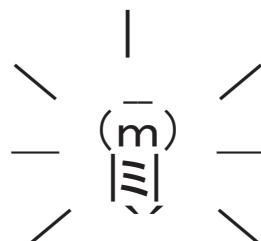


やる夫で学ぶ React、Reduxだお…

知識ゼロから環境構築するお。
Redux-toolkitまで学ぶお。



2021年1月版



まだReduxで消耗してんの? Redux-Toolkitで楽するお□□□

おおおお redux-toolkitやて～ wwwwww
キターネ～(* v 明日からはフロント担当だお…
TypeScript… ちよwww 吹いたwww
! あめでとう👏👏👏👏👏👏👏👏👏👏👏👏👏👏👏👏!
javascript…… 僕も知りたいす!
お勉強は、楽しいお…

やる夫で学ぶ「react-redux」

— redux-toolkit で、簡単・完璧理解だお… —

気分はもう 著

技術書典 10（2021 年夏）新刊

2021 年 6 月 25 日 ver 1.0

■免責

本書は情報の提供のみを目的としています。

本書の内容を実行・適用・運用したことで何が起きようとも、それは実行・適用・運用した人自身の責任であり、著者や関係者はいかなる責任も負いません。

■商標

本書に登場するシステム名や製品名は、関係各社の商標または登録商標です。

また本書では、TM、[®]、[©]などのマークは省略しています。

はじめに

このたびは、「やる夫で学ぶ-Redux- Redux Toolkit は簡単だお…」を、手にしていただき誠にありがとうございます。

本書は、フロントエンド開発で使用される* react * redux(redux-toolkit) を習得するために、開発環境の作成(つまりゼロから)はじめます。

すでに開発環境を整えている方は、

本書は、チュートリアルによくある ToDo リストを 1. Redux なしで作成 2. Redux を導入 3. Redux Toolkit を導入と、書き換えていくことで Redux を用いた「状態管理(アプリケーション全体でのデータ)」を理解してもらえるようになっています。

create-react-app を使用して、ゼロからの手順を解説しています。写経がメンドウな人は、GitHub にあるコードを使ってください。

必要なものは、すべて無料でそろえることができる以下のものです。
* Node.js
* yarn
Microsoft Visual Studio code
* Google Chrome

目次

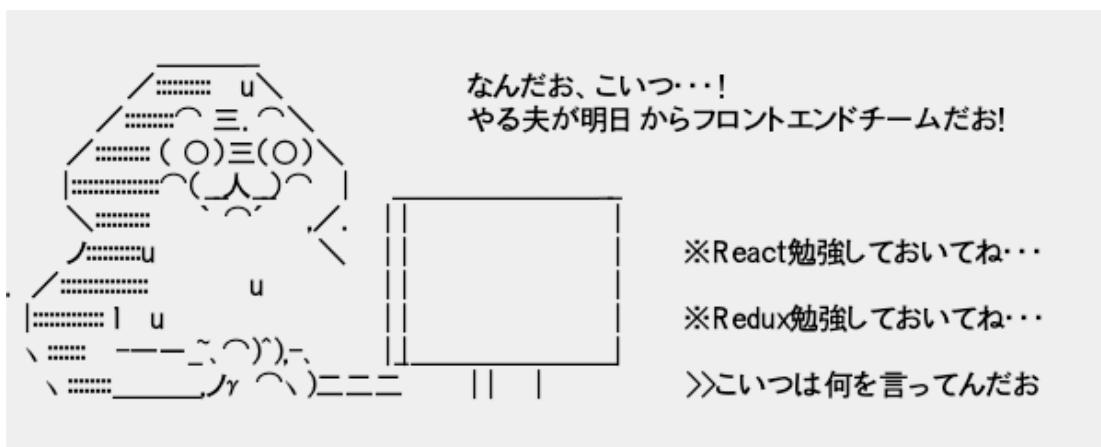
はじめに

i

| | | |
|--------------|---|-----------|
| 第 1 章 | ゼロから始める (開発環境構築) | 1 |
| 1.1 | Node.js | 2 |
| 1.1.1 | Node.js について | 2 |
| 1.1.2 | Node.js のインストールの前に | 8 |
| 1.1.3 | nvm | 9 |
| 1.2 | yarn | 15 |
| 1.2.1 | yarn とは? | 15 |
| 1.3 | Microsoft Visual Studio Code + 拡張機能 | 16 |
| 1.3.1 | VSCODE のインストール | 16 |
| 1.3.2 | VSCODE の拡張機能 | 17 |
| 1.3.3 | Eslint | 18 |
| 1.3.4 | Prettier | 19 |
| 1.4 | Google Chrome + 拡張機能 | 20 |
| 1.4.1 | Google Chrome のインストール | 20 |
| 1.5 | 第 1 章のまとめ | 25 |
| 第 2 章 | スタートプロジェクトの作成 | 26 |
| 2.1 | create-react-app コマンド | 26 |
| 2.2 | アプリケーションを実行 | 28 |
| 2.3 | eslint、prettier のインストールと設定 | 30 |
| 2.4 | eslint、prettier の指摘を修正 | 33 |
| 2.5 | 第 2 章のまとめ | 38 |
| 第 3 章 | Todo リストの作成 (Reactのみ) | 39 |
| 3.1 | React とは? | 39 |
| 3.2 | 表示するデータの型を決める | 40 |
| 3.3 | データ表示画面 | 41 |
| 3.4 | React hooks を使用して、データの追加・編集・削除 | 42 |

第 1 章

ゼロから始める(開発環境構築)



▲図 1.1: やる夫が、突然の移動を命じられたお！

本章では、React、Redux の開発環境を作成します。

「なぜ、これらが必要なのか？」

は、とりあえず置いといて

- * Node.js
- * yarn
- * Microsoft Visual Studio Code + 拡張機能
- * Google Chrome + 拡張機能

をインストールしてください。これらは、すべて無償で提供されています。

なお、これらの準備が整っている方は、本章を読み飛ばしていただいてもかまいません。

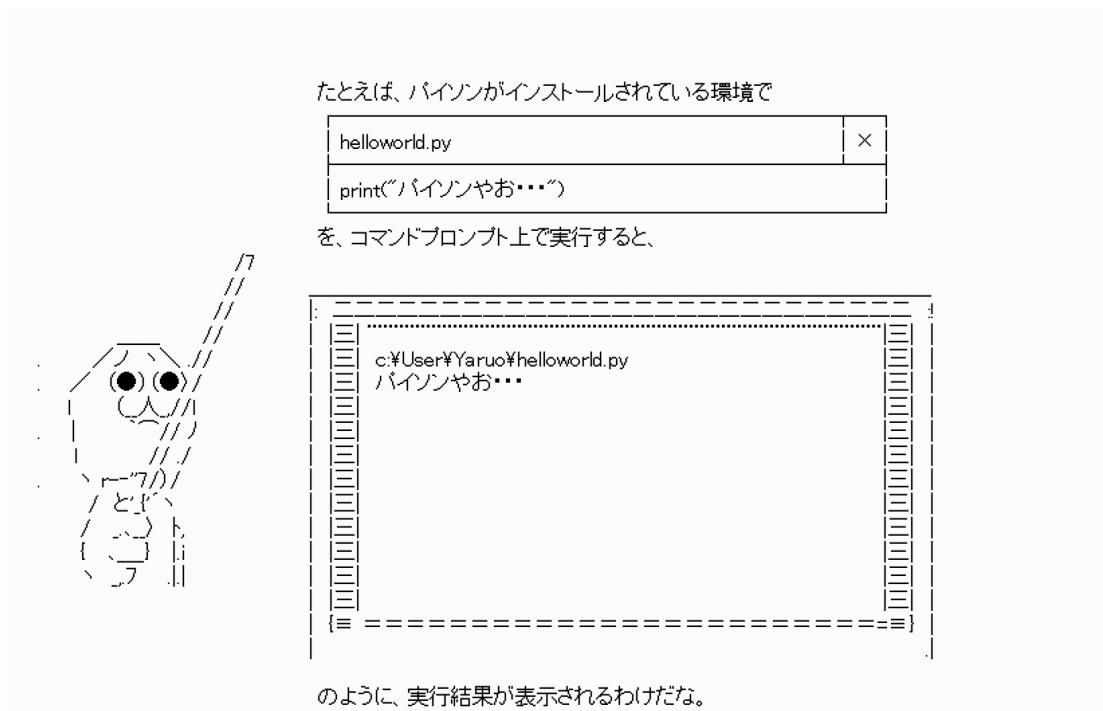
1.1

Node.js

♡| 1.1.1 Node.jsについて

Node.jsとは?

