現役エンジニアで
いるために

和田 卓人

Sep 17, 2020 @NTTコミュニケーションズ

和田卓人

Hatena

t-wada

t_wada

twada

よろしくお願いします

ソフトウエア開発力のさらなる強化へ
及川卓也氏・吉羽龍太郎氏・和田卓人氏がNTT Comの社外技術顧問に就任

カテゴリ Business/Technology

イノベーション　キャリアアップ　社内環境

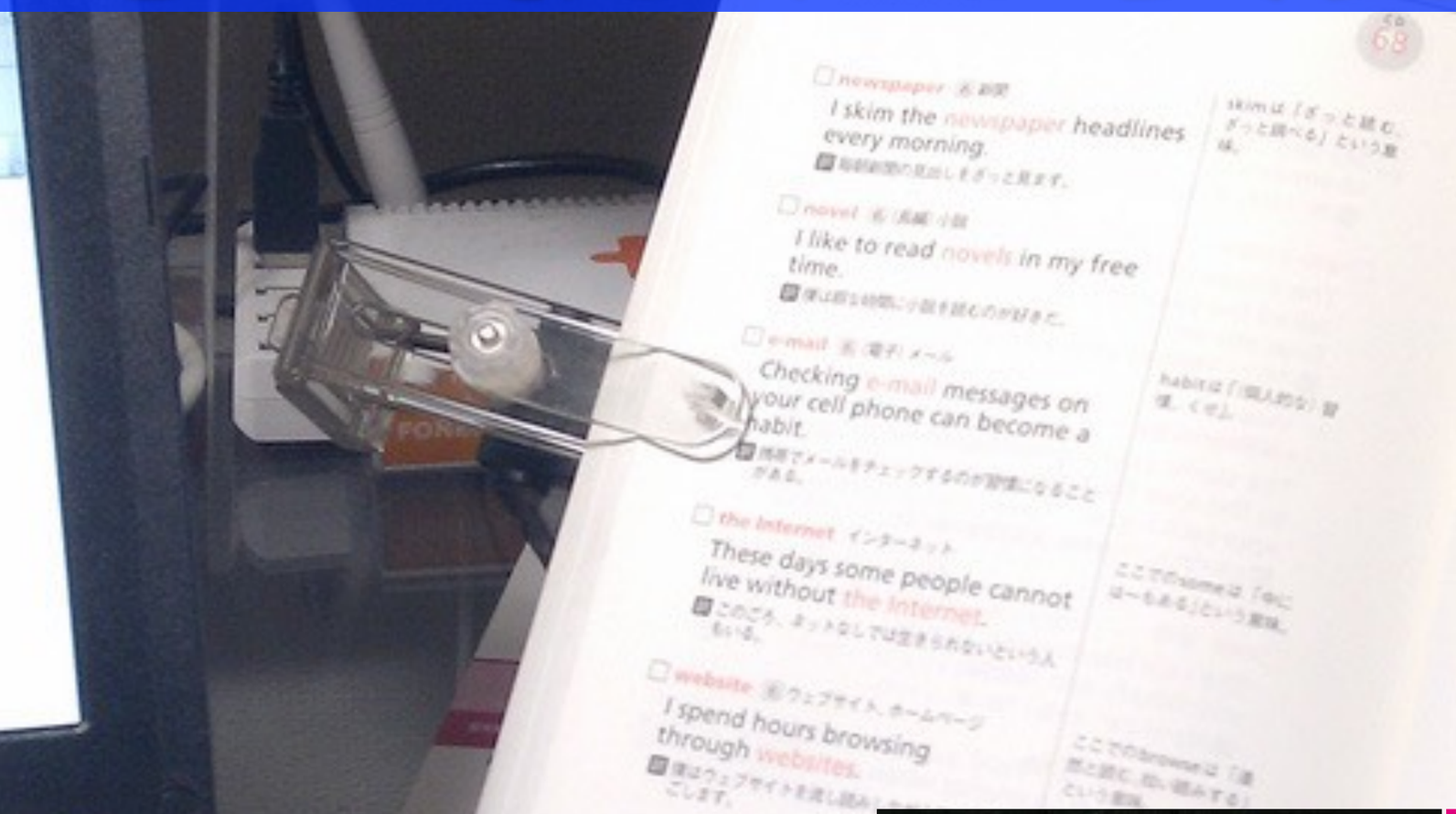
#twada_helps

Agenda

- 👉 [前回] 学び方を学ぶ
- 現役エンジニアでいるために

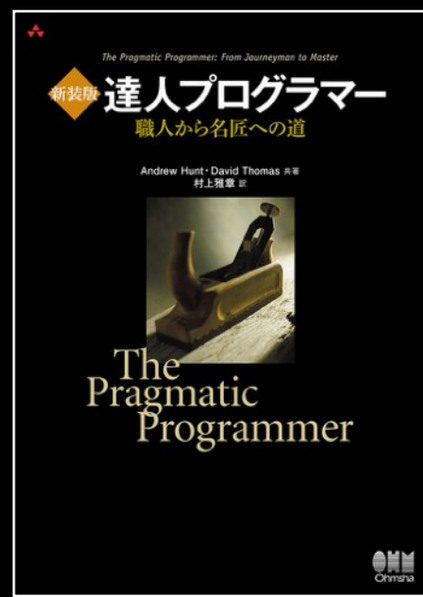
学び続ける姿勢


X200s
Insert
Home
PgUp
Delete
End
PgDn
Back space
Enter
Shift


68
newspaper 名 新聞
I skim the newspaper headlines every morning.
novel 名 長編小説
I like to read novels in my free time.
e-mail 名 電子メール
Checking e-mail messages on your cell phone can become a habit.
the Internet インターネット
These days some people cannot live without the Internet.
website 名 ウェブサイト、ホームページ
I spend hours browsing through websites.
Dialogue
Tom: Hi! I'm back!
Lisa : Oh, hi! I just came home, t
Tom: Did you check the answerin
Lisa : Yeah, but there were no ne
Tom: Turn on the TV, will you? I
weather forecast for tomorro
Lisa : There are no news programs
don't we watch the video we
until then?
Tom: Yeah, but I will check my e-ma
tomorrow's weather on the int
Lisa : OK. I'll watch TV for a while.
finished.


97
Collective Wisdom from the Experts
プログラマが知るべき97のこと
97 Things Every Programmer Should Know
O'REILLY
オライリー・ジャパン
Kevlin Henney 編
和田 卓人 監修
夏目 大 訳

“常にあなたの知識ポートフォリオに投資すること”



技術を学ぶのでは
なく、技術の
学び方を学ぶ

前回のまとめ

四半期毎に技術書を読む

手を動かして学ぶ

毎年少なくとも1つの言語を学習する

身の回りをプログラミング対象にする

アウトプットを行う

か？ (edited)

T **twada**
Sep 10, 2020
1

自分用に書いたコードはほぼ全て GitHub の private repository に管理しています。意識はあまり変わりませんが、公開する際にドキュメントをきちんと書きます。 > 自分用で書いたコードの管理はどうされていますでしょうか?また、プライベート用のコードを書く際に意識することなどはありますでしょうか？

T **twada**
Sep 10, 2020
1

いい質問で、来週の講演にも出てきます。やりたいのはスキルの相対化で、できれば社外のコミュニティに出て自分のスキルの相対化を棚卸しを行いたいです。社内でも同様のことはできると思うので、まずは社内の技術コミュニティに顔を出して他の人を観察しましょう > どのように、自分の知識ポートフォリオの競争力を測ればいいのでしょうか

