型なき言語に付ける型

Type in untyped languages

#virtual_study_group バーチャル空間勉強会#0

お前誰よ

- ・うさみけんた (@tadsan) / Zonu.EXE
 - ・GitHub/Packagistではid: zonuexe
- ・ピクシブ株式会社 pixiv運営本部
- · Emacs Lisper, PHPer, Rubyist
 - ・ Emacs PHP Modeのメンテナ引き継ぎました
 - ・好きなリスプはEmacs Lispです
- ・Qiitaに記事を書いたり変なコメントしてるよ













Friends of Emacs-PHP development

Join us!





Search or jump to...

People 5

Teams 1

Projects 0

Settings 3



Type: All ▼

Language: All ▼

Customize pinned repositories



Manage

5 >

php-runtime.el

PHP-Emacs bridge, call PHP function from Emacs

php

emacs

melpa

₫3 GPL-3.0

Updated 2 days ago

Top languages

Emacs Lisp PHP

phpactor.el

Emacs Lisp

Interface to Phpactor (an intelligent code-completion and refactoring tool for PHP)

■ Emacs Lisp ★ 8 ¥ 4 1 issue needs help Updated 3 days ago

melpa emacs major-mode

Most used topics

People



php





Invite someone



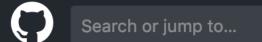




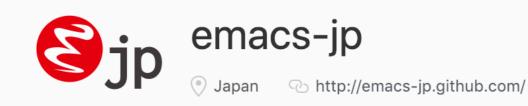
php-mode

A PHP mode for GNU Emacs









Repositories 18 Projects 0 People 37 Teams 8 Settings \$\pi\$

Find a repository...

