Co je F# a jak se mu daří v roce 2019?

Roman Provazník

@rprovaznik | @fsharping

WUG Days 2019 | 14. září 2019

Kdo jsem?

CN GROUP

F# Lead Developer @ CN Group CZ

Zakladatel české **F# komunity** (FSharping.com)

Speaker, F# evangelist

F#

Multiparadigmatický

Silně typový

.NET jazyk

F#

1. verze v **květnu 2005**

Autor: **Don Syme** (Microsoft Research, Cambridge)

Aktuálně ve **verzi 4.7.0**

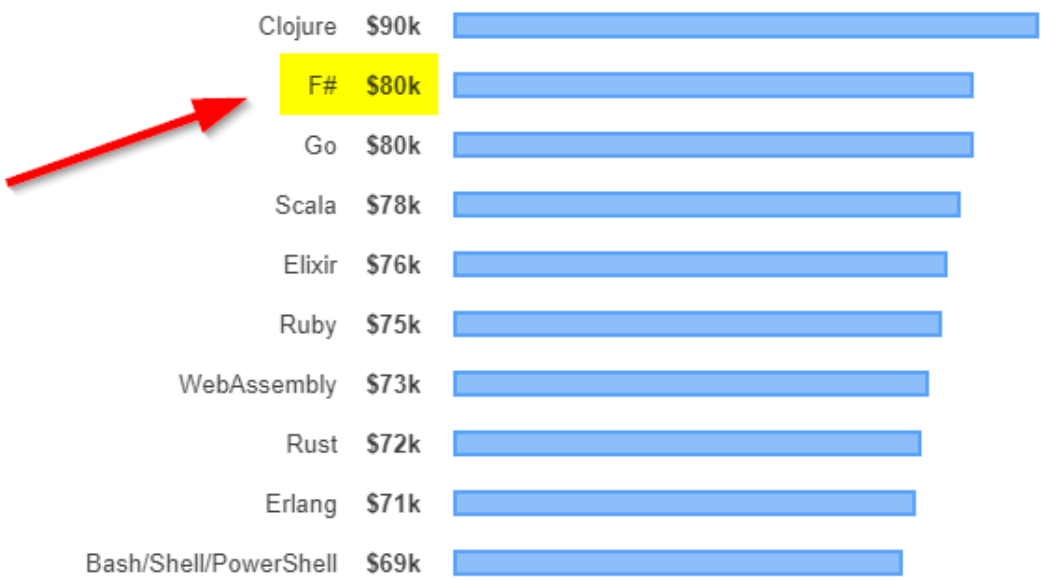
Proč by mě to vůbec mělo zajímat?

Stack Overflow Survey 2019

Top Paying Technologies

What Languages Are Associated with the Highest Salaries Worldwide?

Global | United States



https://insights.stackoverflow.com/survey/2019#top-paying-technologies

F# je multiparadigmatický

OOP!

FP!

F# je multiparadigmatický

F# je multiparadigmatický

F# is bere to **nejlepší z obou** světů

Lze použít **interfaces, abstraktní třídy, dědičnost**...

...stejně jako **high-order functions, anonymní funkce, currying, algebraické datové typy**, atd...

F# je .NET jazyk

Znáte **platformu**

Znáte **NuGet**

Lze použít jakékoliv **oblíbené IDE**

F# ⇔ C# Interop

Lze volat **C# kód** z F# projektu

Lze volat **F# kód** ze C# projektu

Všechny **oblíbené** NuGet **knihovny** a frameworky (ASP.NET, EF Core) **k dispozici**

Ok, kde je háček?

F# vypadá jinak

ML syntaxe (OCaml)

(Skoro) žádné **středníky**

(Skoro) žádné **závorky**

F# vypadá jinak

```
let add x y = x + y
```

Název funkce

Parametry

Tělo funkce

F# vypadá jinak

```
let add x y =
    x + y
```

Odsazení (scope)

Ano, záleží na white-space!

Ale má to i své výhody!