T **twada**
Sep 10, 2020
1

自分用のメモを markdown で書き、 GitHub の個人リポジトリで管理しています。そのなかから要約したものをツイートしたりしています。メモを検索可能にするところが大事です。 > Webや書籍などから学んだ知識はどのように整理（どこかにメモとして残すなど）されていますか？

T **twada**
Sep 10, 2020
1

適切な抽象度で図示し、人に説明する技術を磨くことだと思います。コンポーネントやネットワークの図だけでなく、ロジックツリー、3C、SWOT分析といった説明のスキルです > システムエンジニアとして今後生き残るために、例えばSWでいう「コードを書く」とか「言語を学ぶ」といったアクションに該当するような取り組みはありますか？

Ask

Agenda

学び方を学ぶ

👉 現役エンジニアでいるために

👉 毎日コードを書く

年下から学ぶ

過去から未来を見る

人のつくる渦を見る

大事なことに集中する

1. 毎日コードを

書く

あの John Resig でもうまくいかないこと

- jQuery 作者 John Resig は週末に自分のプロダクト開発を頑張ろうとしたが、失敗。
  - 平日と同じ馬力では書けない
  - 全ての週末が空いているわけではない
  - 一週間 (あるいは二週間) は長い。コードを忘れてしまう
- そこで John Resig が行ったことは……

John Resig
Contact, Subscribe

Write Code Every Day

Last fall, work on my coding side projects came to a head: I wasn't making adequate progress and I couldn't find a way to get more done without sacrificing my ability to do effective work at Khan Academy.

There were a few major problems with how I was working on my side projects. I was primarily working on them during the weekends and sometimes in the evenings during the week. This is a strategy that does not work well for me, as it turns out. I was burdened with an incredible amount of stress to try and complete as much high quality work as possible during the weekend (and if I was unable to it felt like a failure). This was a problem as there's no guarantee that every weekend will be free – nor that I'll want to program all day for two days (removing any chance of relaxation or doing anything fun).

There's also the issue that a week between working on some code is a long time, it's very easy to forget what you were working on or what you left off on (even if you keep notes). Not to mention if you miss a weekend you end up with a two week gap as a result. That massive multi-week context switch can be deadly (I've had many side projects die due to attention starvation like that).

Inspired by the incredible work that Jennifer Dewalt completed last year, in which she taught herself programming by building 180 web sites in 180 days, I felt compelled to try a similar tactic: working on my side projects every single day.

四つのルール

1.毎日コードを書くこと。ブログ、ドキュメント、その他はコードを書いたらやってよい。

2.意味のあるコードを書くこと。インデントやフォーマットの修正、可能ならばリファクタリングもコード書きにはカウントしない。

3.深夜 24 時前に終わらせること。

4.書いたコードを github で全て OSS にすること。

http://ejohn.org/blog/write-code-every-day/

(当時の) @jeresig の github profile

John Resig
jeresig

Block or report user

@Khan
Brooklyn, NY
jeresig@gmail.com
http://ejohn.org/

Organizations

Overview | Repositories 91 | Stars 377 | Followers 13.3k | Following 29

Pinned repositories

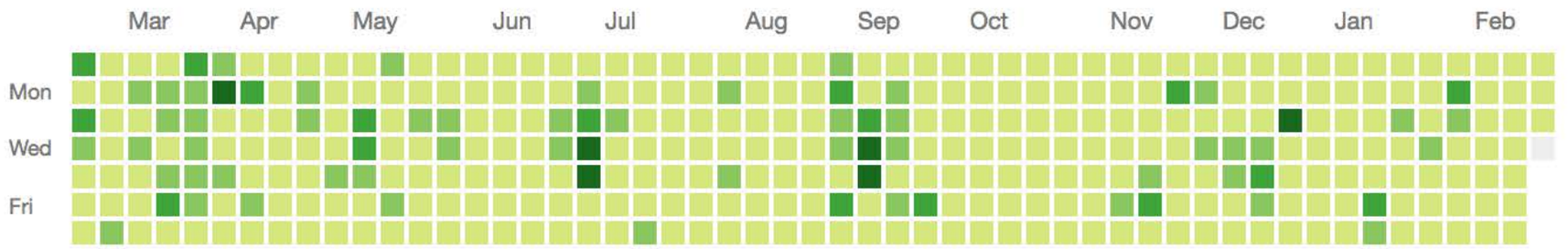
mongaku/mongaku
An image similarity search engine and database.
JavaScript ★ 18 2 1

node-stream-playground
Explore Node.js streams with an interactive playground.
JavaScript ★ 407 34

node-romaji-name
Normalize and fix common issues with Romaji-based Japanese names.
JavaScript ★ 18 2

node-yearrange
Node module for converting year range strings into usable dates.
JavaScript ★ 18 1

910 contributions in the last year



Learn how we count contributions.
Less More

https://github.com/jeresig

John Resig に起こった変化 (1)

- 必要最小限のコードへの集中: 一日30分〜1時間程度で意味のあるコードを書くことが強いられる (休日にはもっとかけられる)

- プログラミングの習慣化: github に草を生やすのが目的ではない。自分で自分自身のために生活習慣を変えるのが大事

- 不安との戦い: 以前は「十分に」進んでいるか、「十分に」完成しているか、不安があった。毎日コードを書いてみて、進んでいるという実感は、実際の進捗と同じくらい重要だという気づきを得た

John Resig に起こった変化 (2)

- 週末の過ごしかた: 以前は開発の全てを週末に賭けて失敗していたが、いまや週末はそれほど重要でなくなり、リアルライフを充実できるようになった

- バックグラウンド処理: 散歩中、シャワー中、常にコードのことをバックグラウンドで考えるようになり、良いアイデアが浮かぶようになった

- コンテクストスイッチ: 以前は週に一回の開発だったのでコンテクストスイッチのコストがあったが、いまは毎日なのでそれがない

John Resig に起こった変化 (3)

