



ErlangとElixirが突き付けるもの
あるいは性能と安全の終わりなき戦いについて
Erlang/OTPとのこの10年を振り返りつつ考える

力武 健次

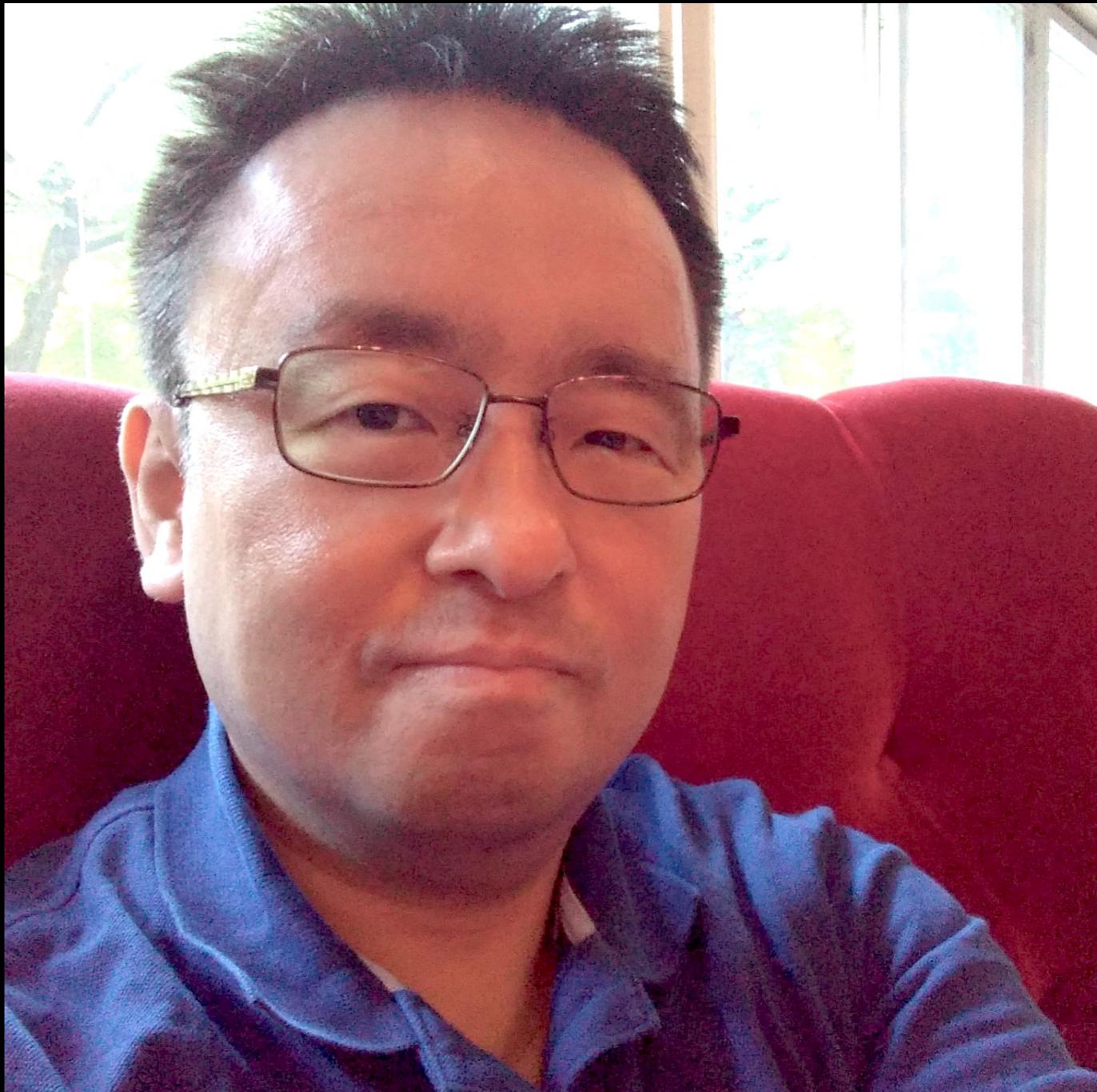
りきたけ けんじ

2018年6月16日

Erlang and Elixir Fest 2018

秋葉原コンベンションホール

@jj1wdx



[PR]