F# syntaxe

Méně vizuálního **„bordelu“**

Píšete **pouze kód = píšete méně**

Vhodné pro **DSL**

```
let add x y = x + y
```

vs

```
namespace Math
{
    public static class Helpers
    {
        public static int Add(int x, int y)
        {
            return x + y;
        }
    }
}
```

F# syntaxe

Čteme kód jako **psaný text**

Čteme ho **méně**

Výrazně **rychlejší refaktoring**

```
[1..100]
|> List.filter (fun x -> x%2 = 0)
|> List.map (fun x -> x * 2)
|> List.rev
```

F# není jen jiná syntaxe*

*je to jiný mindset

Imutabilní přístup

Všechny F# typy jsou **imutabilní** (neměnné)

Lépe **paralelizovatelné**

```
let roman = { Name = "Roman"; Role = "F# freak" }

// nelze
roman.Name <- "Vojta"

let vojta = { roman with Name = "Vojta" }
```

Nativní podpora pro jednoduché „kopírování“ hodnot

Jednodušší **testování**

Operátory – Pipe forward „|>“

Předává parametr **z levé strany** jako **poslední parametr na pravé straně**

```
(List.rev
     (List.map (fun x -> x * 2)
          (List.filter (fun x -> x%2 = 0) [1..100])))
```

vs

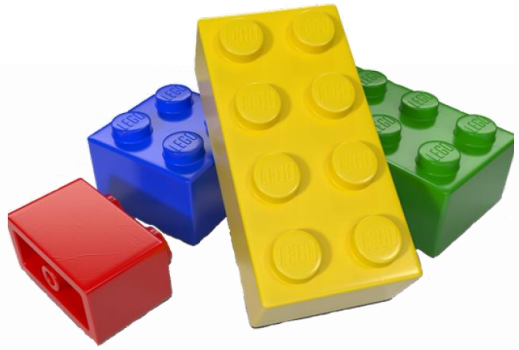
```
[1..100]
|> List.filter (fun x -> x%2 = 0)
|> List.map (fun x -> x * 2)
|> List.rev
```

Operátory – Composition „>>“

Spojuje (komponuje) **menší funkce do větších**

```
let add10 x = x + 10
let add90 x = x + 90
let add100 = add10 >> add90
```

Kód jako LEGO



Funkce jako „first-class citizen“

Funkce lze **spojovat** (compose)

Funkce lze **přijímat jako parametr i vracet** (high-order functions)

S funkcemi lze **pracovat jako s daty**

```
[
    hasValidEmail
    userDoesNotExist
    usernameLongerThan 5
]
|> validate personDTO
```

```
[
    add
    multiply
    subtract
]
|> List.map (fun f -> f 20 10)
```

Funkce jako „first-class citizen“

OO pattern/principle	FP pattern/principle
• Single Responsibility Principle	• Functions
• Open/Closed principle	• Functions
• Dependency Inversion Principle	• Functions, also
• Interface Segregation Principle	• Functions
• Factory pattern	• Yes, functions
• Strategy pattern	• Oh my, functions again!
• Decorator pattern	• Functions
• Visitor pattern	• Functions

Algebraické datové typy

SUM types

Discriminated unions

```
type Payment =
    | Cash of amount:int
    | BankCard of int * card:int64
    | Barter of description:string
```

```
type Tree<'T> =
    | Branch of 'T * Tree<'T> list
    | Leaf of 'T
```

Product types

Records, Tuples

```
type Person = {
    Name : string
    Role : string
}

let anonRec = {| X = 123.45 |}

let nameAndAge = "Roman", 37
```

Algebraické datové typy

„Essential“ datový typ pro **DDD a domain modeling**

Základ většiny **DSL** v jazyce F#

Čitelný i pro zákazníky

```
type Points = Zero | Fifteen | Thirty | Fourty
type Player = A | B
type Game =
    | Score of Points * Points
    | Advantage of Player
    | Victory of Player
```

```
type Result<'a> =
  | Success of 'a
  | ErrorMessage of string
```

```
type UserId = UserId of Guid
```

100% typová bezpečnost už při kompilaci – **nevalidní stav nelze zkompilovat!**

Pattern matching

```fsharp
let showScore (g:Game) =
    match g with
    | Victory p -> sprintf "Vyhral hrac %A" p
    | Advantage p -> sprintf "Hrac %A ma vyhodu" p
    | Score (a, b) when a = Fourty && b = Fourty ->
        "Je to SHODA!"
    | Score (a, b) ->
        sprintf "Je to %A:%A" a b
```

```fsharp
let trollMe value =
    match value with
    | "Roman", 37 -> sprintf "Is that you?"
    | "Roman", _ -> sprintf "Oh, different Roman"
    | _, 100 -> sprintf "You are old as F**K"
    | name, age ->
        sprintf "You are %s, %i years old" name age
```

Umožňuje **větvení kódu podle vzorů** (patterns)

Hlídá **neošetřené větve** (exhaustive pattern matching)

Lze **použít na většinu F# typů** (recordy, tuples, listy, ...)

