

Contents

1. Xây dựng ứng dụng đầu tiên.....	2
2. React component.....	3
Class component.....	3
Functional component.....	4
3. Sử dụng git và push Project lên kho lưu trữ Của github.....	4
4. Importing and Exporting Components.....	6
Import.....	6
Export.....	6
5. State and prop	6
State	6
Prop.....	8
6. Một số hook cơ bản.....	10
useEffect	10
useRef	14
Bài tập 1:.....	16
Bài tập 2.....	16
7. Single Page Application (SPA)	19
8. Routing trong ReactJS	19
a. Các component cơ bản	20
Ví dụ	21
Chú ý:.....	21
Cách khác:.....	26
Bài tập	27
9. Controlled Forms	28
Đặc điểm quan trọng của Controlled Forms	28
Lợi ích của controlled forms	31
Bất lợi của controlled forms.....	31
Ví dụ cho bài CRUD (step by step)	31
Bài tập	41
10. Uncontrolled Forms	44
Đặc điểm quan trọng của Uncontrolled Forms	44
Lợi ích của Uncontrolled forms	45
Bất lợi của Uncontrolled forms.....	45
11. Template cho app	45
Cài đặt Bootstrap 4.....	45

Cấu hình sử dụng Bootstrap 4.....	46
Sử dụng: Font Awesome Icons and Bootstrap-Social	47
Cài đặt React Router	47
12. Step by step tạo project với bootstrap 4:	47
Tạo app:	47
Cài Bootstrap 4	47
Cấu hình sử dụng Bootstrap 4.....	47
Cài đặt React Router	47
Chạy app	47
Sử dụng: Font Awesome Icons.....	47
Cài react bootstrap:	47
App có cấu hình thư mục như sau:.....	48
Tạo components, tạo Header.js có dạng.....	48
Tạo Slider có dạng:	51
Phần ListCart	55
Phần Promotion.....	57
Phần Footer	59
Xử lý validate đơn giản với React Hook Form	60
Module Login.....	61
Module Signup.....	66
13. Redux	73
Nguyên lý vận hành	74
Cài đặt và ví dụ	75
Áp dụng Redux vào bên trong project ReactJS	77
14. Redux form	79
Cài đặt và sử dụng Redux form	80
Các ví dụ khác về việc sử dụng Redux Form.....	82
15. JSON Server.....	83
16. React Animation.....	84

1. Xây dựng ứng dụng đầu tiên

```
npx create-react-app my-app
```

Khởi chạy project

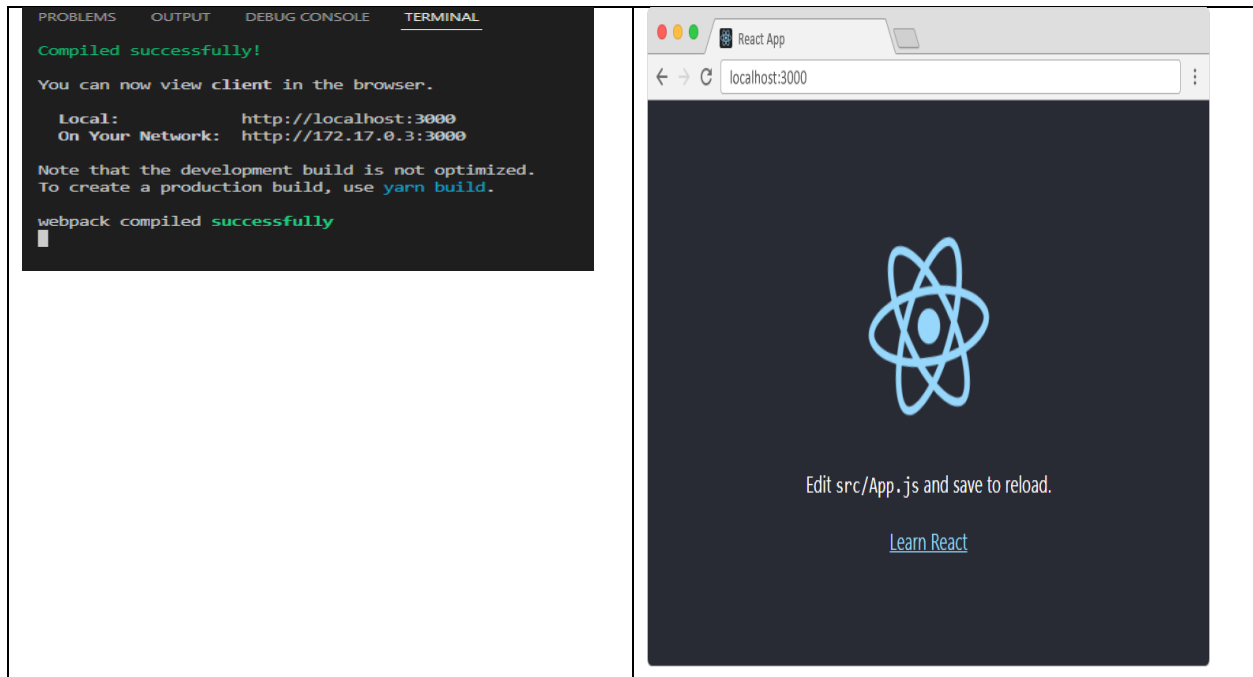
npm start

Runs the app in the development mode.

Open <http://localhost:3000> to view it in your browser.

The page will reload when you make changes.

You may also see any lint errors in the console.



2. React component

React component hay component là một block code độc lập, có khả năng tái sử dụng. Việc chia UI thành các component giúp cho việc tổ chức và quản lý code dễ dàng hơn

React component bao gồm 2 loại:

- **Class component**
- **Functional component**

Class component

Đặc điểm

Các Class component đơn giản là những **class ES6** kế thừa từ class tên 'Component' của React

Mọi Class component sẽ **phải chứa method render()**, nơi **return một JSX template** hoặc **null**

Trong docs mới nhất của React, Class component đã được đưa vào danh mục '**Legacy React APIs**' và đội ngũ phát triển React cũng khuyến khích việc sử dụng Functional Component để thay thế. Tuy nhiên, Class component hiện vẫn sẽ được hỗ trợ sử dụng.

<https://react.dev/reference/react/Component>

Syntax

```
import React, { Component } from "react";
class ClassComponent extends Component {
  render() {
    return (
      <div>
        <h1>Hello Class component!!</h1>
      </div>
    );
  }
}
export default ClassComponent;
```

Functional component

Đặc điểm

Xuất hiện từ phiên bản React 16.8

Functional component được tạo bởi một hàm JavaScript, return một JSX template hoặc null

Vì được tạo bởi một hàm JavaScript, ta có thể viết Functional component dưới dạng JS function thông thường hay ES6 arrow function

Với việc đội ngũ phát triển React đang chú trọng phát triển các hooks và cách viết, triển khai code khá dễ hiểu, Functional component được khuyến khích sử dụng

Syntax

JS function

```
import React from "react";
const FunctionComponent = () => {
  return (
    <div>
      <h1>Function
      component example!!!</h1>
    </div>
  );
}
export default
FunctionComponent;
```

```
import React from "react";
function FunctionComponent{
  return (
    <div>
      <h1>Function
      component example!!!</h1>
    </div>
  );
}
export default
FunctionComponent;
```

3. Sử dụng git và push Project lên kho lưu trữ Của github

git init (để khởi tạo git)

Branch

Khi sử dụng Git, bạn có thể tạo ra nhiều nhánh (branch) khác nhau. Câu lệnh Git này dùng để kiểm tra branch hiện tại:

git branch

Để tạo mới một branch:

git branch <name_branch>

Xóa một branch:

git branch -d <name_branch>


Để đưa subsidiary vào branch master, thì trước hết sẽ di chuyển đến branch master. Sau đó làm các bước tiếp theo:

git checkout master	Để cập nhật tất cả các file thì ta sử dụng câu lệnh:
git branch new-branch	git add .
git checkout new-branch	Còn muốn cập một file nào đó thì sử dụng câu lệnh:
git add .	git add .<tên file>
git commit -m "first-commit"	sử dụng câu lệnh Commit để ghi lại việc thay đổi và đẩy thông tin thay đổi lên Local Repository:
git checkout master	git commit -m 'Commit 1'
git merge new-branch	

Tạo Repository: my-app

Create a new repository



A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner *
 anhttv20 / my-app

✓ my-app is available.

Great repository names are short and memorable. Need inspiration? How about [potential-barnacle?](#)

Description (optional)

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

...or create a new repository on the command line

```
echo "# my-app" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/anhttv20/my-app.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/anhttv20/my-app.git
git branch -M main
git push -u origin main
```

4. Importing and Exporting Components

Module

Một module là một file. Hay là “One script is one module”. Những module có thể load nhiều function bởi hai keywords đặc biệt đó là Import và Export. Và đặc biệt module này có thể gọi và sử dụng một module khác.

Import

Cho phép import các functionality từ các module khác. Export: Khai báo những variables hoặc function cho phép những module khác truy cập và sử dụng

Export

Có 2 loại export đó là named và default:

Named Export: Trong JavaScript ES6, named export được sử dụng để xuất nhiều thứ từ một module bằng cách thêm keyword export vào khai báo của chúng. Những thứ được export sẽ được phân biệt bằng tên. Sau đó import những thứ chúng ta cần sử dụng bằng cách bao quanh chúng cặp dấu ngoặc nhọn { }. Tên của module đã nhập phải giống với tên của module đã xuất.

Default Export: Trong Javascript ES6 chỉ cho phép xuất một mặc định cho mỗi file. Default Export có thể cho một function, class hoặc một object.

Không thể export khi định nghĩa tên: export default const name = 'value';

```
import React from "react";
import Taylor from '../assets/images/Taylor.png'
export function Profile() {
  return (
    <img
      src={Taylor}
      alt="Taylor Swift"
      width="200px" height="200px"
    />
  );
}
export default function Gallery() {
  return (
    <section>
      <h1>Amazing singers</h1>
      <Profile />
      <Profile />
      <Profile />
    </section>
  );
}

import Gallery from './components/Gallery';
import { Profile } from './components/Gallery'
```

5. State and prop

State

Được sử dụng để quản lý dữ liệu của riêng component chứa nó

Thay đổi được giá trị thông qua hook useState (Functional Component), khi state thay đổi giá trị, Component chứa state sẽ được re-render

Syntax:

```
import { useState } from "react";
.....
const [state, setState] = useState([giá trị khởi tạo]);
```

Ví dụ bộ đếm (counter) trong StateExample.jsx, ta có 1 state mang tên 'counter', phương thức cập nhật lại giá trị của state này là 'setCounter' và state này có giá trị ban đầu = 0

```
const [counter, setCounter] = useState(0);
```

Khi thực hiện bấm nút Increase, function handleIncrease sẽ được gọi ở onClick của button, setCounter sẽ được thực hiện và thay đổi tăng state counter, dẫn tới việc Component StateExample được render lại và số tương ứng sẽ xuất hiện

Trường hợp tương tự xảy ra với nút Decrease

```
const handleIncrease = () => {
  setCounter(counter + 1);
};
const handleDecrease = () => {
  setCounter(counter - 1);
};
```

Một số css

box.css	stateExample.css
<pre>.box-wrapper { width: 100px; height: 100px; margin: 10px; } //propExample.css .box-container { display: flex; flex-direction: row; justify-content: center; }</pre>	<pre>.state-example-wrapper { margin-bottom: 40px; } .counter-text { font-size: 30px; } .state-change-btn { font-size: 15px; padding: 5px 10px; margin: 10px; }</pre>

Demo như sau

State Example

Counter: 5

Decrease

Increase

```
StateExample.jsx
import React, { useState } from "react";
import "../assets/css/stateExample.css";
const StateExample = () => {
```

```

const [counter, setCounter] = useState(0);
const handleIncrease = () => {
  setCounter(counter + 1);
};
const handleDecrease = () => {
  setCounter(counter - 1);
};

return (
  <div className="state-example-wrapper">
    <h1>State Example</h1>
    <h2 className="counter-text">Counter: {counter}</h2>
    <button className="state-change-btn"
onClick={handleDecrease}>
      Decrease
    </button>
    <button className="state-change-btn"
onClick={handleIncrease}>
      Increase
    </button>
  </div>
);
};
export default StateExample;

```

Prop

Được sử dụng để truyền dữ liệu giữa các Component của React theo hướng từ Component cha sang Component con dưới dạng một object.

Không thể thay đổi được, prop là dạng dữ liệu 'chỉ đọc' tức là props không thể thay đổi ở Component con, nếu cần có sự thay đổi thì phải thay đổi ở Component cha sau đó truyền lại xuống.