- ワークライフバランス: 仕事/生活/自分のプロジェクトのバランスの取り方が分かったのが最大の収穫だった。毎日やるということは、バランスを取るということ

- まわりからの理解: 「毎日コードを書く」という習慣を公言したことで、パートナーからの理解も得られるようになった

- どれだけコードを書いたか: この習慣を続けると書くコードやアウトプットは自分でも覚えられないくらいの量になり、充実感を得られる

「いま、小さなことを多く積み上げることが、とんでもないところへ行くただひとつの道なんだなというふうに感じています」

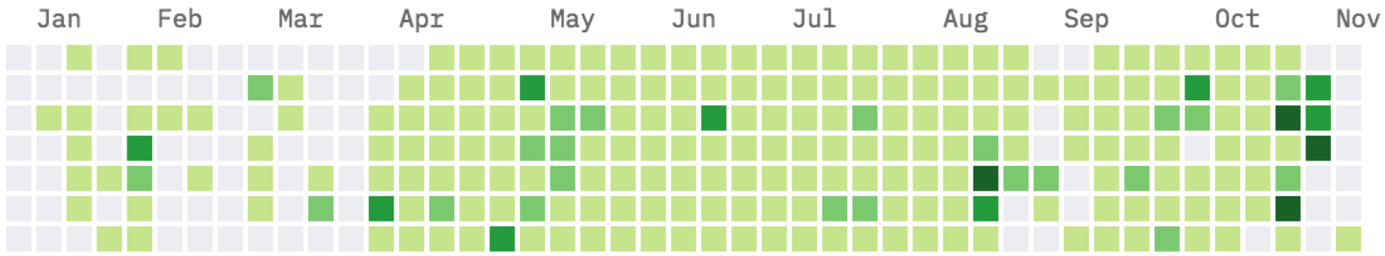
イチローが2004年にNBA年間最多安打を更新したときの言葉

私も結構続けました

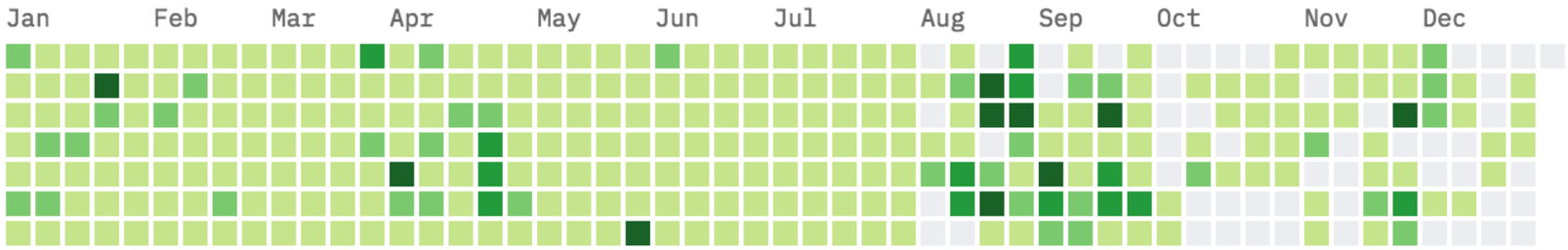
@twada on GitHub

Less More

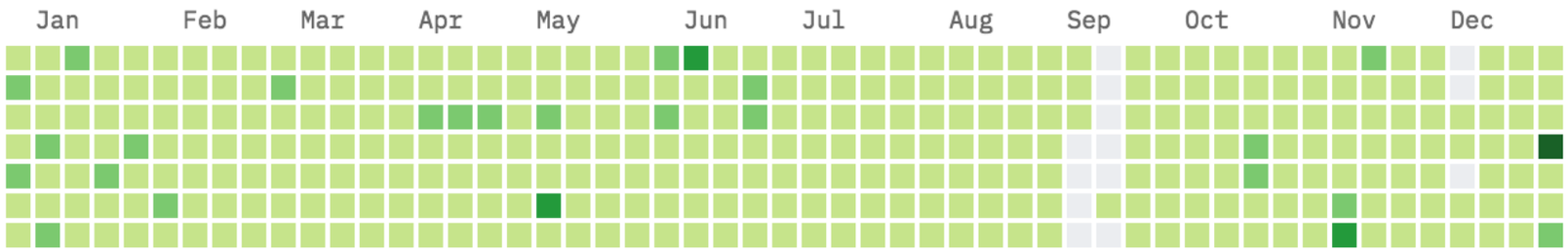
2018: 1,290 Contributions (so far)



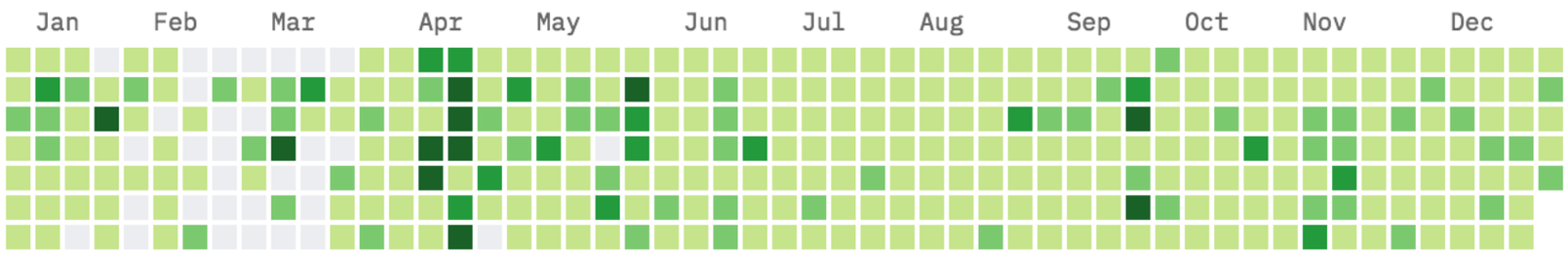
2017: 1,583 Contributions



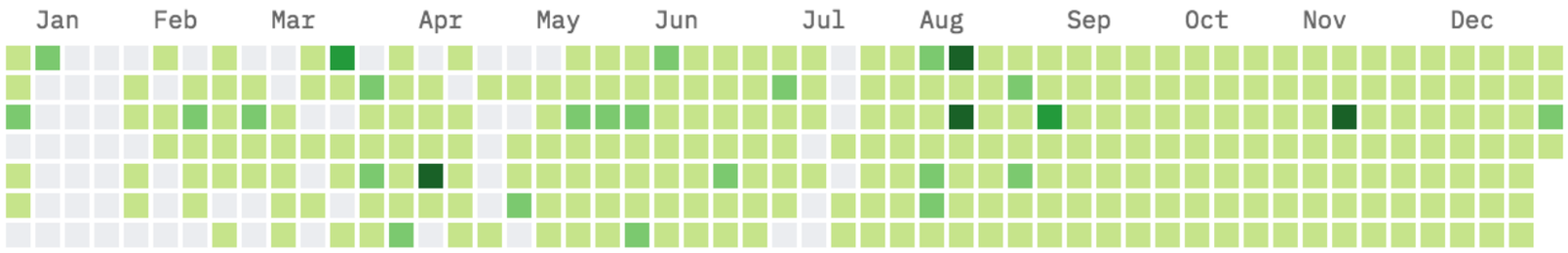
2016: 1,616 Contributions



2015: 2,529 Contributions



2014: 2,740 Contributions



事例

Overview Repositories 101 Stars 345 Followers 44 Following 16

Pinned repositories

bqspec

SQL testing tool for Google BigQuery.

Python ★ 6

embexpr

embedded python expression parser (for mainly easy DSL or config file)

Python

N YONASHIRO

or

ne

Blo

Recruit Technologies Co.,Ltd.

Tokyo, Japan

owan.orisano@gmail.com

趣味

4月に@t_wadaさんの講義を聞いてから`write code every day`を実践していていろんなコードを趣味で書いてきたので紹介させてください.