Pattern matching

```
type Permissions = Read | Write
type Access = Admin | Limited of Permissions
type User = {
  Login : string
  Rights: Access }

let displayLogin user =
    match user with
    | { Login = "Jakub" } -> "Hi Jakub!"
    | { Rights = Admin } -> "Please be generous god!"
    | { Login = l; Rights = Limited(perm) } ->
        match perm with
        | Read -> sprintf "You are %s and you can read only" l
        | Write -> sprintf "You are %s and you can write" l
```

Lze aplikovat i na **vnořené datové struktury**

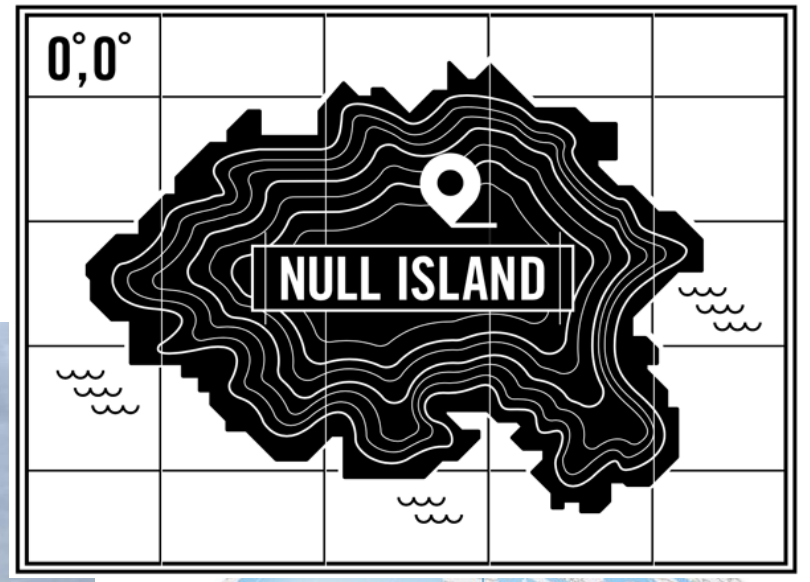
Pattern matching + algebraické datové typy jsou pro F# naprosto zásadní

Stejně jako NULL*

*jeho absence

NULL v Minecraft

NULL v zeměpise



NULL v F#

F# typy jsou **non-nullable**

Pro ne vždy přítomnou hodnotu se používá **union typ Option** (klíčové slovo „**option**“)

Pattern matching místo defenzivy

```
type Person = {
    Name : string
    Age : int option
}
```

```
type Option<'a> =
    | Some of 'a
    | None
```

```
match moznaHodnota with
| Some x -> sprintf "Hura, mame hodnotu %A" x
| None -> "Nemam nic"
```

NULL v F#

NULL je jeden z **největších omylů v dějinách programování**!

Option typ tento problém řeší - **méně defenzivy, méně kódu, větší bezpečnost**

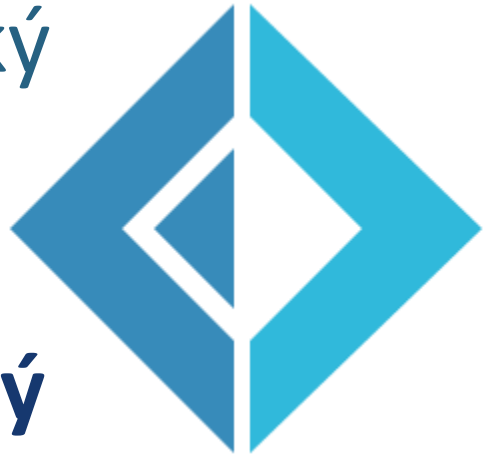
Možnost Option hodnoty **funkcionálně kombinovat** – bind, map, fold, iter...

Takže co je tedy F#?

Multiparadigmatický

Přehledný

Komponovatelný



Silně typový

Syntakticky **úsporný**

Bezpečný

.NET jazyk

A je toho víc!

Partial function application

Computation expressions

Quotations

Currying

Active patterns

Custom operators

Units of measure

Statically resolved types

Jak se F# daří v roce 2019?

