**Basic exercises jsx**

**Slot 1-2-3-4**

**Created by traltb@fe.edu.vn**

**Bước 1: Cài đặt nodejs:**

<https://nodejs.org>

Cài đặt xong kiểm tra thử node đã cài đặt ok,

Mở command line, gõ node -v

Xem phiên bản node

Tiếp tục gõ: npm -v

Xem phiên bản của npm (Node Package Manager)

Node và npm là 2 tool để phát triển, chạy ứng dụng javascript, nodejs

**Bước 2: Tạo ứng dụng react**

Mở command line, gõ lệnh:

npm install -g create-react-app

Lệnh này gõ 1 lần thôi, để tạo ứng dụng react.

**Bước 3: Bắt đầu xây dựng ứng dụng react**

Vào ổ D:\, tạo thư mục FER202, Code, Exercises để tạo ra đường dẫn trên máy tính như sau: D:\FER202\Code\Exercises

Tạo mới 1 thư mục AtClass trong thư mục này. Click phải ở thư mục này, chọn Open In Terminal

Gõ lệnh:

npx create-react-app exercise1

Sau khi chạy xong lệnh này, sẽ tạo ra 1 thư mục exercise1 trong thư mục AtClass, tiếp tục thực hiện lệnh chuyển thư mục, về làm việc với thư mục exercise1, Gõ lệnh:

Cd exercise1

Gõ lệnh sau nếu gói web-vitals nếu bị lỗi:

npm install web-vitals

Chạy ứng dụng, gõ:

npm start

Kết quả theo hình:

A blue and white symbol with a circle in the middle

AI-generated content may be incorrect.

Nếu trong quá trình chạy có bị lỗi này:

npm.ps1 cannot be loaded because running scripts is disabled on this system

Vào tìm kiếm trên máy tính, gõ PowerShell

Click phải, chạy với quyền Admin, chọn Run as Administrator:

Gõ lệnh sau:

Set-ExecutionPolicy RemoteSigned

Sau đó chạy lại lệnh: npm start

**Bước 4: Hiệu chỉnh file code jsx**

Mở Visual studio code, Chọn File->Open Folder. Duyệt đến thư mục exercises1

Vào Terminal, chọn New Terminal,

Nhấn Ctrl+ C để dừng server

Mở file App.js: hiệu chỉnh code như sau:

**Cách 1**: Thêm thuộc tính style vào thẻ h1, lưu ý dùng 2 cặp {}, và thuộc tính Viết hoa chữ cái thứ 2:

import "./App.css";

function App() {

  return (

    <div>

      <h1 style={{ color: "blue", textAlign: "center" }}>Hello world!</h1>

    </div>

  );

}

export default App;

Cách 2:

import "./App.css";

function App() {

  const h1Style = {

    color: "blue",

    textAlign: "center",

  };

  return (

    <div>

      <h1 style={h1Style}>Hello world!</h1>

    </div>

  );

}

export default App;

Cách 3: Tạo class trong file App.css

.h1Style {

  color: blue;

  text-align: center;

}

Khi đó nội dung file App.js là:

import "./App.css";

function App() {

  return (

    <div>

      <h1 className="h1Style">Hello world!</h1>

    </div>

  );

}

export default App;

Bài tập tương tự:

**Exercise 1: Display a list of names**

Cho trước 1 mảng các tên của người, hiển thị danh sách này dưới dạng danh sách không sắp xếp thứ tự (Given an array of names, display them in an unordered list)

Tạo mới 1 tập tin NamePerson.js trong thư mục src, lưu ý chỉ có 1 thẻ div cha

function NamePerson() {

  const names = ["Alice", "Bob", "Charlie"];

  return (

    <div>

      <ul>

        {names.map((name, index) => (

          <li key={index}> {name}</li>

        ))}

      </ul>

    </div>

  );

}

export default NamePerson;

Sau đó thêm component NamePerson vào trong file index.js

import React from "react";

import ReactDOM from "react-dom/client";

import "./index.css";

import App from "./App";

import reportWebVitals from "./reportWebVitals";

import NamePerson from "./NamePerson";

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(

  <React.StrictMode>

    <App />

    <NamePerson />

  </React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

Reload lại trang web.

A white background with black and white clouds

AI-generated content may be incorrect.

Có thể viết file NamePerson.js theo cách này:

function NamePerson() {

  const names = ["Alice", "Bob", "Charlie"];

  const element = (

    <ul>

      {names.map((name, index) => (

        <li key={index}>{name}</li>

      ))}

    </ul>

  );

  return element;

}

export default NamePerson;

**Exercise 2: Display a person's details**

Given a person object, display their name, age, and occupation.

