

君ならどう書く～OCamlでプレゼン～

IT Planning 小笠原

☆発表の要点

- OCaml の特徴を紹介
- OCaml でどう書くか？
- マルチスレッド
- 実装のポイント

☆OCamlの特徴

- 静的な型チェックと型推論
- 関数型とオブジェクト指向の融合
- 高速なコードを出力するコンパイラと豊富なライブラリ
(日本語もOK)

☆コードの例

```
Printf.printf "こんにちは %s" "世界";;
```

```
let rec fact = function  
  0 -> 1  
| n when n > 0 ->  
    n * fact (n - 1)  
| _ -> failwith "error";;
```



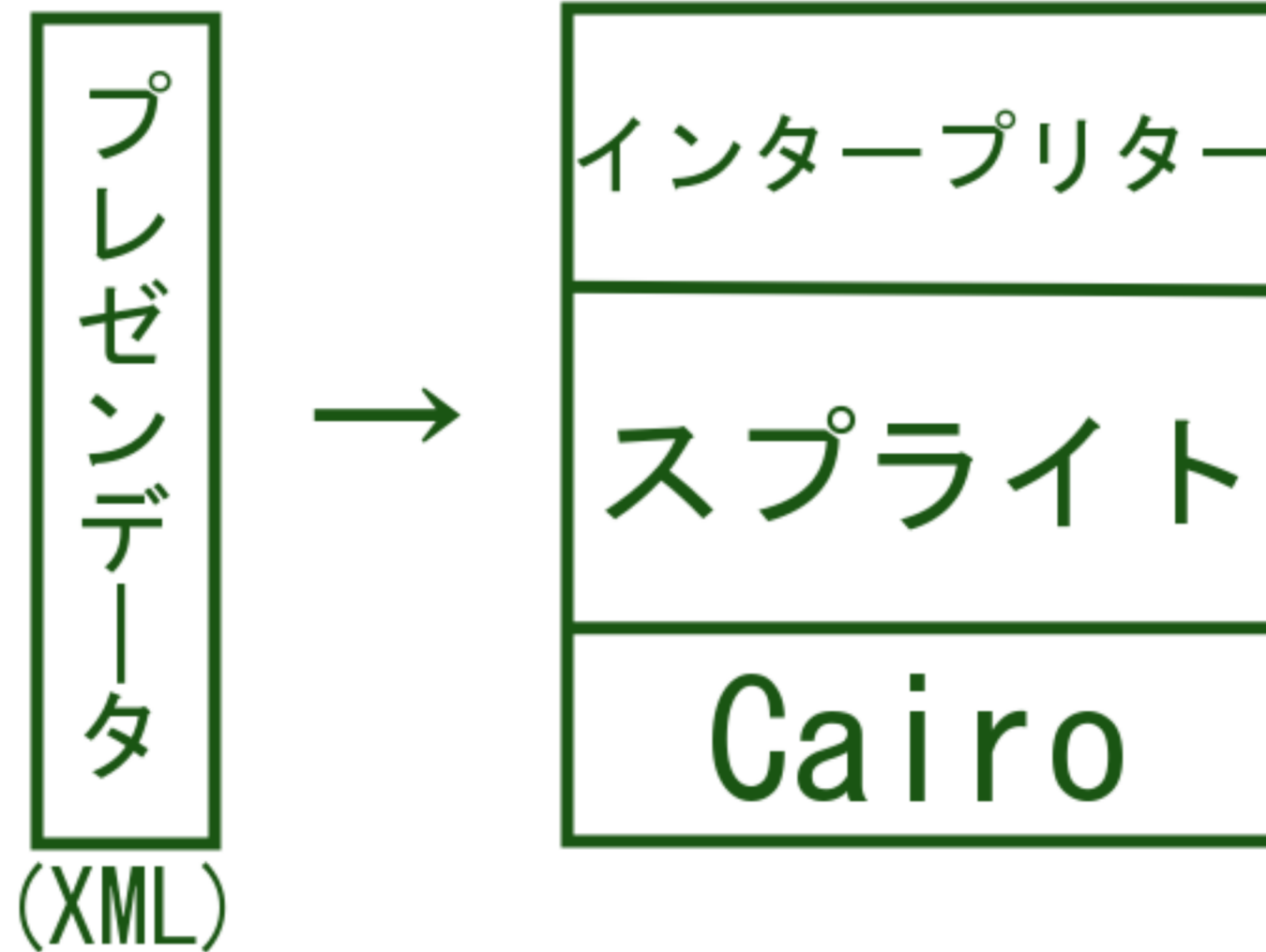
☆プレゼンソフト、どう書く？

- Lablgtk
 - 本格的
- 標準のGraphicsライブラリ
 - ちょっと力不足
- Cairo
 - ちょどいい(かも)

☆設計のもうひとつのポイント

- アニメーションを手軽に表現できるよう
マルチスレッドプログラミングにしたい
 - チャンネルスタイルの
ロックフリーなマルチスレッド
プログラム

☆設計まとめ



☆Concurrent Programming

- メモリを共有するマルチスレッドプログラミングでは、共有変数への書き込みロックが必要。
これはデッドロックを注意深く回避しなくてはならない



スレッド同士がチャンネルを通じてのみ情報を交換。
(同期、非同期含む)
これにより、ロック操作が不要なマルチスレッド
プログラミングができる

☆Concurrent Programming

open Event

```
type 'a cell = {          ロックフリーな共有変数"Cell"  
  getCh : 'a Dir_chan.in_channel;  
  putCh : 'a Dir_chan.out_channel }  
let get { getCh = getCh } = sync (Dir_chan.receive getCh)  
let put { putCh = putCh } x = sync (Dir_chan.send putCh x)  
let cell x =  
  let (getIn, getOut) = Dir_chan.new_channel () in  
  let (putIn, putOut) = Dir_chan.new_channel () in  
  let rec loop x = select [  
    wrap (Dir_chan.send getOut x) (fun () -> loop x);  
    wrap (Dir_chan.receive putIn) loop  
  ]  
in
```



☆まとめ

- 2DグラフィックレンダリングにCairoを使って、高性能レンダリング
- チャンネルスタイルのConcurrencyで手軽にマルチスレッド
- 色々使えるOCaml



Any Question?