Clean

lethevert

開発効率も
実行効率も
Lightweightな
言語です。

Q.1 どんな言語？

- Haskellとよく似た言語です

例）Quick Sort

```
sort [] = []
sort [a:aa] = sort [x \\ x <- aa | x < a]
              ++ [a]
              ++ sort [x \\ x <- aa | x >= a]
```

Q.2 誰が作ったの？

- オランダのNijmegen大学の
  Rinus Plasmeijer教授とそのチームです
- 1984年から継続的に開発
  されています

Q.3 Haskellとは何が違うの？

- 入出力にモナドを使いません
- 一意型を使います

```
dialog :: *File -> *File
dialog f #f = fwrites "your name?" f
            (s,f) = freadline f
            s = chop s
            f = fwrites ("Hello, "++s++"\n") f
            = f
```

Q.4 一意型ってどういうもの？

- 一意型を持つオブジェクトには、
  ２回以上アクセスできません

```
f = fwrites "your name?" f
```

次のアクセスのために、
毎回、新しいファイルハンドル
を返します
変数の名前は同じですが、
違うオブジェクトです

Q.5 毎回 f って書かなければダメ？

- プリプロセッサを作りました

```
dialog :: *File -> *File
dialog f = for f
             fwrites "your name?"
             s <- freadline
             s = chop s
             fwrites ("Hello, "++s++"\n")
             return
```

Q.6 実行効率はいいの？

- Language Shootoutでトップクラスです

x	language	mean	-
1.0	C **gcc**	1.47	1
1.1	D **Digital Mars**	1.56	
1.1	**Clean**	1.62	3
1.1	C++ **g++**	1.68	
1.2	Pascal **Free Pascal**	1.81	1
1.4	Eiffel **SmartEiffel**	2.02	3
1.5	**OCaml**	2.16	2
1.5	SML **MLton**	2.26	2
1.7	Ada 95 **GNAT**	2.48	2
1.8	Haskell **GHC**	2.68	

Calculate Reset

multipliers	
Full CPU Time	1
Memory Use	0
GZip Bytes	0

benchmark	weight
binary-trees	1
chameneos	1
cheap-concurrency	1

Q.7 他に証拠はないの？

- ベンチマークではありませんが・・・
- このPCでコンパイラの日本語化をしましたが実行効率にストレスは感じませんでした

マシンスペック
6年前に購入したノートパソコン
（購入当時のOSはWindows Me）
CPU Intel Celeron 600MHz
Memory: 192MB

Q.8 実行効率は重要なの？

- 処理系に付属のコレクションとは別のアルゴリズムで新しいコレクションを実装しなければいけなくなったらどうしますか？
- 全文検索用のインデックス処理を実装しなければならなくなったらどうしますか？
- 特殊な画像処理を実装しなければならなくなったらどうしますか？

効率が必要なプログラムを書くには C/C++ しか選択肢はないのですか？

Q.9 使えるプラットフォームは？

Clean 2.2 （最新バージョン）で

- IA32 Windows
- AMD64 Windows
- Macintosh (PPC)
- IA32 Linux
- AMD64 Linux
- Solaris 8 (Sparc)

Q.10 Intel Macは？

- 次のバージョンでサポート予定です

Q.11 実行方式は？

- 基本的には、コンパイル方式です
- Windows環境には、Famkaという実験的なシェルがあります

Q. 12 どんなものが作れるの？

- 例えば、こんなゲームが3000行未満で作れるらしいです

Q.13 Webアプリは作れるの？

- Clean TeamがiTaskというframeworkを開発しています
- それとは別に、最近CGIでClean Wikiを作りました

Q. 14 コードを見せてください

```
//Clean Wiki の更新処理
mapUpdatePage :: MModel
mapUpdatePage
    = model checkUser      \b =
      if b (
        model updatePage     \pg =
        model readTitleList \tl =
        view$ VHttpSeeOther pg.pi_title
      )(
        model readCGIUpdatePageInfo \pg =
        model readTitleList          \tl =
        view$ VHttpUnauthorized
              [vHeader pg.pi_title, vTitles tl
              ,vEditAgain pg]
      )
```