Ví dụ trong Component cha PropExample.jsx, ta truyền dữ liệu là màu của box xuống Component con là Box.jsx

```
<Box boxColor="green" />
```

Component Box sẽ nhận được prop và dữ liệu sẽ được sử dụng để set màu như trong style ở dưới.

```

import React from "react";
import "../assets/css/box.css";
const Box = (props) => {
  return (
    <div>
      <div className="box-wrapper"
style={
  { backgroundColor: props.boxColor }
}>
      </div>
    </div>
  );
};

```



```
}

```

Ta hoàn toàn có thể thử các giá trị màu khác để nhận được các box với màu tương ứng

```
...
<Box boxColor="black" />
<Box boxColor="blue" />
...
```

Để ví dụ cho trường hợp **prop không thể thay đổi được**, ta sẽ thay đổi màu của box bằng cách đặt thêm một button 'Change prop color', khi click ta sẽ thực hiện set lại giá trị boxColor của prop

```
const Box = (props) => {
  return (
    <div>
      <div className="box-wrapper"
        style={
          { backgroundColor: props.boxColor }
        }>
        /* button được thêm ở đây */
        <button
          onClick={() => {
            props.boxColor = "yellow";
          }}
        >
          Change prop color
        </button>
      </div>
    </div>
  );
}
```

Một lỗi sẽ được hiển thị thông báo ta không thể gán giá trị ở đây là boxColor (prop) từ đối tượng 'không thể mở rộng'

Uncaught runtime errors:

ERROR

State Example

Cannot add property boxColor, object is not extensible

TypeError: Cannot add property boxColor, object is not extensible

```
at onClick (http://localhost:3000/static/js/bundle.js:169:26)
at HTMLUnknownElement.callCallback (http://localhost:3000/static/js/bundle.js:11224:18)
at Object.invokeGuardedCallbackDev (http://localhost:3000/static/js/bundle.js:11268:20)
at invokeGuardedCallback (http://localhost:3000/static/js/bundle.js:11325:35)
at invokeGuardedCallbackAndCatchFirstError (http://localhost:3000/static/js/bundle.js:11339:29)
at executeDispatch (http://localhost:3000/static/js/bundle.js:15483:7)
at processDispatchQueueItemsInOrder (http://localhost:3000/static/js/bundle.js:15509:11)
at processDispatchQueue (http://localhost:3000/static/js/bundle.js:15520:9)
```

Để thay đổi được màu của box, ta sẽ phải thực hiện ở Component cha PropExample.jsx:

Ta đặt thêm 2 button ở Component cha lần lượt là 'Change to red' và 'Change to yellow' và dùng 1 state để quản lý màu (Cần state thay vì biến thường để re-render Component và thấy được sự thay đổi)

Ban đầu, với giá trị mặc định của state `boxColor` là `green`, Box sẽ được render và hiển thị với màu xanh tương ứng

Khi click 1 trong 2 button trên, state `boxColor` sẽ được thay đổi, đồng nghĩa với việc Component cha là `PropExample` chứa state sẽ được re-render, Box lúc này cũng sẽ được render lại với giá trị `boxColor` mới tương ứng với nút vừa bấm

```
import React, { useState } from "react";
import "../assets/css/propExample.css";
import Box from "../Box";
const PropExample = () => {
  const [boxColor, setBoxColor] = useState("green");

  const handleChangeToRed = () => {
    setBoxColor("red");
  };

  const handleChangeToYellow = () => {
    setBoxColor("yellow");
  };
  return (
    <div>
      <h1>Prop Example</h1>
      <div className="box-container">
        { /* <Box boxColor="green" /> */ }
        <Box boxColor={boxColor} />
      </div>
      <button className="state-change-btn"
onClick={handleChangeToRed}>
        Change to red
      </button>
      <button className="state-change-btn"
onClick={handleChangeToYellow}>
        Change to yellow
      </button>
    </div>
  );
};
export default PropExample;
```

6. Một số hook cơ bản

useState:

Dùng để khai báo state của component (Đã đề cập phía trên)

useEffect

Được sử dụng để xử lý các side effects:

- Thao tác API
- Thao tác với DOM
- Thêm, xóa event listeners ('click', 'scroll', ...)

- setTimeout, setInterval

Syntax

```
import {useEffect} from 'react'
...
useEffect(callback, dependency array nếu có);
```

Ví dụ về các trường hợp của dependency array

- Trong trường hợp dependency array không xuất hiện, useEffect sẽ chạy với mỗi lần component re-render
- Trong trường hợp dependency array rỗng, useEffect sẽ chạy đúng một lần khi component render lần đầu tiên
- Trong trường hợp dependency array có giá trị, useEffect sẽ chạy khi component render và một trong các giá trị truyền vào có sự thay đổi

effectExample.css	refExample.css
<pre>.effect-example { margin-top: 20px; }</pre>	<pre>.increaseBtn { font-size: 30px; width: fit-content; } .count-text-wrapper { margin-bottom: 50px; }</pre>

Ví dụ 1:

EffectExample.jsx
<pre>import React, { useEffect, useState } from "react"; const EffectExample = () => { const [countA, setCountA] = useState(0); const [countB, setCountB] = useState(0); useEffect(() => { console.log("Không có dependency array"); }); useEffect(() => { console.log("Dependency array rỗng"); }, []); useEffect(() => { console.log("Dependency array phụ thuộc vào countA"); }, [countA]); useEffect(() => { console.log("Dependency array phụ thuộc vào countA và countB"); }, [countA, countB]); const handleIncreaseCountA = () => { setCountA(countA + 1); }; const handleIncreaseCountB = () => { setCountB(countB + 1); }; }</pre>

```
};
return (
  <div className="effect-example">
    <div className="count-text-wrapper">
      <h1>Count A: {countA}</h1>
      <h1>Count B: {countB}</h1>
    </div>
    <div>
      <button onClick={handleIncreaseCountA}>Increase Count
A</button>
      <button onClick={handleIncreaseCountB}>Increase Count
B</button>
    </div>
  </div>
);
};
export default EffectExample;
```

Ví dụ 2: ban đầu

```
import React, { useState } from 'react';

export const EffectDemo = () => {
  //State
  const [fullName, setFullName] = useState({ name: 'name',
familyName: 'family' });
  const [title, setTitle] = useState('useEffect() in Hooks');

  return (
    <div>
      <h1>Title: {title}</h1>
      <h3>Name: {fullName.name}</h3>
      <h3>Family Name: {fullName.familyName}</h3>
    </div>
  );
};
```

Sau đó khai báo useEffect()

```
import React, { useState, useEffect } from 'react';

export const EffectDemo = () => {
  //State
  const [fullName, setFullName] = useState({ name: 'name',
familyName: 'family' });
  const [title, setTitle] = useState('useEffect() in Hooks');

  //useEffect
  useEffect(() => {
    setFullName({ name: 'TrungHC', familyName: 'HCT' });
  });
};
```

```

    return (
      <div>
        <h1>Title: {title}</h1>
        <h3>Name: {fullName.name}</h3>
        <h3>Family Name: {fullName.familyName}</h3>
      </div>
    );
  };
};

```

Sau đó

```

useEffect(() => {
  console.log('useEffect has been called!');
  setFullName({ name: 'TrungHC', familyName: 'HCT' });
}, [fullName.name]);

```

Như vậy hàm `useEffect()` chỉ được gọi 2 lần: 1 lần khi render components, 1 lần khi set name thành "TrungHC".

Vậy nếu chúng ta muốn hàm `useEffect()` chỉ gọi 1 lần khi render components (tương đương với `componentDidMount`) thì như thế nào? Trong trường hợp này ta chỉ cần truyền tham số thứ 2 của `useEffect()` là 1 hàm rỗng []

```

useEffect(() => {
  console.log('useEffect has been called!');
  setFullName({ name: 'TrungHC', familyName: 'HCT' });
}, []);

```

Ví dụ về clean up function

Khi truyền callback vào `useEffect`, callback có thể return một clean up function. Clean up function mang vai trò 'dọn dẹp' những gì đã thực thi trong `useEffect` nếu cần thiết

Ở ví dụ có bộ đếm (component Counter) tăng theo từng giây sử dụng `setInterval` và một button ẩn hiện bộ đếm này. Trong một số trường hợp ta cần bộ đếm bị ẩn đi (hay component Counter bị loại bỏ khỏi view), ta sẽ muốn bộ đếm này dừng lại để tránh một số hành vi không mong muốn. Việc này sẽ được xử lý ở clean up function bằng cách `clearInterval`

```

EffectIntervalExample.jsx
import React, { useState } from "react";
import Counter from "../Counter";

const EffectIntervalExample = () => {
  const [showCounter, setShowCounter] = useState(true);

  const toggleCounter = () => {
    setShowCounter((prevShowCounter) => !prevShowCounter);
  };

  return (
    <div>
      {showCounter && <Counter />}
      <button onClick={toggleCounter}>Toggle Counter</button>
    </div>
  );
};

```

```
);
};
export default EffectIntervalExample;
```

```
Counter.jsx
import React, { useEffect, useState } from "react";

const Counter = () => {
  const [count, setCount] = useState(0);

  useEffect(() => {
    const interval = setInterval(() => {
      setCount((prevCount) => prevCount + 1);
      console.log("Increase count");
    }, 1000);

    return () => {
      clearInterval(interval);
    };
  }, []);
  return (
    <div>
      <h1>Count: {count}</h1>
    </div>
  );
};

export default Counter;
```

Trong trường hợp không có clean up function hoặc clean up function không xử lý việc clearInterval, kể cả khi ấn bộ đếm, ta vẫn sẽ thấy trong console liên tục in ra 'Increase count' => Đây là điều ta không mong muốn xảy ra

useRef

Được sử dụng để

- Lưu trữ tham chiếu giữa các lần re-render của component
- Truy xuất đến thành phần DOM

Việc thay đổi giá trị của useRef không gây ra sự re-render của component như useState, vì vậy, các biến liên quan đến hiển thị UI thì ta nên sử dụng useState, còn lại có thể sử dụng useRef

Syntax

```
import {useRef} from "react"
...
const ref = useRef([Giá trị khởi tạo]);
```

Ví dụ 1: lưu trữ tham chiếu giữa các lần re-render của component

```
RefExample.jsx
```

```
import React, { useRef, useState } from "react";
import "../assets/css/refExample.css";

const RefExample = () => {
  const countRef = useRef(0);
  const countObj = {
    current: 0,
  };

  const [count, setCount] = useState(0);

  const increaseCount = () => {
    countRef.current++;
    countObj.current++;
    setCount(count + 1);
  };

  console.log("count", count);
  console.log("countRef", countRef);
  console.log("countObj", countObj);

  return (
    <button className="increaseBtn" onClick={increaseCount}>
      Rerender
    </button>
  );
};

export default RefExample;
```

Sử dụng 1 state count và với mỗi lần bấm nút, state này sẽ bị thay đổi dẫn tới việc component RefExample sẽ được re-render

Quan sát giá trị của countRef và countObj :

- Với countRef, giá trị current sẽ bị thay đổi sau mỗi lần component re-render do giá trị trước đó vẫn được lưu lại
- Với countObj, giá trị current sẽ giữ nguyên do mỗi lần component re-render thì giá trị của các object thường như này sẽ được khởi tạo lại hoàn toàn

Ví dụ 2: truy xuất đến thành phần DOM:

Trong trường hợp này, ref chính là một element trên DOM, hay cụ thể hơn là input, chúng ta hoàn toàn có thể thao tác với element này như lấy giá trị của input hay click button để focus input này

```
RefDomExample.jsx
import React, { useRef } from "react";

const RefDomExample = () => {
  // Thường khi dùng ref truy xuất DOM giá trị khởi tạo sẽ là null
  const inputRef = useRef(null);

  const getInputValue = () => {
```

```

    console.log("Giá trị của input:", inputRef.current.value);
  };

  const focusInput = () => {
    inputRef.current.focus();
  };

  return (
    <div>
      <input type="text" ref={inputRef} />
      <button onClick={getInputValue}>Log Input Value</button>
      <button onClick={focusInput}>Focus Input</button>
    </div>
  );
};

export default RefDomExample;

```

Bài tập 1: Viết Countdown có dạng như sau:



Bài tập 2: Viết 1 List of todos (có Thêm, xóa, hiển thị)

Hiện thị	Thêm

Bài 1:

```

Timer
import React, { useState, useEffect } from 'react';

const Timer = (props) => {
  const [count, setCount] = useState(10);

  useEffect(() => {

```



```

                                <button type="button"
                                    onClick={ () =>
handleDelete(todo.id) }>
                                Remove</button>
                                </li>
                            </div>
                        )
                    }
                <hr />
            </div>
        )
    }
    export default Task;

```

App.js

```

import Task from './components/ex/Task'
import { ListTask } from './components/ex/ListTask';
import { useState, useEffect } from 'react';

const [todos, setTodos] = useState(ListTask);
const [input, setInput] = useState('');
useEffect(() => {
    // console.log('use effect todos')
}, [todos]);
const handleClick = (event) => {
    if (!input) {
        alert('empty input')
        return;
    }
    //hook not merge state
    //...spread syntax array js
    let newTodo = {
        id: Math.floor((Math.random() * 100000) + 1),
        title: input,
        type: 'me'
    }
    setTodos([...todos, newTodo])
    setInput('')
}

const handleChangeInput = (event) => {
    setInput(event.target.value)
}

const deleteDataTodo = (id) => {
    let currentTodos = todos;
    currentTodos = currentTodos.filter(item => item.id !== id)
    setTodos(currentTodos)
}

<Task
    todos={todos}
    title={'List of todos'}
    deleteDataTodo={deleteDataTodo}

```

```
    />
    <input type="text" value={input}
      onChange={ (event) => handleOnChangeInput (event) } />
    <button type="button"
      onClick={ (event) => handleEventClick (event) }>Add a
      todo</button>
```

7. Single Page Application (SPA)

Hiện nay khi lướt web, ta sẽ thường bắt gặp 2 dạng Web Application:

- Single Page Application (SPA): Facebook, Youtube, Netflix, ...
- Multi Page Application (MPA): Các trang báo ở Việt Nam như Dantri, Vnexpress, ...