お仕事募集中です

53歳 / プログラミング歴44年

ソフトウェアエンジニア歴28年目

Erlang/OTP歴11年目

力武健次技術士事務所 所長

ペパボ研究所 客員研究員

先月まで7ヶ月C++とC#とsvnとVisual Studioの仕事してました

この10年を振り返る

Erlang/OTPとの出会い

ERLANG

2008年2月
C言語に絶望していた
東京 日本橋 丸善
手にとってみた
面白い

すぐに通販で買って読んだ



Erlang/OTP コミュニティへ
最初はFreeBSDでのうるう秒パッチから
そのうちWindowsと混ぜたりいろいろ実験
当時はセキュリティ関連の研究をしていた
SSHのRPCとか思いつく
...発表できそう

2010年3月

Erlang Factory SF Bay 2010



長い付き合いが始まる

2011年9月
東京で
Erlang Workshop
実行委員長をやり切る



実は2011年に
すごいE本
...の原典となるプレゼンが
サンフランシスコで行われた

Learn You Some Erlang for Great Good!

A Beginner's Guide

À Kenji, l'une des personnes que j'ai le plus hâte de revoir à chaque conférence

Fred Hébert



Signed March 2014
@ Erlang Factory SF, San Francisco



San Francisco

面白くなってきた

Elixirとの出会い*

* ElixirのロゴはPlataformatecで商標登録作業中のため許可なく使うことはできません。

2014年3月
Erlang Factory SF Bay 2014

José Valim



A photograph of a man with short brown hair and glasses, wearing a dark blue and green striped button-down shirt. He is standing and gesturing with his hands while speaking. The background is a bright blue wall.

Dave Thomas

衝撃的な講演

結構な物議を醸して大変だったみたいですが

Catalyze changes
変化の触媒となれ

講演内容（の要旨）

Erlangは読みにくい!
レコードが難しい!
ドキュメントがない!
なんとかしようぜ!

有言実行

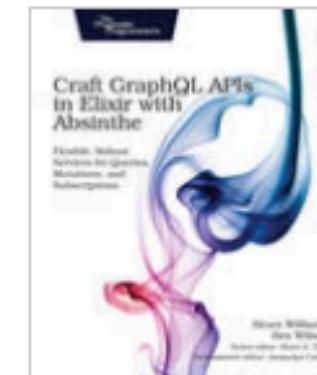
Dave Thomasは
Elixirと関連技術の
プロモーションに
尽力
自分で1冊
PragProgで7冊



Programming
Phoenix 1.4
\$24.95



Programming Elixir
1.6
\$25.95 - \$49.95



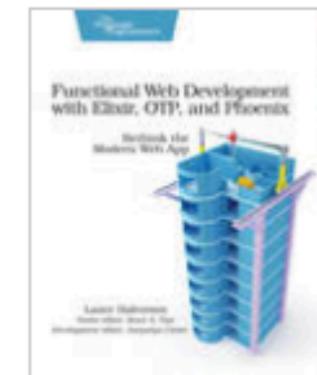
Craft GraphQL APIs
in Elixir with
Absinthe
\$25.95 - \$49.95



Adopting Elixir
\$21.95 - \$43.95



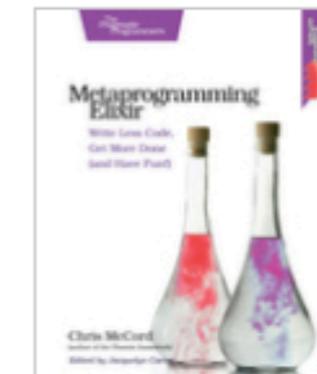
Learn Functional
Programming with
Elixir
\$21.95 - \$43.95