「Node.js」は、通常ブラウザ上で実行される JavaScript をサーバや PC 上で実行できるようする「**JavaScript 実行環境**」です。たとえば、Windows に Python をインストールすると「python.exe」を使い python ファイルを Windows 上で実行できます。



▲図 1.2: python を実行

同じように、**Node.js** をインストールすると、「node.exe」を使い JavaScript ファイルを実行できます。例として、Node.js のドキュメント (Guides) 「How do I start with Node.js after I installed it?」^{*1}にある以下のスクリプトを実行してみましょう。

^{*1} <https://nodejs.org/en/docs/guides/getting-started-guide/>

▼ リスト 1.1:

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

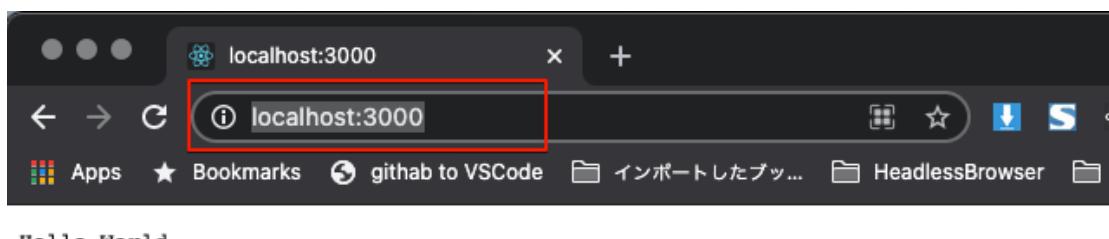
server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

このスクリプトを「app.js」のファイル名で保存し、実行します。

▼ リスト 1.2: app.js を実行

```
> node app.js
Server running at http://127.0.0.1:3000/
```

ブラウザを開き、`http://127.0.0.1:3000`へアクセスすると、ブラウザに実行結果が表示されます。



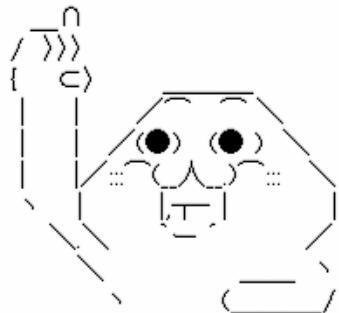
▲ 図 1.3: app.js の実行

通常のHTMLに埋め込まれたJavaScriptをブラウザからで実行すると、OS上の機能を使用する(たとえば、ファイルの書込み・読み込み)などは制限されますが、Node.jsで実行するとOSの機能も使用できます。

詳しくは、本家の「Node.jsとは^{*2}」を参照してください。

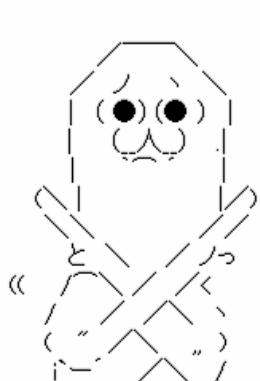
^{*2} <https://nodejs.org/ja/about/>

Node.jsについて



では、さっそくNode.jsを手に入れるお！

▲図1.4: Node.jsをインストールするお！



慌てるでない!

Node.jsには、

- ・最新版
- ・長期サポート版

の2つと

- ・過去にリリースされたバージョン

がある。

ライブラリなどでは、

Node.jsのバージョンが指定されているものもある。

▲図1.5: Node.jsのバージョンには注意

では、Node.jsの本家トップページ:<https://nodejs.org/ja/>へアクセスします。



▲図 1.6: Node.js トップページ

ここでダウンロード可能なのは、「14.17.0 LTS(Long Term Support) 推奨版」と「16.3.0 最新版」^{*3}の2つがあります。

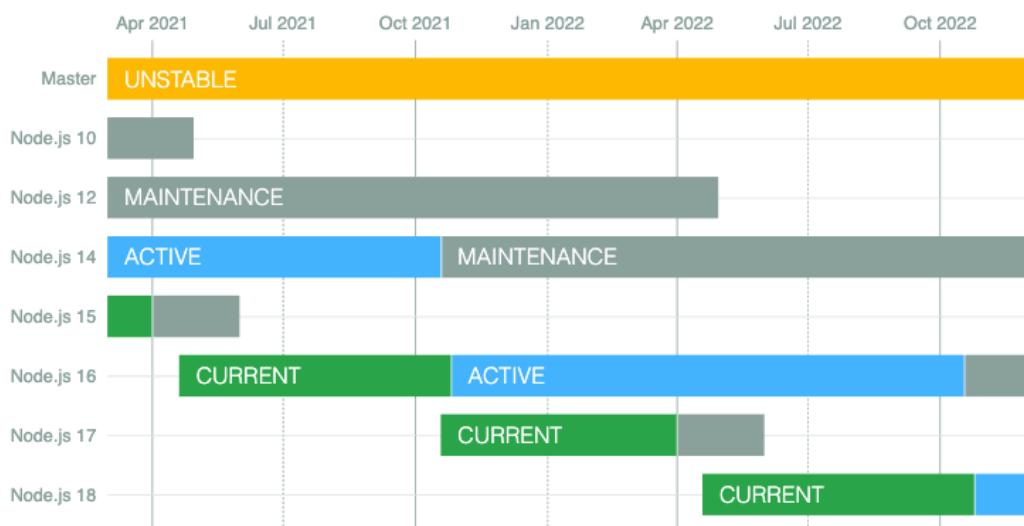
LTS版、最新版は以下のロードマップにより更新されます。

^{*3} 2021/06/10 現在

リリース

[GitHub上で編集](#)

Node.js のメジャーバージョンは 6 ヶ月間 現行リリースの状態になり、ライブラリの作者はそれらのサポートを追加する時間を与えられます。6 ヶ月後、奇数番号のリリース (9, 11 など) はサポートされなくなり、偶数番号のリリース (10, 12 など) はアクティブ LTS ステータスに移行し、一般的に使用できるようになります。LTS のリリースステータスは「長期サポート」であり、基本的に重要なバグは 30 ヶ月の間修正されることが保証されています。プロダクションアプリケーションでは、アクティブ LTS またはメンテナンス LTS リリースのみを使用してください。



| リリース | ステータス | コードネーム | 初回リリース | アクティブ LTS 開始 | メンテナンス LTS 開始 | サポート終了 |
|------|------------|---------|------------|--------------|---------------|------------|
| v12 | メンテナンス LTS | Erbium | 2019-04-23 | 2019-10-21 | 2020-11-30 | 2022-04-30 |
| v14 | アクティブ LTS | Fermium | 2020-04-21 | 2020-10-27 | 2021-10-19 | 2023-04-30 |
| v16 | 現行 | | 2021-04-20 | 2021-10-26 | 2022-10-18 | 2024-04-30 |
| v17 | 次期 | | 2021-10-19 | | 2022-04-01 | 2022-06-01 |
| v18 | 次期 | | 2022-04-19 | 2022-10-25 | 2023-10-18 | 2025-04-30 |

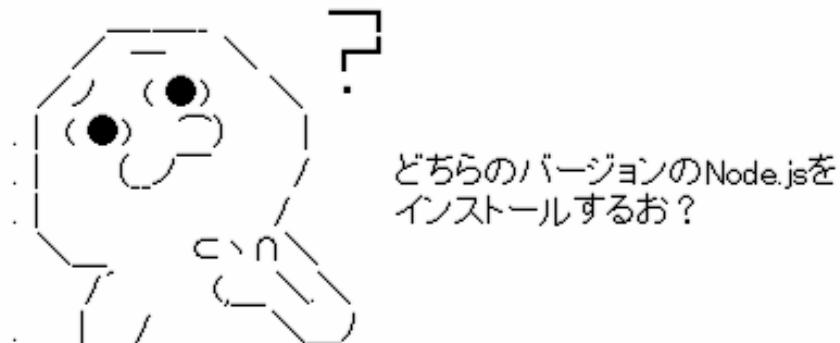
日程は変更される場合があります。

▲図 1.7: Node.js ロードマップ

Node.js の Releases:<https://nodejs.org/ja/about/releases/> にあるように、Node.js は、各年の 4月、10月にリリースされ、* Current * Active * Maintenance のフェーズを経ますが、メジャーバージョン番号が偶数のものだけが、Active 期間を経て長期サポートされます。

上記トップページにある Node.js 14 は、2023/4/30までの長期サポートとなります。実際のプロジェクトで使用する場合は、よほどの理由がない限りは最新の LTS 版を使用します。

♡| 1.1.2 Node.js のインストールの前に



Node.js は、ロードマップにより定期的にバージョンアップされます。又、Node.js 自体の不具合の修正などでマイナーバージョンアップも行われます。

プロジェクト開発中のマイナーバージョンアップでも検証が必要になりますが、メジャーバージョンアップの場合はさらに大きな検証が必要になります。場合によっては、ソースコードの大幅な改良をしなければならなくなります。

それを避けるためにも、プロジェクト毎に Node.js のバージョンは固定して開発します。

通常は、OS にインストールできる Node.js のバージョンはひとつですが、長期にわたるサポートや新規プロジェクト開発のためには、複数の Node.js のバージョンを切り替えて使用できるしくみを用意しましょう。

私が使用しているのは、nvm(node version manager):<https://github.com/nvm-sh/nvm>です。いろいろなバージョンの Node.js を、簡単にインストール・アンインストール・切替ができます。

♥| 1.1.3 nvm

nvm(node version manager)を使えば、複数バージョンのNode.jsを1台のPCにインストールし、バージョンを切り替えることが簡単にできます。