Vậy điểm khác biệt cơ bản giữa chúng là gì?

Với SPA:

- Thông thường, chỉ 1 HTML Page và Content của trang web được trình duyệt tải về khi người dùng truy cập trang web
- Việc định tuyến (Routing) khi người dùng truy cập URL nào đó được thực hiện bởi JavaScript
- Cho người dùng cảm giác web giống như một ứng dụng di động, mượt mà hơn khi sử dụng
- Giảm thiểu gánh nặng xử lý cho Server
- Phân tách rõ ràng FrontEnd (Client) và BackEnd (Server)
- Không tốt cho SEO

Với MPA:

- Mỗi HTML Page, Content của trang web sẽ gắn với một URL, khi người dùng truy cập trình duyệt sẽ tải về các file tương ứng với URL do server cung cấp
- Việc định tuyến được thực hiện bởi Server
- Tốt cho SEO

Ví dụ chúng ta có 2 trang là /user và /product

- Với SPA: Trình duyệt chỉ tải 1 lần 1 file HTML và content của toàn bộ trang web, việc hiển thị UI tương ứng với URL sẽ do JavaScript đảm nhiệm, nếu URL là /user JavaScript sẽ chỉ render phần UI của /user
- Với MPA: Trình duyệt sẽ tải file HTML và content do server trả ra tương ứng với URL, có nghĩa là trình duyệt sẽ tải file HTML và content tương ứng khi truy cập /user và tiếp tục tải file HTML và content tương ứng khi truy cập /product

Tips: Khi chuyển qua lại giữa các tab, navigation, ... của một trang web và thấy URL trình duyệt thay đổi nhưng trang web không reload, đó có thể là dấu hiệu của SPA

8. Routing trong ReactJS

Khi truy cập 1 trang web. Nhập đường dẫn (url) sẽ load(render) ra các trang(Component) tương ứng. Để làm được điều này ta cần quản lý routing trong ứng dụng web của chúng ta. Trong reactjs ta có thể dùng thư viện react-router-dom để làm việc này.

Để cài đặt ta sử dụng lệnh:

```
yarn add react-router-dom
```

Hoặc

```
npm i react-router-dom
```

a. Các component cơ bản

- **Router:** Component bao bọc tất cả component khác của routing. Có rất nhiều loại router:
 - Browser Router: Thông dụng nhất, sử dụng HTML5 history API để đồng bộ UI và URL
 - Hash Router: Loại routing có dấu # ở đầu, công dụng để thấy nhất đó là giúp việc deploy trên những trang có nguồn tài nguyên được chia sẻ giữa nhiều người dùng như Github Pages đơn giản hơn
 - Static Router: Thường sử dụng cho Server Side Rendering
- **MemoryRouter:** Thường sử dụng khi Testing, thành địa chỉ sẽ không được cập nhật URL mà giữ trong memory (Hữu ích khi sử dụng React Native)
- **Route:** Component này dùng để render component nếu đường dẫn url trùng với path props của Route component.
- **Switch:** Bao ngoài Route, chỉ render Route đầu tiên match với path URL
- **Redirect:** Điều hướng từ một path này qua path khác, thường đi kèm với điều kiện cụ thể (VD: người dùng chưa login thì khi truy cập web sẽ redirect qua path **/unauthorized**)
- **Link:** Thẻ chuyển hướng tới path tương ứng
- **NavLink:** Tương tự Link nhưng có thêm activeClassName (Hữu ích trong việc CSS tab nào đang active)

Ví dụ

Có `<BrowserRouter>` bao bọc tất cả component, nếu không có sẽ xảy ra lỗi.

```
function App() {  
  return (  
    <div className="App">  
      <BrowserRouter></BrowserRouter>  
      <Route path="/test" component={<div>test</div>} />  
    </div>  
  );  
}
```

Uncaught runtime errors:

ERROR

```
Invariant failed: You should not use <Route> outside a <Router>
    at invariant (http://localhost:3000/static/js/bundle.js:45213:9)
    at http://localhost:3000/static/js/bundle.js:36647:86
    at updateContextConsumer (http://localhost:3000/static/js/bundle.js:27583:23)
    at beginWork (http://localhost:3000/static/js/bundle.js:27956:18)
    at HTMLUnknownElement.callCallback (http://localhost:3000/static/js/bundle.js:12919:18)
    at Object.invokeGuardedCallbackDev (http://localhost:3000/static/js/bundle.js:12963:20)
    at invokeGuardedCallback (http://localhost:3000/static/js/bundle.js:13020:35)
    at beginWork$1 (http://localhost:3000/static/js/bundle.js:32894:11)
    at performUnitOfWork (http://localhost:3000/static/js/bundle.js:32141:16)
    at workLoopSync (http://localhost:3000/static/js/bundle.js:32064:9)
```

Ví dụ: cơ bản về React Router, ta có các Components bao gồm:

- Một header để điều hướng trang web
- Tab đang active sẽ có màu vàng, tab không active sẽ có màu xanh
- Click tab sẽ render ra UI tương ứng, ngoại trừ tab Unauthorized, khi click tab này sẽ tự động chuyển hướng đến Redirected Page nhờ component Redirect
- Page Product sẽ có 1 vài product, khi click sẽ chuyển đến trang tương ứng và hiện ra id của product

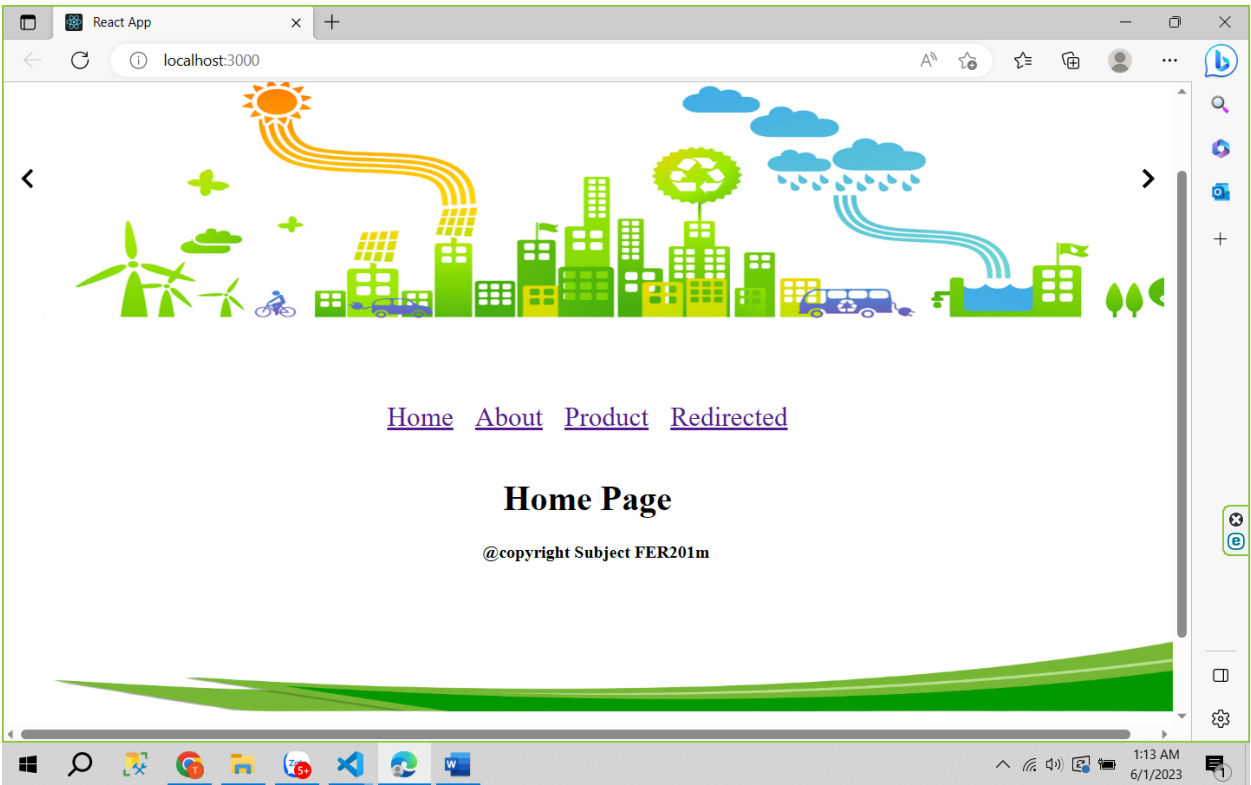
Chú ý:

<Switch> (react-router v5) thì dùng từ khoá 'exact' ở 2 path / và /home vì khi không có từ khóa 'exact', khi qua path /home, nó sẽ match ngay với path / ở Route, khiến cho HomePage được render thay vì AboutPage. Còn khi thay bằng <Routes> (react-router v6) thì không xảy ra như vậy.

Ví dụ:

Switch	Routes
<pre>import {BrowserRouter, Switch, Route} from 'react-router-dom' function App() { return(<BrowserRouter> <Switch> <Route exact path = "/"> <Main Page /> </Route> <Route path = "/user"> <User /> </Route> </Switch> </BrowserRouter>); }</pre>	<pre>import { BrowserRouter, Routes, Route} from 'react-router-dom'; function App() { return (<BrowserRouter> <Routes> <Route path="/" element={<Main Page />} /> <Route path="user/" element={<User />} /> </Routes> </BrowserRouter>); }</pre>

Thực hiện để có kết quả như sau:



(hai-app)

Các Components:

AboutPage	HomePage
<pre>import React from "react"; const AboutPage = () => { return (<div> <h1>About Page</h1> </div>); }; export default AboutPage;</pre>	<pre>import React from "react"; const HomePage = () => { return (<div> <h1>Home Page</h1> </div>); }; export default HomePage;</pre>
RedirectedPage	UnauthorizedPage
<pre>import React from "react"; const RedirectedPage = () => { return (<div> <h1>Redirected Page</h1> </div>); };</pre>	<pre>import React from "react"; import { Redirect } from "react-router"; const UnauthorizedPage = () => { return <Redirect to="/redirected" />; };</pre>

<code>export default RedirectedPage;</code>	<code>export default UnauthorizedPage;</code>
<code>ProductPage</code>	<code>ProductDetailPage</code>
<pre>import React from "react"; import { Link } from "react-router-dom"; const ProductPage = () => { return (<div style={{ display: "flex", flexDirection: "column" }}> <Link to="/product/1">Product 1</Link> <Link to="/product/2">Product 2</Link> <Link to="/product/3">Product 3</Link> </div>); }; export default ProductPage;</pre>	<pre>import React from "react"; import { useParams } from "react-router"; const ProductDetailPage = () => { const { id } = useParams(); return <div>Đây là Product ID: {id}</div>; }; export default ProductDetailPage;</pre>
<code>App.css</code>	
<pre>.App { text-align: center; } .header { list-style: none; display: flex; justify-content: center; flex-direction: row; padding: 0; }</pre>	<pre>.header-item { margin: 20px 10px; font-size: 25px; } .active-header-item { color: rgb(255, 196, 0); }</pre>

Banner:

<code>banner.css</code>
<pre>img { height: 40vh; margin-left: 5vh; width: 180vh; } .each-fade { margin-top: 10vh; margin-bottom: 5vh; }</pre>
<pre>import React from "react"; import { Fade } from 'react-slideshow-image'; import 'react-slideshow-image/dist/styles.css';</pre>

```

import './banner.css'
const fadeImages = [
  {
    url: '/images/slide_1.png',
    caption: 'Slide 1'
  },
  {
    url: '/images/slide_2.png',
    caption: 'Slide 2'
  },
  {
    url: '/images/slide_3.png',
    caption: 'Slide 3'
  },
]

];
const Banner = () => {
  return (
    <div>
      <Fade>
        {fadeImages.map((fadeImage, index) => (
          <div className="each-fade" key={index}>
            <div className="image-container">
              <img src={fadeImage.url} />
            </div>
          </div>
        ))}
      </Fade>
    </div>
  )
}
export default Banner;

