CONTRIBUTING TO THE F# COMPILER

AVI AVNI

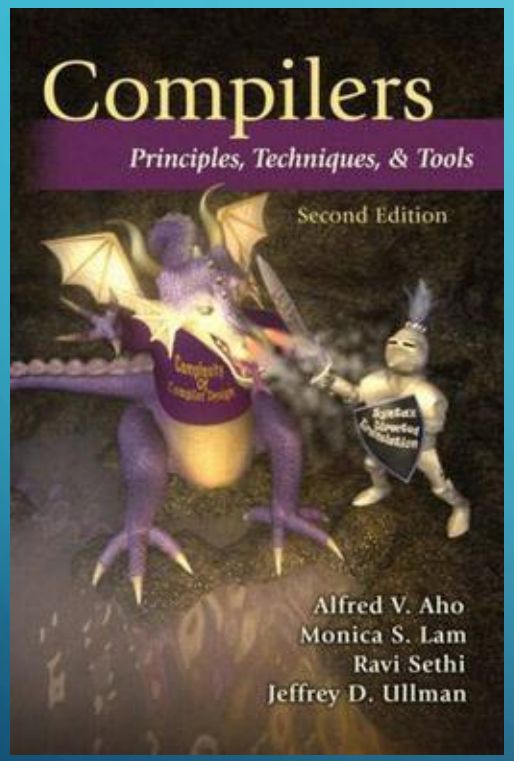
WHO I AM?

- Consultant at SELA
- 8 years experience
- .NET, Cloud, Big Data
- Neo4j Certified Professional
- Contributor to several open source projects