Type: All ▼

Language: All ▼

Customize pinned repositories



reading-init.el

init.el読書会のページ

html

dotfiles

emacs

emacs-lisp

Ruby

Updated 6 days ago

helm-c-yasnippet

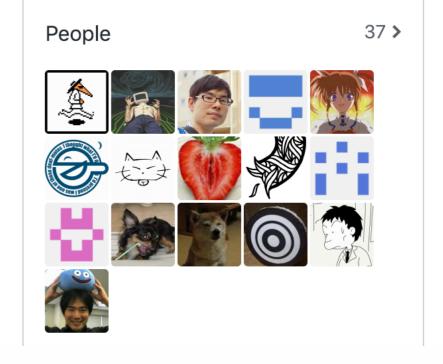
Helm source for yasnippet

● Emacs Lisp ★ 14 🖞 5

Updated on 21 Mar

replace-colorthemes









BE CREATIVE

2020.02.17 MON

本イベントは完全招待制です。



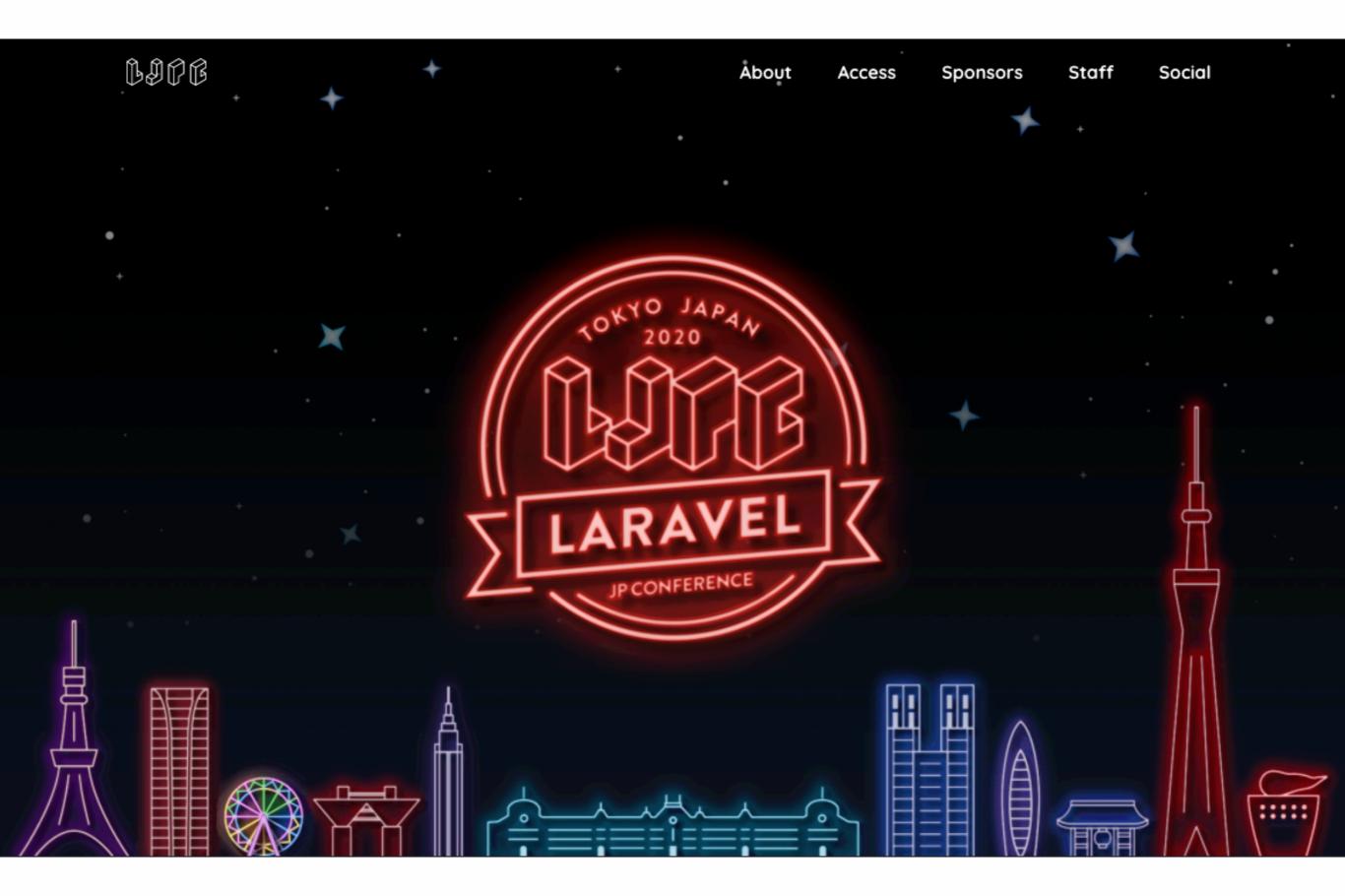
tadsan / 宇佐美健太



「ふつうのPHP」がpixivになるまで前史

2007年に開設されたイラストSNS「pixiv」は当初からフレームワークなしのPHPで構築されています。改善事例などについては過去のPHPカンファレンスなどで紹介してきましたが、今回は私が入社する2012年以前のpixivで既に実践されていた、外部であまり類を見ないテクニックをいくつか紹介いたします。

話しました



ホーム / Laravel JP Conference 2020 / トーク / 速習Composer by うさみけんた

採択 ショートセッション(15分)

速習Composer Laravel JP Conference 2020



現在のLaravelは近年の多くのPHPフレームワークと同様にComposerを基盤として構成されており、 Packagistに登録された多数のPHPライブラリと簡単に相互運用できるようになっています。

この発表ではComposerの基本機能およびLaravelとComposerの関係、そしてComposerの設定方法などをまとめて解説します。

Access

Laravel JP Conference 2020 開催中止のご案内

About

2020-02-21 18:00:00

Laravel JP Conference 2020 代表のblue-goheimochiです。

誠に残念ではございますが、表題の通りLaravel JP Conference 2020の開 催を中止させていただくことに決定いたしました。

2月18日(火)に「Laravel JP Conference 2020の新型コロナウイルス感染 症(COVID-19)への対応に関するお知らせ」をアナウンスさせていただき ました。

上記アナウンス以降引き続き、新型コロナウイルス感染症(COVID-19)の 国内の動向を慎重に注視しつつスタッフ内で議論を進めておりました。し かし、社会情勢の変化や所属企業によりイベントの参加制限が設けられ る等、参加者およびスピーカーのみなさまのご参加が困難になる可能性が あること、適切な安全確保が困難なことなどを理由に開催中止を決断さ せていただきました。

仕方がないので全然違う話をします

本目のお題

型、ついてますか?