GitHub上のnvmは、Shellscript(sh, dash, zsh, bash)上で動作するため、Linux(UNIX系)、macOSにインストールできます。

Windows版:<https://github.com/coreybutler/nvm-windows>は、別な方がGitHub上で公開されています。コマンドが本家と少し違いますが、複数バージョンのインストール・バージョンの切り替えなど機能は問題ありません。

nvmのインストール

macOS

お使いのTerminalから以下のコマンドを実行してください。

▼リスト1.3: nvmのインストール

```
>curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.37.2/install.sh | bash
```

インストール完了後には、.zshrcへ以下を追加してください。

▼.zshrcへ追加

```
export NVM_DIR="`( [ -z "${XDG_CONFIG_HOME-}" ] && printf %s "${HOME}/.nvm" || printf %s "${XDG_CONFIG_HOME}/nvm")`"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
```

Windows

nvm-windowsのリリースページ:<https://github.com/coreybutler/nvm-windows/releases/tag/1.1.7>より、最新版をダウンロードしインストールしてください。

nvmの使い方

macOSではターミナルを起動し、Windowsではコマンドプロンプト、または、Windows Terminalを起動してください。

nvmとnvm-windowsでは、コマンドが少し違いますが、「nvm --help」を入力することで使用できるコマンドが表示されます。

▼Mac OSX

```
> nvm --help
```

```
Node Version Manager (v0.37.2)
```

```
←中略
```

```
Example:
```

```
nvm install 8.0.0
```

```
Install a specific version number
```

```

nvm use 8.0                      Use the latest available 8.0.x release
nvm run 6.10.3 app.js             Run app.js using node 6.10.3
nvm exec 4.8.3 node app.js        Run `node app.js` with the PATH pointing to
>o node 4.8.3
nvm alias default 8.1.0           Set default node version on a shell
nvm alias default node            Always default to the latest available node
>e version on a shell

←中略

Note:
  to remove, delete, or uninstall nvm - just remove the `$NVM_DIR` folder (usually
> `~/.nvm`)

```

もし、32bit 版 Windows をお使いの場合には、インストールの際に 32bit 版を指定する「32」をコマンドの最後につけてください。

▼ windows

```

PS C:\Users\inabakazuya> nvm --help

Running version 1.1.7.

Usage:

  nvm arch                  : Show if node is running in 32 or 64 bit mode.
  nvm install <version> [arch] : The version can be a Node.js version or "latest" >
>for the latest stable version.
                                         Optionally specify whether to install the 32 or 6<
>4 bit version (defaults to system arch).
                                         Set [arch] to "all" to install 32 AND 64 bit vers<
>ions.
                                         Add --insecure to the end of this command to bypass
> SSL validation of the remote download server.
  nvm list [available]       : List the Node.js installations. Type "available" >
>at the end to see what can be installed. Aliased as ls.
<中略
  nvm uninstall <version>    : The version must be a specific version.
  nvm use [version] [arch]     : Switch to use the specified version. Optionally s<
>pecify 32/64bit architecture.
                                         nvm use <arch> will continue using the selected v<
>ersion, but switch to 32/64 bit mode.
  nvm root [path]            : Set the directory where nvm should store differen<
>t versions of Node.js.
                                         If <path> is not set, the current root will be di<
>spayed.
  nvm version                : Displays the current running version of nvm for W<
>indows. Aliased as v.

```

インストール可能な Node.js を表示

まずは、インストール可能な Node.js のバージョンを表示してみます。macOS の場合には、古いバージョンから最新バージョンまでが表示されます。

▼ Mac OSX

```
>nvm ls-remote
←古いバージョンから全て表示されるので中略
v14.13.1
v14.14.0
v14.15.0 (LTS: Fermium)
v14.15.1 (LTS: Fermium)
v14.15.2 (Latest LTS: Fermium)
v15.0.0
v15.0.1
v15.1.0
v15.2.0
v15.2.1
v15.3.0
v15.4.0
```

一方、Windows の場合には、表形式で表示されます。新しいバージョンが上に表示されます。

▼ Windows

```
PS C:\Users\inabakazuya> nvm list available
```

| CURRENT | LTS | OLD STABLE | OLD UNSTABLE |
|---------|---------|------------|--------------|
| 15.4.0 | 14.15.2 | 0.12.18 | 0.11.16 |
| 15.3.0 | 14.15.1 | 0.12.17 | 0.11.15 |
| 15.2.1 | 14.15.0 | 0.12.16 | 0.11.14 |
| 15.2.0 | 12.20.0 | 0.12.15 | 0.11.13 |
| 15.1.0 | 12.19.1 | 0.12.14 | 0.11.12 |
| 15.0.1 | 12.19.0 | 0.12.13 | 0.11.11 |
| 15.0.0 | 12.18.4 | 0.12.12 | 0.11.10 |
| 14.14.0 | 12.18.3 | 0.12.11 | 0.11.9 |
| 14.13.1 | 12.18.2 | 0.12.10 | 0.11.8 |
| 14.13.0 | 12.18.1 | 0.12.9 | 0.11.7 |
| 14.12.0 | 12.18.0 | 0.12.8 | 0.11.6 |
| 14.11.0 | 12.17.0 | 0.12.7 | 0.11.5 |
| 14.10.1 | 12.16.3 | 0.12.6 | 0.11.4 |
| 14.10.0 | 12.16.2 | 0.12.5 | 0.11.3 |
| 14.9.0 | 12.16.1 | 0.12.4 | 0.11.2 |
| 14.8.0 | 12.16.0 | 0.12.3 | 0.11.1 |
| 14.7.0 | 12.15.0 | 0.12.2 | 0.11.0 |
| 14.6.0 | 12.14.1 | 0.12.1 | 0.9.12 |
| 14.5.0 | 12.14.0 | 0.12.0 | 0.9.11 |
| 14.4.0 | 12.13.1 | 0.10.48 | 0.9.10 |

```
This is a partial list. For a complete list, visit https://nodejs.org/download/release
```

Node.js 最新 LTS 版をインストール

それでは、最新の LTS 版をインストールします。インストールは、Mac、Windows とも、「nvm install インストールするバージョン番号」でインストールできます。

▼ Mac OSX

```
> nvm install v14.15.2
Downloading and installing node v14.15.2...
Downloading https://nodejs.org/dist/v14.15.2/node-v14.15.2-darwin-x64.tar.xz...
#####
> ##### 100.0%
Computing checksum with shasum -a 256
Checksums matched!
Now using node v14.15.2 (npm v6.14.9)
```

▼ Window

```
PS C:\Users\inabakazuya> nvm install v14.15.2
Downloading Node.js version 14.15.2 (64-bit)...
Complete
Creating C:\Users\inabakazuya\AppData\Roaming\nvm\temp

Downloading npm version 6.14.9... Complete
Installing npm v6.14.9...

Installation complete. If you want to use this version, type

nvm use 14.15.2
```

インストールされている Node.js のバージョンの確認

インストールされている Node.js のバージョンは、「nvm ls」で表示させ確認できます。

▼ Mac OSX

```
$ > nvm ls
      v8.17.0
      v12.19.0
      v14.15.0
->    v14.15.2
      system
default -> v12.18.3
iojs -> N/A (default)
unstable -> N/A (default)
node -> stable (-> v14.15.2) (default)
stable -> 14.15 (-> v14.15.2) (default)
lts/* -> lts/fermium (-> v14.15.2)
lts/argon -> v4.9.1 (-> N/A)
lts/boron -> v6.17.1 (-> N/A)
lts/carbon -> v8.17.0
lts/dubnium -> v10.23.0 (-> N/A)
lts/erbium -> v12.20.0 (-> N/A)
lts/fermium -> v14.15.2
```

▼ Windows

```
PS C:\Users\inabakazuya> nvm ls

  14.15.2
  14.15.1
  12.13.1
* 8.16.1 (Currently using 64-bit executable)
```

使用する Node.js のバージョン切り替え

先ほどの「インストールされている Node.js のバージョン表示」で、現在使われている

Node.js のバージョンも表示されています。使用する Node.js のバージョンを変更する場合には、「nvm use 使用するバージョン番号」で切り替えます。

▼ Node.js のバージョン確認と切替

```
$ > node -v  
v14.15.2  
[~]  
$ > nvm use v12.18.3  
Now using node v12.18.3 (npm v6.14.6)  
[~]  
$ > node -v  
v12.18.3
```

以上で、複数のバージョンの Node.js を切り替えて使える環境が構築できました。

1.2 yarn

♡| 1.2.1 yarn とは？

yarn は、node プロジェクトにおいて必要な機能を簡単に追加できるようにする npm(node package manager) 互換の package management tool(パッケージ管理ツール) です。

npm と比較して* パッケージのインストールが早い* コマンドが npm よりも簡潔* npm よりもパッケージのバージョン管理がしっかりしているなどの利点があります。

yarn のインストール

Node.js をインストールした時点で、Node.js のパッケージ管理ツールである「npm」がインストールされています。

▼ npm のバージョン確認

```
$ > npm -v  
6.14.6
```

yarn をインストールするために、この「npm」コマンドを使用します。

▼ yarn のインストール

```
$ > npm install -g yarn
```

コマンドオプションの「-g」は、global 指定で PC 全体でコマンドが使用できるようになります。-g オプションなしでは、そのプロジェクトでのみ有効となります。

yarn の使い方は、今後使う機会毎に説明します。

1.3

Microsoft Visual Studio Code + 拡張機能

Microsoft 社が無料で提供している「テキストエディタ」です。Electron:<https://www.electronjs.org/> をベースにしたオープンソースで開発されています。

