Keep learning

学び続けるための方法

ken7253

Frontend developer

技術記事を書いたりするのが趣味。

最近はNext.jsを使ったアプリケーションを書いています。

インターフェイス設計やアクセシビリティ・SSG関連の技術に興味があります。

- https://github.com/ken7253
- https://zenn.dev/ken7253
- https://dairoku-studio.com

なぜこのスライドを作ったのか

- そろそろフロントエンド５年目になる
- 持論を展開するようなスライドを書きたくなった
- いちおう中堅エンジニアぐらいにはなれたような気がする
- じゃあ中堅になるまでにどういう勉強方法したっけってまとめを書いてみる

話すこと

- 勉強法を模索して気付いたことをまとめます
- あくまで個人の体感の話です
- 学習方法は人それぞれなので自分に合ったやり方を探すきっかけに

知識のアップデートで気をつけていること

学習における負のループ

学習における負のループ

学習における負のループとは下記のような状態

1. キャリアが増えるにつれ、知らないといけないことが増えてくる気がする
2. 不安から技術的な発言や質問を控えるようになってしまう
3. 学習の機会が減る
4. 学習効率の悪化によって更にキャッチアップが難しくなる

業務が回せるようになった時期、知らないことを**質問する機会が減る**ので陥りやすい。

負のループへの対処

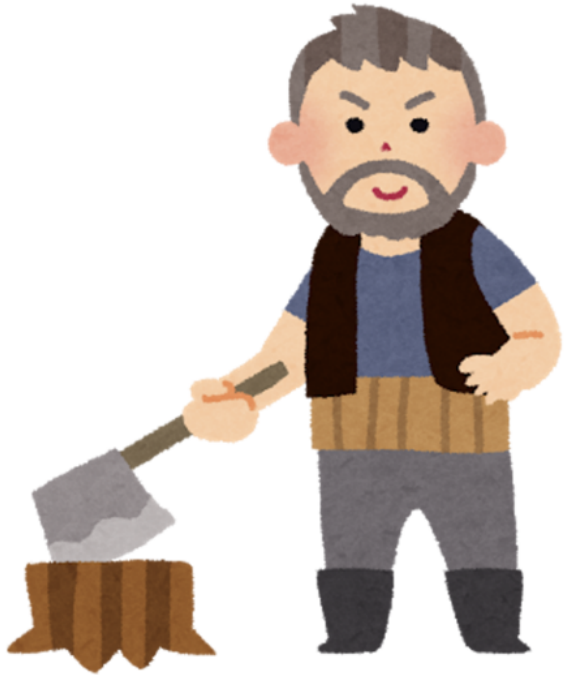
自分が想像しているコミュニティのレベル感と自分自身のレベル感との乖離が原因

「コミュニティのレベルを下げる」ということはできないので基本的に学ぶしかない

- インプットの習慣化で軽減できる（気がする）
- アウトプットの習慣化で軽減できる（気がする）
- どれだけ勉強しても質問はし忘れるし、アウトプットも怖いので諦める
- 恐怖心があるからこそきちんと調べてからアウトプットできる

木こりのジレンマ

木こりのジレンマ



【木こりのジレンマ】

“ある木こりが、がんばって木を切っている。

通りがかった旅人がその様子を眺めていたが、
斧を振るう勢いのわりに、なかなか木が切れていない。

見ると木こりの使っている斧が
刃こぼれしているようなので、旅人は言った。

「斧を研いだほうがいいのでは？」

すると、木こりは言った。

「わかっちゃいるんだけどね、
木を切るのに忙しくて、それどころじゃないよ」”

