Overview libraries using in the project 🐮

This uses some supporting plugins:

- React Libraries (Main Platform): 'react', 'react-dom'.
- React Router V4 (React Plugin): 'react-router', 'react-router-dom'.
- RXJS - Reactive Extensions for JavaScript (handle side effect: asynchronous, timer, share data between components - state management...
- Webpack (Bundling Module support to build project): 'webpack'
- SASS - Pre-Processor: 'sass', 'node-sass'
- Library UI: 'react-bootstrap',
- react-toastify.

Guideline for focusing and developing to project

1. Using command line (CLI) in project

*Note: You can 'yarn' or 'npm' to work with this project.

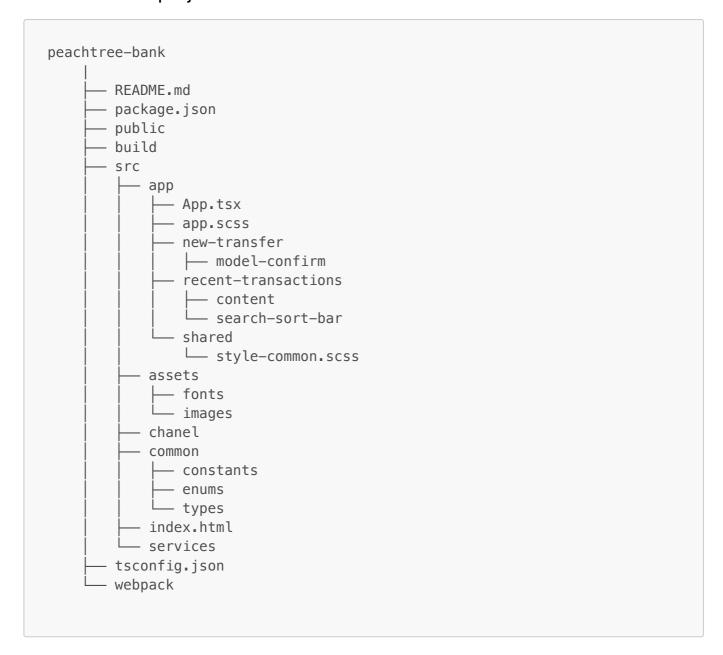Current time, we just use 'start' & 'build' to develop and pack modules in the project:

- Build project for the production environment:

```
npm run build
yarn run build
```

- Start project at dev environment:

```
npm start
yarn start
```

2. Structure of project

```
peachtree-bank
    |
    ├── README.md
    ├── package.json
    ├── public
    ├── build
    ├── src
    │   ├── app
    │   │   ├── App.tsx
    │   │   ├── app.scss
    │   │   ├── new-transfer
    │   │   │   ├── model-confirm
    │   │   ├── recent-transactions
    │   │   │   ├── content
    │   │   │   └── search-sort-bar
    │   │   └── shared
    │   │       └── style-common.scss
    │   ├── assets
    │   │   ├── fonts
    │   │   └── images
    │   ├── chanel
    │   ├── common
    │   │   ├── constants
    │   │   ├── enums
    │   │   └── types
    │   ├── index.html
    │   └── services
    ├── tsconfig.json
    └── webpack
```

- 2.1. **tsconfig.json**:

  File configures for typescript project such as compile decorator.

- 2.2. **package.json**

  File contains all configurations of project (libs-dependencies, script-task, plugins...)

- 2.3. **build/** folder:

  It stores sources of project after building.

- 2.4. **public/** folder:

  It stores sources (css, data-resources, fonts, images, locales) of project after building at the dev environment.

- 2.5. **webpack/** file:

  It includes files using to build and start project.

- 2.6. **src/**

  This is the main folder in project. You can develop anything in here. It separates to 5 sub-folders: **common/ , assets/ , chanel/, service, app, app.tsx and index.tsx file**

  - 2.6.1. **common/** folder

    This includes constant, enum....

  - 2.6.2. **assets/** folder

    This includes images, fonts....

  - 2.6.3. **chanel/** folder

    This includes data share between components.

  - 2.6.4. **service/** folder

    This includes all of apis.

  - 2.6.5. **app/** folder

    This includes ts and scss files of component group by every feature.

    - 2.6.5.1. **shared/** folder

      This includes common files, logic, component... which can re-use more than one time in project.

  - 2.6.6. **app.tsx/** file

    App.tsx is a start-point to any process, and imported out of **index.tsx** to run project.

  - 2.6.7 **index.tsx file**

    This is the first file called from server after running project. All threads of project will begin from here.

3. Basic knowledge and how to apply to this project - chanel - state management

3.1: Create a chanel - to share data between components - we use Subject in RXJS to emit values to be multi-casted to many Observers.

- To define a chanel.

```
import { initialFilter } from './../common/constants/CommonConst';
import { getTransactionsHistoryService } from
'../services/getAccount.service';

import { from, Subject } from 'rxjs';
import { toast } from 'react-toastify';

const subject = new Subject();
const initialState = {
  transactionsHistory: [],
};

let state = initialState;

const transactionsHistoryChanel = {
  subscribe: (setState: any) => subject.subscribe(setState),
  getTransactionsHistory: (filter?: any) => {
      if (!filter) {
      filter = initialFilter;
      }

      from(getTransactionsHistoryService(filter)).subscribe((e: any) =>
{
      state = {
          ...state,
          transactionsHistory: e
      };
      subject.next(state);
      });
  },
};

export default transactionsHistoryChanel;
```

3.2: To use this chanel in component - we subscribe chanel to get data whenever it's changed.

```
import React, { useLayoutEffect, useState } from 'react';
import transactionsHistoryChanel from '../../../chanel/transactions-history.chanel';

function Content() {

  const [state, setState] = useState<any>();

  useLayoutEffect(() => {
      transactionsHistoryChanel.subscribe(setState);
  }, []);
}

export default Content;
```

3.3: Emit value via function getTransactionsHistory to search or sort transactions history.

```
import React, { useEffect, useState } from 'react';
import transactionsHistoryChanel from '../../../chanel/transactions-history.chanel';


function SearchSortBar() {

  const [filter, setFilter] = useState(initialFilter);

  useEffect(() => {
      transactionsHistoryChanel.getTransactionsHistory(filter);
  }, [filter]);
}

export default SearchSortBar;
```

View more:

- Using RXJS: https://www.learnrxjs.io/

4. Run project

1. npm run start (port 4200)
2. Start Nodejs server (default port 4200, Can change port at the webpack devServer port).
3. Open web browser with url: http://your_ip

5. Note commit in project

Don't commit these paths folder and file in the project. Because, they will auto generate when build

\peachtree-bank\target

\peachtree-bank\build

\peachtree-bank\package-lock.json

\peachtree-bank\yarn-lock.json

\peachtree-bank\yarn.lock

\peachtree-bank\yarn-error.log

\peachtree-bank\debug.log