Functional Web
Development with
Elixir, OTP, and
Phoenix
\$24.95 - \$47.95



Build Real-Time
Web Apps with
Phoenix
\$6.75



Metaprogramming
Elixir
\$11.00 - \$27.00

2016年
プログラミング
Elixir日本語版登場

Rubyコミュニティ
の人達をちらほら
と見かけるように
なった

The Pragmatic Programmer
Programming Elixir 1.2

プログラミング Elixir

Functional
|> Concurrent
|> Pragmatic
|> Fun

Dave Thomas 著
笹田耕一・鳥井 雪 共訳



2018年 2月に大事件

erlang-questions メーリングリストにて
ことわざりに差別用語を
自分のレポジトリに付けて
しまった人がいた
すごいE本の著者が抗議

大詫嘆

收拾するのが大変だったらしいです

CODE BEAM
2018年5月31日～6月1日
スウェーデンのストックホルムにて

Code BEAM STO 2018
MÄLARSALEN

DOOR 29



会議のテーマ（の一部）
OpenErlang 20周年
BEAMコミュニティの融合
日常としてのダイバーシティ

長い前振りを終わります

ErlangとElixir

そしてその他のLFE, efene, Alpaca, clojerl, luerl, etc.

どこがおいしいの？

力武の考えるメリット

Immutability

データープロピード
参照を使わない

従来の言語

変数は再利用
参照しかコピーしない
できる限り実体を共有
↑すべてバグの元

従来の言語の原則
極力コピーしたくない
極力メモリを節約したい
安全よりも効率
代入の履歴を取れない

C++で困ったこと
変数は値？ 参照？
const付き？ なし？
コンストラクタはいつ動く？
std::unique_ptr<> あるいは std::shared_ptr<>
→複数の矛盾するセマンティクスが同時に存在

ErlangやElixirでは困りません
変数はすべて実体
一度確定した実体は不变
代入は必ず新しい実体を作る
参照を考えなくていいから楽
(ETSとかプロセス辞書とか使うと大変ですが)

ErlangやElixirの原則
必要ならどんどんコピー
メモリ節約はGCで
効率よりも安全
代入の履歴を取れる

JavaScriptの例

```
// var a = {first: 1, second: 2}
// b = a // 参照のみの共有
{ first: 1, second: 2 }
// a.second = 3
3
// b // 要素の実体は共有
{ first: 1, second: 3 }
// b == { first: 1, second: 3 }
false // なぜ?
// 右側はコンストラクタなので一致しない
```

Erlang/OTPの例

%%% 代入する実体は常に新しくつくられる

%%% 再代入はそもそもできない

```
% A1 = {1,2,3}.
```

```
{1,2,3}
```

```
% B1 = A1.
```

```
{1,2,3} % コピーができる
```

```
% A2 = setelement(3,A1,4).
```

```
{1,2,4}
```

```
% B1 := {1,2,3}.
```

```
true
```

ErlangやElixirのデメリット
とても遅い

発想の転換が必要
RubyやPHPのようには書けません

Erlangの開発者の一人
Joe Armstrongは
どう考えているのか



Joe Armstrong

@joeerl

Following



Also ... the big ideas in Erlang and elixir are processes messages and links - not gen_servers, file structure, and dependency management- the distinguishing feature of Erlang/elixir is the light-weight concurrency and error handling- all the rest is hype.

Dave Thomas @pragdave

My @empexco talk about rethinking the way we write Elixir is up. It's no more than a set of initial thoughts, but I really think there's something to them.

I was just coming off a case of pneumonia, so sorry about the sniffs.

8:31 PM - 3 Jun 2018

77 Retweets 236 Likes



Kenj



3



77



16-

236



39

Erlang/Elixirで大事なこと
軽量並行プロセス
エラーハンドリング
(と、 immutabilityだと、 力武は思っています)

軽量並行プロセス¹

処理過程はそれぞれプロセスに割り当てられて並行に処理



¹ Francesco Cesarini, Simon Thompson, "Erlang Programming", O'Reilly Media, 2009, p. 112, Figure 4-14 より力武健次によって再構成

軽量プロセスのコントロール²
作るのはspawn一発
落とすのはlinkやmonitorでOK
きれいに落とすのは大変
プロセスキューが詰まると終了
処理が遅いので注意

² 昨年のElixir Conf Japan 2017の@voluntasによる講演「なぜ Erlang/OTP を使い続けるのか」を参照

Erlangのエラーハンドリング

リンク (link)

モニター (monitor)