Electron は、Github 社が「Atom(テキストエディタ)」を開発するために構築したフレームワークで、HTML・CSS・JavaScript を使用して、Windows、Mac、Linux のマルチプラットフォームで動作するアプリケーションを開発することができます。

Visual Studio Code(以後は、VSCode と表記します。)は、コードを記述する「テキストエディタ」として非常に優秀ですが、拡張機能(Google Chrome や Firefox などのブラウザ同様に拡張機能が多くの開発者により公開されています。)を追加することで JavaScript 以外の言語(C#、python など)でも使えます。

デバッグなども行えるため、Web 開発を行う上出は、事実上の標準と言っても良いかもしれません。有料では、Jetbrains 社:<https://www.jetbrains.com/> の Webstorm がありますが、今回は、無償の VSCode を使用します。

♡| 1.3.1 VSCode のインストール

VSCode のインストールは、

本家サイト <https://code.visualstudio.com/>

から、ダウンロード後インストールしてください。

または、以下の方法でもインストールすることができます。

Windows

パッケージマネージャーに「Chocolatey」をお使いの方は、

▼ Chocolatey

```
choco install vscode
```

にて、インストールすることができます。

Mac OSX

パッケージマネージャーに「brew」をお使いの方は、

▼ Homebrew

```
$ > brew update  
$ > brew cask install visual-studio-code
```

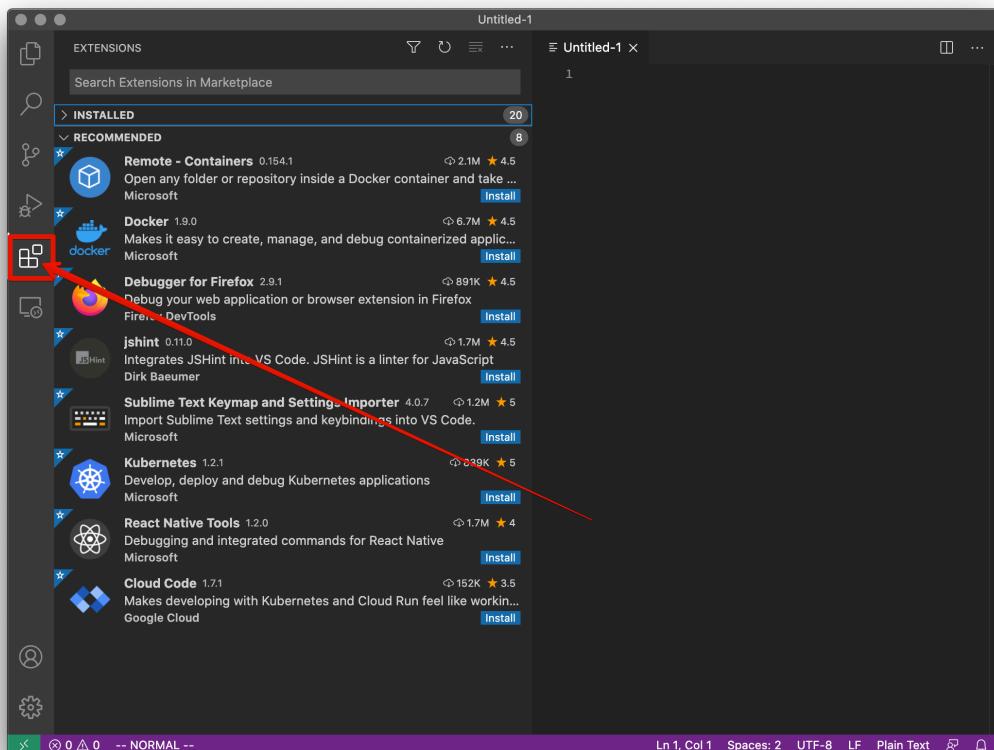
にて、インストールすることができます。

1.3.2 VSCode の拡張機能

VSCode は、プラグイン形式で拡張機能を追加することができます。

React、Redux を使用したプロジェクトでは、以下をインストールすると便利です。

VSCode を起動し、左ツールバーの拡張機能アイコンをクリックします。

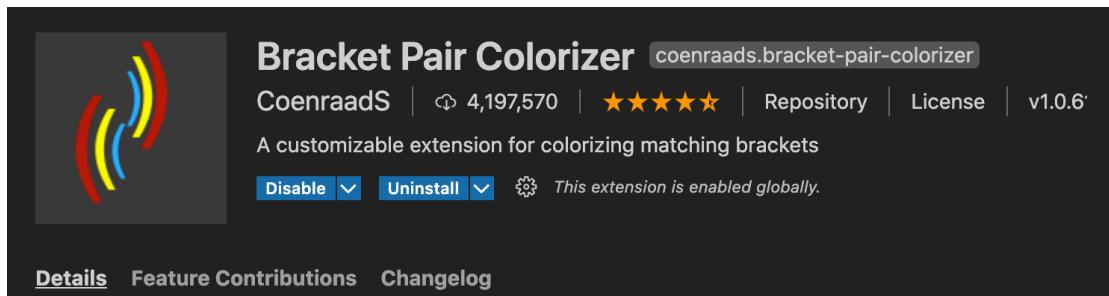


▲図 1.8: VSCode の拡張機能

ここで検索窓に拡張機能の名前、キーワードを入力して検索します。

Bracket Pair Colorizer

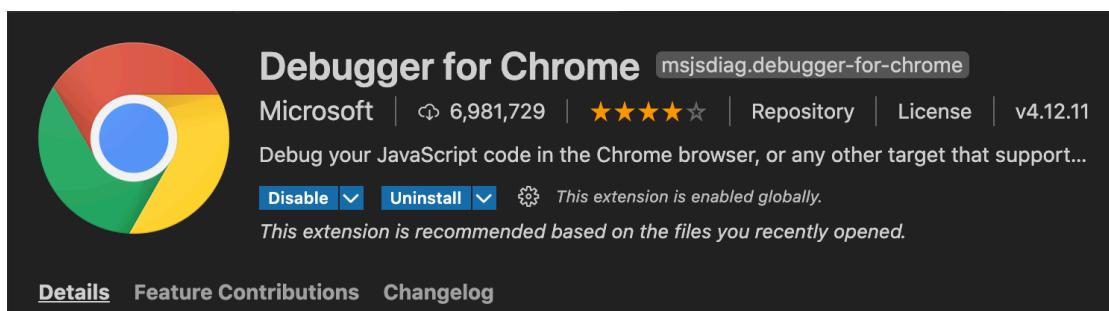
対になるカッコ「{」、「}」を同じ色で表示してくれます。



▲図1.9: Beautify

Debugger for Chrome

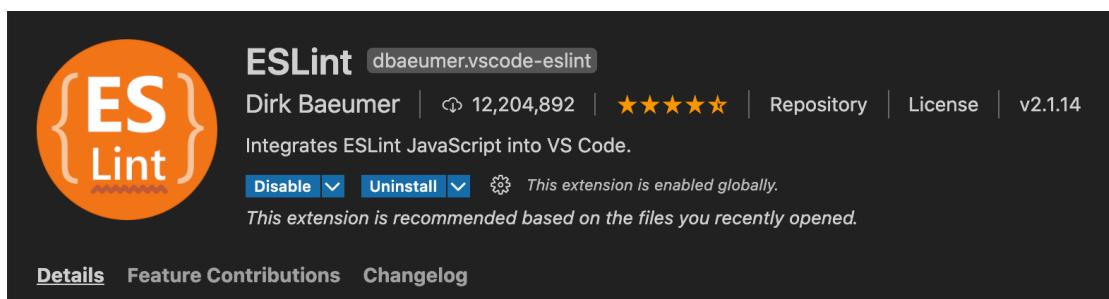
デバッグの際に、PCにインストールされているChromeを自動で起動してくれます。ChromeのDevToolsは、非常に強力です。また、ChromeにもReact、Redux用の拡張機能を追加すると更に便利になります。