型については結構 雑な印象で見解を 語るひとが多い

使书传电

型があるから変更に強い というひとも居れば、 型がないから変更に強い というひとも居る



型とからん

がもだ

intとboolとか でしょ

Q

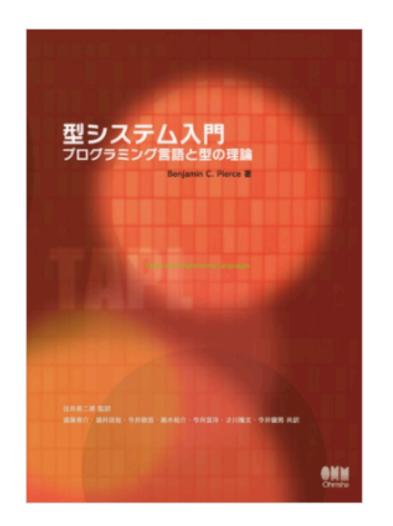
● 書籍・雑誌検索○ サイト内検索





Home > コンピュータ・一般書 > プログラミング・開発 > その他 > 型システム入門 プログラミング言語と型の理論

型システム入門 プログラミング言語と型の理論



著者 : Benjamin C. Pierce 著、住井 英二郎 監

訳、遠藤 侑介 訳、酒井 政裕 訳、今井 敬吾

訳、黒木 裕介 訳、今井 宜洋 訳、才川隆文

訳、今井 健男 訳

定価 : 7,480円 (本体6,800円+税)

判型 : B5

頁 : 528頁

ISBN: 978-4-274-06911-6

発売日 : 2013/03/26

発行元 : オーム社



詩ながまい

はじめに

1.1 計算機科学における型

現代のソフトウェア工学において、多種多様な形式手法は、システムが期待された動作の(明示 的または暗黙の)仕様に照らして正しく振る舞うことを保証する手助けになるものと評価されてい る。広範に及ぶ形式手法の一端は、Hoare 論理、代数的仕様記述言語、様相論理、表示的意味論と いった強力な枠組みである。これらは、ソフトウェアの多種多様な正しさを記述できるが、利用す るのはしばしば煩雑で、プログラマ側にかなり高度な知識を要求する。一方、それよりずっと控え めであるが、控えめであるゆえに、自動検査器をコンパイラやリンカ、あるいはプログラム解析器 に埋め込むことができ、背景となっている理論に馴染みのないプログラマでも使えるような技術が ある。このような軽量形式手法のよく知られた例に、チップ設計や通信プロトコルといった有限状 態システムのエラーを探索するツールである**モデル検査**器と呼ばれるツールがある。別の現在普及 しつつある例としては実行時**モニタリング**と呼ばれる一連の技術があり、これらはシステムの一部 が仕様通りに動いていないことを動的に検知する。しかし、今のところそれらより遥かに普及して いて、最も確立された軽量形式手法は、本書で中心的に扱う型システムである。

「型システム」とは何か、プログラミング言語の設計者や開発者たちが日常的に口にする際の意味 きなコミュニティが共有する多くの用語と同様である。しかし、次のような定義はできよう。

合いをすべて反映し、なおかつ十分に意味のある具体的な定義を与えるのが難しいのは、複数の大





-Benjamin C. Pierce 著/住井英二郎 監訳/遠藤侑介・酒井政裕・今井敬吾・黒木裕介・ 今并宜洋・才川隆文・今井健男 訳 『型システム入門 プログラミング言語と型の理論』 (オーム社、978-4-274-06911-6、2013年) サンプル