b a b y
MILESTONE

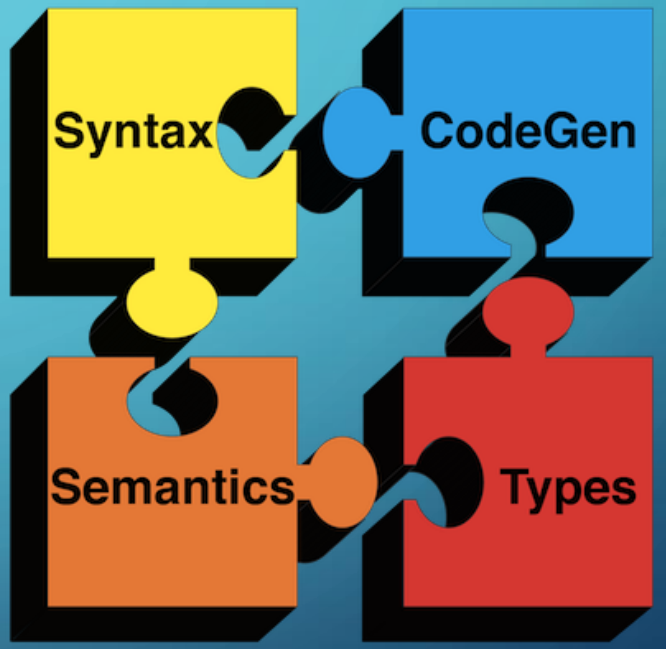

DREAM

LEARNING RESOURCES



https://en.wikipedia.org/w/index.php?curid=11248852

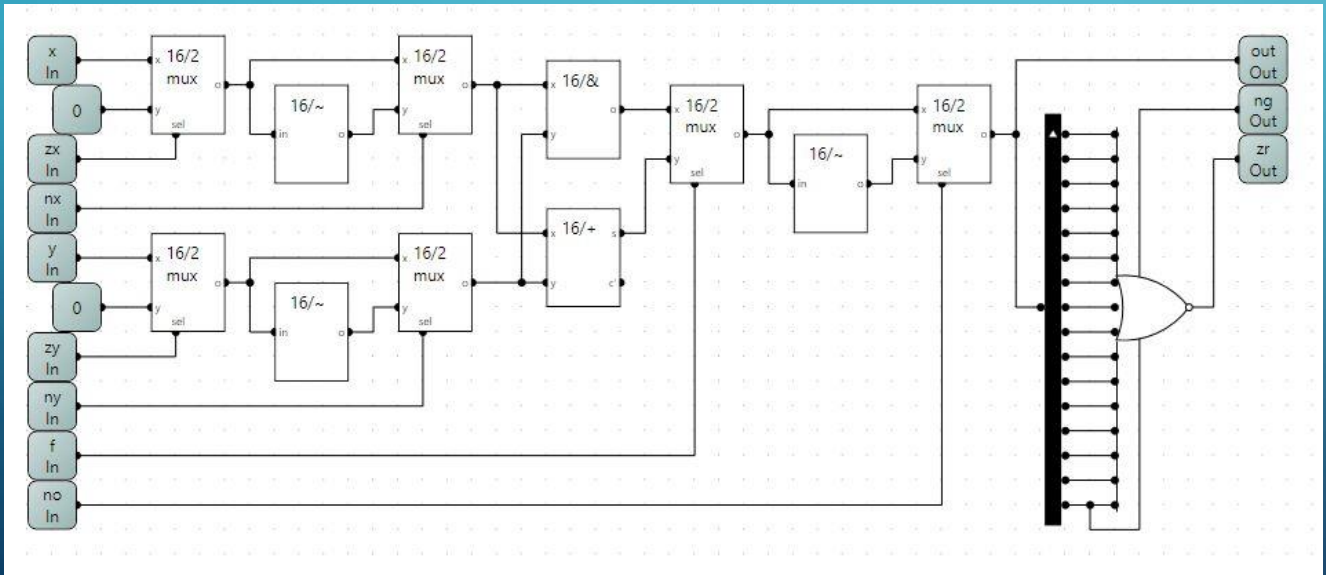
https://www.springer.com/gp/book/9783319607887