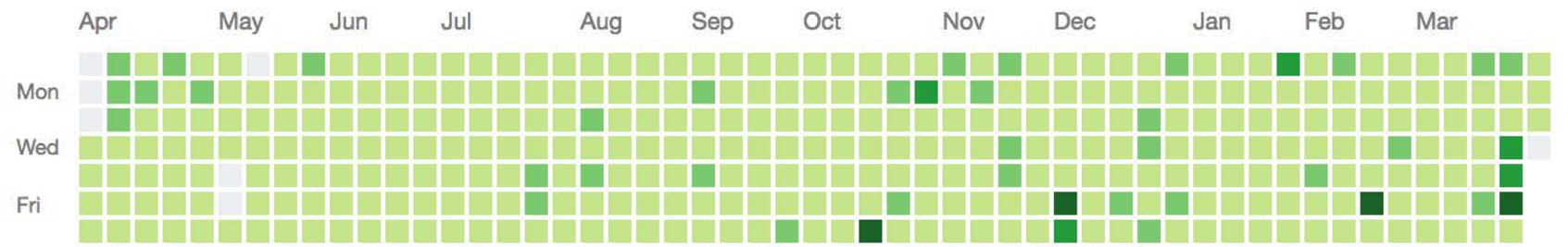
unofficial Active Directory Authentication Library for golang

Go ★ 6

A Python3 implementation of the Wiener attack on RSA

Python ★ 6 1

1,754 contributions in the last year



現在の @jeresig の github profile

John Resig
jeresig

Principal Architect at **@Khan**

@Khan
Brooklyn, NY
Sign in to view email
https://johnresig.com/

PRO
Block or report user

Organizations

Overview | Repositories 107 | Projects 0 | Stars 578 | Followers 15.7k | Following 29

Pinned

mongaku/mongaku
An image similarity search engine and database.
JavaScript ★ 55 9

node-stream-playground
Explore Node.js streams with an interactive playground.
JavaScript ★ 460 42

node-romaji-name
Normalize and fix common issues with Romaji-based Japanese names.
JavaScript ★ 22 3

node-yearrange
Node module for converting year range strings into usable dates.
JavaScript ★ 19 1