型システムとは、プログラムの各部分を、それが計算する値の種類に沿って分類することに より、プログラムがある種の振る舞いを起こさないことを保証する、計算量的に扱いやすい 構文的手法である。

幾つか補足すべき点がある。まず、この定義は型システムを、プログラムに関する推論のためのツールだとみなしている。このような用語の使い方は、本書が、プログラミング言語で用いられる型システムに関するものであるという方向性を反映している。より一般的には、型システム(あるいは型理論)という用語は、論理学や数学、哲学の、もっと広範な研究分野に言及するものである。この意味での型システムは、1900年代初頭、Russellのパラドックス [424] など、数学の基盤を揺るがした論理的パラドックスを回避する手段として、最初に形式化された。20世紀を通じて、型は論理学、特に証明論(Gandy [167] や Hindley [209] を見よ)において標準的に用いられる道具となり、また哲学や科学の言語として浸透していった。この分野における主要な成果として、原点である Russell の分岐階型理論 [493]、Ramsey による単純階型理論 [394] (これは Church の単純型付きラムダ計算 [100] の基礎となった)、Martin-Löf の構成的型理論 [299; 301]、Berardi、Terlouw、Barendregt による純粋型システム [43; 458; 40] などがある。



-Benjamin C. Pierce 著/住井英二郎 監訳/遠藤侑介・酒井政裕・今井敬吾・黒木裕介・今井宜洋・才川隆文・今井健男 訳 『型システム入門 プログラミング言語と型の理論』 (オーム社、978-4-274-06911-6、2013年) サンプル

(一章はサンプルとしてPDFが無料でダウンロードできます)

正しい、完璧に正しいのだが

最初から用語多すぎ

説明の抽象度が 高すぎて現実との ギャップ大きすぎ

この1章の内容を解さほぐして説明するだけで何時間も話せそう

そのほかの型とかプロ グラミング言語論の本 を読むと数式みたいな のが出てきてりかりかり

型について ちょっと知りたい だけなのに

世の中の大型人生

型を使っているが 思索の対象として 向き合ってないひと

特定の言語の型とだけのき合ってる人

抽象概念の型と向き合ってるひと

いろんな背景の人が居る

動的言語を使っていると 静的型付き言語利用者か ら冷ややかに見られがち

や一い型なし型なし

型はあるんだ! 実行時にあるんだ!

そういうことをまじめに 主張すると憐れみを湛え た目で見られはじめる

真の型なし言語も 複数あるが今回は 割愛しました

カかりたく ないですか

このトークに結論はない

実際の言語の型からいきましょう

よくわからんけど型がだるい

```
#include <stdio.h>
long add(long a, long b) {
    return a + b;
int main(void) {
    unsigned int x = 1, y = 2;
    printf("%d + %d = %d\n", x, y, add(x, y));
```

私たちFラン情報科学生は 情弱なのでまともな説明 もないカビの生えた教科 書でC言語を叩き込まれる

私の入門者時代のC 言理解は曖昧 なので曖昧なままで 次の話に進めます

さておき

短くなってべんり

```
// JavaScript

function add(a, b) {
    return a + b;
}

const x = 1, y = 2;

console.log(`${x} + ${y} + ${add(x, y)}`);
```

もっと短くなってべんり

```
# Ruby

def add(a, b)
    a + b
end

x = 1; y = 2

puts "#{x} + #{y} + #{add x, y}"
```

そういう俺たちが 動的言語に触れると こうなる

型とか要らんやろw

そうだな

見ればわかるだろ

```
# Ruby
# to_sオブジェクトが生えてたら何でもいい
def show(obj)
 puts obj.to_s
end
```

常識的に考えて

```
# Ruby
# 足し算できるものならなんでもいいし
# ジェネリクスとかなくてもいいw
def add(a, b)
 a + b
end
def show(obj)
 puts obj.to s
end
```