```

npm i react-slideshow-image -S

Chân trang

```

footer.css
footer {
  display: flex;
  align-items: center;
  position: relative;
  margin-top: -9rem;
  margin-bottom: -2rem;
}

.f-content {
  position: absolute;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
}

```



```

        width: 100%;
        gap: 4rem;
    }

import React from 'react'
import Wave from '../images/wave.png'
import './footer.css'

const Footer = () => {
    return (
        <div className="footer">
            <div className="f-content">
                <span style={{ fontWeight: 'bold' }}>@copyright
Subject FER201m</span>
            </div>
            <img src={Wave} alt=""
                style={{ width: '100%' }}
            />
        </div >
    )
}

export default Footer

```

Các phần tiếp:

```

App
import './App.css';
import React from "react";

import Banner from './components/Banner';
import AboutPage from './components/AboutPage';
import HomePage from './components/HomePage';
import ProductPage from './components/ProductPage';
import RedirectedPage from './components/RedirectedPage';
import { NavLink, Routes, Route } from 'react-router-dom';

function App() {
    return (
        <div className="App">
            <Banner />
            <ul className="header">
                <li className="header-item">
                    <NavLink to="/" activeClassName="active-header-item">
                        Home
                    </NavLink>
                </li>
                <li className="header-item">
                    <NavLink to="/about" activeClassName="active-header-
item">
                        About

```

```

        </NavLink>
      </li>
      <li className="header-item">
        <NavLink to="/product" activeClassName="active-header-
item">
          Product
        </NavLink>
      </li>

      <li className="header-item">
        <NavLink
          to="/redirected"
          activeClassName="active-header-item"
        >
          Redirected
        </NavLink>
      </li>
    </ul>
    <Routes>
      <Route path="/" element={<HomePage />} />
      <Route path="/home" element={<HomePage />} />
      <Route path="/about" element={<AboutPage />} />
      <Route path="/product" element={<ProductPage />} />
      <Route path="/redirected" element={<RedirectedPage />} />
    </Routes>
  </div>
);
}
export default App;

```

Cách khác:

```

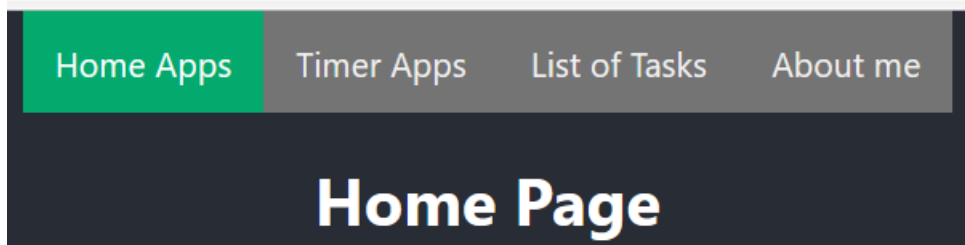
Routes.js
import { createBrowserRouter } from "react-router-dom";
import Home from "../pages/Home/Home";
import SignIn from "../pages/SignIn/SignIn";
import SignUp from "../pages/SignUp/SignUp";

const Routes = createBrowserRouter([
  {
    path: "/",
    element: <Home/>
  },
  {
    path: "/login",
    element: <SignIn />
  },
  {
    path: "/signup",
    element: <SignUp />
  }
]);

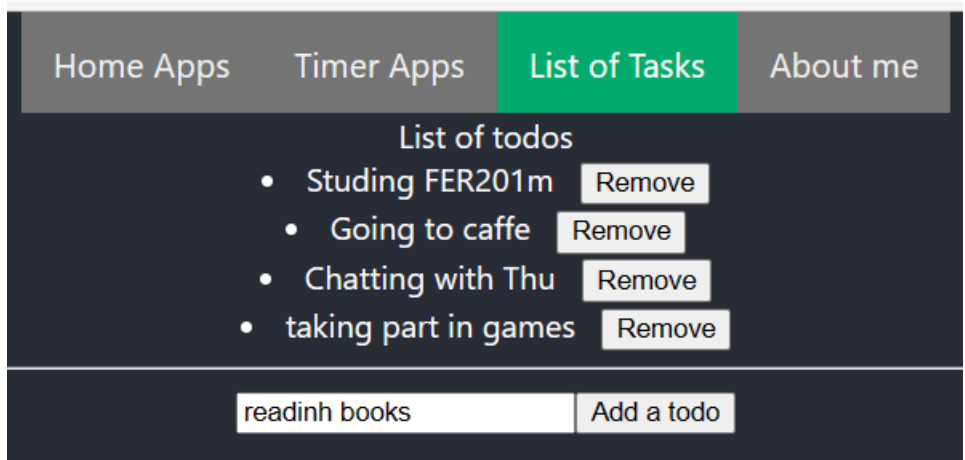
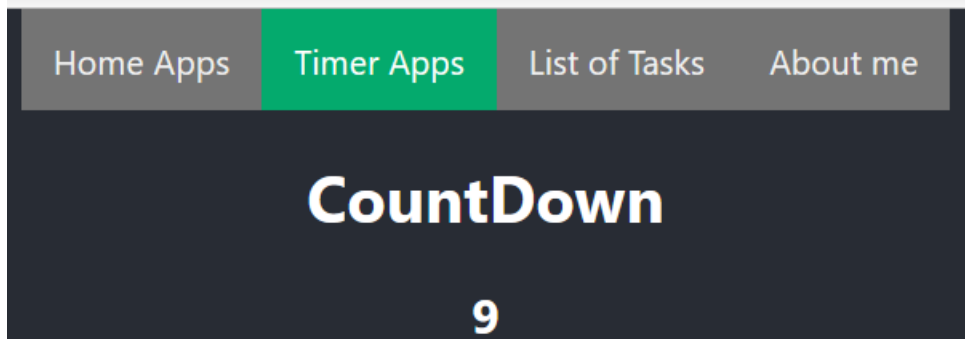
```

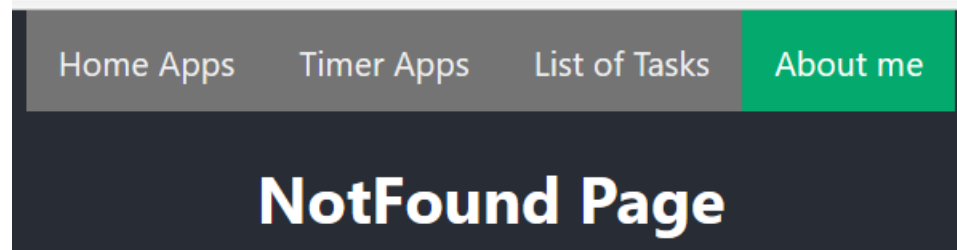
<pre>export default Routes;</pre>
App
<pre>import { RouterProvider } from "react-router-dom"; import routes from "../Router"; const App = () => { return (<RouterProvider router={routes} />); }; export default App;</pre>

Bài tập: Viết 1 app có dạng như sau



Các chức năng tiếp theo lấy từ bài tập 1 và 2:





(my-app)

9. Controlled Forms

Trong React.js, có hai phương pháp để xử lý biểu mẫu: biểu mẫu kiểm soát (controlled forms) và biểu mẫu không kiểm soát (uncontrolled forms). Sự khác biệt chính giữa chúng nằm ở cách quản lý và cập nhật dữ liệu biểu mẫu.

Biểu mẫu kiểm soát là các biểu mẫu mà dữ liệu biểu mẫu được quản lý bởi các **React components**. Trong **Controlled Forms**, các phần tử form input (ví dụ: **input fields, checkboxes, selects**) được liên kết với **component's state**, và bất kỳ thay đổi nào trong các phần tử input đều kích hoạt việc cập nhật **component's state**. Sau đó, **components** được cập nhật và hiển thị dữ liệu biểu mẫu đã được cập nhật.

Đặc điểm quan trọng của Controlled Forms

- Dữ liệu biểu mẫu được lưu trữ trong component's state.
- Các trường nhập liệu nhận giá trị từ state.
- Các thay đổi trong các trường nhập liệu được xử lý bởi các trình xử lý sự kiện, cập nhật state với các giá trị mới.
- Component được cập nhật với dữ liệu biểu mẫu đã được cập nhật.

Ví dụ 1:

```
const [inputa, setInputa] = useState(0);
const handleOnChangeInputa = (e) => {
  setInputa(e.target.value)
}
<div>
  <div>
    <label htmlFor="inputa">Input number 1:</label>
    <input type="number" value={inputa}
      onChange={ (e) => handleOnChangeInputa(e) } />
  </div>
  <h2>Number 1: {inputa}</h2>
</div>
```

Ví dụ 2:

```
.table-student {
  width: 100%;
  border: 1px solid #ccc;
}
.head-student {
```

```

        font-weight: 600;
    }
    .head-student tr,
    .body-student tr {
        display: flex;
        align-items: center;
        padding-left: 20px;
    }
    .head-student td,
    .body-student td {
        display: flex;
        align-items: center;
        justify-content: center;
        padding: 12px 0;
        flex: 1;
    }
    .body-student tr:nth-child(2n+1) {
        background-color: #ddd;
    }
}

```

```

import React, { useState } from "react";
import './form.css'
const ControlledForms = () => {
    const [formData, setFormData] = useState({
        name: "",
        email: "",
        age: 0,
        genderValue: 'male',
    });
    const handleChange = (e) => {
        const { name, value } = e.target;
        setFormData({ ...formData, [name]: value });
    };
    const handleSubmit = (e) => {
        e.preventDefault();
        console.log(formData);
        alert(JSON.stringify(formData));
    };
    return (
        <div>
            <form onSubmit={handleSubmit}>
                <label>
                    Name:
                    <input
                        type="text"
                        name="name"
                        value={formData.name}
                        onChange={handleChange}
                    />
                </label>
                <br />
            </form>
        </div>
    );
}

```

```

<label>
  Email:
  <input
    type="email"
    name="email"
    value={formData.email}
    onChange={handleChange}
  />
</label>
<br />
<label>
  Age:
  <input
    type="number"
    name="age"
    value={formData.age}
    onChange={handleChange}
  />
</label>
<br />
<span className="">Gender:</span>
<input
  value="male"
  onChange={handleChange}
  checked={formData.genderValue === 'male'}
  type="radio" name="genderValue" id="male" />
<label htmlFor="male">Male</label>
<input
  value="female"
  onChange={handleChange}
  checked={formData.genderValue === 'female'}
  type="radio" name="genderValue" id="female" />
<label htmlFor="female">Female</label>
<br />
<button type="submit">Save</button>
</form>
<table className="table-student">
  <thead className="head-student">
    <tr>
      <td>Name</td>
      <td>Email</td>
      <td>Age</td>
      <td>Gender</td>
    </tr>
  </thead>
  <tbody className="body-student">
    <tr>
      <td>{formData.name}</td>
      <td>{formData.email}</td>
      <td>{formData.age}</td>
      <td>{formData.genderValue}</td>
    </tr>
  </tbody>
</table>

```

```
        </tr>
      </tbody>
    </table>
  </div >
);
}
export default ControlledForms;
```

Lợi ích của controlled forms

- Có đầy đủ kiểm soát về dữ liệu biểu mẫu và có thể thực hiện validations, transformations, hoặc xử lý các logic khác trước khi cập nhật state hoặc gửi biểu mẫu.
- Có thể dễ dàng triển khai các tính năng như validations form, hoặc phản hồi thời gian thực cho người dùng.
- Dữ liệu biểu mẫu có thể dễ dàng điều chỉnh hoặc xử lý trước khi gửi đến máy chủ.

Bất lợi của controlled forms

- Controlled forms có thể yêu cầu nhiều code và xử lý sự kiện hơn so với Uncontrolled forms.
- Nếu có một biểu mẫu lớn với nhiều trường nhập liệu, việc quản lý state cho mỗi trường có thể trở nên phức tạp.

