貢献の方法

React は Facebook の最初のオープンソースプロジェクトの１つで現在も非常に活発に開発されており、facebook.com 上のあらゆる人々にコードを届けることにも使用されています。私たちはこのプロジェクトへの貢献をできるだけ簡単かつ透明性の高いものにするために努力していますが、まだ完全ではありません。このドキュメントがプロジェクトへの貢献の手順を明確にし、あなたの持つ疑問を解決できれば幸いです。

行動規範
Facebook が採用するプロジェクト参加者に期待する行動規範があります。全文を読んでください、そうすれば参加者はどのような行動を取ればよいか、またどのような行動が許容されないのか理解できるでしょう。

オープンな開発
React に関する開発作業はすべて GitHub 上で直接行われます。コアチームメンバーと外部のコントリビューターの両方が、同じレビュープロセスを経由するプルリクエストを送ります。

ブランチの構成
私たちは `master branch` を全てのテストが通るベストな状態に保つために努力しています。しかし素早い開発のため、あなたのアプリケーションと互換性がないかもしれない API の変更を行うことがあります。そのため 最新安定版の React を利用することをお勧めします。
プルリクエストを送信する場合は、`master` ブランチに対して行ってください。私たちはメジャーバージョンの安定版ブランチを別々に管理していますが、それらへのプルリクエストは直接受け付けません。代わりに、master から最新の安定したメジャーバージョンへコードを壊さない cherry-pick を行います。

セマンティック・バージョニング
React はセマンティック・バージョニングの原則に従います。バグ修正のためのパッチバージョン、新機能のためのマイナーバージョン、そして重大な変更のためのメジャーバージョンをリリースします。私たちが重大な変更を加えるとき、ユーザーが今後の変更について前もって知り、コードを移行するために、私たちはマイナーバージョンで非推奨警告を行います。
私たちは全てのプルリクエストにラベルを付けます。ラベルは、行われる変更がパッチ、マイナー、メジャー バージョンのどれに該当するかによって決まります。数週間ごとにパッチバージョン、数ヶ月ごとにマイナーバージョン、そして年に 1、2 回メジャーバージョンをリリースします。
重要な変更はすべて changelog file に文書化されています。

バグ

既知の問題を知るには
私たちは公開されるバグの管理に GitHub Issues を使用しています。私たちはこれを注意深く見守り、修正中の作業がある場合はそれを明確にするようにします。新しい Issue を提出する前に、既に同じものが存在しないか確かめてください。

新しい問題の報告
バグを修正するための最善の方法は、バグを再現する最小のテストケースを提供することです。この JSFiddle テンプレート は素晴らしい出発点です。

セキュリティバグ
Facebook にはセキュリティバグの安全な開示のための報奨金制度が存在します。それを念頭において、セキュリティバグは公開の Issues に提出せず、上記のページの手順に従ってください。

連絡方法

- IRC：# freenode の reactjs
- ディスカッションフォーラム：discuss.reactjs.org

また、React に関して助けが必要な場合は、Discord 上の React コミュニティも存在します。

変更の提案
もしあなたがパブリック API に変更を加えたり、実装に些細とはいえない変更を加えたりしたい場合、Issue を提出することをお勧めします。これによって、あなたが大きな労力を割く前に提案について合意に達することができます。
バグを修正するだけの場合は、すぐにプルリクエストを送信しても問題ありませんが、修正したいバグの内容を詳細に記載した Issue を提出することをお勧めします。これは、あなたの修正自体は受け付けないがバグの追跡はしたいという場合に役立ちます。

初めてのプルリクエスト
はじめてのプルリクエストに取り組んでみますか？ この無料ビデオシリーズから手順を学ぶことができます：
GitHub でオープンソースプロジェクトに貢献する方法
あなたが新しい試みをする上で、貢献プロセスに慣れるのを助けるために、私たちは比較的影響範囲の少ないバグを含む **good first issues** のリストを持っています。これはオープンソースプロジェクトへの貢献の入門に最適です。

Issue を解決することにした場合、誰かがすでに修正に取り組んでいる場合に備えて、コメントスレッドを必ず確認してください。現時点で誰も作業していない場合は、他の人が誤って重複して作業をしないように、作業する予定であることを示すコメントを残してください。
誰かが取り組むと宣言した Issue が 2 週間以上放置されている場合、それを引き継ぐことは問題ありませんが、その場合もコメントを残すべきです。

プルリクエストを送信する

コアチームはプルリクエストを監視しています。プルリクエストをレビューしてマージするか、変更を要求するか、説明付きでクローズします。Facebook.com 内部で使用法の検討が必要な可能性のある API の変更については、対応が遅くなることがあります。プロセス全体を通して最新情報とフィードバックを提供するよう最善を尽くします。
プルリクエストを送信する前に、以下のプロセスが行われているか確認してください：

1. リポジトリ をフォークして `master` から新しいブランチを作成します。
2. `yarn` コマンドをリポジトリルートで実行します。
3. バグを修正したり、テストが必要なコードを追加した場合は、テストを追加してください。
4. テストスイートが通ることを確認してください (`yarn test`)。ヒント： `yarn test --watch TestName` コマンドは開発時に役立ちます。
5. `yarn test-prod` コマンドを本番環境でテストするために実行します。これは `yarn test` と同じオプションをサポートします。
6. デバッガが必要な場合は `yarn debug-test --watch TestName` を実行し `chrome://inspect` を開き “Inspect” を押してください。
7. prettier でコードをフォーマットしてください (`yarn prettier`)。
8. リントを行ってください (`yarn lint`)。ヒント： `yarn linc` は変更されたファイルのみに適用できます。
9. Flow による型チェックを行ってください (`yarn flow`)。
10. まだの場合は、先に CLA (Contributor License Agreement) の提出を済ませてください。

