まとめと次のステップ

結城洋志
株式会社クリアコード

実践リーダブルコード
2022-11-02

おさらい

- ✓ 講座の目的？
- ✓ リーダブルコードの必要性？
- ✓ 講座でやったこと？

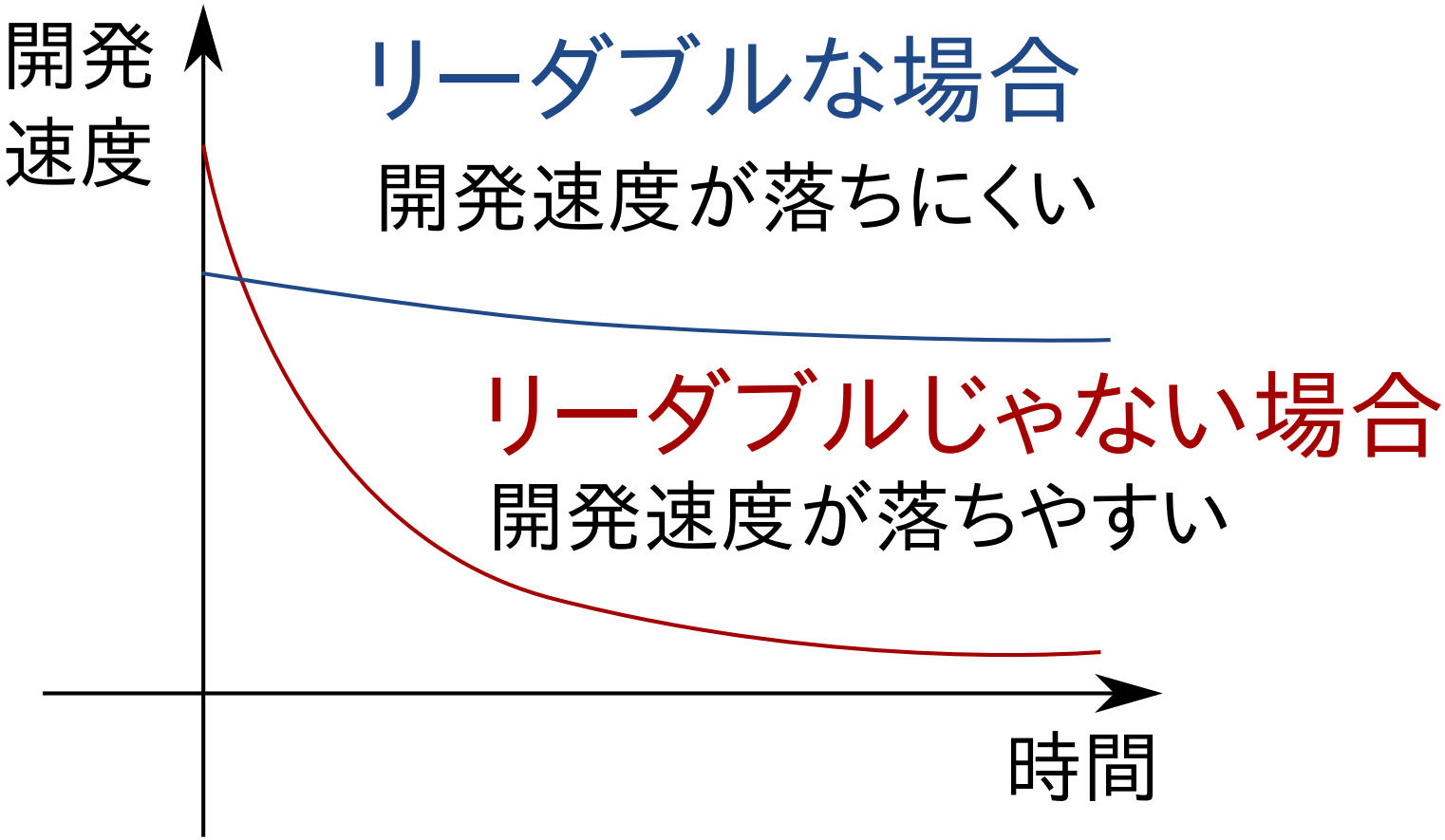
講座の目的

- ✓ 自分の開発チームに
- ✓ **リーダブルなコードが当たり前な文化の作り方**を
- ✓ 持ち帰る

リーダブルコードの必要性

- ✓ 既存のコードを読んで
  素早く内容を把握するため
- ✓ 既存のコードに
  手を加える・続きを書くときの
  効率を落とさないため

変更コストと開発速度



文化の作り方の流れ（1）

- まず自分が
  コードを読む人になる
  - よいコードを共有し、
    リーダブルコードの基準を作る
  - 「読む人」が
    読みやすいなら
    リーダブル