Ví dụ cho bài CRUD (step by step)

Tạo app

```
npx create-react-app crud-demo1
```

Chạy app

```
npm start
```

Dữ liệu (students.js)

```
export const ListStudents =
[
  {
    "id": 1,
    "name": "Nguyễn Văn An",
    "age": 25,
    "gender": "male"
  },
  {
    "id": 2,
    "name": "Trần Thị Bình",
    "age": 30,
    "gender": "female"
  },
  {
    "id": 3,
    "name": "Lê Hoàng Cước",
    "age": 40,
    "gender": "male"
  },
  {
```

```
    "id": 4,  
    "name": "Phạm Thị Diệu",  
    "age": 22,  
    "gender": "female"  
  },  
  {  
    "id": 5,  
    "name": "Võ Thành An",  
    "age": 28,  
    "gender": "male"  
  },  
  {  
    "id": 6,  
    "name": "Đỗ Thị Diệu",  
    "age": 35,  
    "gender": "female"  
  },  
  {  
    "id": 7,  
    "name": "Nguyễn Văn Thành",  
    "age": 50,  
    "gender": "male"  
  },  
  {  
    "id": 8,  
    "name": "Hoàng Thị Hải",  
    "age": 27,  
    "gender": "female"  
  },  
  {  
    "id": 9,  
    "name": "Nguyễn Thành Nam",  
    "age": 33,  
    "gender": "male"  
  },  
  {  
    "id": 10,  
    "name": "Nguyễn Văn Kiên",  
    "age": 26,  
    "gender": "male"  
  },  
  {  
    "id": 11,  
    "name": "Nguyễn Ngọc Bách",  
    "age": 17,  
    "gender": "male"  
  },  
  {  
    "id": 12,  
    "name": "Nguyễn Thu Hiền",  
    "age": 19,
```



```
        "gender": "female"
    }
]
```

Dùng form

LIST OF STUDENT (12)
List student

Student id

Enter ID

Student Name

Enter Name

Age

Enter Age

Gender: ☒ Male ☐ Female

Có css

```
* {
    margin: 0;
    padding: 0;
    font: inherit;
}
body {
    min-height: 100vh;
    font-family: 'Roboto', sans-serif;
    background-color: #ebefff;
}
/* content */
.content {
    max-width: 1230px;
    margin: 30px 50px;
    border: 1px solid #ccc;
    background-color: #fff;
    padding: 25px;
    box-shadow: 0 15px 30px 0 rgba(0, 0, 0, 0.2);
}
.title h1 {
    font-size: 20px;
    font-weight: 600;
    text-transform: uppercase;
}
.title h3 {
    margin-bottom: 20px;
}
.info {
    display: flex;
    flex-direction: column;
```

```

    gap: 16px;
  }
  .field {
    display: flex;
    gap: 6px;
    flex-direction: column;
  }
  .field input {
    outline: none;
    border: 1px solid #bbb;
    border-radius: 4px;
    padding: 8px 12px;
  }
  .info-gender {
    display: flex;
    gap: 12px;
    align-items: center;
  }
  .male-gender,
  .female-gender {
    display: flex;
    gap: 4px;
    align-items: center;
  }
}

```

Component CrudDemo.js

```

import { ListStudents } from '../data/students';
import React, { useState } from 'react';

const init = {
  id: '',
  name: '',
  age: '',
  gender: 'male',
}

const [students, setStudents] = useState(ListStudents);
const [info, setInfo] = useState(init);
const [update, setUpdate] = useState(false);
const handleInfo = (e) => {
  setInfo({
    ...info,
    [e.target.name]: e.target.value
  })
}

<div className="content">
  <div className="title">
    <h1>
      List of student
    (<span>{students.length}</span>)
    </h1>
    <h3>List student</h3>
  </div>

```

```

        </div>
        <div className="info">
            <div className="field">
                <label>Student id</label>
                <input
                    readOnly={update}
                    value={info.id}
                    onChange={handleInfo}
                    name='id'
                    id="studentID" type="text"
placeholder="Enter ID" />
            </div>
            <div className="field">
                <label>Student Name</label>
                <input
                    value={info.name}
                    onChange={handleInfo}
                    name='name'
                    id="studentName" type="text"
placeholder="Enter Name" />
            </div>
            <div className="field">
                <label>Age</label>
                <input
                    value={info.age}
                    onChange={handleInfo}
                    name='age'
                    id="studentAge" type="number"
placeholder="Enter Age" />
            </div>
            <div className="info-gender">
                <span>Gender:</span>
                <div className="male-gender">
                    <input
                        value="male"
                        onChange={handleInfo}
                        checked={info.gender === 'male'}
                        type="radio" name="genderValue"
id="male" />
                    <label>Male</label>
                </div>
                <div className="female-gender">
                    <input
                        value="female"
                        onChange={handleInfo}
                        checked={info.gender === 'female'}
                        type="radio" name="genderValue"
id="female" />
                    <label>Female</label>
                </div>
            </div>
        </div>
    </div>

```

```

    </div>
  </div>

```

Xử lý thêm-xoá-sửa

```

//tìm 1 sinh viên qua id
const getStudentById = (id) => {
  for (let i = 0; i < students.length; i++) {
    if (students[i].id.toString() === id.toString()) return
    students[i]
  }
  return null;
}
//Hàm xử lý thêm 1 sinh viên
const addStudent = (student) => {
  const newStudents = [...students]
  newStudents.push(student)
  setStudents(newStudents)
}
//Hàm xử lý Xóa 1 sinh viên
const deleteStudent = (id) => {
  const confirmDelete = prompt('Bạn muốn xóa sản phẩm có id là '
+ id + '?', 'ok')
  if (confirmDelete) {
    const newStudents = students.filter(student =>
student.id.toString() !== id.toString());
    setStudents(newStudents)
  }
}
//Hàm xử lý cập nhật 1 sinh viên
const updateStudent = (student) => {
  const fineIndex = students.findIndex(studentTmp =>
studentTmp.id.toString() === student.id.toString())
  const newStudent = [...students]
  newStudent[fineIndex] = student
  setStudents(newStudent)
}

```

Cập nhật lại việc thêm (nếu lưu trữ thì không cần):

```

const studentsCurent = useRef(null)
//Hàm xử lý thêm 1 sinh viên
const addStudent = (student) => {
  const newStudents = [...students]
  newStudents.push(student)
  setStudents(newStudents)
  studentsCurent.current = newStudents;
}

```

Tương tự với xoá và sửa cũng vậy

Thêm css cho Button

```

.btn {
  display: flex;

```

```
    align-items: center;
    gap: 8px;
    margin: 16px 0 28px;
  }
  button {
    padding: 8px 16px;
    border-radius: 4px;
    color: #fff;
    margin-right: 10px;
    transition: all linear 0.3s;
  }
  button:hover {
    filter: brightness(0.8)
  }
}
```

Thêm nút Thêm:

```
<div className="btn">
  <button
    onClick={handleSaveStudent}
    style={{
      backgroundColor: '#007bff'
    }} className="btn-save">Save</button>
</div>
```

Thêm css cho bảng

```
.table-student {
  width: 100%;
  border: 1px solid #ccc;
}
.head-student {
  font-weight: 600;
}
.head-student tr,
.body-student tr {
  display: flex;
  align-items: center;
  padding-left: 20px;
}
.head-student td,
.body-student td {
  display: flex;
  align-items: center;
  justify-content: center;
  padding: 12px 0;
  flex: 1;
}
```

Thêm bảng

```
<table className="table-student">
  <thead className="head-student">
    <tr>
      <td>ID</td>
      <td>Name</td>
```

```

        <td>Age</td>
        <td>Gender</td>
        <td>Action</td>
    </tr>
</thead>
<tbody className="body-student">

    </tbody>
</table>

```

LIST OF STUDENT (12)

List student

Student id

Student Name

Age

Gender: ☒ Male ☐ Female

ID	Name	Age	Gender	Action
----	------	-----	--------	--------

Hiển thị dữ liệu ra bảng, xử lý chọn dòng (để cập nhật)

```

{students.map(student => (
    <tr
        onClick={ (e) => {
            if (e.target.textContent !==
'Delete') {
                setInfo({
                    id: student.id,
                    name: student.name,
                    age: student.age,
                    genderValue:
student.gender,
                })
                setUpdate(true)
            }
        }}
        key={student.id}>
        <td>{student.id}</td>
        <td>{student.name}</td>

```

```

                <td>{student.age}</td>
                <td>{student.gender}</td>
                <td>
                    <button
                        style={{
                            backgroundColor: '#dc3545'
                        }}
                        onClick={() =>
deleteStudent(`${student.id}`)}
                        className="btn-
delete">Delete</button>
                </td>
            </tr>
        ))}

```

Thêm sinh viên

```

const handleSaveStudent = () => {
    const student = {
        id: info.id,
        name: info.name,
        age: info.age,
        gender: info.gender
    }
    if (info.id == "" || info.name == "" || info.age == "") {
        alert('Vui lòng nhập hết các ô bên dưới')
    }
    else if (getStudentById(info.id)) {
        alert('ID đã tồn tại, vui lòng chọn ID khác')
    }
    else {
        setInfo(init)
        addStudent(student)
    }
}

```

Button thêm

```

<button
    onClick={handleSaveStudent}
    style={{
        backgroundColor: '#007bff'
    }} className="btn-save">Save</button>

```

Xử lý cập nhật

```

const handleUpdateStudent = () => {
    if (update) {
        const student = {
            id: info.id,
            name: info.name,
            age: info.age,
            gender: info.genderValue
        }
    }
}

```

```
        setInfo(init)
        setUpdate(false)
        updateStudent(student)
    }
    else {
        alert('Vui lòng click vào student để thực hiện
update')
    }
}
```

Button cập nhật

```
<button
                                onClick={handleUpdateStudent}
                                style={{
                                    backgroundColor: '#6c757d'
                                }} className="btn-update">Update</button>
```

Thay đổi khi cập nhật xong

```
useEffect(() => {
    setUpdate(false)
    studentsCurent.current = students
}, [])
```

Tô màu các dòng lẻ

```
.body-student tr:nth-child(2n+1) {
    background-color: #ddd;
}
```

Sắp xếp theo tên

```
const handleSortStudent = () => {
    const newStudents = [...students].sort((studentA,
studentB) => {
        return studentA.name.localeCompare(studentB.name)
    })
    setStudents(newStudents);
}
```

Button sắp xếp

```
<button
                                onClick={handleSortStudent}
                                style={{
                                    backgroundColor: '#28a745'
                                }} className="btn-sort">Sort by name</button>
```

Sắp xếp theo tuổi


```
const handleSortAgeStudent = () => {
  const newStudents = [...students].sort((studentA,
studentB) => {
    return studentA.age - studentB.age;
  })
  setStudents(newStudents);
}
```

Button sắp xếp

```
<button
  onClick={handleSortAgeStudent}
  style={{
    backgroundColor: '#28a745'
  }} className="btn-sort">Sort by age</button>
```

Tìm kiếm

```
const handleSearchStudent = () => {
  const search = prompt('Nhập vào tìm kiếm');
  if (search) {
    const searchStudents =
[...studentsCurent.current].filter(student =>
  student.name.toLowerCase().includes(search.toLowerCase
Case()))
    setStudents(searchStudents);
  }
}
```

Button tìm kiếm

```
<button onClick={handleSearchStudent}
  style={{
    backgroundColor: '#17a2b8'
  }} className="btn-search">Search</button>
```

Show all

```
const handleShowStudent = () => {
  const allStudents = [...studentsCurent.current];
  setStudents(allStudents);
}
```

Button show all

```
<button onClick={handleShowStudent}
  style={{
    backgroundColor: '#17FF34'
  }} className="btn-show">Show all</button>
```

(crud-demo)

Bài tập

Cho dữ liệu sinh viên như sau:

```
export const data =
[
  {
    "id": 1,
```

```
        "name": "Lê Bá Trọng",
        "dob": "12/06/2000",
        "gender": "Male",
        "score": 9
    },
    {
        "id": 2,
        "name": "Lại Thị An",
        "dob": "05/04/2002",
        "gender": "Female",
        "score": 6.5
    },
    {
        "id": 3,
        "name": "Vũ Tuấn Anh",
        "dob": "17/11/2001",
        "gender": "Male",
        "score": 4
    },
    {
        "id": 4,
        "name": "Trịnh Thị Châu",
        "dob": "12/04/2002",
        "gender": "Female",
        "score": 5.5
    },
    {
        "id": 5,
        "name": "Lê Vân Anh",
        "dob": "17/07/2003",
        "gender": "Female",
        "score": 9
    }
]
```

Viết chương trình thực hiện các chức năng như hình dưới:

←

↻

localhost:3000

🏠

🔍

☆

⌵

List of Students

Fullname	Date of Birth	Gender	Score	Actions
Lê Bá Trọng	12/06/2000	Male	9	<button>Delete</button>
Lại Thị An	05/04/2002	Female	6.5	<button>Delete</button>
Vũ Tuấn Anh	17/11/2001	Male	4	<button>Delete</button>
Trịnh Thị Châu	12/04/2002	Female	5.5	<button>Delete</button>
Lê Văn Anh	17/07/2003	Female	9	<button>Delete</button>

Sort by name

Sort by DOB

Sort by score

Add a student

Css (có thể tự làm)

```

table {
  border-collapse: collapse;
  width: 80%;
  border-color: red;
  margin-left: 5px;
}
th,
td {
  padding: 8px;
}
th {
  background-color: #ff6200;
  color: #000;
}
tr:hover {
  background-color: #ddd;
}
#buttonDelete {
  width: 100%;
  height: 30px;
}
.inputAdd {
  width: 10%;
  height: 25px;
  margin-right: 5px;
}
#buttonAdd {
  width: 5%;
  height: 30px;
}

```

```
h1,
h2,
#input-wrap {
  margin-left: 5px;
}
#button-wrap {
  text-align: right;
  margin-top: 20px;
  margin-right: 280px;
}
.buttonOther {
  width: 150px;
  height: 30px;
  margin-left: 10px;
}
```

Chú ý:

- Mã số sinh viên không nhập vào (gợi ý lấy max của các id +1)
- Khi lựa chọn type='date' thì thời gian có dạng 'yyyy-MM-dd' phải chuyển sang 'dd/mm/yyyy'
- Sắp xếp theo tên (tên rồi đến họ)
- Sắp xếp theo DOB thì năm, tháng, ngày

(solution-crud-react-bail)

10.Uncontrolled Forms

Uncontrolled Forms là biểu mẫu trong đó dữ liệu biểu mẫu được xử lý bởi các phần tử DOM, thay vì được điều khiển bởi các React component. Trong Uncontrolled Forms, các trường nhập liệu không được liên kết một cách rõ ràng với component's state. Thay vào đó, chúng ta sử dụng các tham chiếu (refs) để truy cập giá trị trường nhập liệu khi cần thiết.