665 contributions in the last year

2020 2019 2018 2017 2016 2015 2014

Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar

Learn how we count contributions. Less More

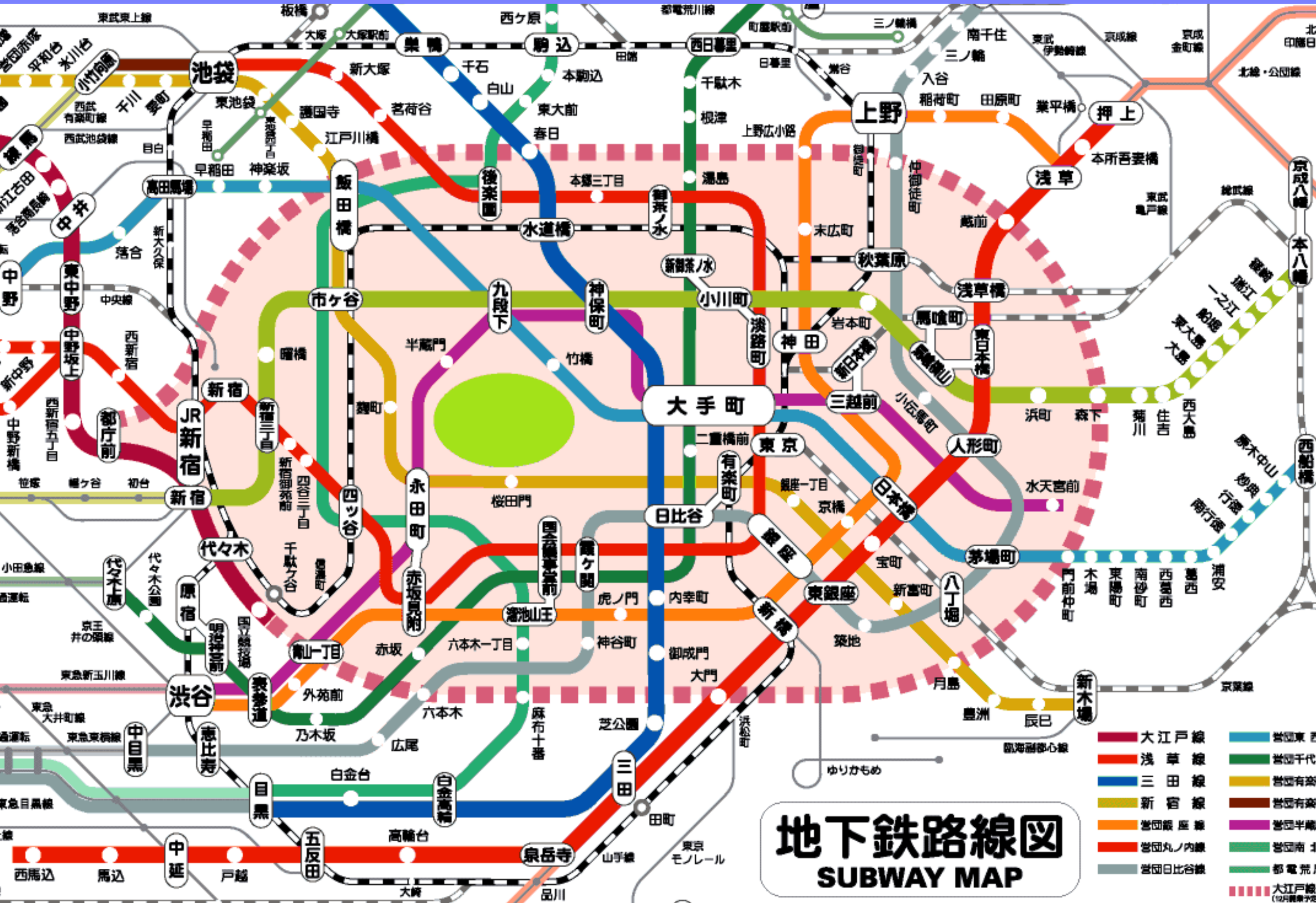
@Khan @mongaku @webpack

Activity overview

32%
Code review

https://github.com/jeresig

始発駅近くに住み、座れる可能性をコントロールする



毎日コードを書く

👉 年下から学ぶ

過去から未来を見る

人のつくる渦を見る

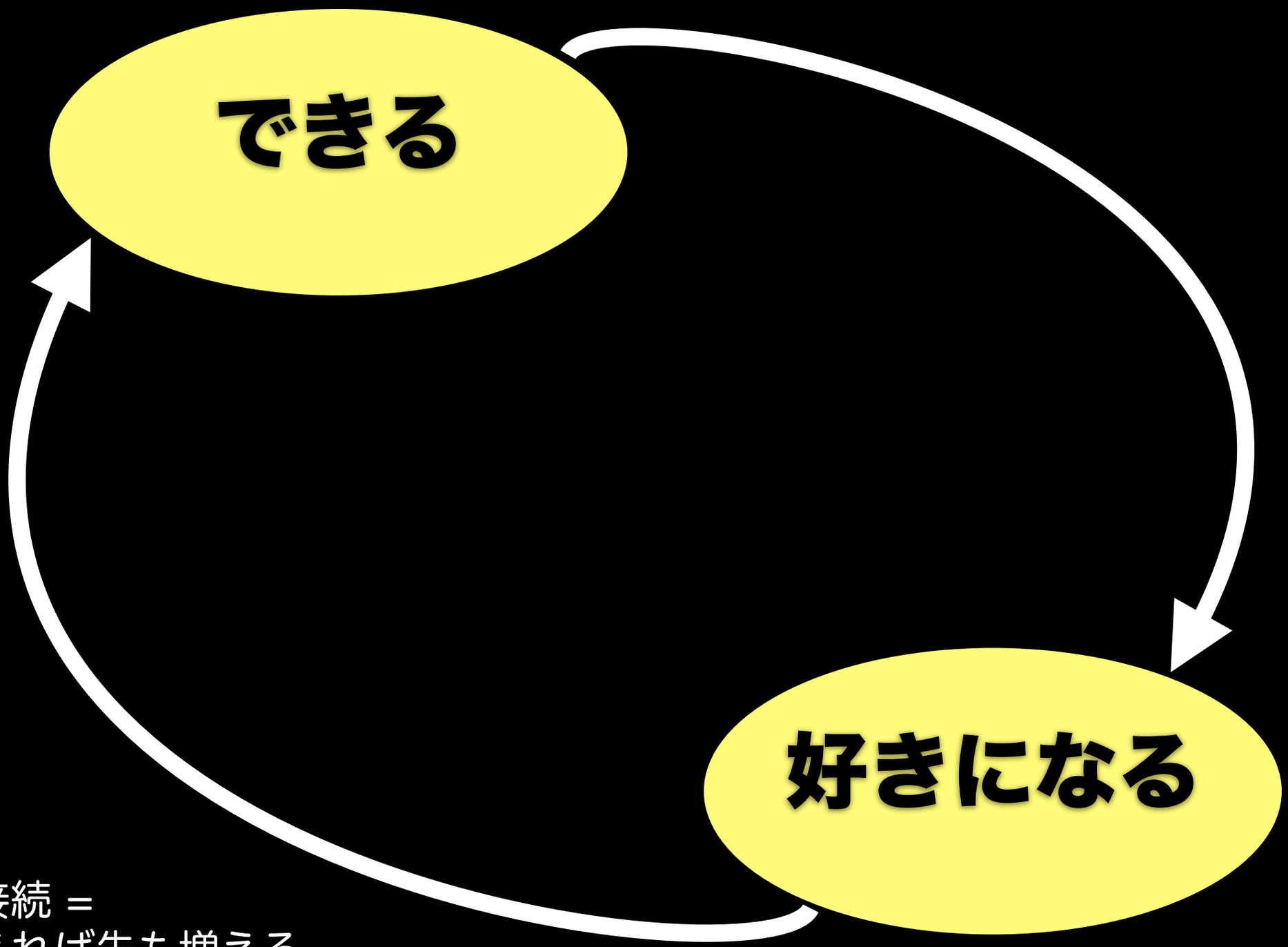
大事なことに集中する

2. 年下から

学ぶ

“一生プログラマーでいれるかどうかは、言い換えれば年下から学べるか否か。”

過剰適合とタコツボ化



矢印: 正の接続 =
　根元が増えれば先も増える。
　根元が減れば先も減る

ベンチマークとアンラーニング

- 定期的に自分のスキルを棚卸しする
- 積極的に外部に出て、自分のスキルを相対化する
- 使う道具を定期的に変える
- 未知のコミュニティに参加する
- 若者から学ぶ
- 若者と同じ土俵で競う

ペアプログラミング

ベテランにはアンラーニングのチャンス

毎日コードを書く

年下から学ぶ

👉 過去から未来を見る

人のつくる渦を見る

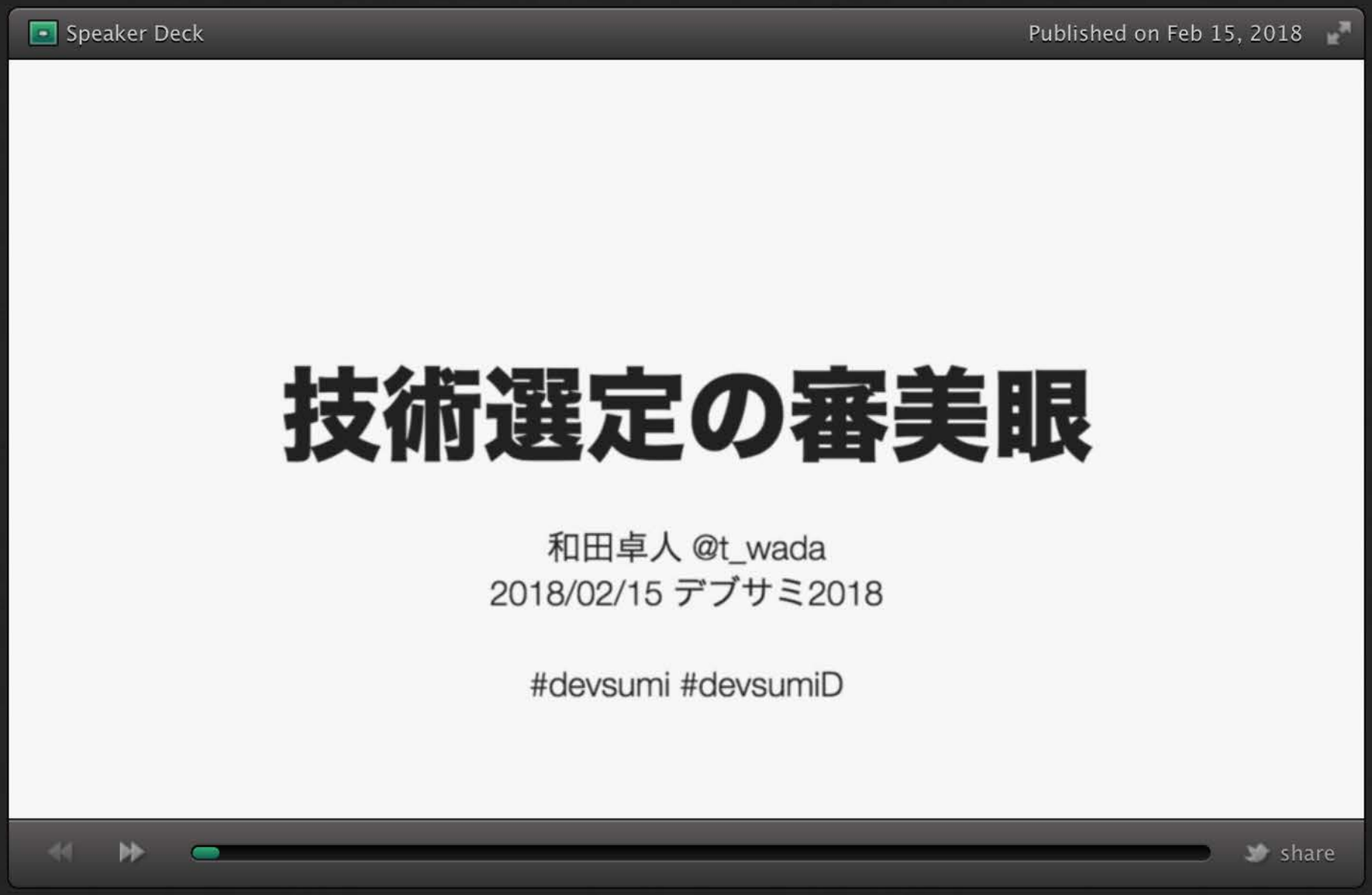
大事なことに集中する

3. 過去から

未来を知る

技術は「振り子」

技術は「らせん」



技術選定の審美眼 / Understanding the Spiral of Technologies

by Takuto Wada

Published February 15, 2018 in Programming

初演: 2018/02/15 デブサミ2018 15-D-1
ハッシュタグ: #devsumi #devsumiD

技術選定の審美眼。時代を超えて生き続ける技術と、破壊的な変化をもたらす技術を見極める（前編）。デブサミ2018

2018年2月20日

IT分野の技術はつねに速いスピードで変化し続けています。そうしたなかで登場する新しい技術には、スルーしてもかまわないものと、スルーすべきでない重要な技術があります。