文化の作り方の流れ（2）

- ✓ **コードを読みあうチーム**になる
- ✓ 基準の育て方
  - ✓ 各メンバーがコードを読む
  - ✓ リーダブルだと思ったコードを共有
  - ✓ チームとしてリーダブルかを判断
  - ✓ →チームの基準に加わる

基準の育て方

- コードを読む文化を作る
  - まず自分が読み始める
  - リーダブルなコードを探す
  - 見つけたリーダブルなコードを他のメンバーに伝える
  - →コードが読まれるという自覚がチームに浸透

今日やったのはここまで

コードを読むきっかけを増やす！

次のステップの例

- ✓ **コミット単位**で読む
- ✓ **スタックトレース**を手がかりに読む
- ✓ **出力されるメッセージ**を手がかりに読む

例1：コミット単位で読む

```
commit 4d516212f14ddcf04a8ead386104dab1d37ab8f4
Author: YUKI "Piro" Hiroshi <piro.outsider.reflex@gmail.com>
Date:   Tue Oct 4 14:12:49 2022 +0900

    Support backquote

diff --git a/common/placeholder-parser.js b/common/placeholder-parser.js
index fe6d4df..d9c82ee 100644
--- a/common/placeholder-parser.js
+++ b/common/placeholder-parser.js
@@ -31,14 +32,17 @@ export function process(input, processor, processedInput...

     if (escaped) {
       if ((inDoubleQuoteString && character == '"') ||
-          (inSingleQuoteString && character == "'")) {
+          (inSingleQuoteString && character == "'") ||
+          (inBackQuoteString && character == '`')) {
         if (inArgsPart)
           rawArgs += '\\';
       }
```

コミットの読み方

- ✓ コード全体ではなく差分を読む
- ✓ コードの中身・設計の仕方ではなく
  コードの書き方・開発の仕方に注目する
- ✓ **文脈なしでも読みやすい**
  →コードがリーダブルな証拠

コミット単位で読む負担を減らす

- pull型よりpush型にする
  （メール、チャットへの自動投稿など）
  - 読むのに取りかかるコストが下がる
  - 流し読む（負担が多いと続かない）
- 問題探し視点では読まない
  （必要ならコードレビューを実施）

コードを読むと気付きが増える

- ✓「詰まらずに読めるなあ」
- ✓「詳しくは分からないけど
  流れを追いやすいなあ」
- ✓「コミットの差分から
  趣旨を読み取りやすいなあ」
- ✓ →コードやコミットが
  リーダブルな証拠

コミット単位で読むのが難しい場合

- ✓ 文脈がないと読むのが辛い
- ✓ 知らない言語、
  知らないプロジェクトだと
  目が滑る

→文脈を伴った読み方をしてみる

例2：スタックトレース

```
SyntaxError: /var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/parser/ext/enscript.rb:120:
 syntax error, unexpected keyword_end, expecting end-of-input
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/parser/rd/ext/block-verbatim.rb:25:in
`<top (required)>'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/parser/rd/rd2rabbit-lib.rb:9:in `<top
(required)>'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/parser/rd.rb:12:in `<top (required)>'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/utils.rb:30:in `require_safe'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/utils.rb:48:in `block (2 levels) in
require_files_under_directory_in_load_path'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/utils.rb:43:in `glob'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/utils.rb:43:in `block in
require_files_under_directory_in_load_path'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/utils.rb:41:in `each'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/utils.rb:41:in
`require_files_under_directory_in_load_path'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/parser.rb:4:in `<top (required)>'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/canvas.rb:8:in `<top (required)>'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/command/rabbit.rb:48:in `run'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/command/rabbit.rb:29:in `run'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/task/slide.rb:238:in `rabbit'
/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib/rabbit/task/slide.rb:100:in `block in define_run_task'
/var/lib/gems/2.5.0/gems/rake-12.3.2/exe/rake:27:in `<top (required)>'
Tasks: TOP => default => run
(See full trace by running task with --trace)
```

恐れることはない

```
SyntaxError:
  /var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib
    /rabbit/parser/ext/enscript.rb:120:
      syntax error, unexpected keyword_end, expecting end-of-input

/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib
  /rabbit/utils.rb:48:
    in `block (2 levels) in require_files_under_directory_in_load_path'

/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib
  /rabbit/utils.rb:43:in `glob'

/var/lib/gems/2.5.0/gems/rabbit-2.2.1/lib
  /rabbit/utils.rb:43:
    in `block in require_files_under_directory_in_load_path'
```