引数が少ないうちはな…

```
# Ruby
user = User.findBy(id: params[:user_id])
```

引数が増えてくると

```
# Ruby
obj = palpunte(
  user id,
  ????,
  ????,
  ????,
p obj
```

実装を直接見に行く機会が増える

引数が増えてくると

```
# Ruby

# 実装を見ないと期待するオプションに何を渡せばいいのか

def palpunte (id, foo, bar, buz)
    p [id, foo, bar, buz]

end

# というか見てもよくわからない
```

素朴に書くと引数がわからん

```
# Ruby
# 実際にはキーワード引数だけで表現できる
def palpunte (id: nil, **options)
   raise TypeError if id.nil?
   foo | | = "default foo"
  bar | = "default bar"
  buz | = "default buz"
  p [id, foo, bar, bar]]
end
```

だんだん型チェックが増えてく

```
raise TypeError unless foo.instance_of?(String)
raise TypeError unless bar.instance_of?(String)
raise TypeError unless buz.instance of?(String)
```

Ruby

いわゆる動的言語にも型が付く流れ

Flowtype TypeScript

PEP484 mypy

Ruby3 rbi Sorbet Steep

PHP型宣言 PHPDoc PhpStorm,Phan

身体は型を対象の数の

型が付いていない ということは どういうことか

可能性は無限大

```
<?php
/**
  @param mixed $a 可能性は無限大
  @param mixed $b
 * @return mixed
function add($a, $b)
   return $a + $b;
```

仮引数宣言がない言語

無限大じゃ困るんだよ

とはいえ+できる 値という前提がある ので案外困らない

ほんとにう

PHPの演算子の暗黙的型変換

```
<?php
error reporting(0);
var dump(add(1, 2)); // => int(3) 想定通り
var dump(add(1.0, 2.0)); // => float(3.0) 許容範囲
var dump(add('1', '2')); // => int(3) 許容範囲
var dump(add('', '')); // => int(0) ?????
var dump(add(1, 2, 3)); // => int(3) ?????
var_dump(add([], [])); // => [] えっ?????
var dump(add(PHP INT MAX, 1));
// => float(9.2233720368548E+18)
```

実行時に型を調べよう

いわゆる動的言語は 実行時に型があるの でそれを検査できる

JavaScriptでは typeof 演算子で 型名文字列がとれる

Python type() Ruby Object#class

PHPではis_int(), is_string()などの関数 (gettypeは使いにくい)

可能性を絞り込もう

```
<?php
/**
   @param mixed $a 可能性は無限大
   @param mixed $b 可能性は無限大
   @return mixed
function add($a, $b)
{
   if (!is_int($a)) {
       throw new TypeError('Error');
   if (!is_int($b)) {
       throw new TypeError('Error');
   // 実装するときに int 以外の値が渡ってくる可能性を考えなくて良い
   return $a + $b;
```

実行時に変な値に対して処理が続行する可能性は避けられる

例外で処理が停止するので

PHPとかいう 静的型付き プログラミング言語

静的型付きPHP

```
<?php declare(strict_types=1);

// このメソッドを実装するときに int 以外の値が

// 渡ってくる可能性は考えなくてよい

function add(int $a, int $b): int
{
   return $a + $b;
}</pre>
```