※【神様がいれば「ふりかえり」はいらない。 - ユニファ開発者ブログ】より引用

木こりのジレンマ

目の前の作業をこなすのに精一杯 で余裕がない状況

先程とは違い中堅以降で起こりがちな現象だと思っています。

- 仕事に精一杯取り組むのは偉い
- 休み方や斧の研ぎ方を教えるのも仕事のうち
- 「業務が忙しくて勉強ができない」という理由づくりを（自分に）させない

学びと同時に業務の効率化と自動化を進めてチームに余裕を持たせないといけない。

どう学ぶか

業務時間内で学ぶ習慣づけをする

いきなり業務時間外でも「バリバリコード書いてアウトプット」は正直敷居が高い。

できないことを目標にしてはいけない、**できる範囲で確実に**

- 毎日30分、技術動向を追うために記事を見る
- 基礎知識を深めるために毎日MDNを読む
- 集中力が切れたらX（Twitter）を見る

毎日30分学習している人とそうでない人では年間120時間程度の差が出る

```
30(m) * 245(d) / 60(m) = 122.5(h) / 8(h) = 15.31(営業日)
```

必ず業務時間より先に学習時間を確保する

パーキンソンの法則 第一法則 より業務の空き時間で勉強することはできない。

そのため、学習時間は業務時間よりも先に確保する必要がある。

パーキンソンの法則とは、イギリスの歴史学者・政治学者であるシリル・ノースコート・パーキンソンが当時の行政組織を研究するなかで、組織・運営と人間の心理作用に関する非合理的な行動の分析を説いた法則を指す。

パーキンソンの法則は以下の2つの法則から成り立つ。

- 第一法則：「仕事の量は、完成のために与えられた時間をすべて満たすまで膨張する」
- 第二法則：「支出の額は、収入の額に達するまで膨張する」

※ パーキンソンの法則 | 三菱UFJリサーチ&コンサルティング より引用。

アウトプットを行う

インプットの習慣がついたらアウトプットもする。

アウトプットも絶対にできることから始めてみる。

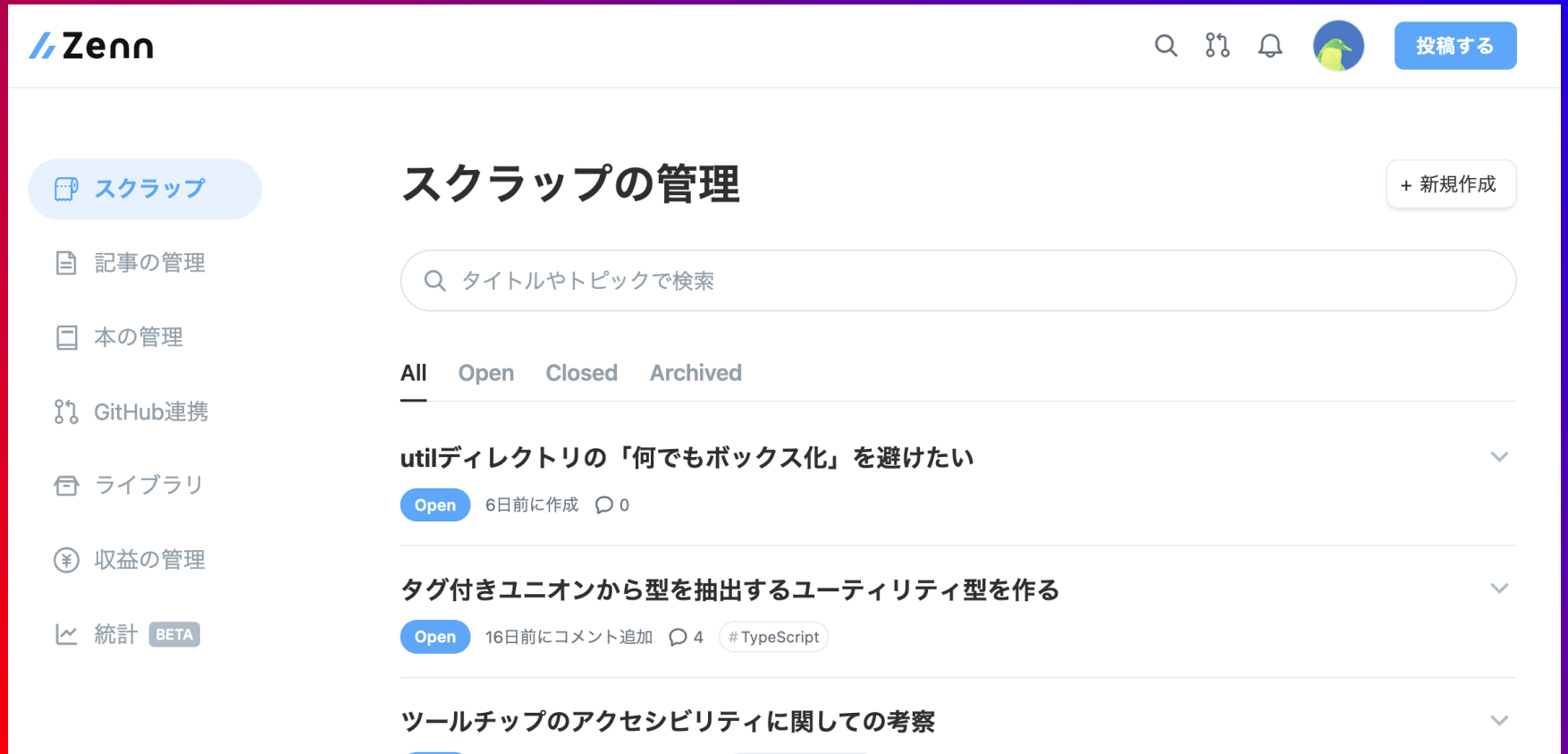
まずは、読んだ記事のURLとかをX（Twitter）や自分のtimesにポストするだけでいい。