Parent child



linked



when B dies

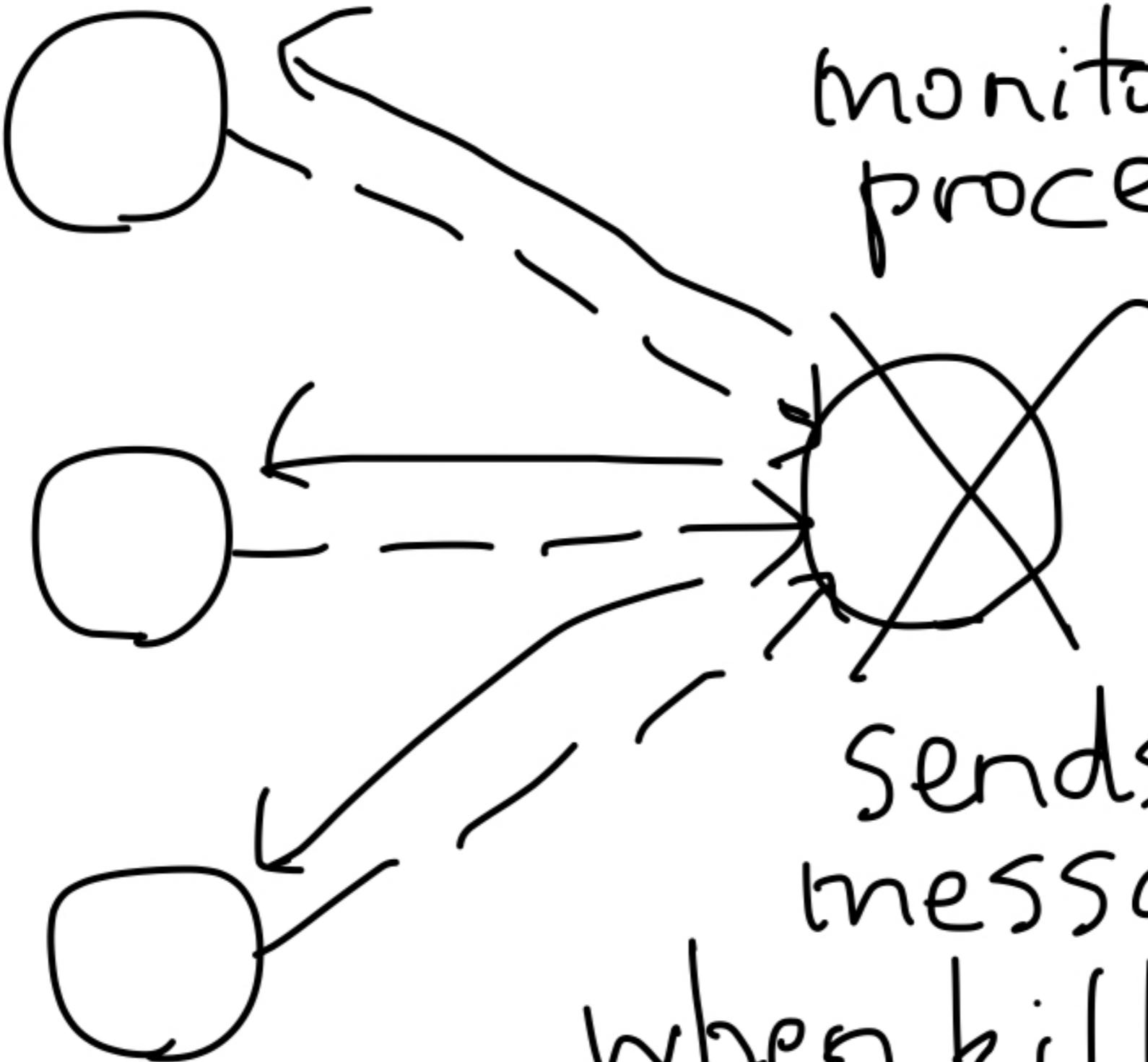


exit signal



{'EXIT', B, why }

monitoring
processes



monitored
process

sends
message
when killed

Erlang/Elixirの今後

基本理念

Lagom är bäst

ほどほどなのが一番良い

ほどほど、とは
安全は高速化に優先
手を抜かずに高速化
ケンカしない/させないコミュニティ
急がずできることを着実にやっていく

組み込み分野への応用

GRiSP / Nerves / 各種組み込みボードでの実行



今後の発展の予想
大規模(>1000ノード)クラスタ
ブロックチェーン(Aeternity)
Erlang/Elixir以外の各種言語
Gradual Type System
Language Server Protocol