▲図1.10: Beautify

♥| 1.3.3 Eslint

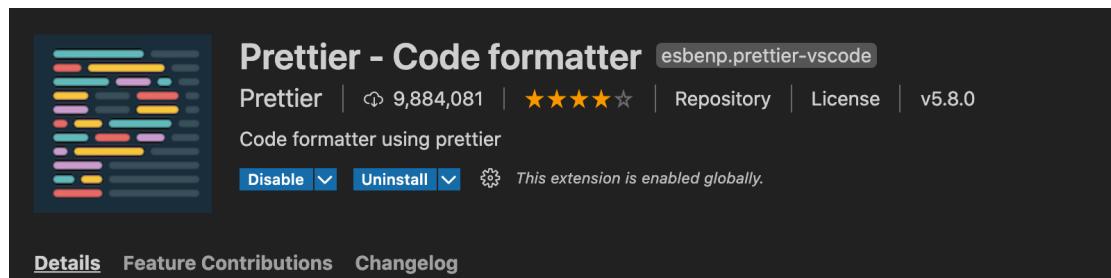
コード記法の間違いを指摘・修正してくれます。



▲図1.11: Beautify

♥| 1.3.4 Prettier

Eslint と同じように、コード記法の間違いの指摘・修正やコードフォーマットを行います。Eslint を合わせて使うと最強です。



▲図 1.12: Beautify

1.4 Google Chrome + 拡張機能

既に、ご存じだと思いますが、ブラウザシェアで1位になっているGoogle社が提供するブラウザです。

PCへインストールされていない方は、

♡ 1.4.1 Google Chrome のインストール

Google Chrome:<https://www.google.com/intl/ja/chrome/>



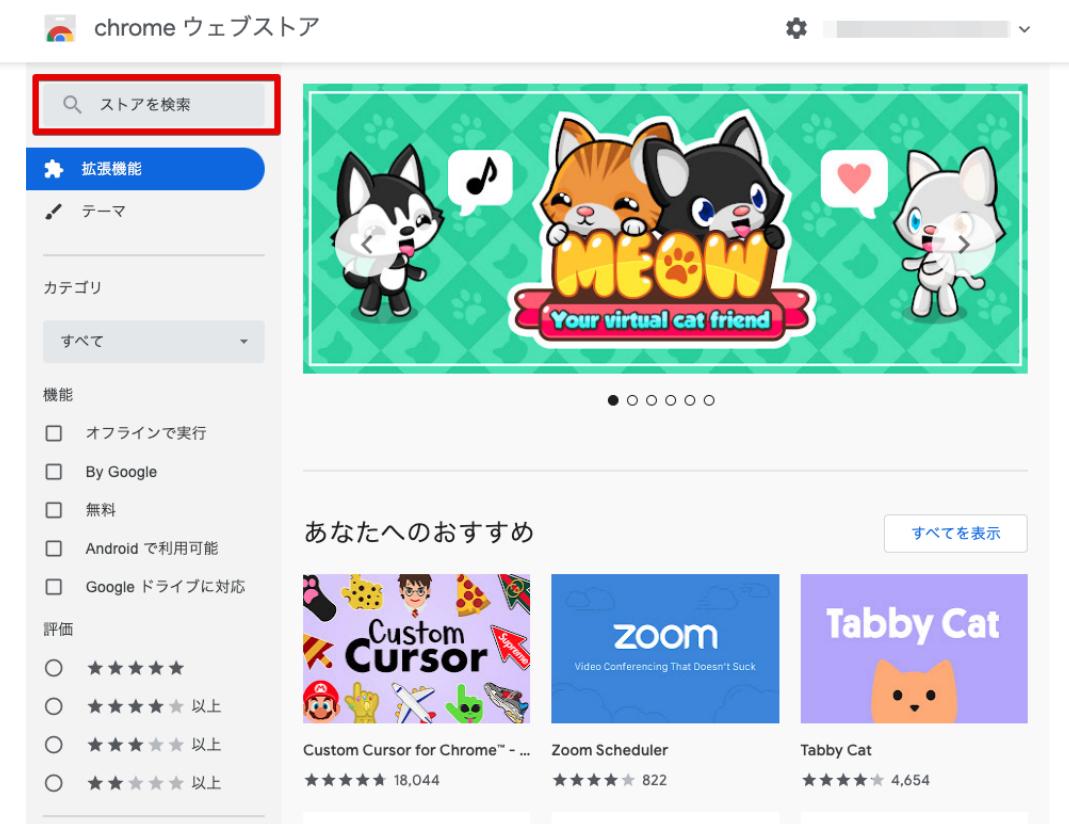
▲図 1.13: Google Chrome

==== Google Chrome の拡張機能こちらも、VSCode と同様に拡張機能を追加することで、更に便利に使うことができます。

React、Redux の開発では、以下の拡張機能は必須と言っても良いほどです。

拡張機能のインストールは、chrome ウェブストアで検索してください。

chrome ウェブストア:<https://chrome.google.com/webstore/category/extensions?hl=ja>

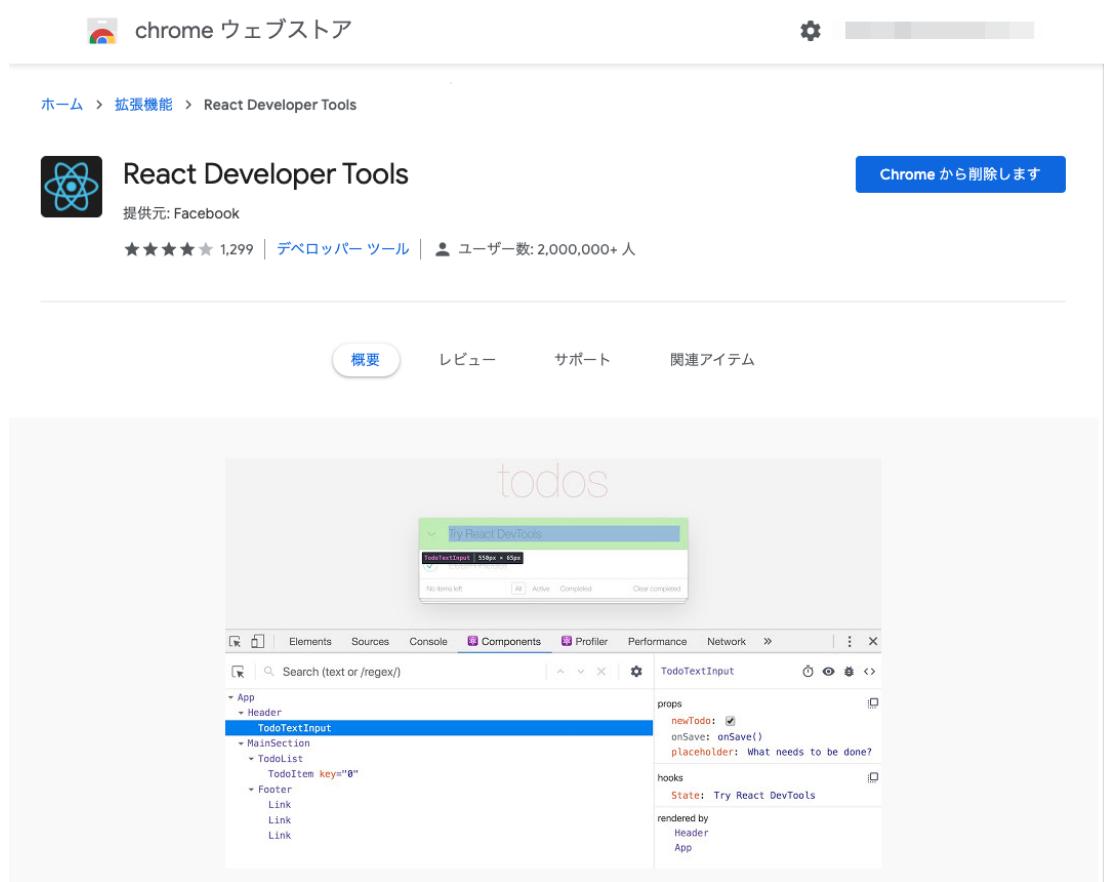


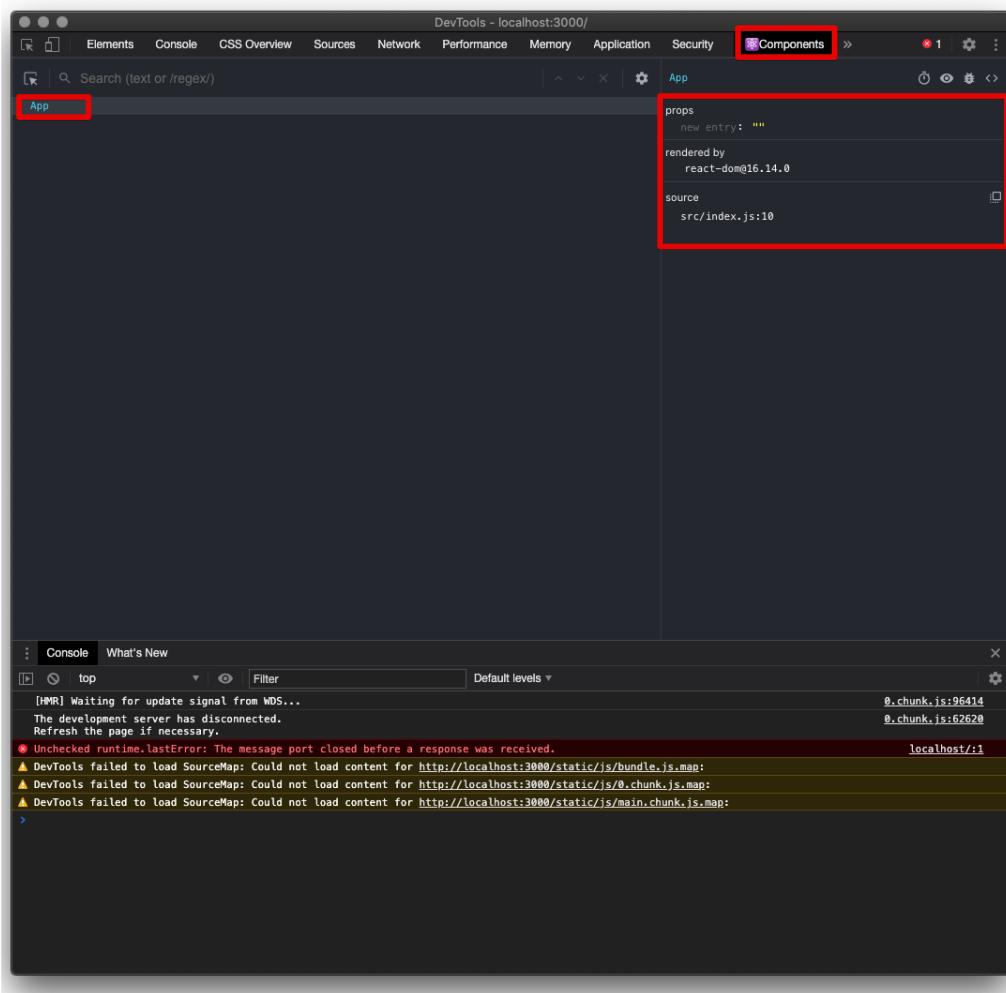
▲図1.14: chrome ウェブストア

React Developer Tools

Reactを使用して作成したページは、最終的にはページ出力用JavaScriptに変換され、ブラウザで表示されるときにはHTMLとして出力されます。この拡張機能を使うと、Google ChromeのDevToolsにComponentsタブが作成され、Props、Stateを確認することができます。

React Developer Tools



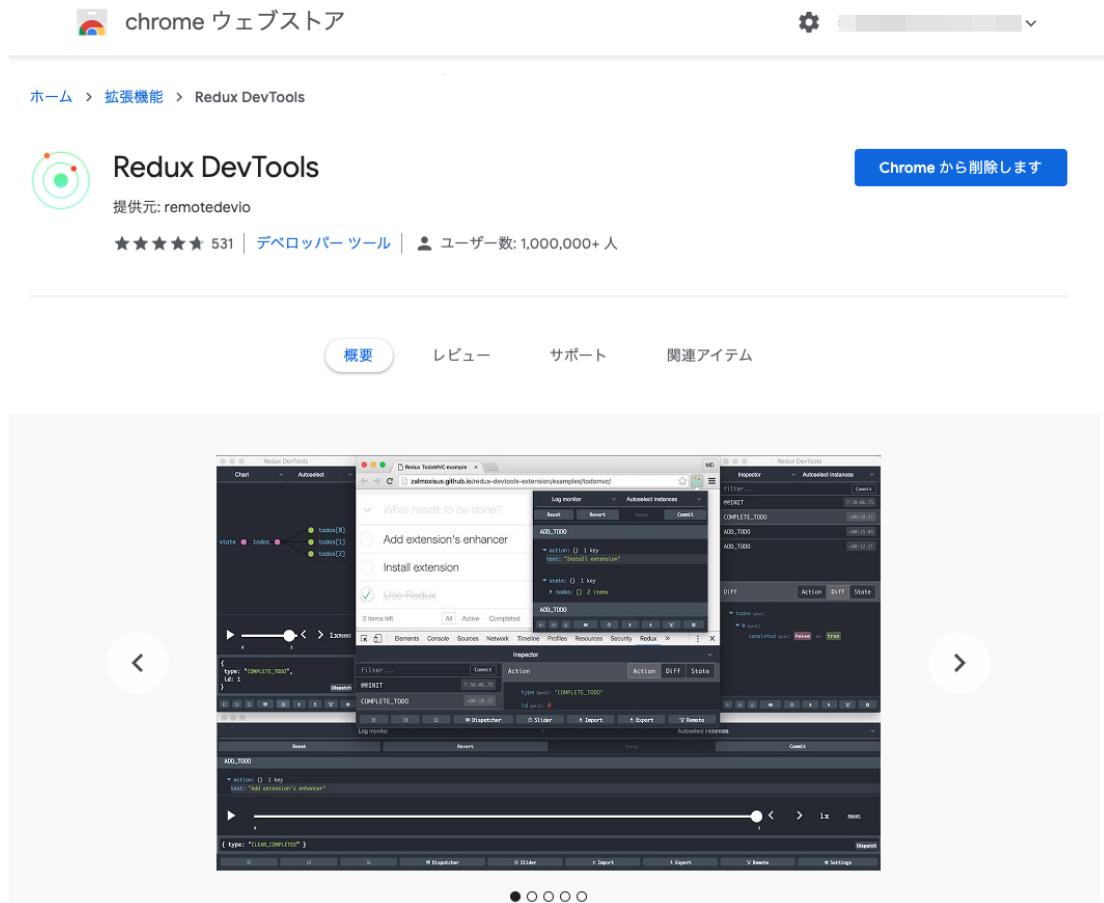


▲図 1.15: React DevTools で App を表示

Redux DevTools

後ほど、Redux の章で改めて説明しますが、「タイムトラベルデバッグ(実行されたアクションを遡る)」が簡単にできるようになります。また、実行されたアクション、変更された State が「新」「旧」とあり、どの部分が変更されたのかも確かめるのも簡単です。

Redux DevTools



1.5 第1章のまとめ

React、Redux の開発環境は、できましたでしょうか？

- nvm
- node
- yarn
- VSCode + 拡張機能
- Google Chrome + 拡張機能

のインストールを完了してください。

第2章

スタートプロジェクトの作成