http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=Compilers

LEARNING RESOURCES

- Nand2Tetris project
- Created by Professor Shimon Schocken



Tetris

Compiler

Virtual Machine

Assembler

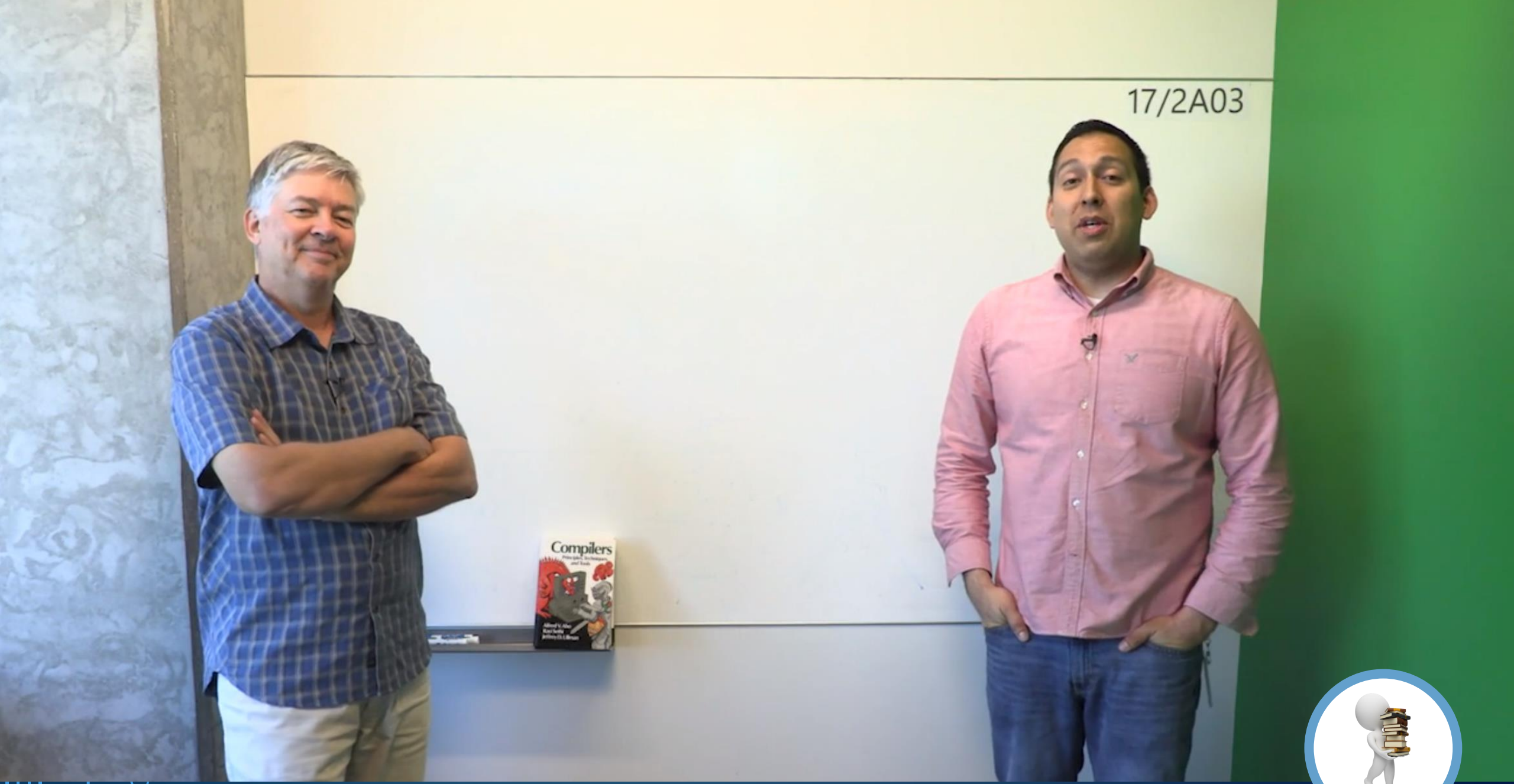
CPU

Logic Gates

NAND

COMPILER SUMMARY

Tokenizer	Parser	Type Checker	Optimizer	Code Generator
Source -> Tokens	Tokes -> AST	AST -> TAST	TAST -> TAST	TAST -> IL



https://channel%20nredom=mret?noitcurtsnoC-relipmoC-nredoM-no-grebsljeH-srednA/zerauJ-hteS/sgolB/moc.ndsm.9compiler%20construction

COMPILER EDITOR FEATURES

Tokenizer	Parser	Type Checker	Optimizer	Code Generator
Source -> Tokens	Tokes -> AST	AST -> TAST	TAST -> TAST	TAST -> IL
Basic Syntax Highlighting	Breakpoints	Tooltips		FSI
	Code outlining	IntelliSense		
		Rename		
		Find References		

MENTORSHIP PROGRAM

DEVELOPMENT PROCESS - WIN

- How to compile the F# compiler?
  - Fork and Clone the repository
  - Install prerequisite software (DEVGUIDE.md)
  - Run “build.cmd debug” in CLI
- How to debug the compiler?
  - Write some test code
  - Run “fsc.exe test.fs --pause“ in CLI
  - Open VisualFSharp.sln in Visual Studio 2017
  - Attach the debugger to fsc.exe process
  - Add breakpoint

DEVELOPMENT PROCESS

VisualFSharp (Debugging) - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Architecture Test AI Tools R Tools Analyze Window Help

Debug | Any CPU | FSharp.LanguageService | Continue | Application Insights

Process: [8040] fsc.exe | Lifecycle Events | Thread: [18420] Main Thread | Stack Frame: Microsoft.FSharp.Compiler.Driver.mainCo

fsc.fs

```
2103        |> main4 dynamicAssemblyCreator
2104
2105
2106    let compileOfAst (ctok, legacyReferenceResolver, reduceMemoryUsage, assemblyName, target, outFile, pdbFile, dllReferences,
2107        main1OfAst (ctok, legacyReferenceResolver, reduceMemoryUsage, assemblyName, target, outFile, pdbFile, dllReferences, no
2108        |> main2a
2109        |> main2b (tcImportsCapture, dynamicAssemblyCreator)
2110        |> main3
2111        |> main4 dynamicAssemblyCreator
2112
2113    let mainCompile (ctok, argv, legacyReferenceResolver, bannerAlreadyPrinted, reduceMemoryUsage, defaultCopyFSharpCore, exite
2114        //System.Runtime.GCSettings.LatencyMode <- System.Runtime.GCLatencyMode.Batch
2115        typecheckAndCompile(ctok, argv, legacyReferenceResolver, bannerAlreadyPrinted, reduceMemoryUsage, defaultCopyFSharpCore
2116
2117
```

Diagnostic Tools

Diagnostics session: 12 seconds (12.919 s selected)

10s

Events

Process Memory (MB)

43 43

0 0

CPU (% of all processors)