Erlang/Elixir コミュニティ

他の言語にくらべると小さなコミュニティです
すごい人だけで勉強できます
Elixirのおかげで新しい人も増えました
難しい言語システムなのでレベルが高いです
飛び込んでいくと親切してくれます

**性能よりも安全を
優先できる人ならば
ErlangやElixirに
向いていると思します**

最後に昨年に引き続き
お原稿い

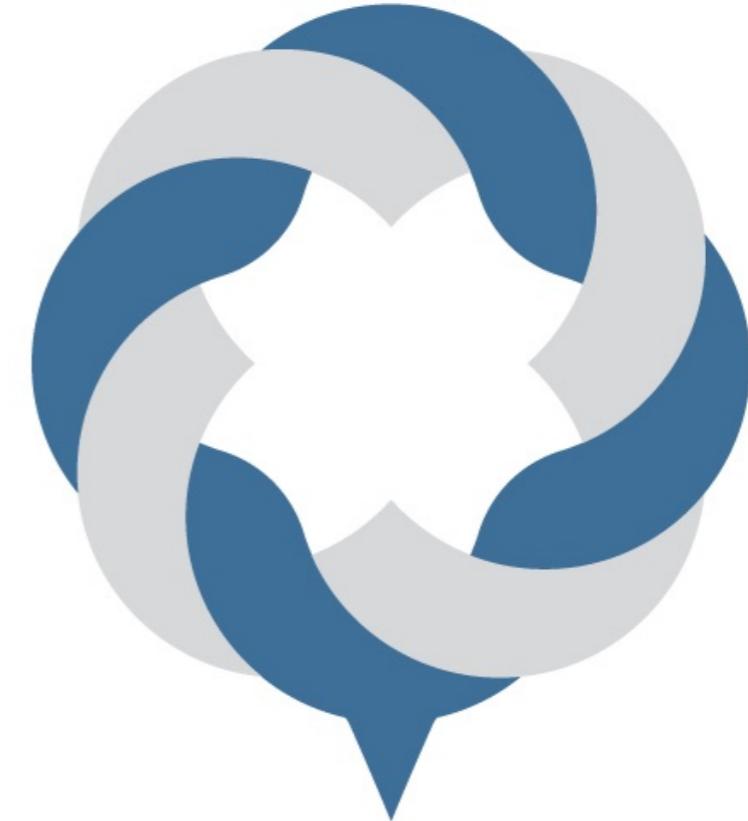
日本の人気が
ユーティに
日本の人気が
いません

日本国外で
英語の
成果発表を!

よろしく
お願ひします！

謝辞

この講演は
GMOペパボ株式会社
ペパボ研究所の
ご支援で実現しました
ありがとうございます



ペパボ研究所

Pepabo R&D Institute, GMO Pepabo, Inc.

[PR]

力武健次技術士事務所では
お仕事募集中です
ご相談歓迎致します
ぜひ懇親会でお声がけください

おしまい
ありがとうございました
ご質問をどうぞ



写真等クレジット

- Title: [Samuel Wong](#) on [Unsplash.com](#)
- José Valim: By Augie De Blieck from USA (elixirconf-47), [CC BY 2.0](#), [via Wikimedia Commons](#)
- Dave Thomas: By James Davidson (Flickr: Dave Thomas) [CC BY 2.0](#), [via Wikimedia Commons](#)
- [Pragmatic Bookshelf Elixir/OTP/Phoenix books page](#)
- リンクによる例外シグナル送信例
- モニターによる例外メッセージ送信例: 力武健次、「Erlangで学ぶ並行プログラミング第8回」、Software Design 2015年11月号、技術評論社、p. 124の図を再構成。
- スウェーデン国旗: By Jon Harald Søby and others. Public domain, [via Wikimedia Commons](#)
- ペパボ研究所ロゴ: GMOペパボ株式会社
- スライド中の書籍の表紙と中表紙についてはオーム社、No Starch Press, Pragmatic Bookshelfの書籍情報を専ら書籍紹介の目的で使用しています。
- 上記に注釈のない写真、図、絵については力武健次が撮影、編集、制作しています。