

# リーダブルコードを 読んだ後

須藤功平

株式会社クリアコード

DevLOVE 2012

2012/12/16

やること

# クリアコードの 開発方法を体験

# 目標

リーダーブルコードを  
当たり前にする

# イベントのテーマ

世界を変えるのは他の誰かではない、  
世界を変えるのは、**自分自身だ。**

# 目標

あなたが  
リーダーブルコードを  
当たり前にする

# やること

- ✓ クリアコードの開発方法を体験
  - ✓ 対象：野生のフリーソフトウェア
- ✓ 読む人を「想像」しない
- ✓ 読む人を「思い出す」

# 想像上の誰か

“想像上の誰かが自分のコードを理解しやすいかなんて考えるのは大変なことだ。”

[「リーダブルコード 1.5 でもやるんだよ」より引用]

# 想像しない

- ✓ 読む人を「想像」しない
- ✓ 読む人を「思い出す」

# 思い出すには？

# 読む人になる

# クリアコードの開発方法

- ✓ コミットメールが全員に届く  
(diff入り)
- ✓ 全員が読む
- ✓ 気になったらコメント  
(プロジェクト外の人でも)
- ✓ 「昨日印象に残ったコミット」  
を朝会で報告

# ポイント

- ✓ 全員が書く人で読む人
- ✓ 書く時：
  - ✓ 読む時の事を思いながら  
(こう書くと読みやすいんだよな。)
- ✓ 読む時：
  - ✓ 書く時の事を思いながら  
(これ読みやすいから今度同じように書こう。)

当日は

省略

なぜリーダブルコード？



開発を  
続けるため

# メリット

- ✓ **メンテナンスコストを低く保つ**  
(開発を続けるわりにあわなくなってしまう)
- ✓ **機能追加・変更をしやすい**  
(使われないものになって開発する必要がなくなる)
- ✓ **プログラマーの意欲を保つ**  
(プログラマーも人です)

# 注意

- ✓ すべてを解決するわけではない
  - ✓ よりよくする重要な方法の1つ
- ✓ トレードオフ

# メンテナンスコスト

“メンテナンスコストを低く保つためにずっとリファクタリングをし続けると…高いメンテナンスコスト”

”

# 素早い変更

“素早く機能追加・変更できる  
よう、リーダブルコードにする  
ために必要な時間を確保して開  
発しているのに、機能を追加し  
たいということと必ず1週間以上は  
必要だと言われる…素早くない  
変更”

”

# ゴール

- ✓ 本末転倒にならないように！
- ✓ リーダブルコードを書くことはゴールじゃない
- ✓ 効率よく目的を実現するのがゴール  
(今だけ効率がよければよいのではない。念のため。)
- ✓ 手早くリーダブルコードを書けるようになろう！

# なぜコミットメール？

# ペアプロより スケールする

# ペアプロ

- ✓ 常にコードレビュー
- ✓ 2人だけ
- ✓ 時間をあわせる
- ✓ 距離をあわせる
- ✓ プロジェクトをあわせる

# コミットメール

- ✓ 常にコードレビュー
- ✓ 2人以上も読める
- ✓ 時間がずれてもよい
- ✓ 距離がずれてもよい
- ✓ プロジェクトがずれてもよい

# コミット

ペアプロしているように！

- ✓ 小さくコミット
- ✓ 意図が伝わるコミット

# ペアプロの方がよいこと

## 全体の設計とかもみれる

- ✓ コミットメールだけだとムリ
- ✓ pull requestだけでもムリ
- ✓ 一緒に書いていないとムリ  
(同じプロジェクトだといけそう)

ここままで  
飛ばす

# 確認

# 事前課題

# 事前課題

1. リーダブルコードを読む
2. Rubyコードの感想戦を読む
3. [github://clear-code/git-utils](https://github.com/clear-code/git-utils)をfork
4. `commit-email.rb`を読む
  - ✓ Rubyコードの感想戦のように読む

# 事前課題確認1

1. リーダブルコードを読む
2. Rubyコードの感想戦を読む
3. `github://clear-code/git-utils`をfork
4. `commit-email.rb`を読む
  - ✓ Rubyコードの感想戦のように読む

# 事前課題確認2

1. リーダブルコードを読む
2. Rubyコードの感想戦を読む
3. `github://clear-code/git-utils`をfork
4. `commit-email.rb`を読む
  - ✓ Rubyコードの感想戦のように読む

# 事前課題確認3

1. リーダブルコードを読む
2. Rubyコードの感想戦を読む
3. [github://clear-code/git-utils](https://github.com/clear-code/git-utils)をfork
4. `commit-email.rb`を読む
  - ✓ Rubyコードの感想戦のように読む

# 事前課題確認4

1. リーダブルコードを読む
2. Rubyコードの感想戦を読む
3. `github://clear-code/git-utils`をfork
4. `commit-email.rb`を読む
  - ✓ Rubyコードの感想戦のように読む

# むずかしい？

- ✓ 「うまくできない」は今のうち
- ✓ 聞けば答えます

# 進め方

- ✓ 読み手になる：15分
  - ✓ 事前課題をやっていない人：Rubyコードの感想戦を読む
- ✓ 読み手目線で書く：15分
  - ✓ 事前課題をやっていない人：読み手になる
- ✓ 質疑応答：10分

# 読み手になる：15分

1. 数人のグループを作る
2. グループ用のgit-utilsをclone
3. 読んでcommit & push
  - ✓ リーダブルなら理由をコメントとして書く
  - ✓ リーダブルじゃないなら直してコミットメッセージで理由を説明

# 読み手目線で書く：15分

## 1. 他のグループのリポジトリのコミットを確認

コミットメールが届いているはず

- ✓ 自分達にピンとこないものを選ぶ
- ✓ 直してメッセージで理由を説明
- ✓ コードで伝える

## 2. 自分たちのリポジトリを確認

## 3. 繰り返す

# 質疑応答：10分

- ✓ 気になったコミットは？
- ✓ やってみて困ったことは？
- ✓ 実践すると困りそうなことは？

# 宣伝

## コミットへのコメントサービス

<https://www.clear-code.com/services/#commit-comment>

“ お客様の開発プロジェクトのコミットを読んでコメントすることにより、お客様の開発チームが「当たり前のようにクリアなコードを書く文化」を育むことを支援します。

”