Using the above people array, do the following exercises:

**Exercise 3:** Display a list of people (PeopleList.js)

**Exercise 4:** Display a table of people

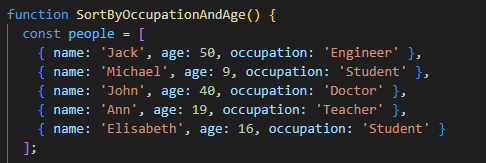
**Exercise 5:** Find the first teenager (FirstTeenager.js)

**Exercise 6:** Check if all are teenagers (AreAllTeenagers.js)

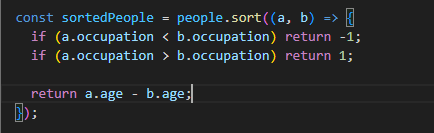
**Exercise 7:** Sort by Occupation and Then by Age

Write a function to sort the people array first by occupation alphabetically and then by age in ascending order within each occupation group.

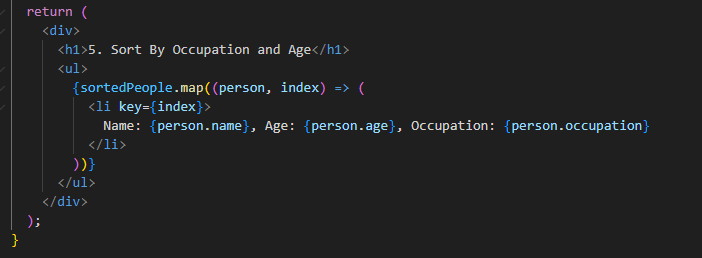
B1: Khai báo component SortByOccupationAndAge và các dữ liệu trong mảng people

  
B2:

* Đầu tiên sẽ sắp xếp theo nghề nghiệp trước. So sánh 2 đối tượng nghề nghiệp. Nếu nghề nghiệp a đứng trước b trong bảng chữ cái, trả về -1 để a đứng trước b. Nếu a đứng sau b thì trả về 1 để a đứng sau b.
* Nếu như nghề nghiệp của a và b như nhau thì sẽ sắp xếp theo tuổi. Đối tượng nào có tuổi nhỏ hơn sẽ đứng trước



B3: Sử dụng return để trả về JSX để render giao diện. Sử dụng <h1> để đặt tiêu đề và <ul> để tạo một danh sách hiện thị các phần tử sau khi đã sắp xếp. sortedPeople.map(…) sẽ tạo một mảng mới từ sortedPeople, trong đó mỗi phần tử là một <li> chứ thông tin phần tử. Bên trong mỗi <li> sẽ hiện thị thông tin tên, tuổi và nghề nghiệp của phần tử đó. Key={index} sẽ giúp React nhận diện các phần tử trong danh sách và tối ưu hoá việc render.



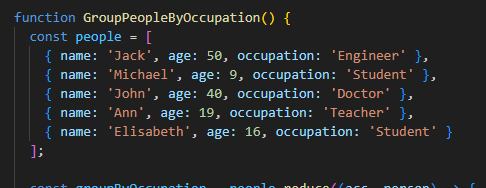
B4: Xuất component



**Exercise 8:** Group People by Occupation

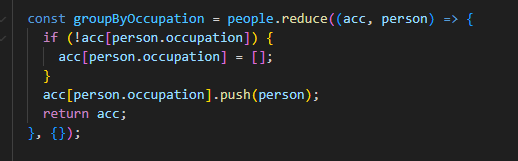
Write a function that groups people by their occupation into an object. Each key is an occupation, and the value is an array of people with that occupation.

B1: Khai báo component GroupPeopleByOccupation và các dữ liệu trong mảng people

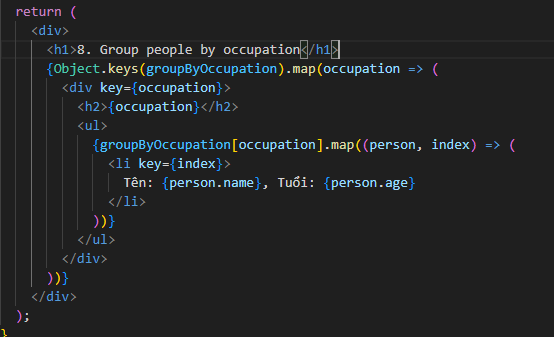


B2: Nhóm các đối tượng theo nghề nghiệp bằng reduce. Sử dụng acc (accumulator) để lưu trữ các nhóm nghề nghiệp và person là đối tượng trong mảng people đã được duyệt.

* Kiểm tra nếu nghề nghiệp của đối tượng hiện tại trong (person.occupation) chưa tồn tại trong acc sẽ tạo một mảng rỗng cho nghề nghiệp đó bằng acc[person.occupation] =[];
* Sau đó sẽ thêm đối tượng hiện tại vào mảng của nghề nghiệp đó bằng acc[person.occupation].push(person);
* Tiếp theo sẽ trả về acc sau mỗi vòng lặp.



B3: Sử dụng return để trả về JSX để hiện thị lên giao diện các thông tin về các nghề nghiệp và danh sách người nằm trong nghề nghiệp đó.



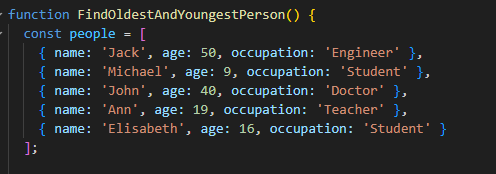
B4: Xuất component



**Exercise 9:** Find the Oldest and Youngest Person

Create a function to find and display the oldest and youngest person in the array.

