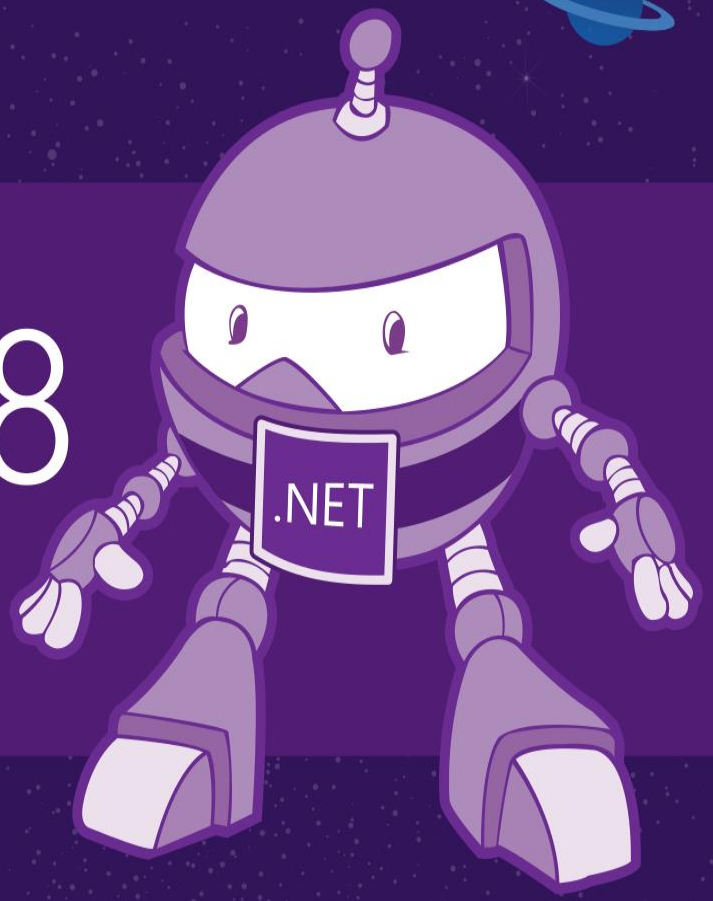
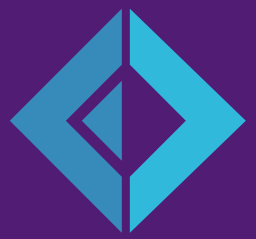
.NET Conf 2018

Discover the world of .NET



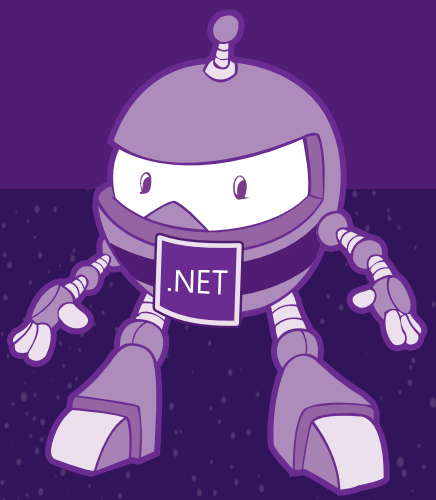
www.dotnetconf.net

初探 F# 函數式程式設計



Functional Programming in F#

Carl Su
Microsoft MVP



STUDY4.TW
為學習而生

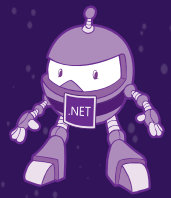
.NET Conf 2018

講者簡介

- 微軟最有價值專家 (Development Technologies)
- COSCUP 開源人年會 講者、議程協調人
- Hacking Thursday 社群值日生
- Docker Taipei 共同發起人
- Arch Linux Taiwan 共同發起人

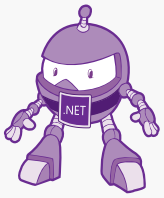
為何您該考慮用 F#？

.NET



因為選 F# 就對了！

- F# 重視資料不可變
- 函數可組合，具通用性
- 絕對不會遇到 `NullReferenceException`



因為 F# 非常簡潔

- 程式碼寫更少，省去多餘符號和型別標注
- 具備型別推導能力，自動認出正確的型別

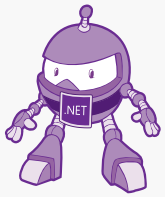
因為 F# 相容性很好

- F# 是 .NET 家族的成員之一
- 相容現有 C# 函式庫和框架
- 支援 Azure Functions 等多項服務

火力展示

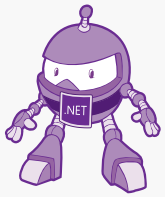
Conciseness

```
// one-liners
[1..100] |> List.sum |> printfn "sum=%d"

// no curly braces, semicolons or parentheses
let square x = x * x
let sq = square 42

// simple types in one line
type Person = {First:string; Last:string}

// complex types in a few lines
type Employee =
    | Worker of Person
    | Manager of Employee list

// type inference
let jdoe = {First="John";Last="Doe"}
let worker = Worker jdoe
```

