Theory of Programming Languages

A practical perspective

Vincenzo Ciancia

Istituto di Scienza e Tecnologie dell’Informazione
Consiglio Nazionale delle Ricerche, Pisa

Lecture 1: Short Intro, and Setup

Quick intro

“Programming languages?”

You know what a programming language is!

Do you know?

Do you **actually** know?

Then what is a “programming language”?

“Programming languages?”

A programming language *can be* a lot of things, actually.

Syntax	**Semantics**
Interpreter	**Compiler**

“Syntax”

Syntax can be a lot of things (maybe) but for our purposes:

Context free grammars

Syntax tree

“Semantics”

Semantics can be a lot of things (for sure) among which:

Operational	**Denotational**	**Logical**
Transition systems	**Rewrite Rules**	**Term/Graph Rewriting**
Fixed point	**Category Theory**	**Equivalences**
Coalgebras	**Dialgebras**	**Bialgebras**
Bisimilarity	**Simulation**	**Many other things…**

“Interpreter“

Execute a program, written in a formal language.

REPL:

Read **Eval** **Print** **Loop**

“Compiler”

Translate a program from a source language to a target language

The result will be run later

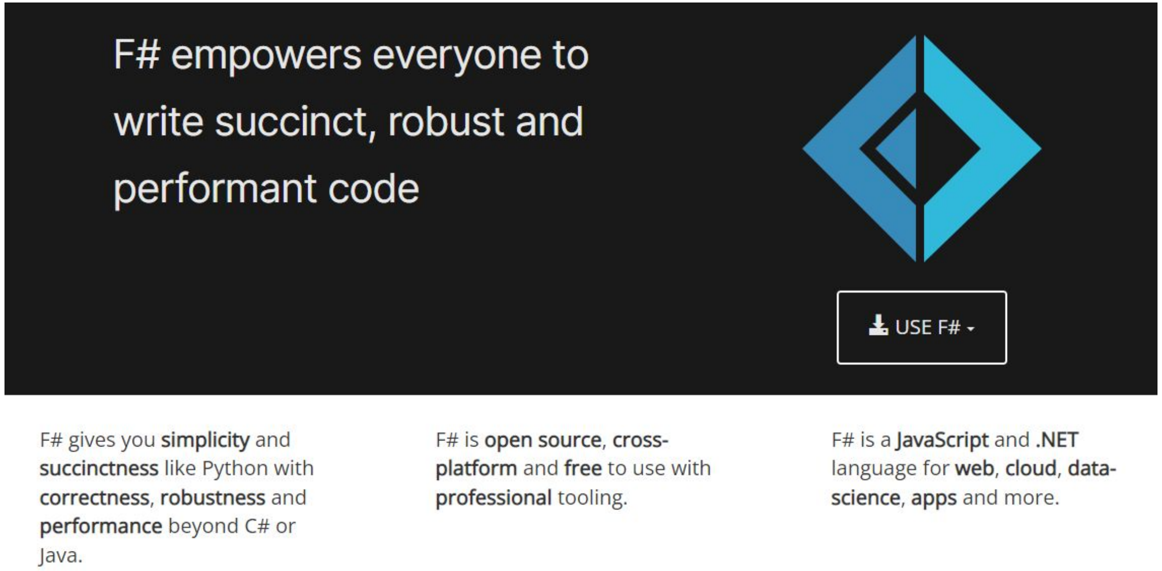
Let us name a few compilers

TO BE CONTINUED...

Setup (FSharp & C)

What is FSharp?

https://fsharp.org/



F# empowers everyone to write succinct, robust and performant code

F# gives you **simplicity** and **succinctness** like Python with **correctness**, **robustness** and **performance** beyond C# or Java.

F# is **open source**, **cross-platform** and **free** to use with **professional** tooling.

F# is a **JavaScript** and **.NET** language for **web**, **cloud**, **data-science**, **apps** and more.

Why FSharp

We are going to implement toy-like **interpreters**

Main concepts: Abstract Syntax, Recursive Functions, Functions as Values

A functional programming language is perfect for the job

Do we even know what a functional language is?

Why FSharp

We are going to implement toy-like **interpreters**

Main concepts: Abstract Syntax, Recursive Functions, Functions as Values

A functional programming language is perfect for the job

Do we even know what a functional language is?

Do we even know what a functional language is?