めまぐるしい変化の中で、どこに着目することで重要な技術を見極めるのか。一方で、長年にわたって変わらず現役で使われ続けている技術にはどのような特徴があるのでしょうか。

2月15日と16日の2日間、都内で開催されたイベント「Developers Summit 2018」では、こうした変化する技術、変化しない技術をテーマにした、和田卓人氏の講演「技術選定の審美眼」が行われました。

本記事では、その講演の内容をダイジェストで紹介します。

技術選定の審美眼

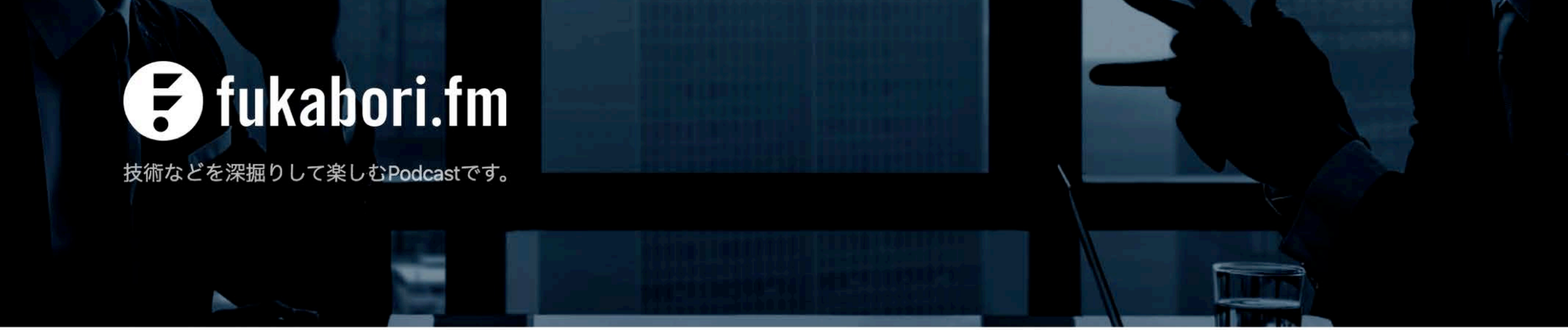
和田卓人といいます。

カテゴリ

- クラウド
- Docker / コンテナ / 仮想化
- 開発ツール / 言語 / プログラミング
- DevOps / アジャイル開発
- 運用ツール / システム運用
- Web技術 / JavaScript
- Windows / Linux / OS
- サーバ / ストレージ / ネットワーク
- RDB / NoSQL / ミドルウェア
- 機械学習 / AI / ビッグデータ
- 業務アプリケーション / Office
- 働き方 / 給与 / 学び
- 業界動向 / IoT / その他
- 編集後記 / おもしろ
- 殿堂入り / オススメ

Blogger in Chief

Junichi Niino（jniino）
IT系の雑誌編集者、オンラインメデ



29. 技術選定の審美眼(2) w/ twada

2020年03月22日

twadaさんをゲストに、前回に引き続き技術選定の審美眼、集中と分散の螺旋について語っていただいたエピソードです。

28. 技術選定の審美眼(1) w/ twada

2020年03月16日

twadaさんをゲストに、技術選定の審美眼およびUnix、REST/Web、RDMBS/SQLの共通点について語っていただいたエピソードです。

27. 論理削除とは何か？どのような解法があるのか？ w/ twada

2020年03月02日

twadaさんをゲストに、RDBにおける論理削除や、その解決方法について語っていただいたエピソードです。

「T字型」ではなく複数の柱を

毎日コードを書く

年下から学ぶ

過去から未来を見る

👉 人のつくる渦を見る

大事なことに集中する

4. 人のつくる

渦を見る

組織の時代から個人の時代へ

Features Business Explore Marketplace Pricing

Search GitHub

Sign in or Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside millions of other developers.

Username

Pick a username

Email

you@example.com

Password

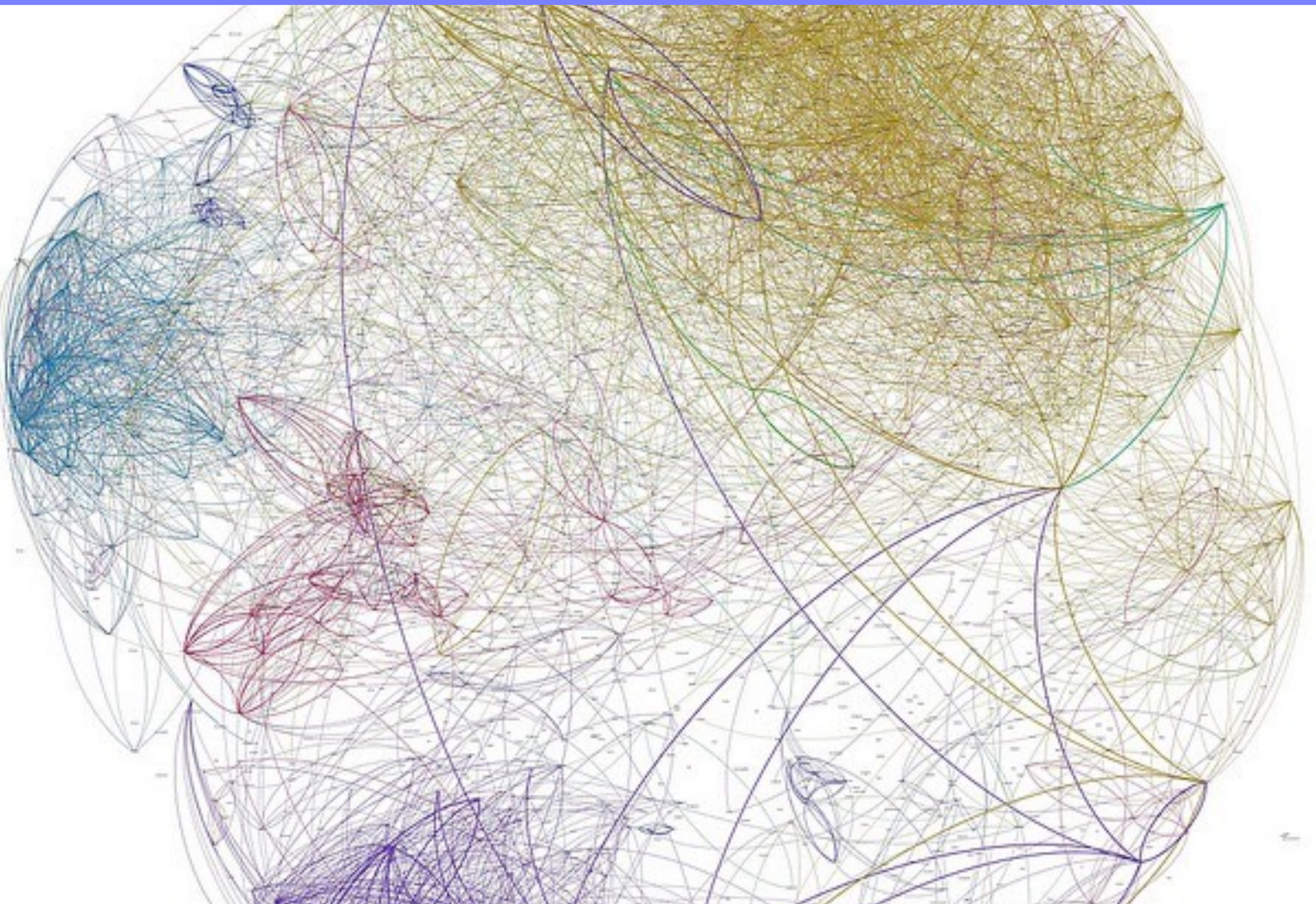
Create a password

Use at least one letter, one numeral, and seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We'll occasionally send you account related emails.