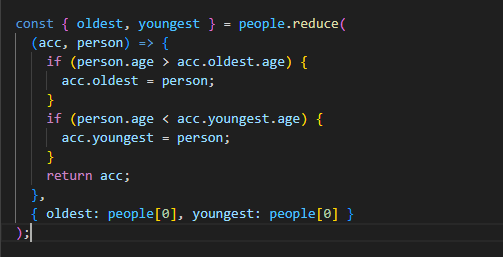
B1: Khai báo component FindOldestAndYoungestPerson và các dữ liệu trong mảng people



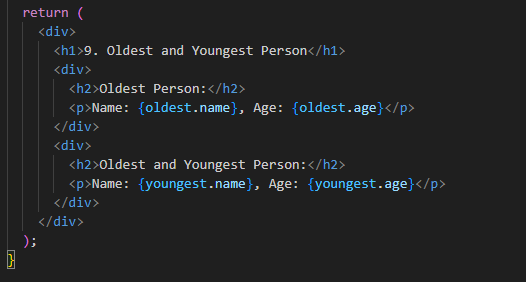
B2: Sử dụng reduce để duyệt qua toàn bộ mảng để xác định người già nhất oldest và người trẻ nhất youngest. Khởi tạo các giá trị accumulator (acc) là acc.oldest và acc.youngest trong mảng people sau đó so sánh.

* Nếu person.age > acc.oldest.age thì sẽ cập nhật acc.oldest = person
* Nếu person.age < acc.youngest.age thì sẽ cập nhật acc.youngest = person.

Sau đó trả về acc sau mỗi lần lặp



B3: Sử dụng return để trả về JSX để hiện thị các kết quả về thông tin người lớn tuổi nhất và nhỏ tuổi nhất trên giao diện



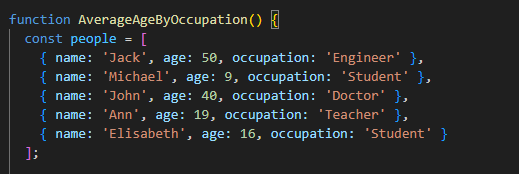
B4: Xuất component



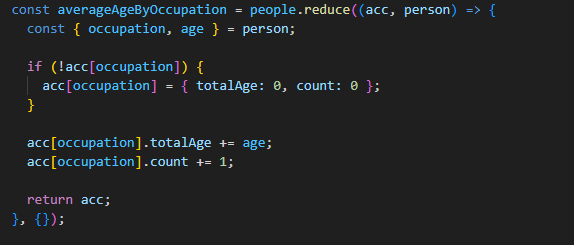
**Exercise 10:** Calculate the Average Age of Each Occupation

Write a function that calculates and displays the average age for each occupation.

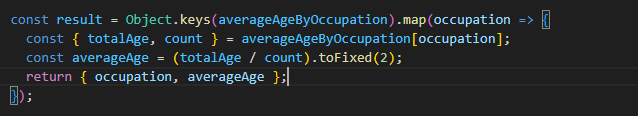
B1: Khai báo component SortByOccupationAndAge và các dữ liệu trong mảng people.

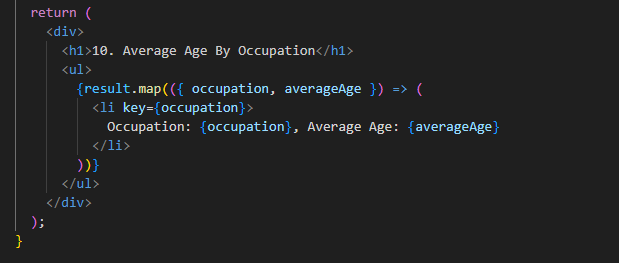


B2: Nhóm và tính tổng tuổi theo nghề nghiệp. Sử dụng reduce() để duyệt qua mảng people và nhóm các đối tượng theo nghề nghiệp. Với mỗi nghề nghiệp, lưu lại tổng tuổi những người có nghề nghiệp đó bằng (totalAge) và số lượng người ở trong nghề nghiệp đó (count). Với mỗi đối tượng person trong people, kiếm tra xem nghề nghiệp đã tồn tại trong đối tượng acc chưa. Nếu chưa có sẽ tạo một đối tượng {totalAge: 0, count:0} cho nghề nghiệp đó. Sau đó cộng dồn tuổi vào totalAge và tăng count lên 1.



B3: Sauk hi có tổng tuổi và số lượng người trong từng nghề nghiệp, chuyển đổi thành averageAgeByOccupation thành mảng. Sử dụng Object.keys() để duyệt qua từng nghề nghiệp. Sau đó tính tuổi trung bình bằng cách chia totalAge cho count. Sử dụng .toFixed(2) để làm tròn đến 2 chữ số thập phân.



B4: Sử dụng return để trả về JSX để hiện thị danh sách nghề nghiệp và tuổi trung bình

B5: Xuất component