どこを見ればいいか書いてあるだけ

エラーに遭遇した時に深掘りしてみる

- ✓ どういう経緯でエラーになったかが分かる
- ✓ 回避方法や解決方法が分かる
  （当てずっぽうで色々試さなくて済む）
- ✓ **経緯を辿りやすい**
  →コードがリーダブルな証拠

例3：画面に出ているメッセージ

Firefox のファイルを他のアプリケーションが使用しているため、
ブックマークと履歴のシステムが無効化されます。この問題は
セキュリティソフトが原因で生じることがあります。

ソースコード内を検索する

```
https://searchfox.org/l10n
  /source/ja/browser/chrome/browser/places/places.properties


...
# LOCALIZATION NOTE (lockPrompt.text)
# %S will be replaced with the application name.
lockPrompt.text    =%S のファイルを他のアプリケーションが
                    使用しているため、...
...
```

変数名や関数名などを辿っていく

```
https://searchfox.org/mozilla-central
  /source/browser/components/BrowserGlue.jsm

/**
 * Show the notificationBox for a locked places database.
 */
_showPlacesLockedNotificationBox:
    function BG__showPlacesLockedNotificationBox() {
  ...
  var text = placesBundle.formatStringFromName("lockPrompt.text", [
    applicationName,
  ]);
  ...
```

目に見えるものを手がかりに深掘りしてみる

- ✓ どういう経緯で表示されているかが分かる
- ✓ 回避方法や解決方法が分かる
  （当てずっぽうで色々試さなくて済む）
- ✓ **経緯を辿りやすい**
  →コードがリーダブルな証拠

OSSのコードを読んで学ぶ

- OSSのコードは
  リーダブルになるように
  気をつけて書かれることが多い
  - ×　すごい技術力があるから
    リーダブルに書けている
  - ○　並の技術力なので
    リーダブルにしないと
    やってられない

共有の仕方の例

- ✓ リーダブルコードの共有方法
  - ✓ Wikiに書く（全チームで有効）
  - ✓ コードで伝える（上級チーム向け）

Wikiに書いて共有する

- ✓ Wikiもdiffを通知するようにするとなお良い
  （RedmineとGitHub用はツールあり）
- ✓ 後で参照できる
- ✓ 更新もできる
  （リーダブルコードの基準は変わることもある！）

実際のコードで共有する

- 上級チーム向け
  （チームにコードを読む文化が根付いた後）
- リーダブルコードを
  真似てコミット
  - →他の人：「またこの書き方だ」
  - →真似する人増加→チームが合意
  - →チームが合意→Wikiにまとめる

コードを読む文化

- 新人にも真っ先に
  その文化に馴染んでもらう
  - 新人のスムーズな受け入れに有用
  - 人の入れ替えがチームのリーダブル
    基準の見直しの機会になる

新人のスムーズな受け入れに有用

開発を通じて↓を共有できる

- ✓ チームが大事にしていること
  - ✓ チームのリーダブル基準
  - ✓ チームの開発スタイル

（ただし、上級チームになっていればだが。）

新人のスムーズな受け入れに有用

チームが大事にしていることが
開発を通じて共有される

↓

- ✓ 新人に手取り足取り教えずに済む
  - ✓ 開発速度低下を抑えられる

リーダブル基準の見直しの機会に

チームが大事にしていることが
開発を通じて共有される

↓

- ✓「もっとこっちの方が
  リーダブルでは？」
  - ✓チームのリーダブル基準の
    見直しのよい機会

これからやること

- この講座をチームでもやる
  - 資料はすべて再利用可能
- 自分がコードを読み始める
  - 自分が変更するコードの周辺から
リーダブルコードを探す
  - 見つけたリーダブルなコードを
他のメンバーに伝える

読みにくいコードに打ち克つ！

- ✓ × 読むコストが勿体ないから読むのをやめる
- ✓ △ 「読みにくい」と指摘する
- ✓ ○ 読みやすいコードにしていく
- ✓ ○ 読みやすくする知見を共有していく

サポート

- ✓ 今日の資料はすべて再利用可能
  https://github.com/clear-code/readable-code-workshop/tree/master/20221102a
  （ https://slide.rabbit-shocker.org/authors/Piro/ ）
  - ✓ 迷ったら読み返せる

クリアコード

- ✓ クリアなコードが大切
  - ✓ クリア == clear == 意図が明確
  - ✓ クリアなコードはリーダブルコード

みなさんのコーディングライフで
リーダブルコードが当たり前に
なることを応援します！