静的 = 実行しなくても 型が決まってる状態

静的型付き Statically typed 既に型が決定してる

動的型付<u>け</u> Dynamic typing 実行時に型を付ける

よくある誤解 静的型=コンパイラ 動的型=インタプリタ

ただしプログラム実行時 のデータの表現方法が そういう関係になりがち

信語は静的型付き

必ずしも排他の関係ではない

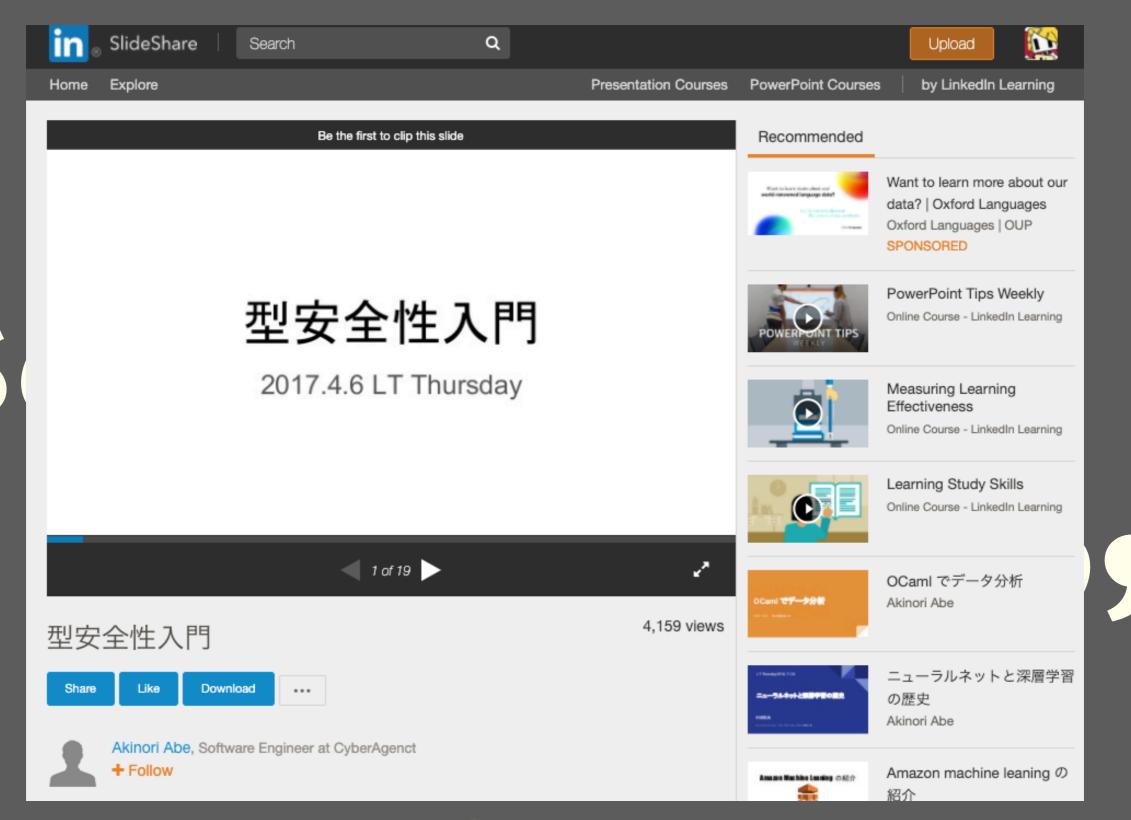
お使いの言語ランタイム(ruby, python, phpなど)のC言語実装がどうなっているか、変数がC言語の型でどう表現されてるか読んでみるとおもしろいと思います

PHPは実行時に 型をチェックする

ただし関数と プロパティに静的に 型を付けられる

ただし静的に型を付けて も実行時にクラッシュす る可能性があります

型で守られてるとはどういう状態が



-Akinori Abe 『型安全性入門』(Apr 6, 2017)

Be the first to clip this slide

健全性と完全性

型安全性 (type safety), 健全性 (soundness)

- 「型が付くならば、ある種のバグは絶対にない」(偽陰性の排除)
- 「型エラーならば、バグがあるかもしれない」(偽陽性の許容)
 - o バグがないのに、間違って型エラー出しちゃうかも
 - バグがあるのに型検査を通過させるリスクが大きい

完全性 (completeness)

- 「型エラーがあれば、バグが絶対にある」(偽陽性の排除)
- 健全かつ完全な型検査は難しい
 - チューリング完全な言語では決定不能問題





4,159 views

型安全性入門

-Akinori Abe 『型安全性入門』(Apr 6, 2017)