100 100

0 0

Summary | Events | Memory Usage | CPU Usage

Events

Show Events (1 of 1)

Exceptions (0

IntelliTrac

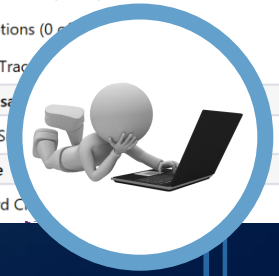
Memory Usa

Take S

CPU Usage

Record C

Solution Explorer | Team Explorer | Properties



DEVELOPMENT PROCESS - LINUX

- How to compile the F# compiler?
  - Fork and Clone the repository
  - Install prerequisite software (DEVGUIDE.md)
  - Run “scr/buildfromsource.sh” in CLI
  - Copy fsc.deps.json and fsc.runtimeconfig.json from sdk folder to binaries folder
- How to debug the compiler?
  - Write some test code
  - Run “dotnet fsc.exe test.fs --pause“ in CLI
  - Open VisualFSharp folder with Visual Studio Code
  - Attach the debugger to fsc.exe process
  - Add breakpoint

DEVELOPMENT PROCESS

- F# compiler source code is approximately 200K LOC
- No need to learn all code at once
- Focus on small task each time
- Where is the right place to change something?
- Don’t forget to write some tests

DEVELOPMENT PROCESS

- Improving performance
- More readable error message
- Fixing bugs
- Implement new features

F# COMPILER - DEMO

Tokenizer	Parser	Type Checker	Optimizer	Code Generator
Source -> Tokens	Tokes -> AST	AST -> TAST	TAST -> TAST	TAST -> IL
lex.fsl	pars.fsy	TypeChecker.fs	Optimizer.fs	IlxGen.fs
	ast.fs	tast.fs		

1ST TASK

```
C:\Windows\System32\cmd.exe - fsi.exe
Microsoft Windows [Version 10.0.17134.191]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Repos\visualfsharp\debug\net40\bin>fsi.exe

Microsoft (R) F# Interactive version 10.1.0 for F# 4.1
Copyright (c) Microsoft Corporation. All Rights Reserved.

For help type #help;;

> type I3 = int * * 3;;
type I3 = int * int * int

> let i3 : I3 = 1, 2, 3;;
val i3 : I3 = (1, 2, 3)

>
```

WHERE THIS FEATURE BELONG?

Tokenizer	Parser	Type Checker	Optimizer	Code Generator
Source -> Tokens	Tokes -> AST	AST -> TAST	TAST -> TAST	TAST -> IL
lex.fsl	pars.fsy	TypeChecker.fs	Optimizer.fs	IlxGen.fs
	ast.fs	tast.fs		

1ST TASK - PARSING

- Open ast.fs and look at SynType.Tuple discriminated union
- Open pars.fsy and locate tupleType parsing rule
- Add a rule to parse type like: int * * 3 and transform it to : int, int, int
- Build the compiler
- Try in fsi