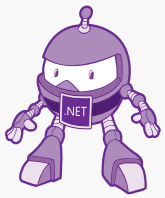
Convenience

```
// automatic equality and comparison
type Person = {First:string; Last:string}
let person1 = {First="john"; Last="Doe"}
let person2 = {First="john"; Last="Doe"}
printfn "Equal? %A"  (person1 = person2)

// easy IDisposable logic with "use" keyword
use reader = new StreamReader(..)

// easy composition of functions
let add2times3 = (+) 2 >> (*) 3
let result = add2times3 5
```

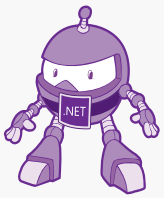
Correctness

```
// strict type checking
printfn "print string %s" 123 // compile error

// all values immutable by default
person1.First <- "new name"  // assignment error

// never have to check for nulls
let makeNewString str =
    // str can always be appended to safely
    let newString = str + " new!”
    newString

// embed business logic into types
emptyShoppingCart.remove   // compile error!

// units of measure
let distance = 10<m> + 10<ft> // error!
```

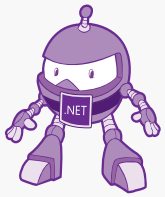
Concurrency

```
// easy async logic with "async" keyword
let! result = async {something}

// easy parallelism
Async.Parallel [ for i in 0..40 ->
    async { return fib(i) } ]

// message queues
MailboxProcessor.Start(fun inbox -> async{
    let! msg = inbox.Receive()
    printfn "message is: %s" msg })
```

Completeness

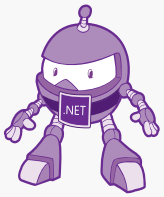
```
// impure code when needed
let mutable counter = 0

// create C# compatible classes and interfaces
type IEnumerator<'a> =
    abstract member Current : 'a
    abstract MoveNext : unit -> bool

// extension methods
type System.Int32 with
    member this.IsEven = this % 2 = 0

let i=20
if i.IsEven then printfn "'%i' is even" i
```

F# 適用情境

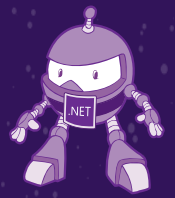
F# 適用於多種場合

- 資料分析與視覺化
  - Excel, R, MATLAB, Mathematica, Python
- Web 開發
  - Fable, ASP.NET Core, WebSharper, SignalR, ASP.NET Blazor, etc…
- 行動 APP / 手機遊戲
  - Fable, Fabulous, Xamarin, Unity3D, etc…
- 機器學習
  - Accord.MachineLearning, ML.NET, Rprovider for F#, DiffSharp, etc…
- 雲端運算
  - Azure, Akka.NET, Kafka, ZeroMQ, Hadoop, Storm, Cassandra, Neo4j, etc…
- 數學統計
  - Math.NET Numerics, ILNumerics, DiffSharp, FsAlg, Alea GPU, etc…



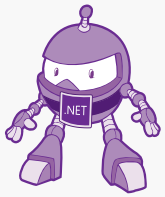
F# 支援平台

.NET



F# 支援所有主流平台

- Mac (Mono, .NET Core)
- Linux (Mono, .NET Core)
- Windows
  - Mono
  - .NET Framework
  - .NET Core
- Android (Xamarin Tools)
- iOS (Xamarin Tools)
- JavaScript / HTML5
  - Fable
  - WebSharper
- CUDA (Alea GPU)
- OpenCL
  - Brahma.Fsharp
  - FSCL
  - GpuLINQ
- FreeBSD (Mono)

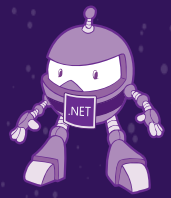


F# 開發環境

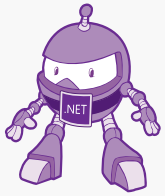
.NET

STUDY4.TW
為學習而生

.NET Conf 2018



F# 支援主流開發環境



Demo 1: Emacs (fsharp-mode)

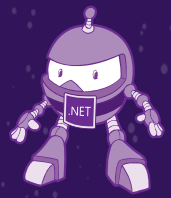
Demo 2: VSCode (Ionide)

