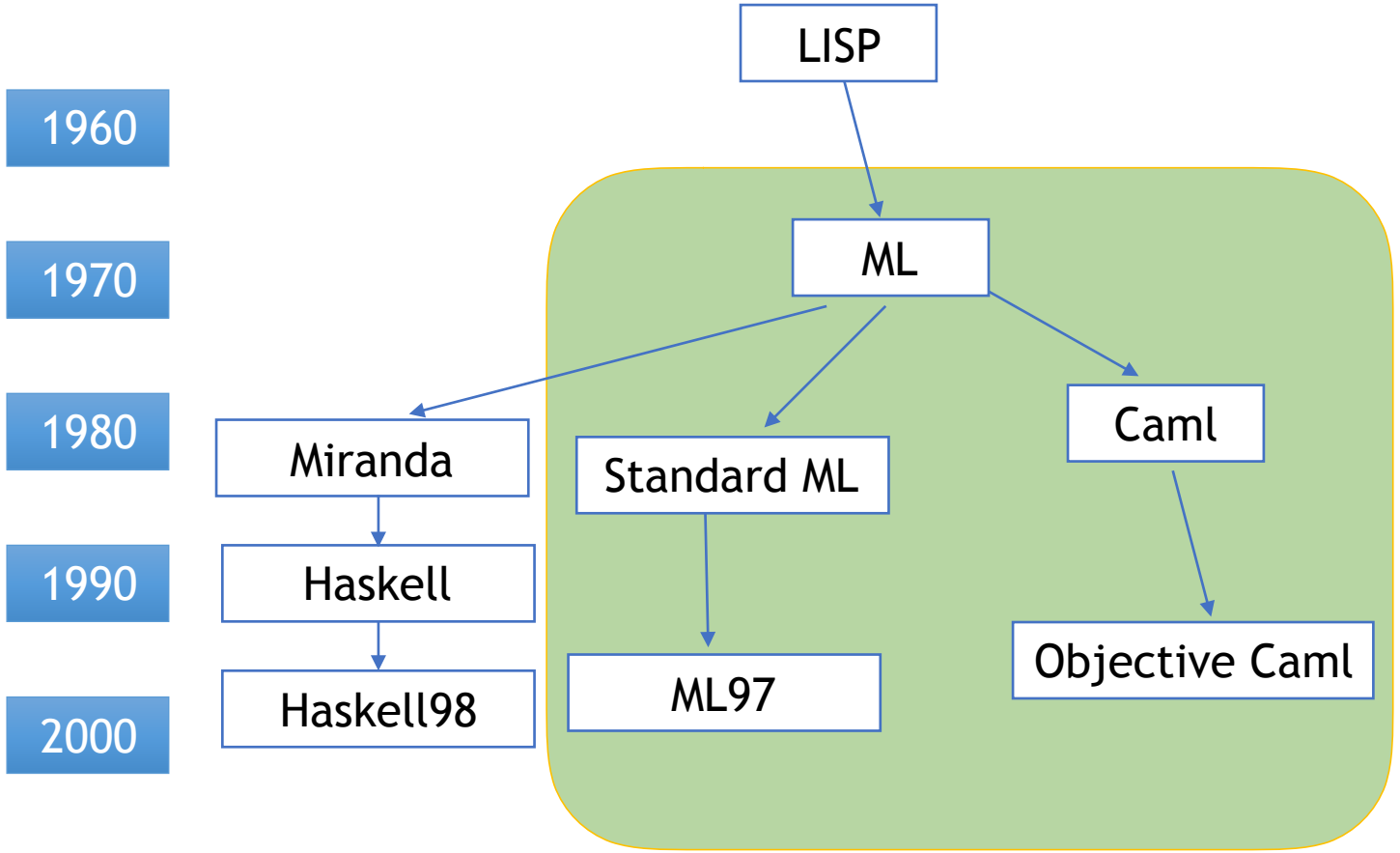
F#

Олег Заимкин, ДатаВоркс, Новосибирск

Что такое F#?


1960
1970
1980
1990
2000
LISP
ML
Miranda
Haskell
Haskell98
Standard ML
ML97
Caml
Objective Caml