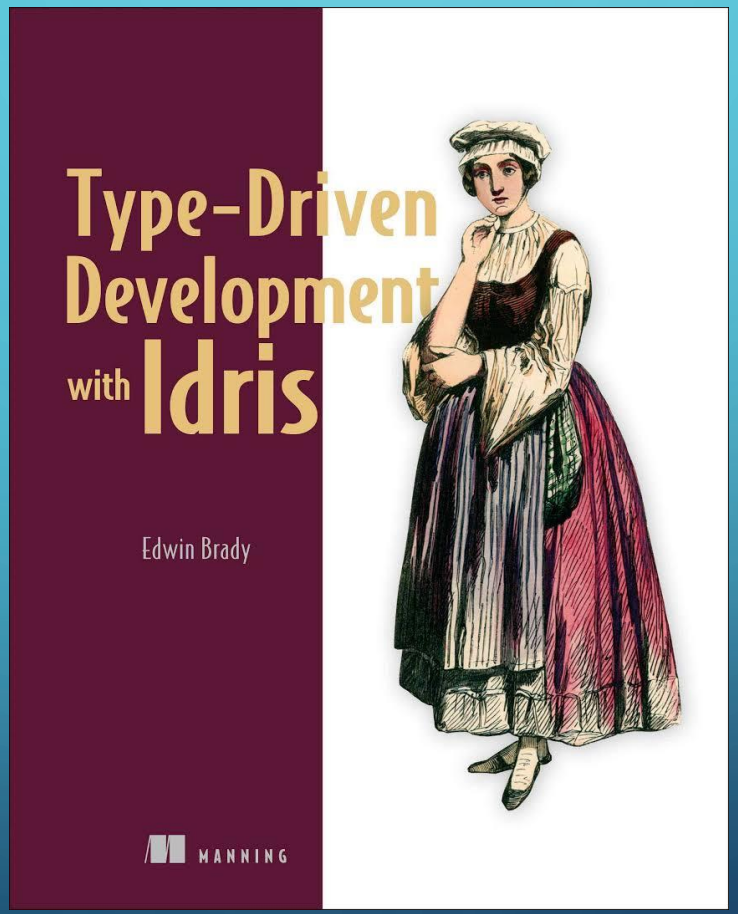
COMMUNITY FEEDBACK

forki commented on Jun 3, 2016 — Contributor

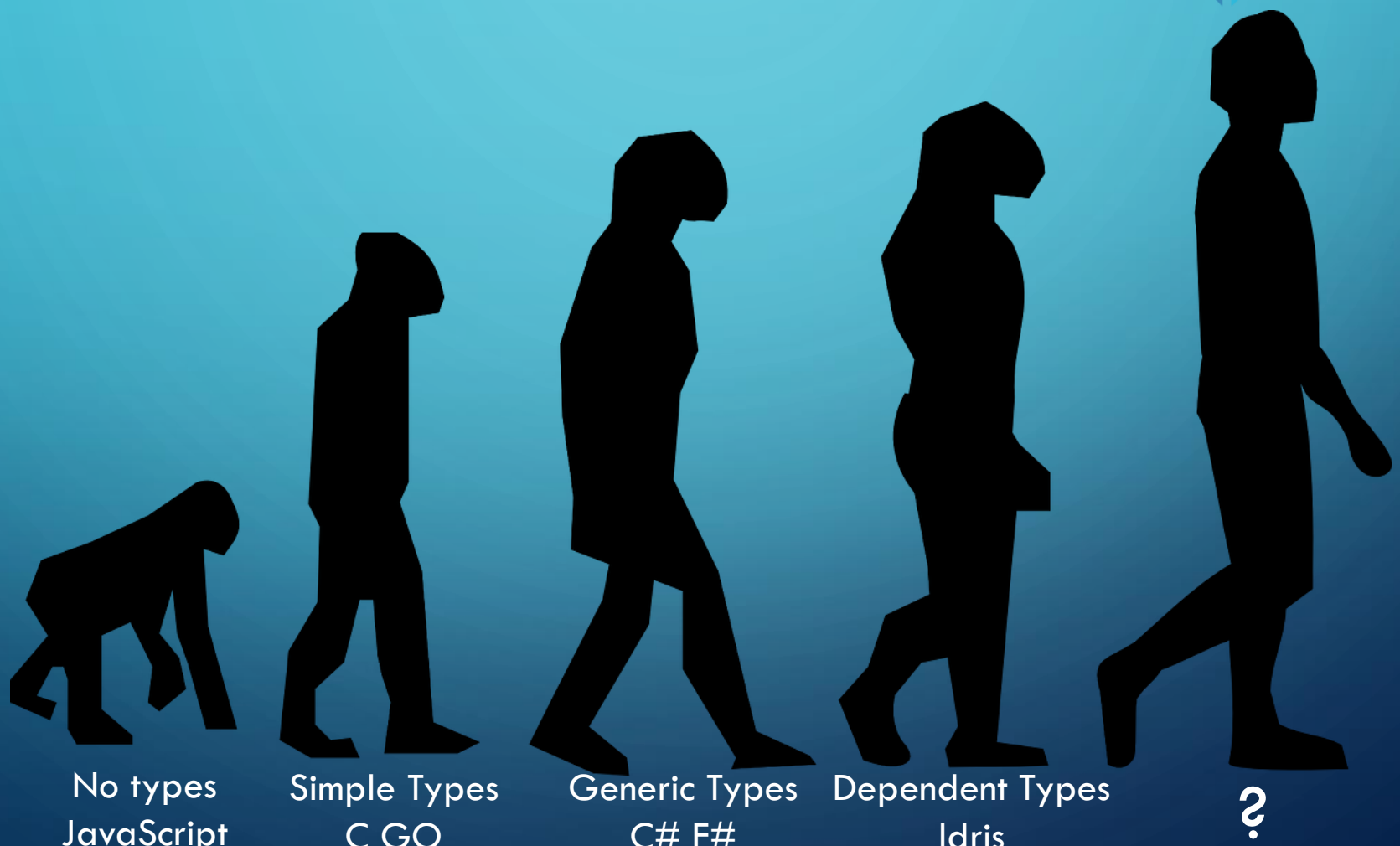
added few comments. apart from that:

1 1

DEPENDENT TYPE



LANGUAGE EVOLUTION - DEMO




CAUTION

2ND TASK

C:\Windows\System32\cmd.exe - fsi.exe

```
C:\Repos\visualfsharp\debug\net40\bin>fsi.exe

Microsoft (R) F# Interactive version 10.1.0 for F# 4.1
Copyright (c) Microsoft Corporation. All Rights Reserved.

For help type #help;;

> type I3 = int * 3;;
type I3 = int * int * int

> let i3 : I3 = 1, 2, 3;;
val i3 : I3 = (1, 2, 3)

> type I33 = int * 3 * 3;;
type I33 = int * int * int * int * int * int * int * int * int

> let i33 : I33 = 1, 2, 3, 4, 5, 6, 7, 8, 9;;
val i33 : I33 = (1, 2, 3, 4, 5, 6, 7, 8, 9)

> _
```

WHERE THIS FEATURE BELONG?

Tokenizer	Parser	Type Checker	Optimizer	Code Generator
Source -> Tokens	Tokes -> AST	AST -> TAST	TAST -> TAST	TAST -> IL
lex.fsl	pars.fsy	TypeChecker.fs	Optimizer.fs	IlxGen.fs
	ast.fs	tast.fs		

2ND TASK – PARSING

- Open ast.fs locate the SynType discriminated union add the Nat option and implement the Range property
- Open pars.fsy locate atomType rule change the result of parsing rawConstant to to return SynType.Nat in case that rawConstant is SynConst.Int32

2ND TASK – TYPE CHECKING

- Open tast.fs locate TType discriminated union add TType_nat option and implement GetAssemblyName and ToString methods
- Open TypeChecker.fs locate TcTypeOrMeasure function and implement the conversion between SynType.Nat to TType_nat
- Then locate TcTyconDefnCore_CheckForCyclicAbbreviations function and implement that TType_nat is ignored in checking cycles in abbreviations
- Then locate TcTypesAsTuple function and implement multiplication of types

2ND TASK – FIX ERRORS

- Open the following files and check how I changed them
  - TastOps.fs
  - TastPickle.fs
  - NicePrint.fs
  - ConstraintSolver.fs
  - PostInferenceChecks.fs
  - IlxGen.fs
  - Symbol.fs

3RD TASK

C:\Windows\System32\cmd.exe - fsi.exe

```
C:\Repos\visualfsharp\debug\net40\bin>fsi.exe

Microsoft (R) F# Interactive version 10.1.0 for F# 4.1
Copyright (c) Microsoft Corporation. All Rights Reserved.

For help type #help;;

> type Vect<'n, 'a> = 'a * 'n;;
type Vect<'n,'a> = 'a * 'n

> type I3 = Vect<3, int>;;
type I3 = Vect<3,int>

> let i3 : I3 = 1, 2, 3;;
val i3 : I3 = (1, 2, 3)

>
```

WHERE THIS FEATURE BELONG?

Tokenizer	Parser	Type Checker	Optimizer	Code Generator
Source -> Tokens	Tokes -> AST	AST -> TAST	TAST -> TAST	TAST -> IL
lex.fsl	pars.fsy	TypeChecker.fs	Optimizer.fs	IlxGen.fs
	ast.fs	tast.fs		

3RD TASK – TYPE CHECKEING

- Open ConstraintSolver.fs locate SolveTypeEqualsType debug around this code
- Refactor the multiplication of types and call it when a tuple with generic type parameter inferred


WHAT IS THE PROBLEM?