個が多く集まると何かが起こる



ロードマップ指向からエコシステム指向へ

2014-03-30

Agile Media Network
パートナーブログ

My Profile

MY PROFILE
POWERED BY iddy

hot entries

- 文教族が専門知を評価しないで利権を失うの巻 - アンカテ
- ドロドロなIT - アンカテ
- 世界を変えているのはコンピュータでなくて数学じゃないだろうか？ - アンカテ
- 片山被告の独善性の謎 - ア

ロードマップ指向とエコシステム指向 ★★18★

IT業界の世代間ギャップを「ロードマップ指向 VS エコシステム指向」という図式でまとめるとうまく整理できるような気がしてきた。

他の業界でも、常に勉強してないと仕事にならない所では、似たような問題があるかもしれない。普通の人は「ロードマップ」の中では真ん中を進むべきで、「エコシステム」の中では真ん中を避けるべきだ、という話。

私は、80年代からずっとプログラマをしていて、今でも現場でコードを書く仕事をしているので、同世代の人から、彼らと現場の若い人との仲裁役というか通訳のようなことを期待されることが多い。

確かにそこには微妙なギャップがあって、自分はどちらの言い分にも共感する所があるので、なんとかそれを言葉にしたいのだが、なかなかうまく言えなかった。

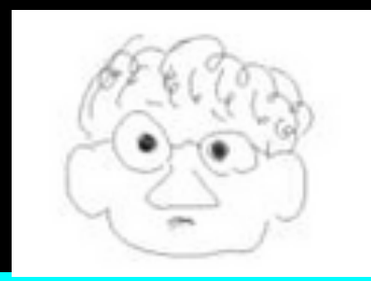
プログラマという仕事は、今も昔も勉強をしてないと普通の仕事も成立しないのだが、その勉強の仕方というか意味づけが、違ってきていると思うのだ。単純に言えば、量の問題である。今は勉強すべきことがずっと多い。

しかし、量の問題としては、何年も前に現場を離れて管理職になっ

http://d.hatena.ne.jp/essa/20140330/p1

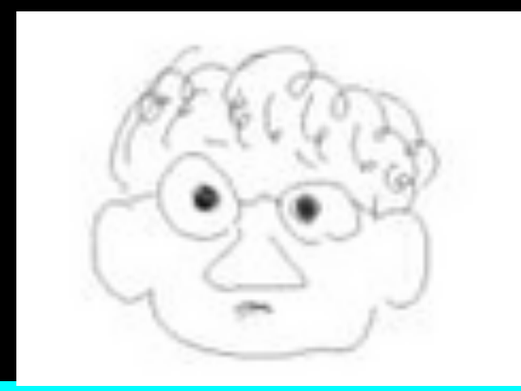
“しかし、今の業界は、「エコシステム」の時代だ。熱帯雨林のように、食いあいつつ共生しあうさまざなタイプのプレイヤーが、自分の為だけの個別の意思決定をして、その相互作用で技術が発展していく。「エコシステム」は矛盾だらけで、ある技術が発展するのと同時に、そのアンチテーゼとなる技術も伸びる”

“「ロードマップ」が指し示す未来の方向と違う方向に進むことは致命的な間違いだが、「エコシステム」はむしろ中心部がレッドオーシャンで、周辺部に生き残りが容易なブルーオーシャンがある”



http://d.hatena.ne.jp/essa/20140330/p1

“普通の人は「ロードマップ」の中では真ん中を進むべきで、「エコシステム」の中では真ん中を避けるべきだ”



http://d.hatena.ne.jp/essa/20140330/p1

The Rise of “Worse is Better”

rpg@lucid.com
日本語訳:
daiti-m@is.aist-nara.ac.jp

私や Common Lisp と CLOS のデザイナーのほとんどは、MIT/Stanford 方式の設計に親しんでいる。 この方式の核心は、「正しい」やり方をせよ、という ことにつきる。デザイナーにとっては、以下の点をすべて正しく満たすことが 重要である。

- 簡潔性
  デザインは実装と使用法の両面において単純でなければならない。 このとき、使用法が単純な方が、実装が単純なことより重要である。
- 正当性
  デザインはすべての点において正しいものでなければならない。 誤りは許されない。
- 一貫性
  デザインは一貫性を欠いたものであってはならない。一貫性を保つ ためには完全性は少しだけ犠牲にしてもよい。一貫性は 正当性と同じぐらい重要である。
- 完全性
  デザインは実際に起こる重要な状況にはすべて対応できなければ ならない。起こることが予想される場合はすべて扱えなければならない。 簡潔さのために完全さが大きく損なわれることがあってはならない。

ほとんどの人がこの条件に同意されることと思う。このデザイン哲学を、以下 「MITアプローチ」と呼ぶことにしたい。Common Lisp (とCLOS) や Scheme はこのデザイン哲学を体現している。

"悪い方がよい"原則はこれと少しだけ異なる：

- 簡潔性
  デザインは実装と使用法の両面において単純でなければ ならない。このとき、実装が単純な方が、使用法が単純なことより重要 である。
  単純さがデザインにおいて最も重視されるべき事柄である。
- 正当性
  デザインはすべての点において正しいものでなければならない。 ただし、どちらかといえば、正しいことよりは単純なことの方が重要である。
- 一貫性
  デザインは一貫性を欠いたものであってはならない。単純さを保つために、一貫性は犠牲にされることがある。しかし、あまり起こら

Rui Ueyama
@rui314

Follow

なんか違うんだよなぁという方向にまとめている人も少なくなかったけど、これはとてもよいまとめ。

[redacted]

Takuto Wada @t_wada
Worse Is Better に関する
する（両立できない）と
る抽象になったとしても
ドルが下がり、進化的な

Show this thread

4:01 PM - 7 Apr 2018

28 Retweets **101** Likes

28

[redacted]

Takuto Wada
@t_wada

Follow

Worse Is Better に関する自分の解釈は「設計の正しさ/美しさと実装の単純さが対立する（両立できない）ときは、実装の単純さを選択した方が、たとえそれが漏れのある抽象になったとしても現実の問題を解決し、実装の単純さによって開発参加のハードルが下がり、進化的な強さを獲得できる」というもの

12:17 AM - 6 Apr 2018

126 Retweets **304** Likes

126 304

https://twitter.com/t_wada/status/982155252643250177

過去を知り、未来に備える
技術選定の審美眼2019

和田卓人 @t_wada

2019/05/17

https://speakerdeck.com/twada/worse-is-better-understanding-the-spiral-of-technologies-2019-edition