FP 設計模式

.NET

STUDY4.TW
為 學 習 而 生

.NET Conf 2018



OOP 原則

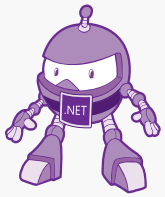
- 單一職責
- 開放封閉原則
- 里氏替換
- 介面隔離
- 依賴反轉

- 工廠模式
- 策略模式
- 修飾模式
- 訪問者模式

FP 世界

- 函數
- 函數
- 函數
- 函數
- 函數

- 函數
- 函數
- 函數
- 函數

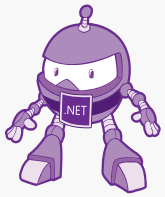


從 C# 到 F#

C# 和 F# 核心概念差異

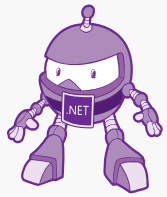
C#	F#
Variables	Immutable values
Statements	Expressions
Objects with Methods	Types and functions

https://blogs.msdn.microsoft.com/dotnet/2017/07/24/get-started-with-f-as-a-c-developer/



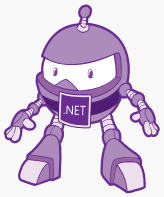
使用表達式，不使用陳述式

```
let getMessage name =
  if name = "Phillip" then   // 'if' is an expression.
    "Hello, Phillip!"        // This string is an expression.
  else
    "Hello, other person!"   // Same with this string.

// getMessage, when called, is an expression.
// Its value is bound to 'phillipMessage'.
let phillipMessage = getMessage "Phillip"

// This expression is bound to 'alfMessage'!
let alfMessage = getMessage "Alf"
```

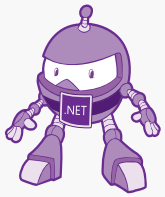
F# 常用集合

- 陣列 (Array)
  - 資料可以修改，立即求值
- 清單 (List)
  - 不可變的單向鏈結串列，適用於模式比對 (pattern matching)，立即求值
- 序列 (Sequence)
  - 不可變的 `IEnumerable<T>`，惰性求值



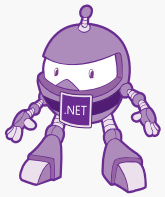
LINQ 對應的 F# 函數

LINQ	F#
`Where`	`filter`
`Select`	`map`
`GroupBy`	`groupBy`
`SelectMany`	`collect`
`Aggregate`	`fold, reduce`
`Sum`	`sum`



函數管道 (Functional Pipelines)

```
let square x = x * x
let isOdd x = x % 2 <> 0

let getOddSquares items =
    items
    |> Seq.filter isOdd
    |> Seq.map square
```

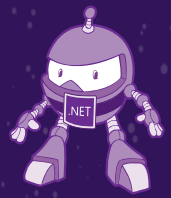



λ 演算 (λ-calculus)

.NET

STUDY4.TW
為 學 習 而 生

.NET Conf 2018

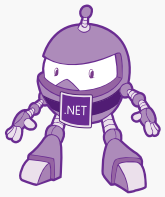


λ 演算概念

語法	名稱
x	變數 (Variable)
(λx.M)	抽象化 (Abstraction)
(M N)	套用 (Application)

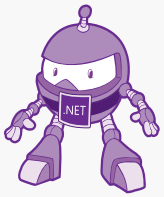
λx.x^2

```
fun x -> x * x
```

λ 演算求值

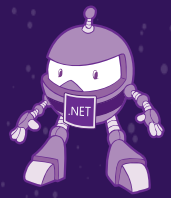
- (+ (* 5 6) (* 8 3))
- (+ 30 (* 8 3))
- (+ 30 24)
- = 54



F# 學習資源

.NET

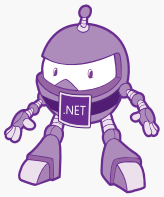
STUDY4.TW
為 學 習 而 生



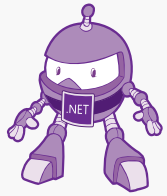
.NET Conf 2018

F# 推薦學習資源

- https://fsharp.org/
- https://fsharpforfunandprofit.com/
- https://en.wikibooks.org/wiki/F_Sharp_Programming
- https://www.youtube.com/watch?v=KPa8Yw_Navk
- https://docs.microsoft.com/en-us/dotnet/fsharp/style-guide/
- https://oomusou.io/tags/#FP (中文)
- https://oomusou.io/tags/#F# (中文)
- https://github.com/fsprojects/awesome-fsharp



特別感謝



.NET Conf 2018