Đặc điểm quan trọng của Uncontrolled Forms

- Dữ liệu biểu mẫu được quản lý trực tiếp bởi các phần tử DOM.
- Trường nhập liệu được truy cập bằng cách sử dụng tham chiếu (refs) để lấy giá trị khi cần thiết.
- Không có quản lý state rõ ràng cho dữ liệu biểu mẫu.
- Các thay đổi trong trường nhập liệu không được theo dõi hoặc xử lý một cách rõ ràng bởi các React components.

Ví dụ:

```
import React, { useRef } from "react";

function UncontrolledForms() {
  const nameInputRef = useRef();
  const emailInputRef = useRef();
  const passwordInputRef = useRef();
  const handleSubmit = (e) => {
    e.preventDefault();
    // Access form data using refs
    const name = nameInputRef.current.value;
    const email = emailInputRef.current.value;
    const password = passwordInputRef.current.value;
    // Perform form submission logic here
  }
}
```

```
        console.log(name, email, password);
        alert(name + "      " + email + "      " + password);
        // Reset the form
        e.target.reset();
    };
    return (
        <form onSubmit={handleSubmit}>
            <label>
                Name:
                <input type="text" ref={nameInputRef} />
            </label>
            <br />
            <label>
                Email:
                <input type="email" ref={emailInputRef} />
            </label>
            <br />
            <label>
                Password:
                <input type="password" ref={passwordInputRef} />
            </label>
            <br />
            <button type="submit">Save</button>
        </form>
    );
}
export default UncontrolledForms;
```

Lợi ích của Uncontrolled forms

- Biểu mẫu không được điều khiển đơn giản hơn và yêu cầu ít code hơn so với controlled forms.
- Chúng hữu ích cho các biểu mẫu đơn giản không yêu cầu quản lý state phức tạp.
- Dễ dàng truy cập trực tiếp giá trị trường nhập liệu khi cần thiết, ví dụ như trong quá trình gửi biểu mẫu.

Bất lợi của Uncontrolled forms

- Bị hạn chế về việc kiểm soát dữ liệu biểu mẫu và không thể dễ dàng thực hiện kiểm tra hoặc biến đổi dữ liệu trước khi gửi biểu mẫu.
- Có thể khó khăn khi triển khai các tính năng biểu mẫu nâng cao như validations thời gian thực hoặc tương tác giữa các trường phụ thuộc.
- Việc thao tác hoặc xử lý dữ liệu biểu mẫu trước khi gửi nó đến máy chủ có thể đòi hỏi các bước bổ sung.

11.Template cho app

Cài đặt Bootstrap 4

```
npm i bootstrap@4.6.2
```

```
npm i reactstrap react react-dom  
npm i popper.js@1.16.1
```

Cấu hình sử dụng Bootstrap 4

index.js

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

Nếu sử dụng:

```
import { Navbar, NavbarBrand } from 'reactstrap';  
Ví dụ:  
<Navbar dark color="primary">  
  <div className="container">  
    <NavbarBrand href="/">Ristorante Con  
Fusion</NavbarBrand>  
  </div>  
</Navbar>
```

```
import { Card, CardImg, CardImgOverlay, CardText, CardBody,  
  CardTitle } from 'reactstrap';
```

Ví dụ:

```
renderDish(dish) {  
  if (dish !== null)  
    return(  
      <Card>  
        <CardImg top src={dish.image} alt={dish.name}  
      />  
        <CardBody>  
          <CardTitle>{dish.name}</CardTitle>  
          <CardText>{dish.description}</CardText>  
        </CardBody>  
      </Card>  
    );  
  else  
    return(  
      <div></div>  
    );  
}  
const menu = this.props.dishes.map((dish) => {  
  return (  
    <div className="col-12 col-md-5 m-1">  
      <Card key={dish.id}  
        onClick={() => this.onDishSelect(dish)}>  
        <CardImg width="100%" src={dish.image}  
alt={dish.name} />  
        <CardImgOverlay>  
          <CardTitle>{dish.name}</CardTitle>  
        </CardImgOverlay>  
      </Card>  
    </div>  
  );  
});
```

Sử dụng: Font Awesome Icons and Bootstrap-Social

```
Npm i font-awesome@4.7.0
```

```
Npm i bootstrap-social@5.1.1
```

```
npm i react-slideshow-image -S
```

Cấu hình

```
import 'font-awesome/css/font-awesome.css';
```

```
import 'bootstrap-social/bootstrap-social.css';
```

Cài đặt React Router

```
npm i react-router-dom
```

12.Step by step tạo project với bootstrap 4:

Tạo app:

```
npx create-react-app reactjs-bootstrap
```

Cài Bootstrap 4

```
npm i bootstrap@4.6.2
```

```
npm i reactstrap react react-dom
```

```
npm i popper.js@1.16.1
```

Cấu hình sử dụng Bootstrap 4

```
index.js
```

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

```
import 'bootstrap/dist/js/bootstrap.js';
```

Cài đặt React Router

```
npm i react-router-dom
```

Chạy app

```
npm start
```

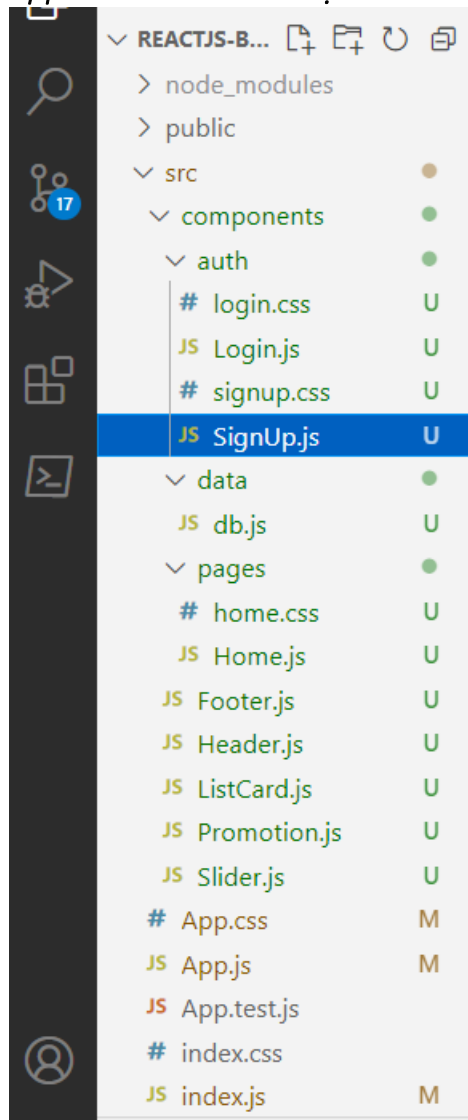
Sử dụng: Font Awesome Icons

```
npm i --save @fortawesome/free-solid-svg-icons @fortawesome/react-fontawesome
```

Cài react bootstrap:

```
npm i --save react-bootstrap
```

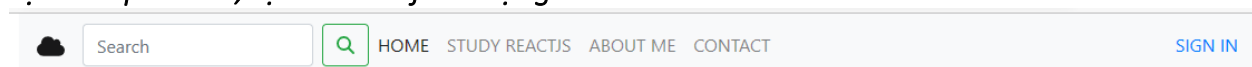
App có cấu hình thư mục như sau:



App.css

```
* {
  box-sizing: border-box;
}
body {
  margin: 0;
  padding: 0;
}
```

Tạo components, tạo Header.js có dạng



```
import React from "react";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
```



```

import { faCloud, faMagnifyingGlass } from "@fortawesome/free-solid-svg-icons";
import { Link } from "react-router-dom";
const Header = () => {
  return (
    <header className="header">
      <div className="container">
        <nav className="navbar navbar-expand-md navbar-light bg-light">
          <Link className="navbar-brand" to="/home">
            <FontAwesomeIcon icon={faCloud} />
          </Link>
          <button
            className="navbar-toggler"
            type="button"
            data-toggle="collapse"
            data-target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent"
            aria-expanded="false"
            aria-label="Toggle navigation"
          >
            <span className="navbar-toggler-icon"></span>
          </button>
          <div className="collapse navbar-collapse"
            id="navbarSupportedContent">
            <form className="form-inline my-2 my-lg-0">
              <input
                className="form-control mr-sm-2"
                type="search"
                placeholder="Search"
                aria-label="Search"
              />
              <button
                className="btn btn-outline-success my-2 my-sm-0"
                type="submit"
              >
                <FontAwesomeIcon icon={faMagnifyingGlass} />
              </button>
            </form>
            <ul className="navbar-nav mr-auto">
              <li className="nav-item active">
                <Link className="nav-link" to="/home">
                  HOME
                </Link>
              </li>
              <li className="nav-item">
                <Link className="nav-link" to="/study">
                  STUDY REACTJS
                </Link>
              </li>
              <li className="nav-item">

```

```
<Link className="nav-link" to="/about">
  ABOUT ME
</Link>
</li>
<li className="nav-item">
<Link className="nav-link" to="/contact">
  CONTACT
</Link>
</li>
</ul>
<Link to="/login">SIGN IN</Link>
</div>
</nav>
</div>
</header>
  );
};
export default Header;
```

Trong components tạo pages, tạo Home.js

```
import "../App.css";
import Header from "../Header";
function Home() {
  return (
    <div>
      <Header />
    </div>
  );
}
export default Home;
```

App.js

```
<BrowserRouter>
  <div className="App">
    <Routes>
      <Route exact path="/" element={<Home />} />
      <Route exact path="/home" element={<Home />} />
    </Routes>
  </div>
</BrowserRouter>
```

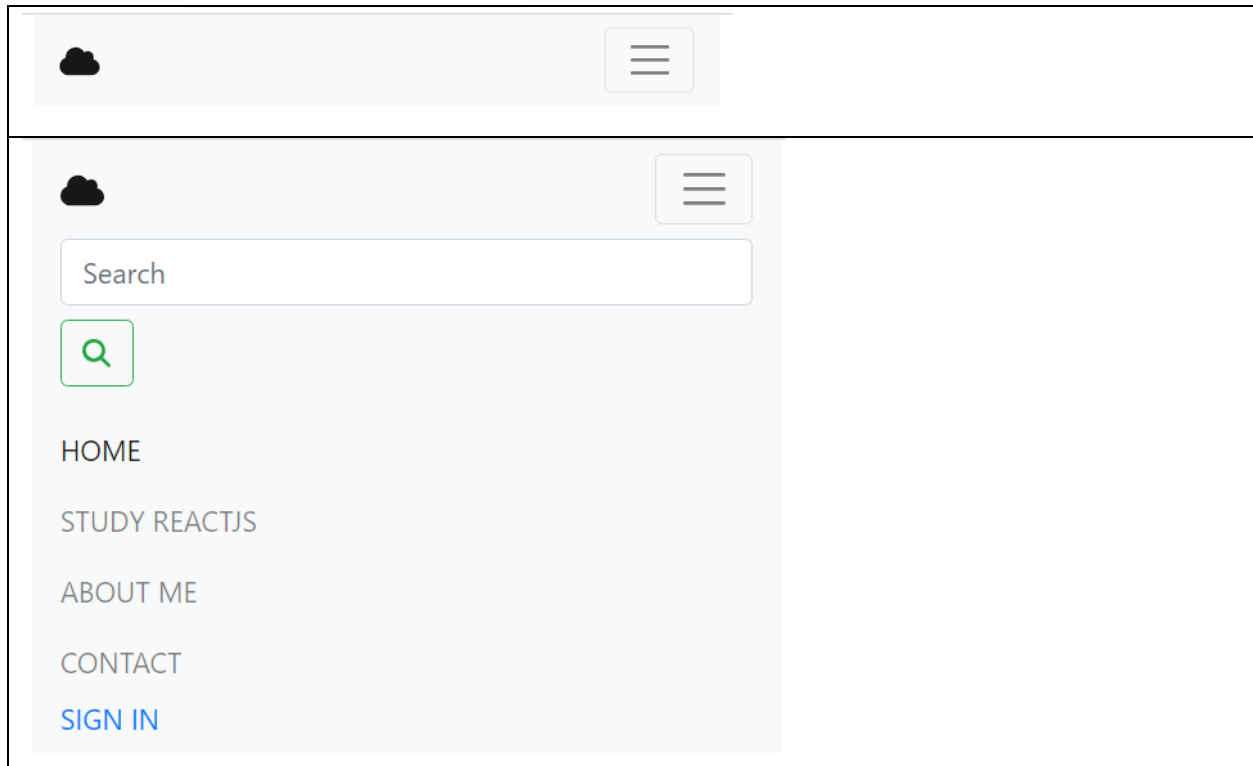
Chạy app

<http://localhost:3000/>



HOME STUDY REACTJS ABOUT ME CONTACT

[SIGN IN](#)