この章では、「create-react-app」と言う、react アプリケーションの雛形がコマンドひとつで作成できる優れものを使用して、スタート用のアプリケーションを作成し、ブラウザで表示するまでを行います。

また、作成するプロジェクトは、TypeScript を使用します。コード記法の指摘・修正を行えるよう「eslint」、「prettier」の設定も行います。

2.1 create-react-app コマンド

React アプリケーションを作成するためには、

- 「node プロジェクト」に必要な package.json を作成
- react など必要なライブラリのインストール
- 作成したアプリケーションが、古いブラウザでも実行できるようにコードを変換 (babel 使用)
- 出力するファイルを纏める (バンドルする-webpack 使用)

など、react ライブラリのインストール以外にも、babel や webpack をインストールして設定ファイルを作成し、使用するライブラリーによっては、babel のプラグインのインストールや設定など、アプリケーションのコードを書き始める前の作業が大変です。

しかし、「そんなメンドウなことは、やってられない。」ので、すぐにでもコードを書き始めるこことできるスタート用アプリケーションが、react の開発元である Facebook から提供されています。

更に、そのスタート用アプリケーションは、コマンド一発でインストールすることができます。

ターミナルを起動し、プロジェクトフォルダを作成するフォルダへ移動します。

▼ create-react-app でスタート用アプリケーション作成

```
$ > yarn create react-app プロジェクト名 --template typescript
```

で、「プロジェクト名」のフォルダが作成され必要なライブラリがインストールされます。

```
Success! Created yourproject at /Users/tmkz/Documents/Devs/playground/yourprojec>ct
Inside that directory, you can run several commands:

yarn start
  Starts the development server.

yarn build
  Bundles the app into static files for production.

yarn test
  Starts the test runner.

yarn eject
  Removes this tool and copies build dependencies, configuration files
  and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

cd yourproject
yarn start

Happy hacking!
```

で、プロジェクト作成が完了します。

github

ここまでのお仕事は、github にあります。以下のコマンドでクローンしてください。

▼ github から

```
$ > git clone -b 00_create-react-app https://github.com/tmkz/yar>uo.git
```

