

ゆめみコーディングテスト

1月31日（1日目）

作業時間 4時間

- 21:00~25:00

やったこと

▼ 環境構築

このブログを参考にしました。

React + TypeScript + ESLint + Prettier の環境構築【yarn】 | Tech・O・Proch
npmよりyarn派の自分用に環境構築の備忘録 パッケージバージョン
node.js18.16.0React18.2.0TypeScript5.1.3ESLint8.43.0@typescript-eslint/eslint-plugin5.

 <https://tech-o-proch.com/programing/react/630>



create-react-app

```
$ yarn create react-app japan-population-graph --template typescript
```

ESLintとPrettier 導入

```
$ yarn add -D @typescript-eslint/eslint-plugin @typescript-eslint/parser
```

▼ RESAS API（都道府県情報の取得）

- APIキーの取得

APIキーを取得するためにアカウントを作成する必要があったためアカウントを作成し、APIキーを取得した。

RESAS-API - 地域経済分析システム（RESAS）のAPI提供情報

地域経済分析システム（RESAS：リーサス）のデータがAPI提供されます。API提供されることにより、従来よりも深く・自由に分析することが可能となります。今までRESASでは見えなかった各地域における真の課題の抽出や地域資源の再発見をサポートし、地域ビジネスに通じるサービスを地域自らが創り

 <https://opendata.resas-portal.go.jp/>



- APIキーを.envから取得できるように変更

.env

```
REACT_APP_RESAS_API_KEY="APIキー"
```

resas_api.ts

```
const API_KEY = process.env.REACT_APP_RESAS_API_KEY;
```

- APIレスポンスから都道府県一覧のチェックボックスを動的に生成

React App

- 北海道
- 青森県
- 岩手県
- 宮城県
- 秋田県
- 山形県
- 福島県
- 茨城県
- 栃木県
- 群馬県
- 埼玉県
- 千葉県
- 東京都
- 神奈川県
- 新潟県
- 富山県
- 石川県
- 福井県
- 山梨県
- 長野県
- 岐阜県
- 静岡県
- 愛知県
- 三重県
- 滋賀県
- 京都府
- 大阪府
- 兵庫県
- 奈良県
- 和歌山県
- 徳島県
- 香川県
- 高松市
- 愛媛県
- 高知県
- 福岡県
- 佐賀県
- 大分県
- 熊本県
- 鹿児島県
- 沖縄県

▼ テストコードの作成

テストコードを用いた開発が初めてだったため、テストコードには何を作ればよいのかがわからないため手探りで実装。

APIから都道府県を取得する関数fetchPrefecturesが正しく動作するか確かめるためのテストコードを作ろうと試みた。

fetch関数のエラー時のテストコードを記述するため、FetchMockを導入するも、リンターによるエラーが発生してしまった。

リンターを使うのも初めてだったため、テストコードを書く際にリンターのエラーの消し方がどうしてもわからなかった。

```
// Import React from 'react';
Unsafe call of an `any` typed value. eslint(@typescript-eslint/no-unsafe-call)
(alias) const fetchMock: FetchMock
import fetchMock
View Problem (Alt+F8) Quick Fix... (Ctrl+.)
fetchMock.enableMocks();
```

↓

```
japan-population-graph > src > App.test.tsx > ...
1  /* eslint-disable @typescript-eslint/no-unsafe-call */
2  /* eslint-disable @typescript-eslint/no-unsafe-member-access */
3  // import React from 'react';
4  // import { render, screen } from '@testing-library/react';
5  // import App from './App';
6  import { fetchPrefectures } from './api/resas_api';
7  import fetchMock from 'jest-fetch-mock';
8
9  fetchMock.enableMocks();
```