毎日コードを書く

年下から学ぶ

過去から未来を見る

人のつくる渦を見る

👉 大事なことに集中する

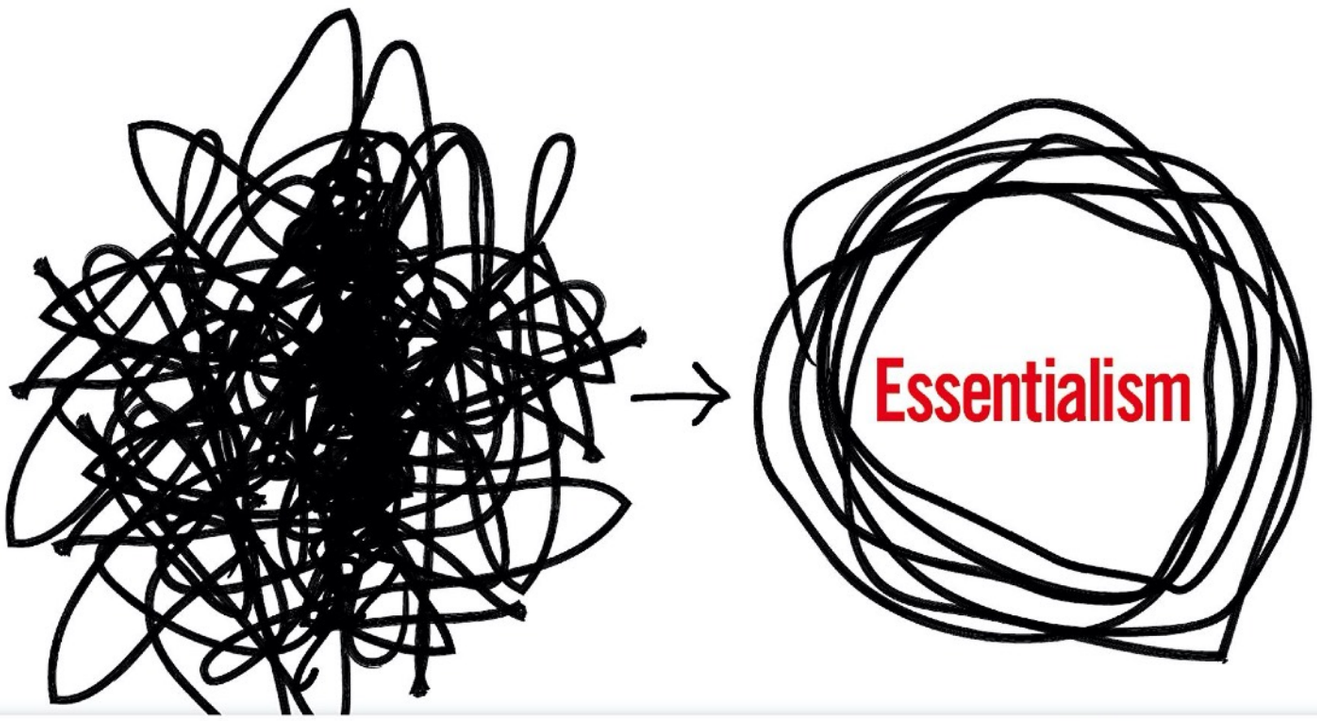
5. 大事なことに

集中する

エッセンシャル思考

最少の時間で成果を最大にする

グレッグ・マキューン＝著　高橋璃子＝訳



ダニエル・ピンク（『モチベーション3.0』著者）
クリス・ギレボー（『1万円起業』著者）
アダム・グラント（『GIVE & TAKE』著者）
絶賛！

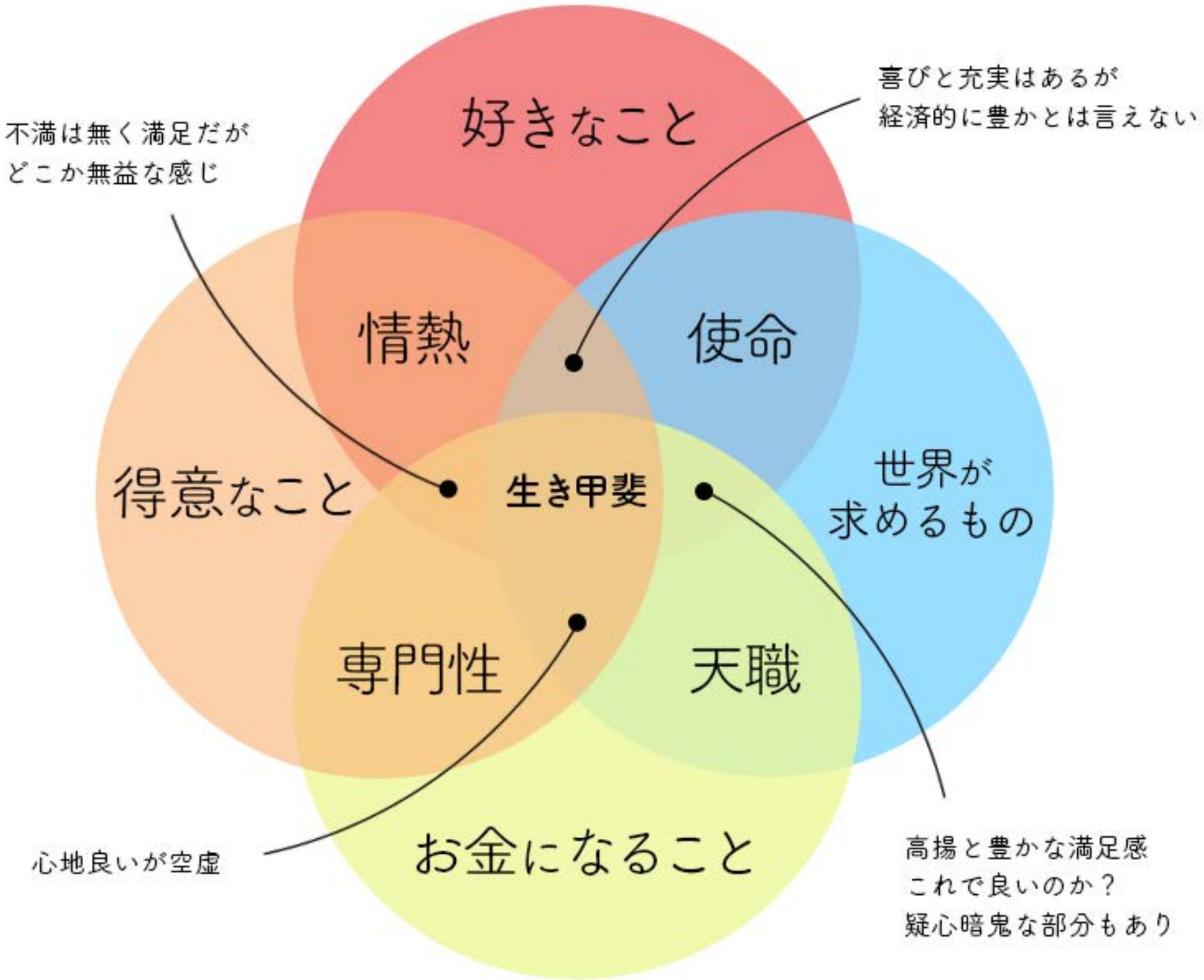
10万部突破！

2014年
全米ベストセラーの邦訳！

apple、google、facebook、Twitterのアドバイザーを務める著者の

99%の無駄を捨て
1%に集中する方法！

生き甲斐の図



ご清聴ありがとうございました

毎日コードを書く

年下から学ぶ

過去から未来を見る

人のつくる渦を見る

大事なことに集中する