Tạo Slider có dạng:



Tạo tệp lưu trữ dữ liệu (categories và images):

```
export const categories =
[
  {
    title: "Category1",
  },
  {
    title: "Category2",
  },
  {
    title: "Category3",
  },
];
export const images = [
  {
    src: "https://images.unsplash.com/photo-1682905926517-6be3768e29f0?ixlib=rb-4.0.3&ixid=MnwxMjA3fDF8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=687&q=80",
  },
];
```

```
    },  
    {  
      src: "https://images.unsplash.com/photo-1508020963102-c6c723be5764?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG90bylwyWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=1170&q=80",  
    },  
    {  
      src: "https://images.unsplash.com/photo-1514477917009-389c76a86b68?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG90bylwyWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=1067&q=80",  
    },  
  ],  
]
```

Css cho slider:

```
.slide-section {  
  margin: 20px 0 20px 0;  
}  
.slide-section .carousel img {  
  height: 150px;  
  object-fit: cover;  
}  
.slide-section .dropdown {  
  display: none;  
}  
@media (max-width: 767.98px) {  
  .slide-list {  
    display: none;  
  }  
}  
@media (max-width: 575.98px) {  
  .slide-section .dropdown {  
    display: block;  
  }  
  .slide-section .list-group {  
    display: none;  
  }  
}
```

Tạo Component Slider.js

```
import { Link } from "react-router-dom";  
import { categories } from '../data/db'  
import { images } from '../data/db'  
const Slider = () => {  
  const active = 0;  
  return (  
    <section className="slide-section">  
      <div className="container">
```

```

<div className="row">
  <div className="col-12 col-md-4">
    <div className="dropdown">
      <button
        className="btn btn-secondary
dropdown-toggle"
        type="button"
        data-toggle="dropdown"
        aria-expanded="false"
      >
        Category
      </button>
      <div className="dropdown-
menu">
        {categories.map((c) => (
          <Link
            to="/category"
            className="list-group-item
list-group-item-action"
            key={c.title}
          >
            {c.title}
          </Link>
        ))}
      </div>
    </div>
    <div className="list-group">
      {categories.map((cate) => (
        <Link
          to="/category"
          className="list-group-item
list-group-item-action"
          key={cate.title}
        >
          {cate.title}
        </Link>
      ))}
    </div>
  </div>
  <div className="col-md-8 slide-list">
    <div
      id="carouselExampleControls"
      className="carousel slide"
      data-ride="carousel"
    >
      <div className="carousel-inner">
        {images.map((c, index) => (
          <div
            className={
              (active === index ? "active" : "")
            }
            "carousel-item " +

```

```

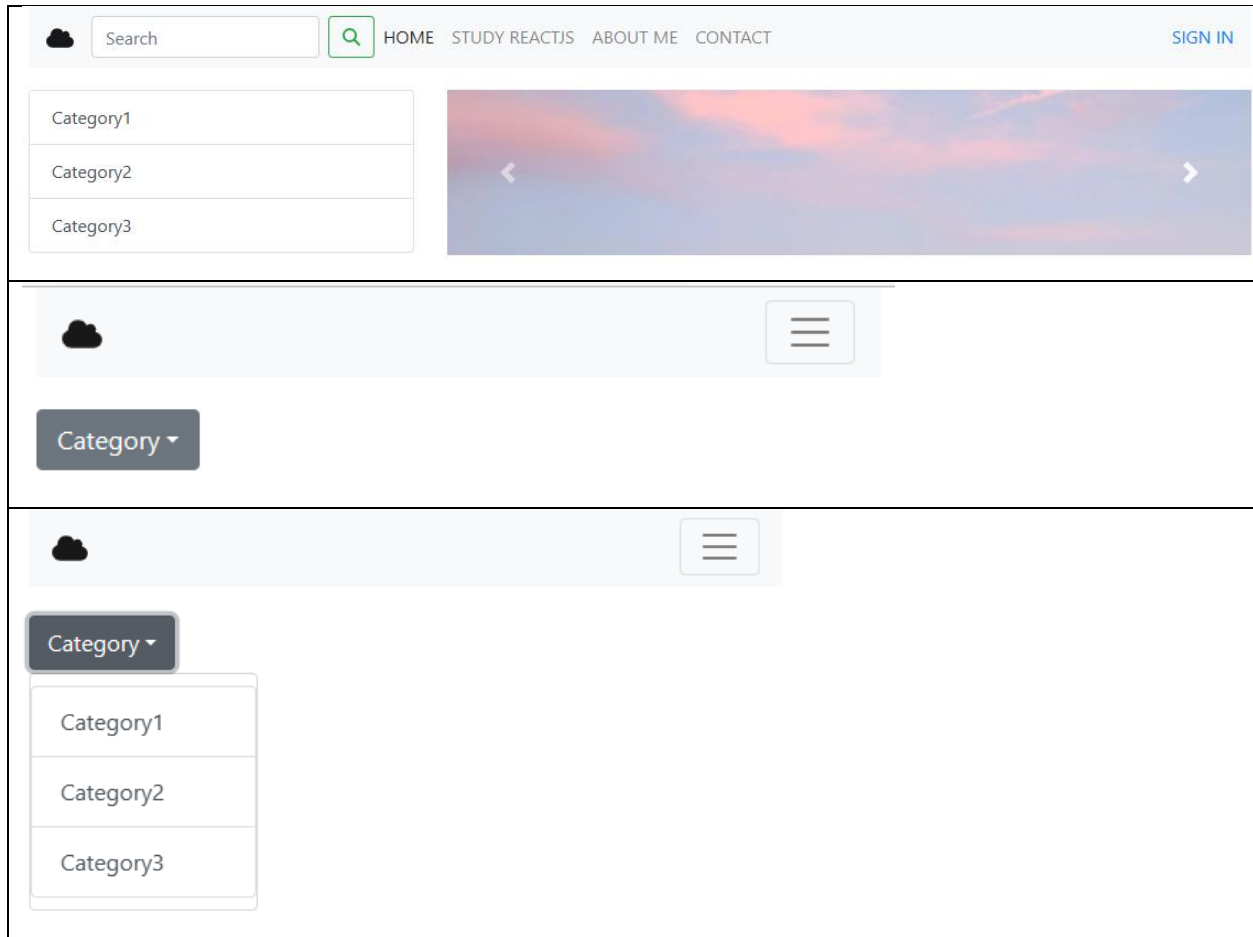
                                key={index}
                                >
                                <img src={c.src}
className="d-block w-100" alt="..." />
                                </div>
                                )}
                                </div>
                                <button
                                    className="carousel-control-prev"
                                    type="button"
                                    data-
target="#carouselExampleControls"
                                    data-slide="prev"
                                >
                                <span
                                    className="carousel-control-
prev-icon"
                                    aria-hidden="true"
                                ></span>
                                <span className="sr-
only">Previous</span>
                                </button>
                                <button
                                    className="carousel-control-next"
                                    type="button"
                                    data-
target="#carouselExampleControls"
                                    data-slide="next"
                                >
                                <span
                                    className="carousel-control-
next-icon"
                                    aria-hidden="true"
                                ></span>
                                <span className="sr-
only">Next</span>
                                </button>
                                </div>
                                </div>
                                </div>
                                </div>
                                </section>
                                );
                                };
                                export default Slider;

```

Đưa vào Home.js

```
<Slider />
```

Chạy app:



Phần ListCart

Css

```
.list-card {  
  margin: 20px 0 20px 0;  
}  
.list-card .card {  
  margin: 10px 0 10px 0;  
}  
.list-card .card img {  
  height: 100%;  
  aspect-ratio: 16/9;  
  object-fit: cover;  
}
```

Phần dữ liệu:

```
export const products = [  
  {  
    name: "Dell G Gaming",  
    describe: "Chi tiết thông số kỹ thuật Dell Gaming G15 5530  
i7-1355U/i7H165W11GR4060",
```

```
      image:
        "https://images.samsung.com/is/image/samsung/p6pim/vn/sm-
        a146pzkgxxv/gallery/vn-galaxy-a14-5g-sm-a146-sm-a146pzkgxxv-
        534780853?$730_584_PNG$"
    },
    {
      name: "Dell Vostro",
      describe: "Laptop Dell Vostro V3510 i5
      1135G7/8GB/512GB/15.6 FHD/GeForce MX350 2GB/Win 11+Office HS21",
      image:
        "https://images.samsung.com/is/image/samsung/p6pim/vn/sm-
        a546elvdxxv/gallery/vn-galaxy-a54-5g-sm-a546-sm-a546elvdxxv-
        535765778?$2052_1641_PNG$"
    },
    {
      name: "MacBook Air 13 2020 M1 256GB",
      describe: "MacBook Air 13 inch M1 2020 8CPU 7GPU
      8GB/256GB",
      image:
        "https://images.samsung.com/is/image/samsung/p6pim/vn/sm-
        a146pzkgxxv/gallery/vn-galaxy-a14-5g-sm-a146-sm-a146pzkgxxv-
        534780853?$730_584_PNG$"
    },
    {
      name: "MacBook Air 13 M1 2020 Ram 16GB",
      describe: "MacBook Air 13 inch M1 2020 8CPU 7GPU
      16GB/256GB",
      image:
        "https://product.hstatic.net/1000370129/product/s22_ultra_den_8745
        2d1a31e241019518f8f97535f104_master_e873eca7c12845f39ec8c99440a3c9
        84_master.png"
    },
  ],
]
```

Phần Component: ListCard.js

```
import { Card } from "react-bootstrap";
import { products } from '../data/db'
const ListCard = () => {
  return (
    <section className="list-card">
      <div className="container">
        <div className="row">
          {products.map((p) => (
            <div className="col-12 col-sm-6 col-lg-3">
              <Card>
                <Card.Img
                  variant="top"
                  src={p.image}
                />
                <Card.Body>
```

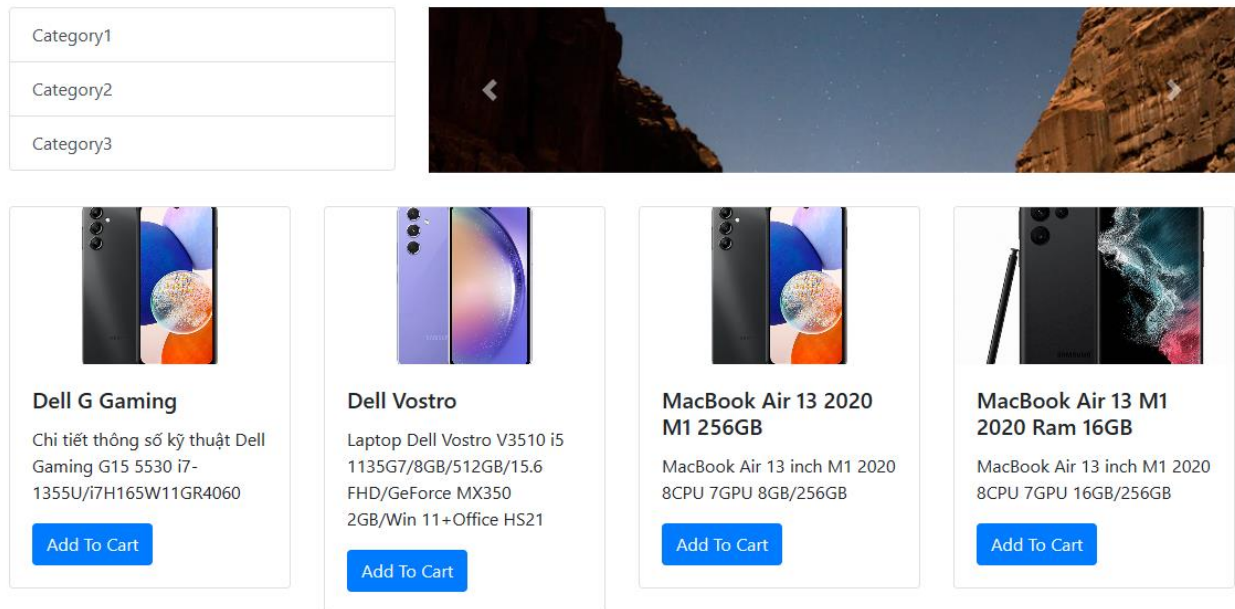


```
le>
Text>
primary">Add To Cart</button>
</Card.Body>
</Card>
</div>
    )}
  </div>
</section>
);
};
export default ListCard;
```