解決策がわからなかったなので、fetchMockを使うテストコードは後で実装

2月3日

作業時間 6時間半

- 10:30~16:00
- 20:00~21:00

やったこと

▼ 都道府県選択部分の機能実装

React App

selected Prefectures:北海道,宮城県,山形県,栃木県,群馬県,

- ☒ 北海道
- ☐ 青森県
- ☐ 岩手県
- ☒ 宮城県
- ☐ 秋田県
- ☒ 山形県
- ☐ 福島県
- ☐ 茨城県
- ☒ 栃木県
- ☒ 群馬県

▼ APIから都道府県の「人口構成」を取得する関数を実装

parameters

Name	Description	Required
prefCode	都道府県コード	true
cityCode	市区町村コード 「すべての市区町村」を選択する場合は「-」を送ります。	true
addArea	追加エリアコード 他地域と合算した値を取得する際に使用するパラメータです。「addArea=都道府県コード_市区町村コード」の形式で指定します。複数指定する場合は「addArea=1_01100,13_13101」などのように、「,」で各地域のパラメータを区切ります。都道府県単位で指定する場合、「addArea=1_,13_」の形式で指定します。最大10個指定でき、11個以上送られてきた場合はステータスコード400を返します。 「cityCode」で「すべての市区町村」(-)を選択した場合、addAreaで合算されるのは、都道府県単位の数値となります。また、「cityCode」でいずれかの市区町村(13101など)を選択した場合、addAreaで合算されるのは、市区町村単位の数値となります。	

今回は都道府県の人口構成なのでcityCodeは"- "を指定

▼ APIから取得した都道府県の「人口構成」を表示

とりあえず直接表示

React App

selected Prefectures:

Population Data

Boundary Year: 2020

総人口

Year: 1960, Value: 5039206

Year: 1965, Value: 5171800

Year: 1970, Value: 5184287

Year: 1975, Value: 5338206

Year: 1980, Value: 5575989

Year: 1985, Value: 5679439

Year: 1990, Value: 5643647

Year: 1995, Value: 5692321

Year: 2000, Value: 5683062

Year: 2005, Value: 5627737

Year: 2010, Value: 5506419

Year: 2015, Value: 5381733

Year: 2020, Value: 5224614

Year: 2025, Value: 5016554

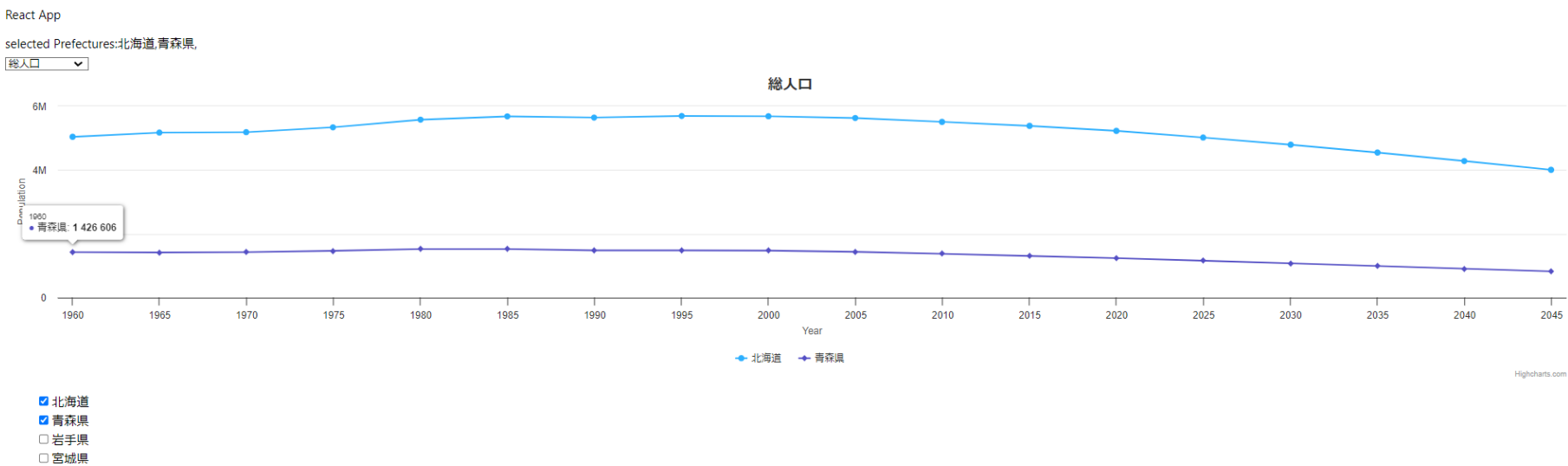
Year: 2030, Value: 4791592

Year: 2035, Value: 4546357

Year: 2040, Value: 4280427

Year: 2045, Value: 4004973

▼ 都道府県の人口構成をhighchartsで表示



▼ テストコードの見直し

jest-fetch-mockを導入しようして数時間溶けてしまった

結局実装はできなかった

2月10日

作業時間 5時間

- 20:30~24:00

• 25:00~26:30


やったこと

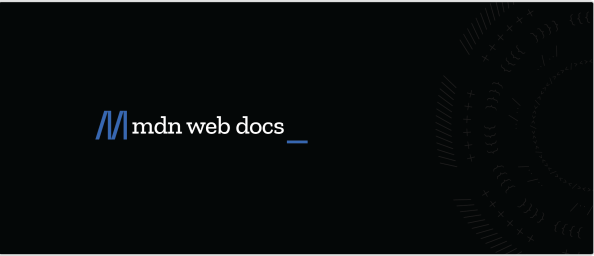
▼ レスポンシブなUIの作成

普段CSSはライブラリを使っているのですが、改めてCSSの基礎について勉強をした。
チェックボックスの改行をレスポンシブにするためにflexを使用した
CSSを学ぶために以下のサイトを参考にした

CSS の構造 - ウェブ開発を学ぶ | MDN

CSS の概要と基本的な使い方について理解できたので、今度は CSS の構造をもう少し詳しく見てみましょう。

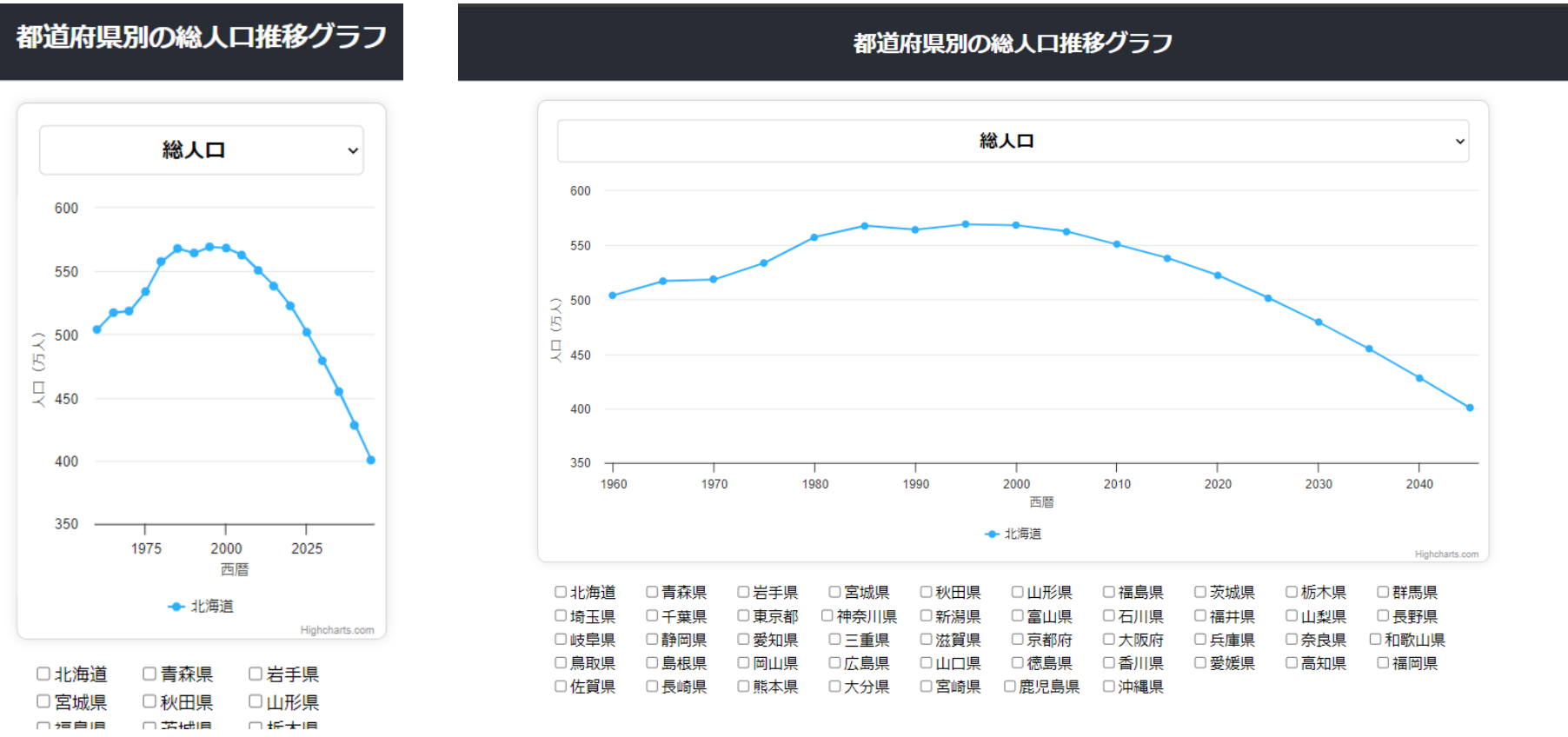
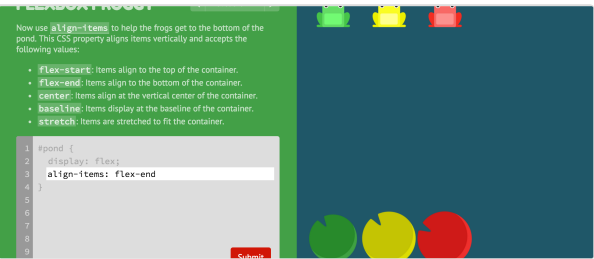
 https://developer.mozilla.org/ja/docs/Learn/CSS/First_steps/How_CSS_is_structured