F# v roce 2019

Silná **F# Software Foundation** + stoupající podpora ze strany **Microsoftu**

Podpora **.NET Core & .NETStandard** + „starý .NET framework“

Lze vyvíjet ve všech dostupných **.NET IDE** – VS, VSCode, Rider

Dostupné knihovny pro **webový vývoj**, **desktop**, **javascript**ové aplikace, **web assembly**, **mobilní aplikace, serverless**

Information-rich programming v F#

Dostupné od **verze 3.0**

Type Providers – rozšiřuje F# compiler o komponenty schopné poskytovat **on-demand typové údaje**

Podpora **JSON, XML, HTML, SQL, YAML**

Dostupné na **F# project incubation space**

https://github.com/fsprojects

DEMO

Type providers

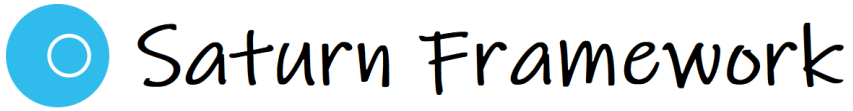
Webový vývoj v F# (Backend)

Většina knihoven staví na **ASP.NET Core**

Funkcionální přístup (Suave, Giraffe)

MVC-like přístup (Saturn)

„Tradiční“ **REST služby** i **backend-generated weby**



Webový vývoj v F# (Backend)

```
let webApp =
    choose [
        route "/ping"   >=> text "pong"
        route "/"       >=> htmlFile "/pages/index.html" ]

let configureApp (app : IApplicationBuilder) =
    app.UseGiraffe webApp

let configureServices (services : IServiceCollection) =
    services.AddGiraffe() |> ignore

[<EntryPoint>]
let main _ =
    WebHostBuilder()
        .UseKestrel()
        .Configure(Action<IApplicationBuilder> configureApp)
        .ConfigureServices(configureServices)
        .Build()
        .Run()
    0
```

Použití **custom operátorů** (kleisli „fish“ operator)

Funkcionální přístup

Inspirováno **Suave.io** syntaxí

Webový vývoj v F# (Backend)

```
let webApp = router {
    get "/ping" (text "pong")
    get "/" (htmlFile "/pages/index.html")
}

let app = application {
    url ("http://0.0.0.0:8085")
    use_router webApp
    memory_cache
    use_static publicPath
    use_gzip
}
run app
```

Saturn Framework

Použití **computation expressions**

MVC-like přístup

Inspirováno knihovnou **Phoenix** z Elixiru

Webový vývoj v F# (Frontend)

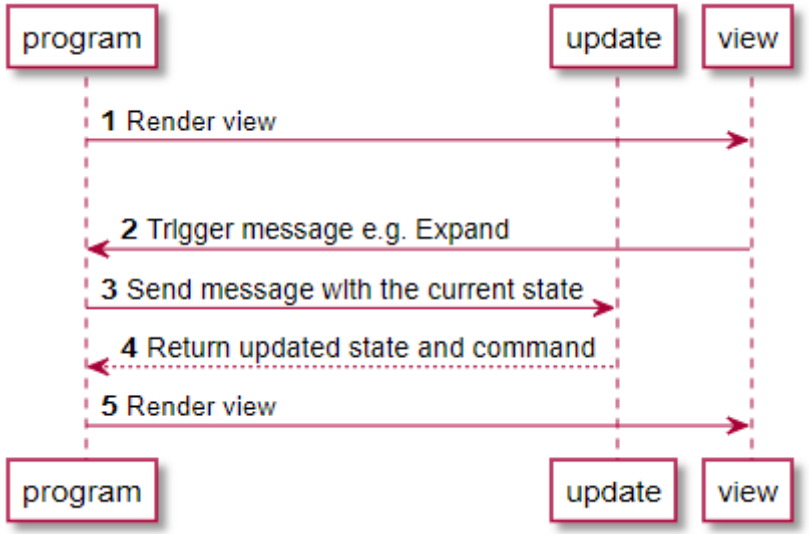
Fable (https://fable.io)
F# to Javascript compiler

Websharper (https://websharper.com)
Web based programming platform, F# to Javascript compiler