- なるべくパブリックなチャンネルにポストする
- 読み始めにURLを貼って読みながらスレッドをつなげるとよい
- 記事以外にも業務の中で学んだことなどを書き留めておく

（忘れるので）何かしらの形で 学んだことを残しておく

アウトプットを行う

Zenn.devのスクラップ機能などもおすすめ。



習慣がついてきたら

習慣化ができてきたら、たまに質を上げてみる。

インプット

- 社外の勉強会に参加してみる
- 難しめの技術書を買ってみる（オライリー・技評）

アウトプット

- LTで話してみる
- 記事を書いてみる

> 💡 インプットの質を先に上げたほうがスムーズに進みやすい。

何を学ぶべきか

3つの軸

勉強する対象は色々あるので分割してみる。

- 仕事に必要なこと
- エンジニアとしての基礎知識
- 自分が興味がある領域・技術

仕事に必要なこと

おそらくみんなもうやってると思うので割愛

エンジニアとしての基礎知識

OSの話であったり、プロトコルの話だったり、プログラミングパラダイムだったり。

基礎知識があるかないかで学習効率がかなり変わってくる。

Reactを例にすると...

- Reactを学習する前に関数型言語の考え方を知っていると理解がかなり短縮される。
- オブジェクト指向しか知らないでReactを触ると勘違いで挙動の理解に苦しむ。

自分の進みたい先が分からない時はとにかく基礎を学ぶといい。

自分の興味ある領域・技術

基礎を徹底的に学ぶのもいいが、興味ある分野が見つかったら調べてみるといい。

小さな知識でも知っている人が少ない知識を持っていると凄そうに見える。

インプットコストや前提知識は大きくなるが安定したポジションを取れるとリターンも大きい。

```
フロントエンドエンジニアで、領域としては${特定の技術領域}などをやってます。
```

```
フロントエンドエンジニアで、${特定の技術領域}が好きです。
```

最終的に**自信を持ってこういう挨拶**ができるように勉強を続ける。

まとめ

まとめ

- まずはインプットの習慣化を業務時間内で
- 学習時間は業務時間より先に確保しておく
- アウトプットも最小限でやってみる
- 慣れてきたら質を上げてみる
- 学ぶべきことが分からない時は、エンジニアとして抑えておくべきことを学ぶ
- 興味が湧く領域があったら自信を持って話せるまで学んでみる