4TH TASK

C:\Windows\System32\cmd.exe - fsi.exe

```
C:\Repos\openfsharp\visualfsharp\debug\net40\bin>fsi.exe

Microsoft (R) F# Interactive version 10.1.0 for F# 4.1
Copyright (c) Microsoft Corporation. All Rights Reserved.

For help type #help;;

> type Vect<'n, 'a> = | Nil | Cons of 'a * Vect<'n - 1, 'a>;;
type Vect<'n,'a> =
  | Nil
  | Cons of 'a * Vect<'n - 1,'a>

> let v2 : Vect<2, int> = Cons(1, Cons(2, Nil));;
val v2 : Vect<2,int> = Cons (1,Cons (2,Nil))

> let v2 : Vect<2, int> = Cons(1, Cons(2, Cons(3, Nil)));;

  let v2 : Vect<2, int> = Cons(1, Cons(2, Cons(3, Nil)));;
  -----------------------------------------------^^^

stdin(3,49): error FS0001: Nat can't be less then 0

> _
```

WHERE THIS FEATURE BELONG?

Tokenizer	Parser	Type Checker	Optimizer	Code Generator
Source -> Tokens	Tokes -> AST	AST -> TAST	TAST -> TAST	TAST -> IL
lex.fsl	pars.fsy	TypeChecker.fs	Optimizer.fs	IlxGen.fs
	ast.fs	tast.fs		

4TH TASK

- Based on 2nd task implement parsing of types like ‘n - 1
- Refactor the logic for calculating multiplication to support additional operations
- Implement type checking
- Raise error when nat is less then 0

5TH TASK

C:\Windows\System32\cmd.exe - fsi.exe

```
Microsoft (R) F# Interactive version 10.1.0 for F# 4.1
Copyright (c) Microsoft Corporation. All Rights Reserved.

For help type #help;;

> type Vect<'n, 'a> = | Nil | Cons of 'a * Vect<'n - 1, 'a>;;
type Vect<'n,'a> =
  | Nil
  | Cons of 'a * Vect<('n - 1),'a>

> let nil<'a> : Vect<0, 'a> = Nil;;
val nil<'a> : Vect<0,'a>

> let v1 = Cons(1, nil<_>);;
val v1 : Vect<1,int> = Cons (1,Nil)

> let v2 = Cons(1, Cons(2, nil<_>));;
val v2 : Vect<2,int> = Cons (1,Cons (2,Nil))

> let v3 = Cons(1, Cons(2, Cons(3, nil<_>)));;
val v3 : Vect<3,int> = Cons (1,Cons (2,Cons (3,Nil)))
```

6TH TASK

C:\Windows\System32\cmd.exe - fsi.exe

```
> type Vect<'n, 'a> = | Nil | Cons of 'a * Vect<'n - 1, 'a>;;
type Vect<'n,'a> =
  | Nil
  | Cons of 'a * Vect<('n - 1),'a>

> let rec map<'n, 'a, 'b> (f:'a -> 'b) (v:Vect<'n, 'a>) : Vect<'n, 'b> =
-   match v with
-   | Nil -> Nil
-   | Cons(a, v) -> Cons(f a, map<'n - 1, 'a, 'b> f v);;
val map : f:('a -> 'b) -> v:Vect<'n,'a> -> Vect<'n,'b>

> let nil<'a> : Vect<0, 'a> = Nil;;
val nil<'a> : Vect<0,'a>

> let v2 = Cons(1, Cons(2, nil));;
val v2 : Vect<2,int> = Cons (1,Cons (2,Nil))

> let add1 x = x + 1;;
val add1 : x:int -> int

> map add1 v2;;
val it : Vect<2,int> = Cons (2,Cons (3,Nil))
```

BONUS

```
let rec append<'n, 'm, 'a> (xs:Vect<'n, 'a>) (ys:Vect<'m, 'a>) : Vect<'n + 'm, 'a> =
  match xs, ys with
  | Nil, ys -> ys
  | Cons(x, xs), ys -> Cons(x, append<'n - 1, 'm, 'a> xs ys)
```

INSIGHTS FROM MY EXPERIENCE

- It's fun ☺
- Easier than you think (for some tasks)
- Makes you a better programmer
- Any contribution is great
- Thanks to Phillip Carter and Channel9 for the promotion