OCaml
Microsoft
.NET

О языке

- FP-first
- Типизация
- АТД + ПП
- Pattern matching
- Null безопасность
- Вывод типов

```
    /// Implementation of "dry run"
    let dryRun ctx options (groups: string list list) =
        let getDeps = getChangeReasons ctx ▷ memoizeRec

        // getPlainDeps getDeps (getExecTime ctx)
        do ctx.Logger.Log Command "Running (dry) targets %A" groups
        let doneTargets = System.Collections.Hashtable()

        let print f = ctx.Logger.Log Info f
        let indent i = String.replicate i "   "

        let rec showDepStatus ii reasons =
            reasons ▷ function
            | ChangeReason.Other reason →
                print "%sReason: %s" (indent ii) reason
            | ChangeReason.Depends t →
                print "%sDepends '%s' - changed target" (indent ii) t.ShortName
            | ChangeReason.DependsMissingTarget t →
                print "%sDepends on '%s' - missing target" (indent ii) t.ShortName
            | ChangeReason.FilesChanged (file:: rest) →
                print "%sFile is changed '%s' %s" (indent ii) file (if List.isEmpty
List.length rest)
            | reasons →
                do print "%sSome reason %A" (indent ii) reasons
            ()
        let rec displayNestedDeps ii =
            function
            | ChangeReason.DependsMissingTarget t
            | ChangeReason.Depends t →
                showTargetStatus ii t
            | _ → ()
```

Алгебраические типы данных (ADT)

```
// simple types
type Nothing = unit
type ChannelId = int
type Message = string

type MyFun = int → string


// Product types
type Timestamp = int * DateTime
let ts = (0, DateTime.Now)
```

Алгебраические типы данных (ADT)

```
type UserInfo = {ident: int; name: string}

type Channel = {
  Messages: Message list
  Info: string
  Users: UserInfo list
  PostText: string
}

// Экземпляр типа
let chan = {Messages = []; Info = "None"; Users = []; PostText = ""}
```

ADT: типы-суммы

```
type Shape =
  | Circle of float
  | Square of float
  | Rectangle of float * float


let shape = Square 42.0
```

FP: Pattern matching

```
type Shape =
  | Circle of float
  | Square of float
  | Rectangle of float * float

let shape = Square 42.0
```

```
let area shape =
  match shape with
  | Circle r → Math.PI * r * r
  | Square x → x * x
  | Rectangle (w,h) → w * h
```

Рекурсия и ПП

```
type List<'T> =
  | Nil
  | Cons of 'T * List<‘T>

let list = Cons(1, Cons(2, Cons (3, Nil)))

let foo = 1 :: 2 :: 3 :: List.Empty
let listToo = [1; 2; 3]


let rec pairProduct ls =
  match ls with
  | Cons (a, Cons (b, tail)) → Cons(a * b, pairProduct tail)
  | other → other
```

FP: Active patterns

```
let (|IsChannelId|_|) s =
  match Int32.TryParse s with
  | true, value → Some value
  | _ → None


let chan = function
  | IsChannelId chanId → printf "%i" chanId
  | any → printf "not a channel identifier"
```

FP: иммутабельность

```
let update (msg: ChannelMessage) (channel: Channel) : Channel =

  match msg with
  | Init (info, userlist) →
    { Info = info; Messages = []; PostText = ""; Users = userlist}

  | Update info →
    { channel with Info = info }

  | AppendMessage message →
    { channel with Messages = state.Messages @ [message] }
```

FP: чистые функции

Чистая функция — это функция, которая при одинаковых аргументах всегда возвращает одни и те же значения и не имеет видимых побочных эффектов.

- Формализм
- Декларативность
- Кеширование
- Параллельное выполнение

View function

```
module Channel.View

let view dispatch (model: ChannelData) =
  [ chatInfo dispatch model
    messageList model.Messages
    div
      [ ClassName "fs-message-input" ]
      [ input
          [ Type «text»; Placeholder "Type the message here..."
            valueOrDefault model.PostText ]
        button
          [ ClassName "btn" ]
          [ i [ ClassName "mdi mdi-send mdi-24px"
                OnClick (fun _ → dispatch PostText) ] [] ]
      ]
  ]
```

FP: статическая типизация и вывод типов

```
let f = (1 = 2)        // bool

let add a b = a + b  // int → int → int

let addFloat (a: float) b : float = a + (b : float)
// float → float → float
```

