin, Elmish.React)
- **Data access** (Type providers, FSharp.Data, Rezoom.SQL)
- **Data science** (Jupyter, FsLab, XPlot, Deedle, Math.NET, ML.NET)

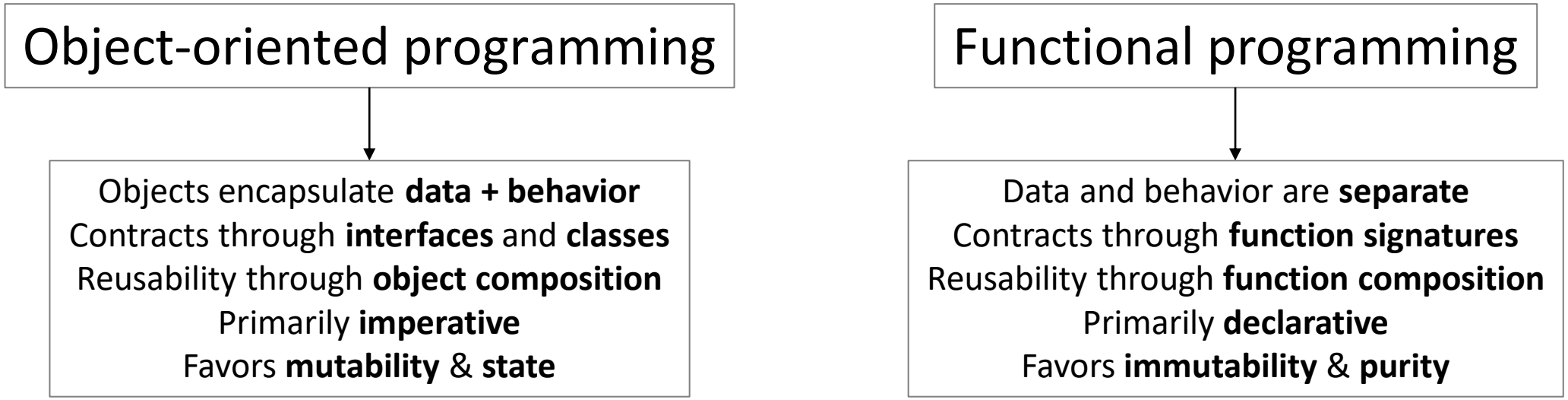
lambda calculus
lenses
sum type
generic operators
nominal typing
hindley-milner type system
monad
catamorphism
category theory
return type inference
impure segregation principle
computation expression
return type inference
side-effect
isomorphism
functor
declarative vs imperative
higher-order polymorphism

What?!

Speaker: Alexey Golub

@Tyrrrz

Paradigm shift



Functional ♥ Purity

Impure

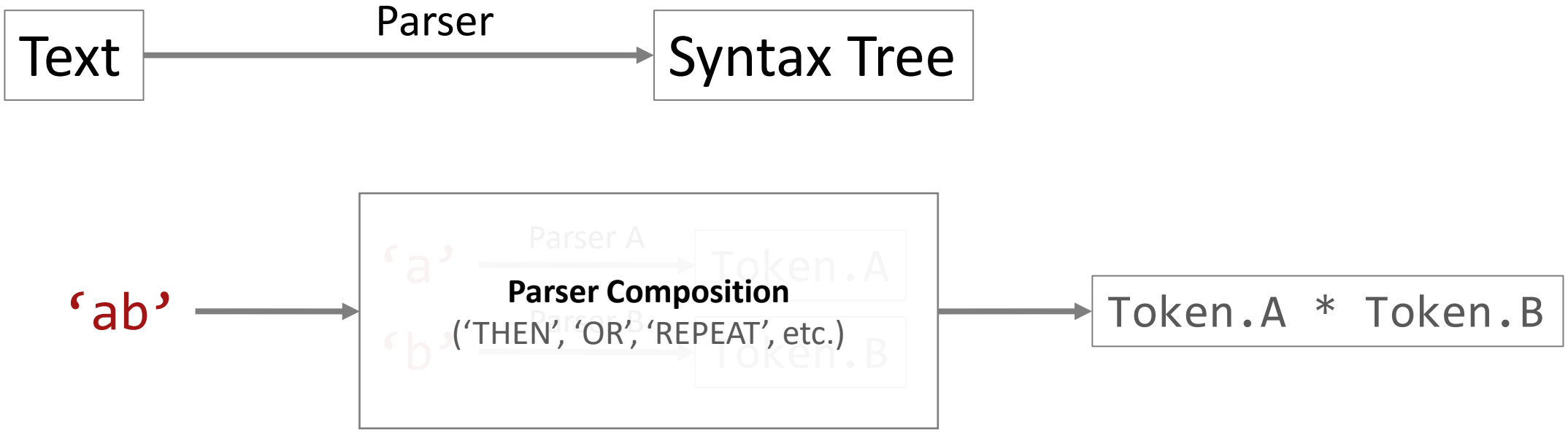
```
isWeekend() {
    var date = now();
    return date.weekday == Saturday ||
           date.weekday == Sunday;
}
```