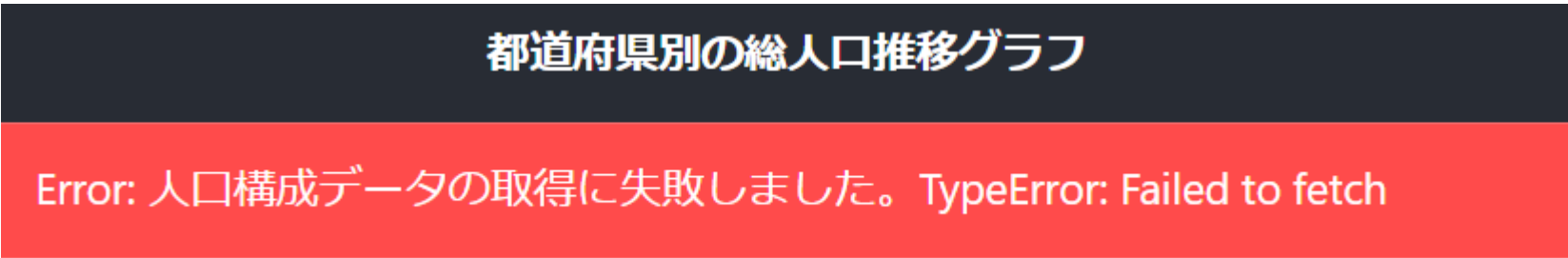
Flexbox Froggy

A game for learning CSS flexbox

 <https://flexboxfroggy.com/#ja>



▼ エラー発生時のUI表示の作成



▼ APIキーを保存した.envファイルが誤ってpushされてしまっていた→APIキーを更新して対処

APIキーを保存していた.envファイルをgitignoreに追加し忘れていて、誤ってpushしてしまっていた。そのため、.envファイルを削除しAPIキーを更新した。

1. **.env** ファイルを削除する

```
git rm --cached .env
echo .env >> .gitignore
git add .gitignore
git commit -m "Remove .env file and update .gitignore"
```