Functions are first-class citizens (can be passed as arguments to other functions)

Recursion is also pivotal

Abstract Data Types

FSharp for this course

Strongly typed language

Parametric types

Easy to try (even online)

FSharp for Life

Functional and Object-Oriented

Fully integrated in **dotnet: interoperable with C#, access to all .net packages**

Free and Open Source

Mature, Production Ready

Easy deployment: cross-compilation from any of linux, windows, osx to any other

How to use it

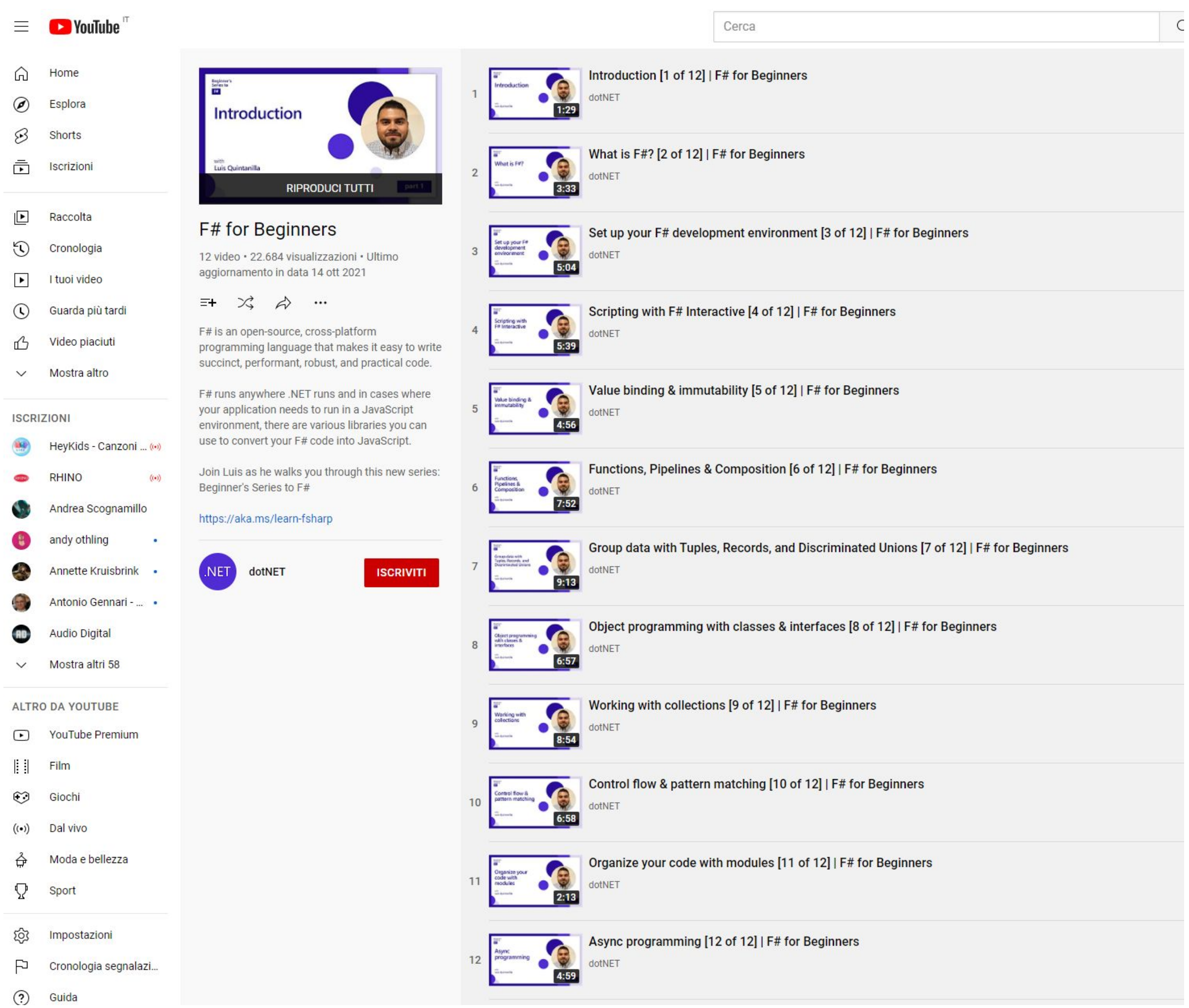
https://dotnet.microsoft.com/en-us/download

(compile, or “dotnet fsi” to run an interpreter)

https://fsharp.org/use/browser/

https://try.fsharp.org/

(interpreter, press alt+enter to evaluate)



F# for Beginners

12 video • 22.684 visualizzazioni • Ultimo aggiornamento in data 14 ott 2021

F# is an open-source, cross-platform programming language that makes it easy to write succinct, performant, robust, and practical code.