Oboje upřednostňuje **Elmish MVU** architekturu

+

Elmish – Model, View, Update



Model – stav vaší aplikace

View – vrací pro daný stav (Model) strom HTML komponent + může odesílat commands

Update – na základě stavu (Modelu) a commandů vrací nový stav

Full stack vývoj v F# - SAFE Stack

Open-source F# **end-to-end development stack**

https://safe-stack.github.io

Sdílený kód pro backend i frontend

Předpřipravené **.NET Core templates**

Saturn

Azure

Fable

Elmish

DEMO

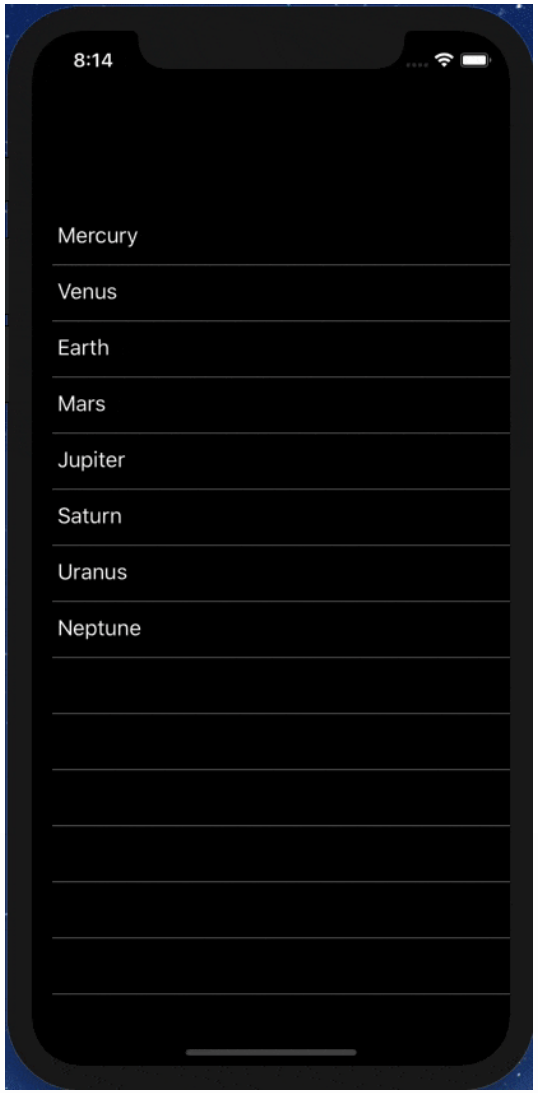
SAFE Stack + MVU ve Fable

Mobilní / desktop vývoj v F#

Fabulous (https://fsprojects.github.io/Fabulous)
Knihovna pro vývoj aplikací pro Android, iOS, UWP, WPF, macOS, GTK

Založena na **Xamarin.Forms**

Aplikuje **Elmish architekturu** na mobilní a desktop vývoj



Webassembly v F#

Bolero (https://fsbolero.io/)

Web assembly knihovna pro F#

Připravený **hot-reload, routing, remoting**, ...

Opět postaven na **Elmish MVU + Blazor**

Dostupné **.NET Core templates**

Serverless v F#

Podpora **Azure Functions**

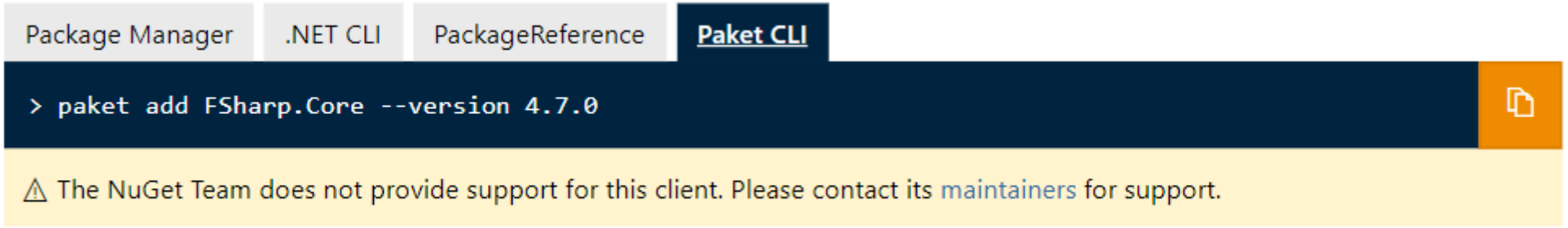
Knihovny pro lepší práci s **Durable functions**

Přirozené použití **F# pro serveless** (stateless přístup)

Nástroje v F#

FAKE (https://fake.build)
DSL pro build tasky

Paket (https://fsprojects.github.io/Paket/)
alternativní **NuGet** client
využívá **lock** file



Aktivní komunita

F# Foundation Slack

Oficiální meetupy (FSharping, F# Sydney, F# Vancouver,…)

Velmi aktivní **Gitter, GitHub**

Konference (F# eXchange London, Lambda Days Krakow, …)

Přátelská a ochotná komunita

F# v roce 2019

Web assembly

Full-stack web

Mobilní vývoj

Serverless

Desktop (WPF, UWP)

Komunita

MVU **(Elmish)**

Kde začít?

https://fsharpforfunandprofit.com/ - **Bible všech F#istů**

https://try.fsharp.org/ - **F# REPL** v prohlížeči

https://fsharping.com – Doražte na **meetup**

Díky a ptejte se! ☺

Roman Provazník

@rprovaznik | @fsharping

WUG Days 2019 | 14. září 2019