関数を呼び出す側が間 違って呼び出すことを PHPだけでは防げない

動的言語で完全かつ 健全にするのは 不可能

偽陽性(動かないのに動き そうなフリをしている)こ とを最小にしていくこと を目指す

コンパイラの代りに 静的型チェッカーを使 おうということになる

PHPでも複数の 型チェッカーが 揃っている

PhpStorm Phan, Psalm PHPStan

PHPのソースコードを解析しても判別できない型は注釈(annotate)してあげる必要がある

人問力 理を書く

人間が型を間違って つけると破綻するの で実際簡単ではない

連想配列(Hash, dict) を構造体のつもりで 多用されると静的解析 を簡単に破滅できる

と思うでしょ? PHPDocでなら 型をつけられます

PHPに型を付けたり CII二載せる話は FANBOXに記事やス ライドを掲載してます

PIXIV FANBOX

```
This buffer is for text that is not saved, and for Lisp evaluation. To create a file, visit it with <open> and enter text in its buffer.
                                                                                                                                                                                                                                                                             init.el --- zonuexe's .emacs -*- coding: utf-8 ; lexical-binding: t -*
  namespace Teto;
                                                                                                                                                                                                                                                                           Filename: init.el
Description: zonuexe's .emacs
Package-Requires: ((emacs "24.4"))
Author: USAMI Kenta <tadsan@zonu.me>
Created: 2014-11-01
Modified: 2016-06-28
Version: 10.10
Keywords: internal, local
Human-Keywords: Emacs Initialization
Namespace: my/
  * Functional toolbox
  * * package
  # gauthor USAMI Kenta <tadsan@zonu.me>
# @copyright 2015 USAMI Kenta
# @license http://www.apache.org/licenses/LICENSE-2.0
 final class Functools
                                                                                                                                                                                                                                                                             URL: https://github.com/zonuexe/dotfiles/blob/master/.emacs.d/init.el
      private function __construct() {}
                                                                                                                                                                                                          [1] (Lisp Interaction Presental
                                                                                                                                                                                        (1,\theta)
                                                                                                                                                                                                                                                                          ; This file is NOT part of GNU Emacs.
                                                                                                                                                                                                                                                                           This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3, or (at your option) any later version.
       * Partial application
                                                                                                                                                New version is available (current: v0.8.18, latest: v0.9.3)
                                                                                                                                                >>> PHP_VERSION
        * * * * param callable $callback
        * Aparam mixed[] $arguments
* Aparam int $pos
                                                                                                                                                >>> []
                                                                                                                                                                                                                                                                           This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
        * preturn \Teto\Functools\PartialCallable
      public static function partial(callable $callback, array $arguments = [], $pos = null)
           return new Functools\PartialCallable($callback, $arguments, $pos);
                                                                                                                                                                                                                                                                          ; You should have received a copy of the GNU General Public License; along with GNU Emacs; see the file COPYING. If not, write to the; Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor,; Boston, MA 82118-1381, USA.
                                                                                                                                                                                        (5,4)
                                                                                                                                                                                                          [1] (PsySH:run Presentation -2
                                                                                                                                                 *** Welcome to IELM ***
ELISP> (emacs-version)
                                                                                                                                                                              ** Type (describe-mode) for help.
        * Returns an indication of the number of arguments accepted by a callable.
                                                                                                                                                 'GNU Emacs 26.1 (build 1, x86_64-apple-darwin17.5.0, NS appkit-1561.40 Version
                                                                                                                                                 9.13.4 (Build 17E199))\n of 2018-04-20"
        * * * * param callable $callback
                                                                                                                                                ELISP>
        * greturn int
                                                                                                                                                                                                                                                                            Nobiscum Sexp. - S-expression is with us.
      public static function arity (callable $callback)
           if (is_array($callback) ||
   (is_string($callback) && strpos($callback, '::') !== false)) {
                                                                                                                                                                                                                                                                          setq-default gc-cons-percentage 0.5)
                                                                          (34,7) [1] (PHP//lw Presentation U:**- *ielm*
                                                                                                                                                                                                          [1] (IELM:run on *ielm* Present U:--- init.el
                                                                                                                                                                                       (4.7)
                                                                                                                                                                                                                                                                                                                 (34,45) [1] (Emacs-Lisp Presentation
M-x com composer | com
                                                            poser-install | composer-find-json-file | composer-run-vendor-bin-command | composer-viem-lock-file | comint-run | comment-dwim | composer-dump-autoload | compile | ...}
```