F# runs anywhere .NET runs and in cases where your application needs to run in a JavaScript environment, there are various libraries you can use to convert your F# code into JavaScript.

Join Luis as he walks you through this new series: Beginner's Series to F#

https://aka.ms/learn-fsharp

dotNET

1	Introduction [1 of 12] | F# for Beginners	dotNET	1:29
2	What is F#? [2 of 12] | F# for Beginners	dotNET	3:33
3	Set up your F# development environment [3 of 12] | F# for Beginners	dotNET	5:04
4	Scripting with F# Interactive [4 of 12] | F# for Beginners	dotNET	5:39
5	Value binding & immutability [5 of 12] | F# for Beginners	dotNET	4:56
6	Functions, Pipelines & Composition [6 of 12] | F# for Beginners	dotNET	7:52
7	Group data with Tuples, Records, and Discriminated Unions [7 of 12] | F# for Beginners	dotNET	9:13
8	Object programming with classes & interfaces [8 of 12] | F# for Beginners	dotNET	6:57
9	Working with collections [9 of 12] | F# for Beginners	dotNET	8:54
10	Control flow & pattern matching [10 of 12] | F# for Beginners	dotNET	6:58
11	Organize your code with modules [11 of 12] | F# for Beginners	dotNET	2:13
12	Async programming [12 of 12] | F# for Beginners	dotNET	4:59

VSCODE USER?

Install the extension called “ionide” for full FSharp support.

You can also run a FSharp interpreter with

```
CTRL+SHIFT+P -> FSI: Start
```

Let’s dive in!

https://github.com/vincenzoml/ProgrammingLanguages

What can ONE do with FSharp

IN THE LARGE:

FSharp is a functional language which is also object oriented and imperative

The language can call native C libraries, manage native memory efficiently

One can even use the GPU

What can YOU do with FSharp

IN THE SMALL:

You can compute expressions: numbers, lists, booleans, records, custom data types...

You can read and write variables and arrays if needed.

You can print, load and save files.

What can you do with FSharp

IN THE SMALL:

You can define functions, anonymous functions, and pass them to other functions.

You can define classes and objects.

You can define modules (libraries).

How to create and run a new project

```
dotnet new console -lang F# -n "Lecture01"

cd Lecture01

dotnet run
```

How to cross-compile from linux to windows

```
dotnet build -c Release -r win-x64 --self-contained # From linux

.\bin\Release\net6.0\win-x64\Lecture01.exe # From windows
```

How to cross-compile from osx to windows

```
dotnet build -c Release -r win-x64 --self-contained # From osx

.\bin\Release\net6.0\win-x64\Lecture01.exe # From windows
```

How to cross-compile from osx to linux

```
dotnet build -c Release -r linux-x64 --self-contained # From osx

.\bin\Release\net6.0\linux-x64\Lecture01 # From linux
```

More on this?

You can also “publish” that is, create a portable executable (to zip and pass around)

List of runtime identifiers: https://docs.microsoft.com/it-it/dotnet/core/rid-catalog

Note: all these features are in common with C#, the “imperative” brother of FSharp.

So far so good, what about coding?

```
printfn("Hello from F#")
```

Wow we are printing!

The “n” in “printfn” means: print also a newline please

Parentheses are optional!

```
printfn "Hello from F#"
```

It’s called printf “because C”…

```
printfn "%d" 3

printfn "%s" "hello"

printfn "The result of %d + %d is %d" 3 2 5
```

https://docs.microsoft.com/it-it/dotnet/fsharp/language-reference/plaintext-formatting

Placeholders here, placeholders there…

```
printfn "%A %A %A" 3.0 3 "3"
```

That’s the “object” placeholder which will print anything…

…using its “`toString()`” method.

Follow us in the repository!!!

This lecture continues as “Lecture01” in the repository (see Program.fs therein)

https://github.com/vincenzoml/ProgrammingLanguages

License

This work is licensed under the Creative Commons Attribution 4.0 International License.

To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/.