F# Compiler Services

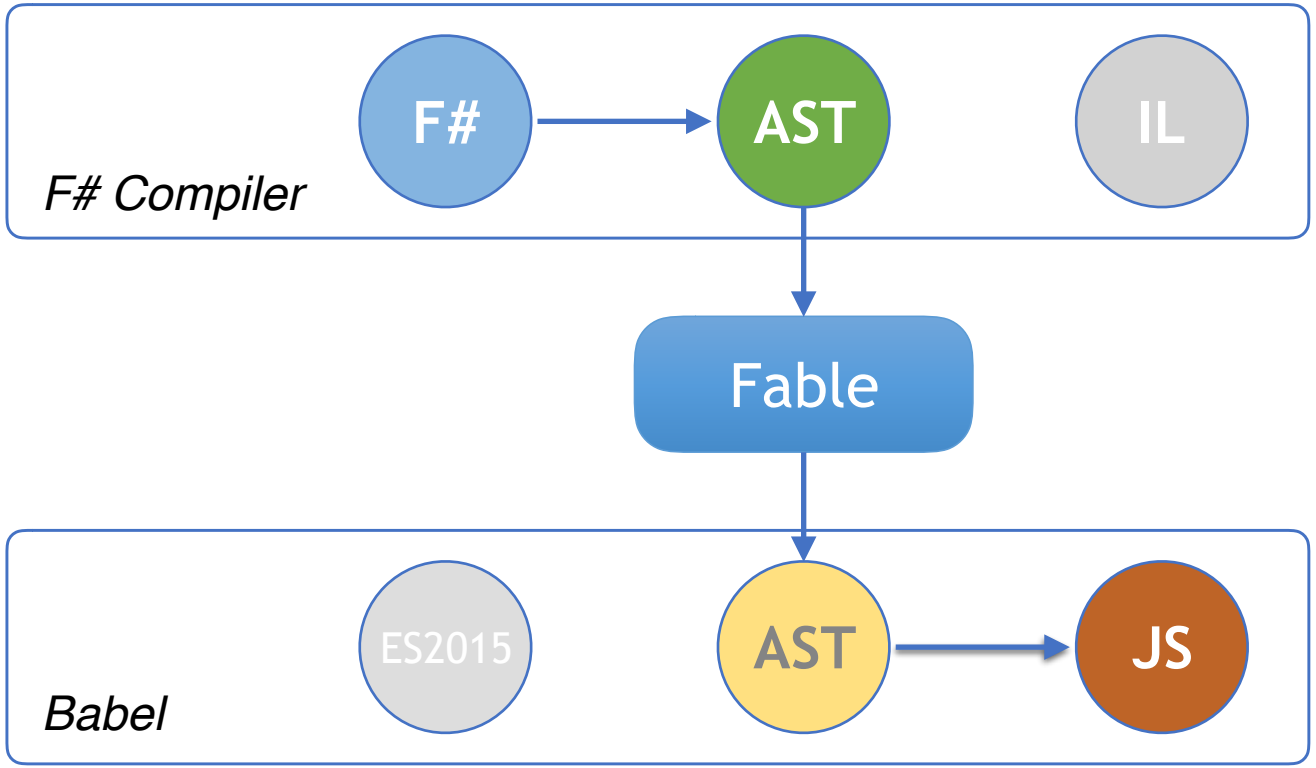
Env.fs — xake

ExecCore.fs Env.fs

```
1   module Xake.Env
2
3   /// <summary>
4   /// Gets true if script is executed under Mono framework.
5   /// </summary>
    bool
6   let isRunningOnMono =
7     System.Type.GetType ("Mono.Runtime")
8     |> function | null -> false | _ -> true
9
10  /// <summary>
11  /// Gets true if running under Unix/OSX (well, linux too).
12  /// </summary>
    bool
13  let isUnix =
14    match System.Environment.OSVersion.| with
15    | System.PlatformID.MacOSX | System
16    | _ -> false
17
18  /// <summary>
19  /// Gets true when started under Wind
20  /// </summary>
    bool
21  let isWindows = not isUnix
22
```

- Platform
- ServicePack
- Version
- VersionString property System.OperatingSyst…
- Clone
- Equals
- GetHashCode
- GetObjectData
- GetType
- ToString

Fable



JS interop

```
type IJQuery = interface end
module JQuery =
  [<Emit(«window['$']($0)")>]  let select (selector: string) : IJQuery = jsNative
  [<Emit(«window['$']($0)")>]  let ready (handler: unit → unit) : unit = jsNative
  [<Emit("$2.css($0, $1)»)>]   let css (prop: string) (value: string) (el: IJQuery) : IJQuery = jsNative
  [<Emit(«$1.addClass($0)")>]  let addClass (className: string) (el: IJQuery) : IJQuery = jsNative


JQuery.ready (fun () →
   JQuery.select "#main"
   ▷ JQuery.css "background-color" "red"
   ▷ JQuery.addClass "fancy-class"
   ▷ ignore
)
```

Tooling

- Dotnet core SDK
- VSCode + Ionide
- Webpack …
- VS
- Rider

Elmish

```
open Elmish.React
open Elmish.Debug
open Elmish.HMR

// App
Program.mkProgram init update root
|> Program.toNavigable (parseHash pageParser) urlUpdate
#if DEBUG
|> Program.withDebugger
|> Program.withHMR
#endif
|> Program.withReact "elmish-app"
|> Program.run
```


safe


mean

F# goodies

- computation expressions
- Type Providers
- REPL

Итоги

- хороший базис
- зрелый язык и компилятор
- короткий learning curve
- работает везде
- хороший инструментарий
- full-stack

Ссылки и контакты

- https://www.youtube.com/watch?v=LBekZt8QB4w
- https://github.com/kunjee17/awesome-fable
- https://github.com/OlegZee
- @olegzee в Telegram

Конец