Contributor License Agreement (CLA)

あなたのプルリクエストを受け付けるために、Contributor License Agreement (CLA) の提出を行って頂く必要があります。これは一度だけ行えば良いので、あなたが他の Facebook オープンソースプロジェクトで既に完了させている場合は必要ありません。初めてプルリクエストを送信する場合は、CLA 提出を完了させたことをお知らせください。そうすれば私たちは GitHub のユーザ名と照らし合わせてチェックを行います。
ここで CLA を完了させてください。

貢献の前提条件

- Node v8.0.0+ と、Yarn v1.2.0+ がインストールされていること。
- `gcc` がインストールされている、または必要に応じたコンパイラをインストールすることができること。依存関係の中にはコンパイルステップが必要なものもあります。OS X では Xcode のコマンドラインツールが役立つでしょう。Ubuntu では `apt-get install build-essential` コマンドで必要なパッケージをインストールできます。他の Linux ディストリビューションでも似たようなコマンドで実現できるでしょう。Windows では追加の手順が必要になり、詳しくは `node-gyp` installation instructions を参照してください。
- Git について精通していること。

開発ワークフロー

React リポジトリをクローンしたあと、`yarn` コマンドで依存関係のパッケージを取得してください。 そうすれば、いくつかのコマンドが実行可能になります：

- `yarn lint` コードスタイルをチェックします。
- `yarn linc` `yarn lint` と似ていますが、変更されたファイルのみをチェックするのでこちらの方が速いです。
- `yarn test` 全てのテストスイートを実行します。
- `yarn test --watch` 対話式のテストウォッチャーを実行します。
- `yarn test <pattern>` 指定したパターンにマッチするファイルのみテストを実行します。
- `yarn test-prod` 本番環境でテストを実行します。`yarn test` と同じオプションをサポートしています。
- `yarn debug-test` は `yarn test` に似ていますがデバッガ付きです。`chrome://inspect` を開き “Inspect” を押してください。
- `yarn flow` Flow による型チェックを行います。
- `yarn build` 全てのパッケージを含む `build` フォルダを作成します。
- `yarn build react/index,react-dom/index --type=UMD` React と ReactDOM だけの UMD ビルドを作成します。

`yarn test`（またはそれに近い上記のコマンド）を実行し、あなたの行った変更によって何らかの異常を引き起こしていないか確認することをお勧めします。とはいえ実際のプロジェクトで自分の React のビルドを使ってみることも役に立つでしょう。
まず `yarn build` を実行します。これによってビルド済みのバンドルファイルが `build` フォルダ内に作られ、同時に `build/packages` 内に npm パッケージも用意されます。
変更を試す一番簡単な方法は `yarn build react/index,react-dom/index --type=UMD` を実行し、`fixtures/packaging/babel-standalone/dev.html` を開くことです。このファイルは `build` フォルダの `react.development.js` を既に使用しているので、変更が反映されます。
あなたの加えた変更を既存の React プロジェクトで試したい場合、`build/dist/react.development.js`、`build/dist/react-dom.development.js`、もしくは他のビルドされたファイルをあなたのアプリケーションにコピーして安定版の代わりに使用することができます。もし、npm 版の React を使用している場合は `react` と `react-dom` を依存関係から削除し、`yarn link` を使用してそれらがローカルの `build` を指すようにしてください：

```
cd ~/path_to_your_react_clone/build/node_modules/react
yarn link
cd ~/path_to_your_react_clone/build/node_modules/react-dom
yarn link
cd /path/to/your/project
yarn link react react-dom
```

`yarn build` を React フォルダで実行するたびに、あなたのプロジェクトの `node_modules` フォルダに更新されたバージョンが現れるでしょう。その後、プロジェクトをビルドし直して変更を試すことができます。
ただしプルリクエストにあなたの新機能に応じたユニットテストを含めることは必須です。これによって将来あなたのコードを壊してしまわないことが担保されます。

スタイルガイド

Prettier と呼ばれる自動コードフォーマッタを使います。 `yarn prettier` コマンドをコードを変更した後に実行してください。
そうすれば、後はリンターがあなたのコードに存在するほとんどの問題を捕らえるでしょう。 自分の書いたコードのスタイルをチェックしたい場合は単に `yarn linc` を実行してください。
しかしながら、リンターでもチェックしきれないいくつかのスタイルがあります。何か分からないことがあれば Airbnb's Style Guide が正しい方向に導いてくれるでしょう。

紹介ビデオ

React への貢献方法について紹介している この短いビデオ（26 分）にあなたは興味があるかもしれません。

ビデオの概要：

- 4:12 - ローカルで React のテストとビルドを行う
- 6:07 - プルリクエストの作成と送信
- 8:25 - コードを整理する
- 14:43 - React npm レジストリ
- 19:15 - React に新しい機能を追加する

React へ初めて貢献することについてのよりリアルな**感覚**については、この面白い ReactNYC talk をチェックしてください。

Request for Comments (RFC)

バグ修正やドキュメンテーションの改善を含む多くの変更は、通常の GitHub プルリクエストのワークフローを通して行われレビューされます。
ただし、いくつかの "大きめの" 変更は、多少の設計プロセスと React コアチームの合意を経ることをお願いします。
"RFC" (request for comments) プロセスは、新機能がプロジェクトに取り込まれるまでの一貫性があり整備された筋道を提供することを目的としています。rfcs リポジトリ を訪れれば貢献することができます。

ライセンス

React に貢献するにあたって、あなたの貢献は MIT ライセンスの元にあることに同意したとみなします。

次のセクション

次のセクション を読んで、コードベースの構成方法について知ることができます。

このページを編集する