Pure

```
isWeekend(date) {
    return date.weekday == Saturday ||
           date.weekday == Sunday;
}
```

- Domain is modelled as data and pure functions that transform it
- Pure functions are composed to create data flow pipelines
- Impure side-effects are pushed towards the edges of the system

Parsers are functional



FParsec

- Library for building **recursive-descent** parsers
- Lets us express **grammar rules** with **functions**
- Parser is a **function** of **input** that returns **produced result**
- Complex parsers are built by **combining** less complex parsers
- **Hierarchy** of parsers resembles the target **syntax tree**
- Incredibly **fast performance**

https://github.com/stephan-tolksdorf/fparsec

Let’s build our own **query language**!

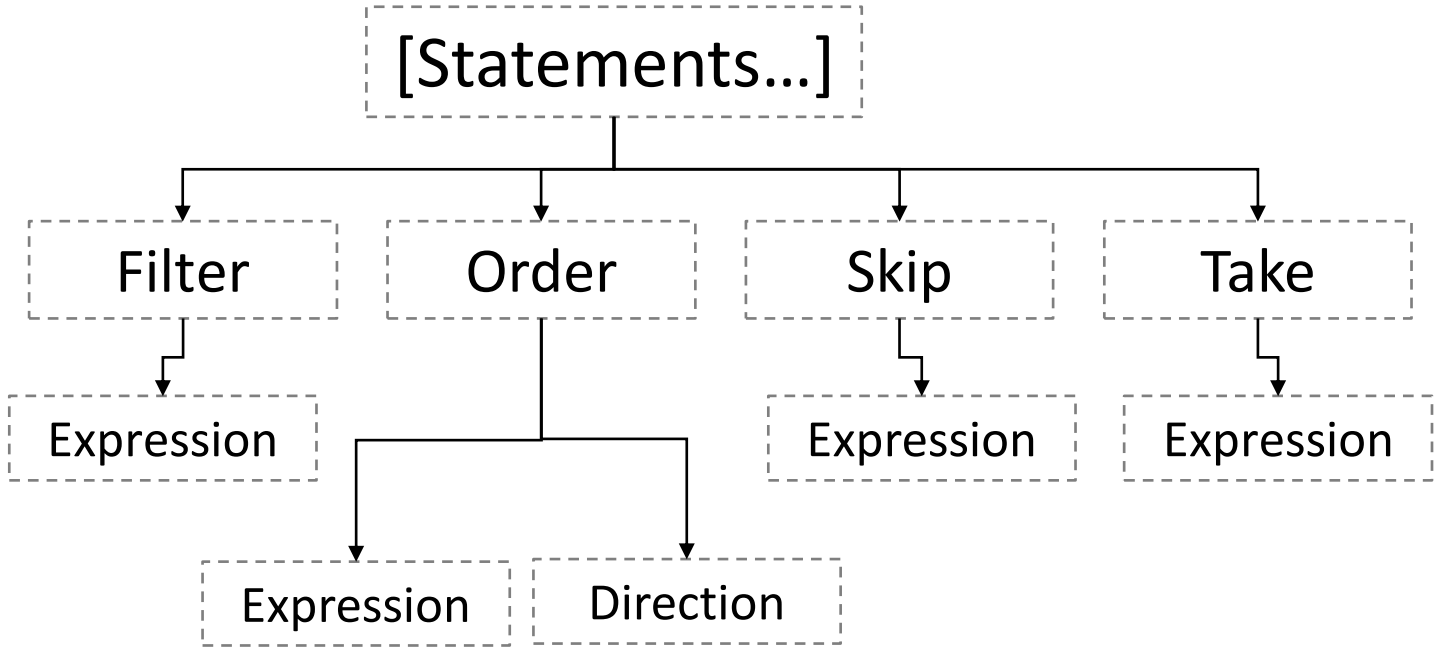
Speaker: Alexey Golub

@Tyrrrz

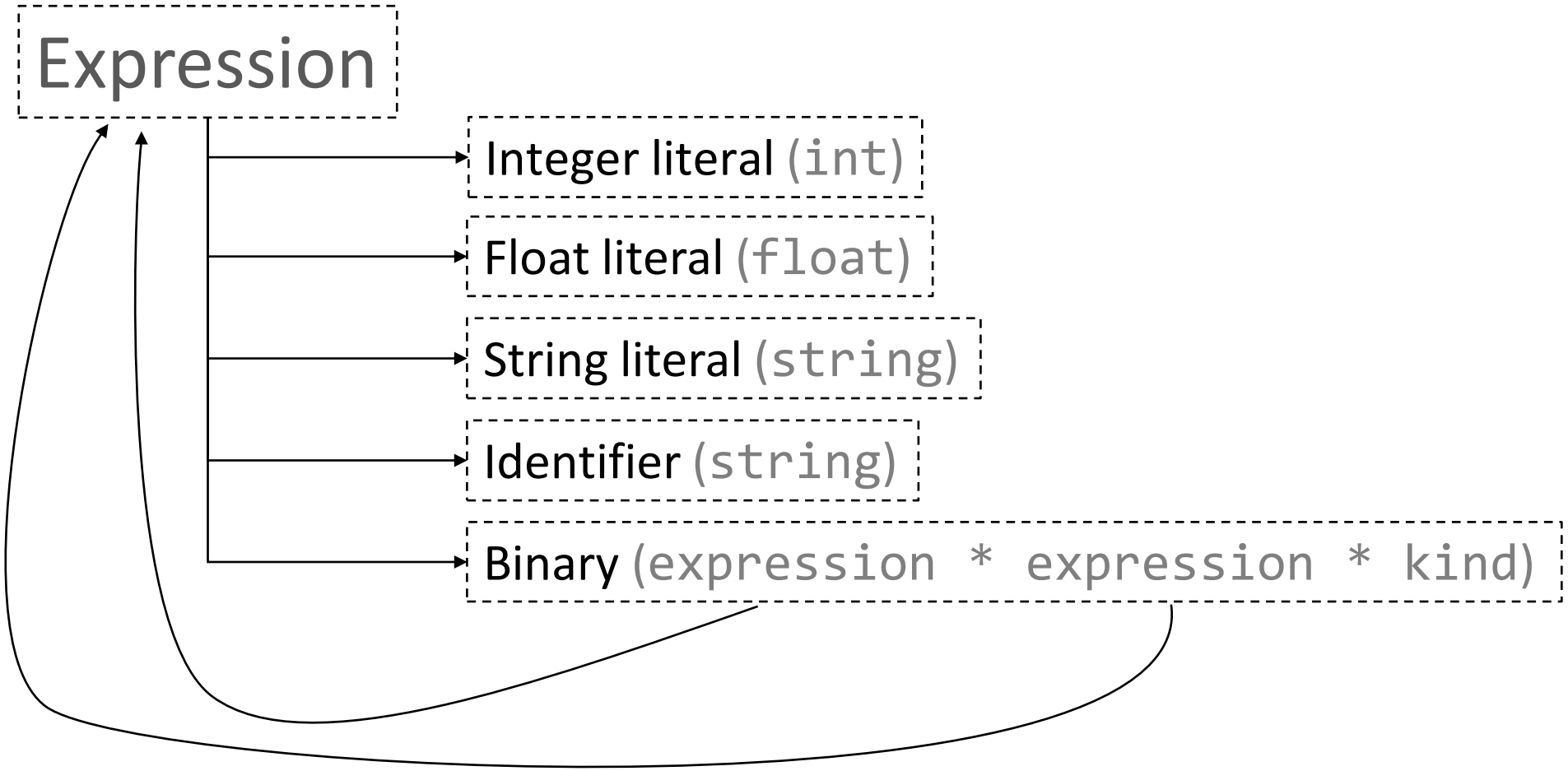
Language syntax

```
filterby Category = 'Fantasy'
orderby Rating desc
skip 5
take 10
```

- keywords
- identifiers
- literals
- expressions
- statements



Language syntax



Operator precedence parser

- Automatically parses **recursive operators**
- Supports **infix**, **prefix**, **postfix**, and **ternary** operators
- Handles **associativity** and **precedence** rules

```
type OperatorPrecedenceParser<'TTerm, 'TAfterString, 'TUserState> =
  member ExpressionParser: Parser<'TTerm,'TUserState>
  member TermParser: Parser<'TTerm,'TUserState> with get, set

  member AddOperator: Operator<'TTerm, 'TAfterString, 'TUserState> -> unit
  member RemoveOperator: Operator<'TTerm, 'TAfterString, 'TUserState> -> bool
```

Things we've learned

- F# syntax & project structure
- Data types, records, discriminated unions, tuples
- Pattern matching
- Function composition
- Writing language parsers

Source code

https://github.com/Tyrrrz/JetBrainsDotnetDay2020

Contains the demo project and the presentation

Learn more

- F# for Fun and Profit by Scott Wlaschin
  https://fsharpforfunandprofit.com
- FParsec tutorial by Stephan Tolksdorf
  https://quanttec.com/fparsec/tutorial.html
- Parsing in F# with FParsec by Alexey Golub
  https://tyrrrz.me/blog/parsing-with-fparsec

Thank you!

Speaker: Alexey Golub

@Tyrrrz