2.2 アプリケーションを実行

アプリケーションが作成出来ましたので、実行してみます。

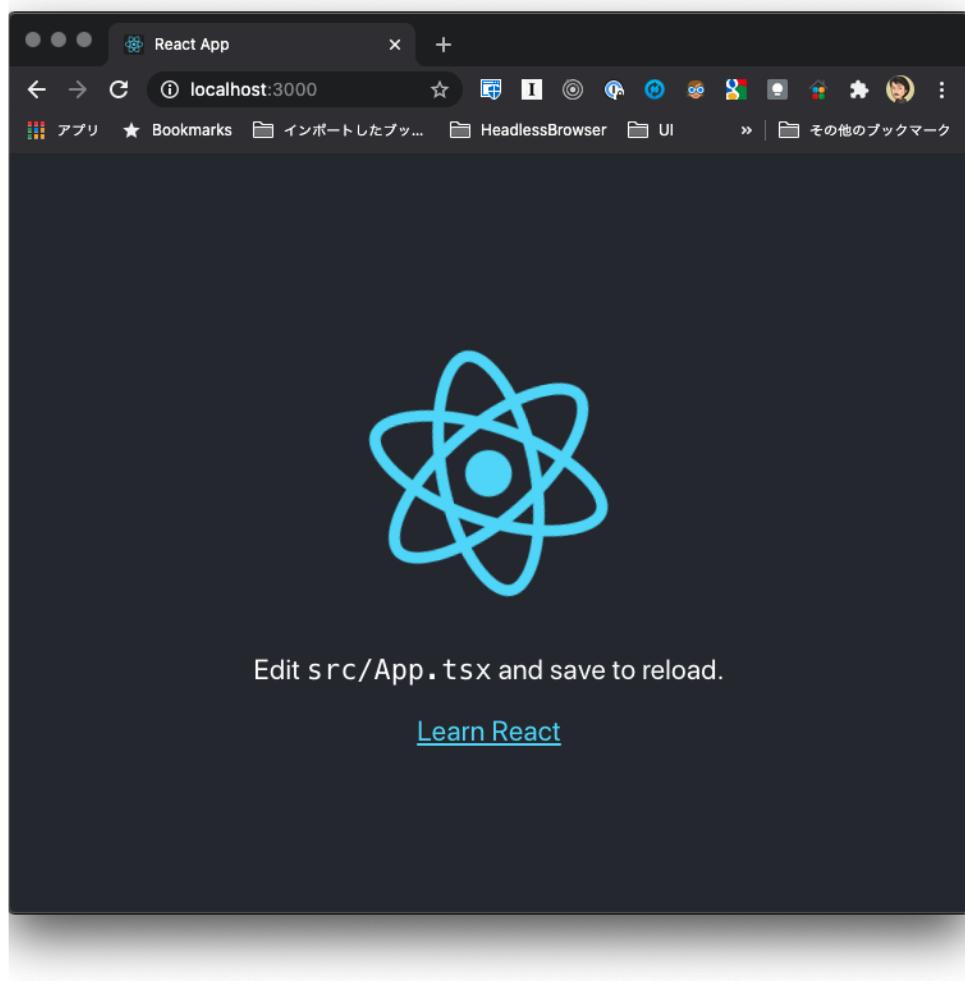
ターミナルに表示されているように、プロジェクトフォルダへ移動し、スタート用のコマンドを入力します。

```
$ > cd プロジェクト名  
$ > yarn start
```

すると、webpack に同梱されている開発用の web server が起動し、デフォルトでは、port:3000 でアプリケーションへアクセスできます。

```
Compiled successfully!  
  
You can now view your project in the browser.  
  
Local:          http://localhost:3000  
On Your Network:  http://192.168.1.10:3000  
  
Note that the development build is not optimized.  
To create a production build, use yarn build.
```

Google Chrome が起動し、http://localhost:3000 へアクセスし以下のページが表示されます。



▲図 2.1: create-react-app の画面

このページが表示されれば成功です。

2.3 eslint、prettier のインストールと設定

あなたがプロジェクトチームに参加しているのであれば、そのチームのコーディング規約に沿うことが求められます。

例えば、

- ファイル、コンポーネント、関数などの命名規則
- 文末のセミコロン
- 改行位置

など、チーム毎、プロジェクト毎にコーディングがあります。

また、JavaScript では、「es5」などの標準仕様がありますので、どの標準仕様に沿ってコーディングするのかも指定されます。

自分でコーディング規約に沿っているつもりでも、レビューを受けるまで知らなかったり、見落としていたりします。

そのため、コーディング規約に沿っているかを、

- 自動でチェック
- 自動で修正
- 修正方法の提案

などをしてくれる便利なものがありますので、積極的に使うようにしましょう。

この便利な機能こそが、第1章で紹介した

- Eslint
- Prettier

になります。

しかし、残念ながら Eslint も Prettier も

- 沢山のプラグインがある
- Eslint と Prettier が競合する部分もある

など、メンドウなことが沢山あります。

オープンソースの世界では、メンドウなことは大抵を先人が解決してくれている、もしくは、解決しようとしている最中です。もし、解決しようとしている最中で出来ることがあれば手伝いたいものです。

この「Eslint、Prettier」のインストール・設定についても以下の方法で簡単に解決ができます。

eslint-config <https://www.npmjs.com/package/@abhijithvijayan/eslint-config>

こちらは、自分で使われている「Eslint、Prettier」のプラグイン・設定を公開し、コマンド一発でインストール完了するようにしてくれています。

▼ コマンド一発でインストール

```
$ > npx install-peerdeps @abhijithvijayan/eslint-config --dev --yarn
```

このプロジェクトは、

- React
- Typescript

を使っていますので、

プロジェクトのルートフォルダに、「.eslintrc.json」ファイルを作成し、以下のように設定をしてください。

私の好みで、Prettier にて

- 必要な箇所には、セミコロンを付ける
- 文字列は、ダブルクオートで囲む

と、しています。

それ以外では、airbnb を元に基本的なルールが設定しております。ルールは、「.eslintrc.json」で上書きできますのでご自分の好みで変更してください。

▼ .eslintrc.json

```
{
  "extends": [
    "@abhijithvijayan/eslint-config/typescript",
    "@abhijithvijayan/eslint-config/react"
  ],
  "parserOptions": {
    "project": "./tsconfig.json"
  },
  "rules": {
    "react/jsx-props-no-spreading": "off",
    "prettier/prettier": [
      "error",
      {
        "semi": true,
        "singleQuote": false
      }
    ]
  }
}
```

Eslint、Prettier の設定が完了しましたので、src フォルダにある「App.tsx」を開いてみ

ると、コーディング規約に外れるものは指摘されています。

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: Shows the project structure with files like package.json, App.tsx, tsconfig.json, etc.
- EDITOR**: The code editor displays `App.tsx` with several red arrows pointing to specific lines:
 - Line 3: `import './App.css';`
 - Line 5: `function App() {`
 - Line 10: `<div className="App">`
 - Line 14: `<header className="App-header">`
 - Line 15: ``
 - Line 19: `<p> Edit <code>src/App.tsx</code> and save to reload.`
 - Line 21: `</p>`
 - Line 22: ``
 - Line 23: `Learn React`
 - Line 24: ``
 - Line 25: `</header>`
 - Line 26: `</div>`
 - Line 27: `export default App;`
- PROBLEMS** tab: Shows a list of ESLint and Prettier errors for `App.tsx`. A red box highlights this tab and the error list:
 - Replace 'react' with "react" eslint(prettier/prettier) [1, 19]
 - Replace './logo.svg' with "./logo.svg" eslint(prettier/prettier) [2, 18]
 - Replace './App.css' with "./App.css" eslint(prettier/prettier) [3, 8]
 - Missing return type on function. eslint(@typescript-eslint/explicit-function-return-type) [5, 1]
 - Missing return type on function. eslint(@typescript-eslint/explicit-module-boundary-types) [5, 1]
- STATUS BAR**: Shows the file name (00_create-react-app), line count (4), and various developer tools.

▲図 2.2: Eslint、Prettier に怒られています

2.4 eslint、prettier の指摘を修正

ESlint、Prettier は指摘するだけではなく、修正案の提示・修正（できるものだけですが・・・）までしてくれます。

VSCode 側で設定を行うと、ファイルを保存する度に自動で修正をいれることもできます。

私は、修正を自分のタイミングで行いたいので VSCode 側の設定は行っていません。

もし、VSCode 側の設定を行いたい場合には、VSCode で

[File] -> [Preferences] -> [Settings] にて、以下の各項目を検索して設定するか、settings.json へ追加してください。