モナドパターンを使って、例外処理の記述を省いています

Q. 15 他のコードを見せてください

```
//Clean Wiki サイドバーの表示
viewTitles tis = body $ HtmlData $ map (\{ti_title} = title ti_title) tis
  where
    body data =
      __HTMLSTART__
      <div id="sidebar">
      <% data %>
      </div>
      __HTMLEND__

    title t = let urlTitle = mkUrlTitle t in
      __HTMLSTART__
      <a href="<% ROOT %>view/<% urlTitle %>">
      <% HtmlText t %></a><br />
      __HTMLEND__
```

プリプロセッサを使って
HTMLをプログラムに
埋め込んでいます

Q. 16 もっとコードを見せてください

//プリプロセッサのトークン分割処理

```
tokenize str = tokenize 0 where
    tokenize s #n = countSpace s
                  | n >= sz = []
                  #c = str.[n]
                  | c == 'o'
                    #e = s+1
                    | e >= sz = finally n sz
                    | str.[e] == 'x' = proc_finally countHex n
                    = proc_finally (countDecimal True) n
                  | isDigit c = proc_finally (countDecimal True) n
                  | isNormal c = proc_finally countNormal n
                  | isMember c ['\'','\"']//']
                    = proc_finally (countString c) n
                  | isSpecial c = proc_finally countSpecial n
                  | otherwise = finally n (n+1)
```

Cleanの典型的な文字列処理です
文字列は文字の配列で、インデックスアクセスします

Q. 17 開発効率はいいの？

- とてもよいです

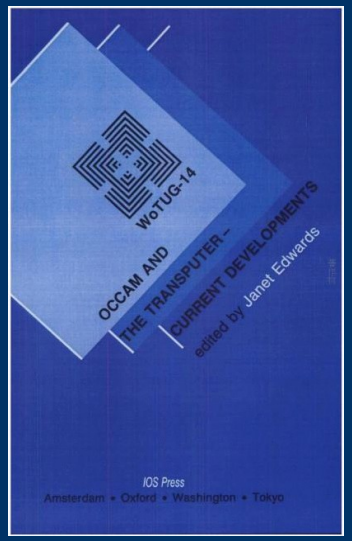
- 自動カリー化と遅延評価のコンボ
  - 共通部分を字面で見つけて置き換えるだけ
- パターンマッチ
  - 複雑な条件も簡単
- 型推論
  - 型チェッカの安心感
  - 型を書く責務からの開放

Q.18 プリプロセッサって何？

- 私が個人的に開発しているCleanの文法を拡張するツールです
- Cleanのコンパイラに組み込んで使います

Q.19 Concurrent Cleanとの関係は？

- 初期のCleanはTransputerというプラットフォームで、分散・並列処理をシンプルに記述できる特徴を持っていました
- その後、Transputerの衰退とともに、Cleanのその機能も失われました



Q.20 今のCleanの並列処理は？

- Hildeというライブラリで、
  ライブラリレベルの協調スレッドを
  提供しています
- でも、遅延評価なので、
  協調スレッドが本当に欲しくなることは
  あまりありません
- ネイティブスレッドとマルチプロセッサは
  今のところ対応していません

Q.21 他に特徴は？

- 純粋関数型
- 遅延評価
- 型推論
- 正格性解析
- 自動カリー化
- 高階型システム
- 柔軟な演算子定義
- Object I/O
  GUIライブラリ
- Generics
  総称プログラミング
- Dynamics
  動的型システム
  動的な式の永続化
- Sparkle
  定理証明器
- GAST
  自動テストシステム

第一線の研究者が開発する
最先端の技術を組み込んだ言語処理系

Q. 22 未来のユーザーに一言

- 今はユーザーが少ないので、ライブラリのラインナップに穴が多い
- フレームワークとかもまだまだ
- でも、逆にいえば・・・

今なら何をやっても
世界初の試みになる！