プロフィール

投稿

tadsan

ショップ

型について学ぶにはどうすればいい

Q

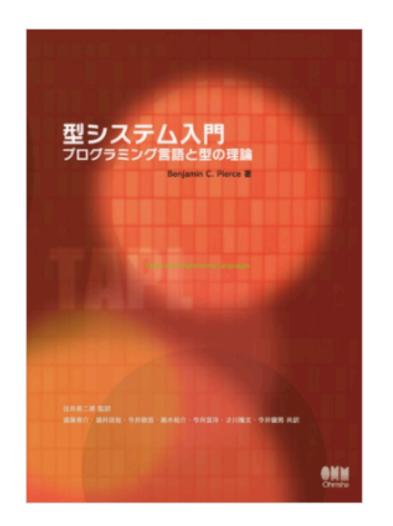
● 書籍・雑誌検索○ サイト内検索





Home > コンピュータ・一般書 > プログラミング・開発 > その他 > 型システム入門 プログラミング言語と型の理論

型システム入門 プログラミング言語と型の理論



著者 : Benjamin C. Pierce 著、住井 英二郎 監

訳、遠藤 侑介 訳、酒井 政裕 訳、今井 敬吾

訳、黒木 裕介 訳、今井 宜洋 訳、才川隆文

訳、今井 健男 訳

定価 : 7,480円 (本体6,800円+税)

判型 : B5

頁 : 528頁

ISBN: 978-4-274-06911-6

発売日 : 2013/03/26

発行元 : オーム社

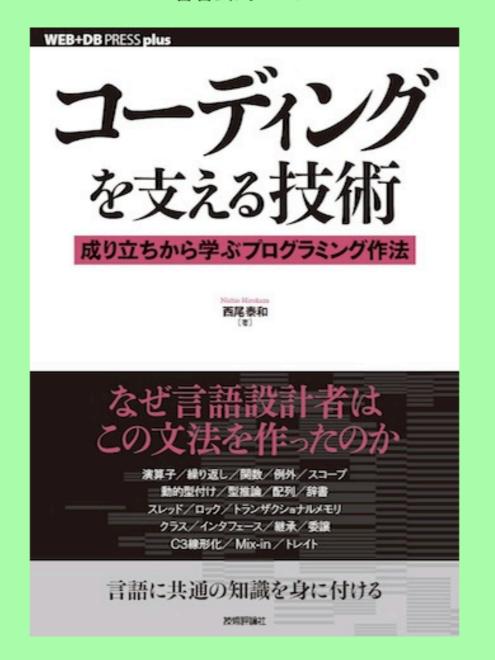


ハードルが高い (そもそも値段が…)

一章のPDFは無料なので 読んでほしいが難しいの で理解できなくてもいい

コーディングを支える技術

著者公式ページ



世の中にはたくさんのプログラミング言語があります。そしてプログラミングに関する概念も、関数、型、スコープ、クラス、継承など、さまざまなものがあります。多くの言語で共通して使われる概念もあれば、一部の言語でしか使われない概念もあります。これらの概念は、なぜ生まれたのでしょうか。本書のテーマは、その「なぜ」を理解することです。

そのために本書では、言語設計者の視点に立ち、複数の言語を比較し、そして言語がどう変化してきたのかを解説します。いろいろな概念が「なぜ」生まれたのかを理解することで、なぜ使うべきか、いつ使うべきか、どう使うべきかを判断できるようになるでしょう。そして、今後生まれてくる新しい概念も、よりいっそう理解しやすくなることでしょう。

型のみならずプログラミング言語についてのさま さまな概念を学べる

まとめ

仕事で使ってる言語を好 きになるために目を逸ら さず向き合っていこう