▼ VSCode の設定

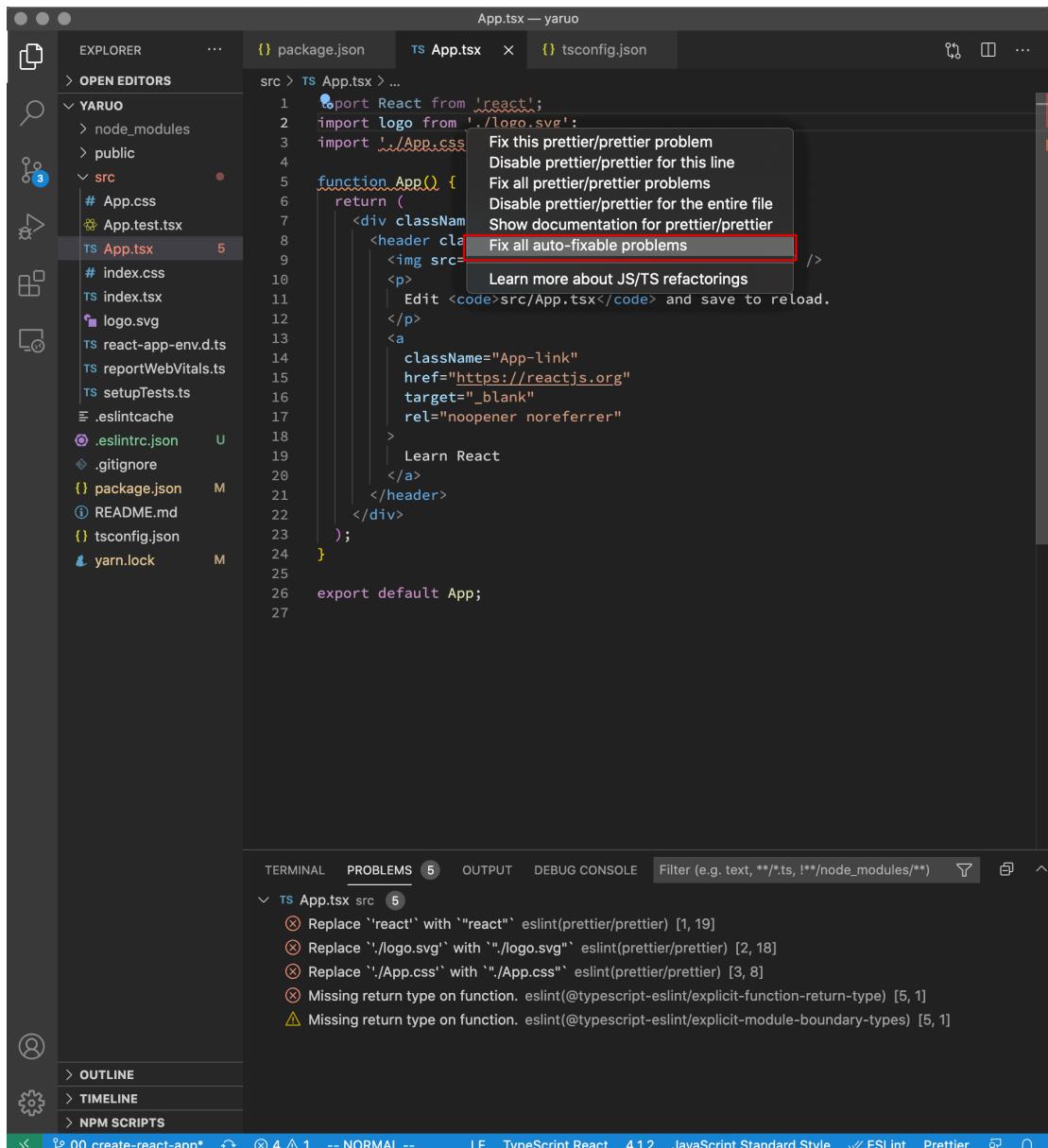
```
"editor.formatOnSave": true,  
"[JavaScript)": {  
    "editor.formatOnSave": false  
},  
"[JavaScriptreact)": {  
    "editor.formatOnSave": false  
},  
"[typescript)": {  
    "editor.formatOnSave": false  
},  
"[typescriptreact)": {  
    "editor.formatOnSave": false  
},  
"editor.codeActionsOnSave": {  
    "source.fixAll": true,  
    "source.fixAll.eslint": false  
},  
"prettier.disableLanguages": ["JavaScript", "JavaScriptreact", "typescript", "typescriptreact"],
```

VSCode 上で、

- 赤波線で指摘されている
- 問題タブひ表示されている

ものを修正します。

App.tsx の赤波線の上で「コマンドキー（Windows では、ctrl） + ピリオド」を押すと、画面のようにポップアップが表示されます。



▲図 2.3: ポップアップが表示

「Fix all auto-fixable problems」を選択すると、自動修復可能なものを修正してくれます。

筆者が VSCode を日本語化していないのは、エラーメッセージでググる場合を考えてのことです。英語での情報の方が的確なページを見つけやすいと思います。

以下のように、修正されました。

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure with files like package.json, App.tsx, tsconfig.json, and various CSS and test files.
- Code Editor:** Displays the content of App.tsx. The code defines a functional component named App that returns a div with a header containing a logo and a link to reactjs.org.
- Terminal:** Shows the command "yarn start" being run.
- Problems View:** Shows two ESLint errors:
 - Missing return type on function. eslint(@typescript-eslint/explicit-function-return-type) [5, 1]
 - Missing return type on function. eslint(@typescript-eslint/explicit-module-boundary-types) [5, 1]
- Status Bar:** Shows the file path "00_create-react-app*", line numbers, and toolbars for LF, TypeScript React 4.1.2, JavaScript Standard Style, ESLint, and Prettier.

▲図 2.4: 修正後

自動修正できない問題が残りました。問題点は、このファイルは Typescript を使用していますので、

「関数 App の戻り値の型が指定されていない！」
と言ふことです。

ついでに、Arrow 関数へ書き換えておきます。

▼ App.tsx

```
import React, { ReactElement } from "react";
import logo from "./logo.svg";
import "./App.css";

const App = (): ReactElement => {
  return (
    <div className="App">
```

このように、戻り値の型を指定することで指摘を修正することができました。

The screenshot shows the VS Code interface with the following details:

- Code Editor:** The main window displays the file `App.tsx` with the following content:

```
App.tsx — yaruo
src > TS App.tsx > [?] App
1 import React, { ReactElement } from "react";
2 import logo from "./logo.svg";
3 import "./App.css";
4
5 const App = (): ReactElement => {
6   return (
7     <div className="App">
8       <header className="App-header">
9         <img src={logo} className="App-logo" alt="logo" />
10        <p>
11          Edit <code>src/App.tsx</code> and save to reload.
12        </p>
13        <a
14          className="App-link"
15          href="https://reactjs.org"
16          target="_blank"
17          rel="noopener noreferrer"
18        >
19          Learn React
20        </a>
21      </header>
22    </div>
23  );
24
25  export default App;
26
27
```

- Explorer:** Shows the project structure with files like `App.tsx`, `index.css`, `index.tsx`, and `logo.svg`.
- Problems Panel:** Located at the bottom, it displays the message: "No problems have been detected in the workspace so far." (This message is highlighted with a red box).
- Status Bar:** Shows the file name `01 eslint_prettier`, line numbers, and various extension icons.

▲図 2.5: 全ての問題の修正完了

2.5 第2章のまとめ

React を使用したアプリケーションは、スタートアップ用のアプリケーションがコマンド一発でインストールできます。

より良いコーディングをするためにも、Eslint、Prettier を導入しましょう。

ここまで的内容は、github 上で、以下のコマンドでクローンできます。

▼github

```
$ > git clone -b 01_eslint_prettier https://github.com/tmkkz/yaruo.>  
> git
```

第 3 章

Todo リストの作成 (React のみ)

この章では、第 2 章で作成したスタートアップ用のアプリケーションを魔改造し、
Redux:<https://redux.js.org/tutorials/essentials/part-1-overview-concepts>
のチュートリアルにある Todo リストを作成します。
していきます。

3.1 React とは？

3.2**表示するデータの型を決める**

3.3

データ表示画面

3.4**React hooks を使用して、データの追加・編集・削除**

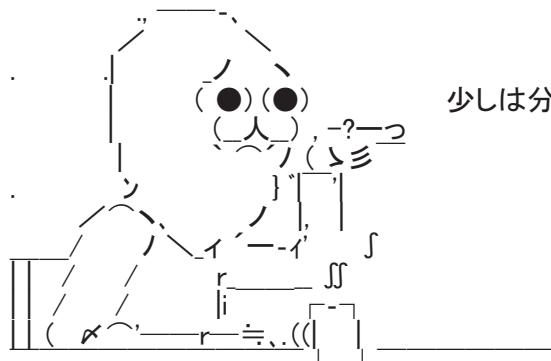
やる夫で学ぶ「react-redux」

redux-toolkit で、簡単・完璧理解だお…

2021年6月25日 ver 1.0

著 者 気分はもう

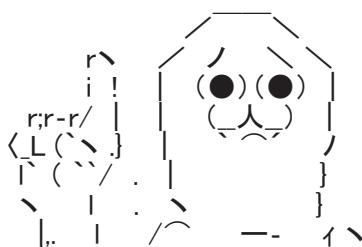
© 2020 気分はもう



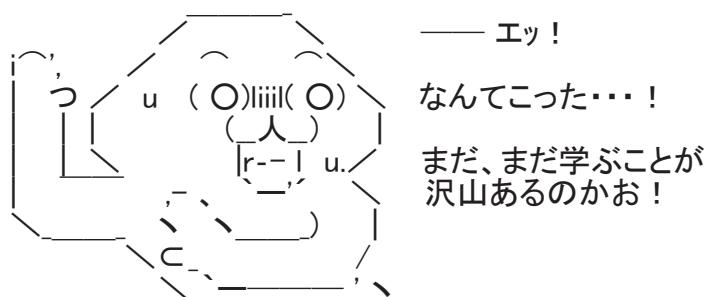
少しは分かったのか?



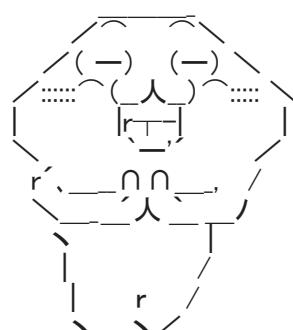
ヒヤヒヤヒヤ!
何とかなるお…



ほんとうは、非同期でのデータの取得や
UI関連の説明もしたかったのじゃが
紙面の関係で次回にする。



——エッ!
なんてこった…!
まだ、まだ学ぶことが
沢山あるのかお!



——というわけで
ここまで読んでくれて
ありがとうございます!