▼ 初期状態で北海道のグラフを表示するように変更

useEffectを用いて、prefecturesを取得したときに何も選択されていない状態だったら、北海道を表示するプログラムを追加

```
useEffect(() => {
  // 初期状態として北海道のグラフを表示
  const init = async (): Promise<void> => {
    if (prefectures.length > 0 && selectedPrefectures.length === 0) {
      setSelectedPrefectures([prefectures[0]]);
      await updatePrefecturePopulations();
      setSelectedPrefectures([]);
    }
  };
  void init();
}, [prefectures]);
```

2月11日

作業時間 5時間

- 9:00~10:00
- 15:00~17:00
- 23:00~25:00

やったこと

▼ デプロイ

netlifyを使ってデプロイした。

1. githubとnetlifyを連携
2. プロジェクトのリポジトリを選択
3. 環境変数にAPIを設定
4. ビルド設定

Build settings	
Runtime:	Not set
Base directory:	japan-population-graph
Package directory:	japan-population-graph/public
Build command:	yarn build
Publish directory:	japan-population-graph/build
Functions directory:	japan-population-graph/netlify/functions
Deploy log visibility:	Logs are public
Build status:	Active
Learn more about configuring builds in the docs ↗	
Configure	

▼ テストモックの作成

以下の記事を参考にしたら、fetchのモックを作ることができた

非同期リクエストを扱うコンポーネントのテスト：fetch そのものをモック、実装編

<https://zenn.dev/tkdn/books/react-testing-patterns/viewer/testing-with-fetchmock>



Zenn

Jestでfetchを簡単にモックする方法 | Angular | TypeScript

◇この記事でわかること・Jestでfetchを簡単にモックする方法UNOどうも、こんにちは。widen uno (@WidenUno) です。先日、Angularプロジェクト上のfetchをJestでモック化する際、手こずったため、今回は、簡単

<https://widen.tokyo/jest-fetch-mock-test/>

Jestでfetchを簡単にモックする方法

```
test('API error', async () => {
  global.fetch = jest.fn().mockResolvedValueOnce(
    Promise.resolve({
      json: async () => await Promise.resolve(mockAPIError),
    }),
  );
  await expect(fetchPrefectures()).rejects.toThrow('都道府県データの取得に失敗しました。Error: API');
});
```



```
PASS src/App.test.tsx (9.161 s)
  API:fetchPrefectures
    ✓ success (13 ms)
    ✓ API error (88 ms)
    ✓ 400 error (3 ms)
    ✓ Network error (5 ms)
  getPrefectureName
    ✓ success (2 ms)
    ✓ notfound error (8 ms)
  API:fetchPrefecturePopulation
    ✓ success (7 ms)
    ✓ API error (4 ms)
    ✓ 400 error (2 ms)
    ✓ Network error (3 ms)
  API:fetchPrefecturePopulations
    ✓ success (4 ms)
    ✓ API error (6 ms)
    ✓ 400 error (2 ms)
    ✓ Network error (4 ms)
  render: App
    ✓ render: App (426 ms)
    ✓ snapshot App (63 ms)
  render: prefecture
    ✓ render: loading (40 ms)
    ✓ render: prefecture (786 ms)
    ✓ render: API error (22 ms)
    ✓ render: 400 error (20 ms)
    ✓ render: Network error (25 ms)
  render: population-graph
    ✓ render: loading (26 ms)
    ✓ render: population-graph (659 ms)
    ✓ render: API error (132 ms)
    ✓ render: 400 error (129 ms)
    ✓ render: Network error (167 ms)
  PopulationGraph
    ✓ render: PopulationGraph (341 ms)
  PrefectureCheckBoxList
    ✓ render: PrefectureCheckBoxList (108 ms)
```

最終的にはテストコードを正しく実装することができた。

本来であれば開発と同時にテストコードを作るべきだが、テストコードの作成が最後となってしまった。

今回の開発でテストコードの使い方を掴む事が出来たので次回の開発からは開発と同時進行でテストコードを実装していきたい。

▼ リファクタリング

コンポーネントの細分化

ビュー部分とロジック部分の分割

不要なコメント・デバックログの削除