Cho vào Home.js

```
<ListCard />
```

Chạy app ta được:



Phần Promotion

Css

```
.promotion {
  margin: 20px 0 20px 0;
}
@media (max-width: 767.98px) {
  .promotion {
    display: none;
  }
}
```

Phần Component: Promotion.js

```

import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import {
  faCheck,
  faShippingFast,
  faExchangeAlt,
  faPhoneVolume,
} from "@fortawesome/free-solid-svg-icons";
const Promotion = () => {
  return (
    <section className="promotion">
      <div className="container">
        <div className="row">
          <div className="col-3 col-md-6 col-sm-12 pb-1">
            <div className="d-flex align-items-center bg-light mb-4 p-5">
              <FontAwesomeIcon icon={faCheck} size="3x" color="blue" />
              <h5 className="font-weight-semi-bold m-0 ml-3">
                Sản phẩm chất lượng
              </h5>
            </div>
          </div>
          <div className="col-3 col-md-6 col-sm-12 pb-1">
            <div className="d-flex align-items-center bg-light mb-4 p-5">
              <FontAwesomeIcon icon={faShippingFast} size="3x" color="blue" />
              <h5 className="font-weight-semi-bold m-0 ml-3">
                Miễn phí giao hàng
              </h5>
            </div>
          </div>
          <div className="col-3 col-md-6 col-sm-12 pb-1">
            <div className="d-flex align-items-center bg-light mb-4 p-5">
              <FontAwesomeIcon icon={faExchangeAlt} size="3x" color="blue" />
              <h5 className="font-weight-semi-bold m-0 ml-3">
                7 ngày hoàn trả
              </h5>
            </div>
          </div>
        </div>
      </div>
    </section>
  );
};

```

```

    <div className="d-flex align-items-center
bg-light mb-4 p-5">
        <FontAwesomeIcon icon={faPhoneVolume}
size="3x" color="blue" />
        <h5 className="font-weight-semi-bold
m-0 ml-3">Hỗ trợ 24/7</h5>
    </div>
</div>
</div>
</div>
</section>
);
};
export default Promotion;

```

Chạy app ta được:



Sản phẩm chất lượng



Miễn phí giao hàng



7 ngày hoàn trả



Hỗ trợ 24/7

Phần Footer

Css

```
.footer .copyright {
  margin: 0;
  padding: 20px;
  background-color: #f8f9fa;
}

.footer .back-to-top {
  width: 100%;
  height: 100%;
  font-size: 30px;
  background-color: aliceblue;
  display: flex;
  justify-content: center;
  align-items: center;
}

.footer .back-to-top:hover {
  cursor: pointer;
}
```

Component Footer.js

```
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faUpLong } from "@fortawesome/free-solid-svg-icons";
const Footer = () => {
  const scrollToTop = () => {
    window.scrollTo({
      top: 0,
      behavior: "smooth",
    });
  };
  return (
    <section className="footer mb-5">
      <div className="container">
        <div className="row">
          <div className="col-10">
            <p className="copyright">Copyright @ 2023
AnhTTV20 </p>
          </div>
          <div className="col-2" onClick={scrollToTop}>
            <div className="back-to-top">
              <FontAwesomeIcon icon={faUpLong}
color="blue" />
            </div>
          </div>
        </div>
      </div>
    </section>
  );
};
export default Footer;
```

Cho vào Home.js

```
<Footer />
```

Chạy app

Copyright @ 2023 AnhTTV20



Xử lý validate đơn giản với React Hook Form

Tham khảo

<https://techmaster.vn/posts/37367/xu-ly-validate-don-gian-voi-react-hook-form>

Cài đặt React Hook Form

```
npm i react-hook-form
```

Module Login

Css

```
* {
    box-sizing: border-box;
}
.body {
    height: 100vh;
    margin: 0;
    padding: 0;
    background: #F0F2F5;
    display: flex;
    align-items: center;
    flex-direction: column;
    justify-content: center;
}
a {
    font-size: 18px;
}
a:hover {
    text-decoration: none;
}
.back-to-home {
    position: absolute;
    top: 0;
    left: 0;
}
.back-to-home a {
    display: inline-block;
    padding: 10px;
    background-color: #367051;
    color: #fff;
}
.back-to-home a:hover {
    text-decoration: none;
    color: #212529;
}
.inner-wrap {
    padding: 0 30px 20px 30px;
    background-color: #fff;
    display: flex;
    flex-direction: column;
}
.form-group {
    margin-bottom: 20px;
    position: relative;
}
.form-group.pw {
    margin-bottom: 0;
}
.sign-up {
```

```
        align-self: flex-end;
        margin: 10px 0;
    }
    .sign-up a {
        padding: 5px 10px;
        color: #fff;
        background-color: #367051;
    }
    .sign-up a:hover {
        color: #212529;
    }
    .form-control:focus {
        border: 1px solid #367051;
        box-shadow: none;
    }
    .form-control.success {
        border: 1px solid #367051;
        box-shadow: none;
    }
    .form-control.error {
        border: 1px solid #e74c3c;
        box-shadow: none;
    }
    .btn {
        background-color: #367051;
        color: #fff;
        font-size: 18px;
    }
    .header {
        text-align: center;
        margin: 20px 0;
    }
    .header h1 {
        color: #232323;
        padding: 0;
        font-size: 34px;
    }
    span .fa-xmark {
        display: inline-flex;
        justify-content: center;
        align-items: center;
        width: 24px;
        height: 24px;
        position: absolute;
        right: 10px;
        top: 7px;
        color: #fff;
        font-size: 12px;
        background-color: #e74c3c;
        border-radius: 50%;
    }
```

```
span .fa-check {
  display: inline-flex;
  justify-content: center;
  align-items: center;
  width: 24px;
  height: 24px;
  position: absolute;
  right: 10px;
  top: 7px;
  color: #fff;
  font-size: 12px;
  background-color: #2ecc71;
  border-radius: 50%;
}
span {
  visibility: hidden;
}
span.visible {
  visibility: visible;
}
span.hidden {
  visibility: hidden;
}
small {
  visibility: hidden;
  font-size: 16px;
  color: #e74c3c;
  padding: 0 10px;
  margin: 5px 0;
}
small.visible {
  visibility: visible;
}
small.hidden {
  visibility: hidden;
}
```

Component Login.js

```
import { Link } from "react-router-dom";
import "../login.css";
import { useForm } from "react-hook-form";
const Login = () => {
  const {
    register,
    handleSubmit,
    formState: { errors },
  } = useForm();

  return (
    <div className="body">
```

```

    <div className="back-to-home">
      <Link to="/">Home</Link>
    </div>
    <div className="container">
      <div className="row justify-content-center">
        <div className="inner-wrap col">
          <div className="header">
            <h1>Login form</h1>
          </div>
          <div className="data w-100">
            <form
              id="form"
              className="form"
              onSubmit={handleSubmit((data) =>
                console.log(data))}
            >
              <div className="form-group">
                <input
                  type="text"
                  placeholder="Email"
                  className="form-control"
                  id="email"
                  name="email"
                  {...register("email", {
                    required: "Email không
được để trống",
                    pattern: {
                      value: /^[\\w-
.]+@([\\w-]+\\.)+[\\w-]{2,4}$/ ,
                      message: "Không
đúng định dạng",
                    },
                  })}
                </div>
                <div className="form-group">
                  <input
                    type="password"
                    placeholder="Password"
                    className="form-control"
                    id="password"
                    name="password"
                    {...register("password", {
                      required: "mật khẩu
không được để trống",
                      minLength: {
                        value: 6,

```



```
        message: "Tối  
        thiếu 6 kí tự",  
        },  
        maxLength: {  
            value: 20,  
            message: "Tối đa  
        20 kí tự",  
        },  
    })}  
    />  
    {errors.password && (  
        <p className="text-  
danger">{errors.password.message}</p>  
    )}  
    </div>  
    <div className="form-group">  
        <button className="btn form-  
control btn-success">  
            Login  
        </button>  
    </div>  
    </form>  
    </div>  
    <div className="sign-up">  
        <Link to="/signup">Sign Up</Link>  
    </div>  
    </div>  
    </div>  
    </div>  
    </div>  
    );  
};  
export default Login;
```

Khai báo trong App.js

```
<Route exact path="/login" element={<Login />} />
```

Chạy app

Home

Login form

Email

Password

Login

Sign Up

Home

Login form

Email

Email không được để trống

Password

mật khẩu không được để trống

Login

Sign Up

Module Signup

Css

```
* {  
  box-sizing: border-box;  
}  
.body {  
  height: 100vh;  
  margin: 0;  
  padding: 0;  
  background: #F0F2F5;  
  display: flex;  
  align-items: center;  
  flex-direction: column;  
  justify-content: center;  
}  
.back-to-home {  
  position: absolute;  
  top: 0;  
  left: 0;  
}  
.back-to-home a {  
  display: inline-block;  
  padding: 10px;
```

```
        background-color: #367051;
        color: #fff;
    }
    .back-to-home a:hover {
        text-decoration: none;
        color: #212529;
    }
    .inner-wrap {
        padding: 0 30px 20px 30px;
        background-color: #fff;
        display: flex;
        flex-direction: column;
    }
    .form-group {
        margin-bottom: 20px;
        position: relative;
    }
    .form-control:focus {
        border: 1px solid #367051;
        box-shadow: none;
    }
    .form-control.success {
        border: 1px solid #367051;
        box-shadow: none;
    }
    .form-control.error {
        border: 1px solid #e74c3c;
        box-shadow: none;
    }
    .btn {
        background-color: #367051;
        color: #fff;
        font-size: 16px;
    }
    .header {
        text-align: center;
        margin: 20px 0;
    }
    .header h1 {
        color: #232323;
        padding: 0;
        font-size: 34px;
    }
    span .fa-xmark {
        display: inline-flex;
        justify-content: center;
        align-items: center;
        width: 24px;
        height: 24px;
        position: absolute;
        right: 10px;
```

```
    top: 7px;
    color: #fff;
    font-size: 12px;
    background-color: #e74c3c;
    border-radius: 50%;
}

span .fa-check {
    display: inline-flex;
    justify-content: center;
    align-items: center;
    width: 24px;
    height: 24px;
    position: absolute;
    right: 10px;
    top: 7px;
    color: #fff;
    font-size: 12px;
    background-color: #2ecc71;
    border-radius: 50%;
}

span {
    visibility: hidden;
}

span.visible {
    visibility: visible;
}

span.hidden {
    visibility: hidden;
}

small {
    visibility: hidden;
    font-size: 16px;
    color: #e74c3c;
    padding: 0 10px;
    margin: 5px 0;
}

small.visible {
    visibility: visible;
}

small.hidden {
    visibility: hidden;
}

.sign-in {
    align-self: flex-end;
    margin: 10px 0;
}

.sign-in a {
    padding: 5px 10px;
    color: #fff;
    background-color: #367051;
```

```

}
.sign-in a:hover {
  text-decoration: none;
  color: #212529;
}

```

Component SignUp.js

```

import { Link } from "react-router-dom";
import "./signup.css";
import { useForm } from "react-hook-form";
import { useEffect } from "react";
const SignUp = () => {
  const {
    register,
    handleSubmit,
    formState: { errors },
  } = useForm();
  useEffect(() => {
    console.log(errors.phone);
  }, [errors]);
  const submit = (data) => {
    console.log(data);
  };
  return (
    <div className="body">
      <div className="back-to-home">
        <Link to="/">Home</Link>
      </div>
      <div className="container">
        <div className="row justify-content-center">
          <div className="inner-wrap col-xl-7">
            <div className="header">
              <h1>Create New Account</h1>
            </div>
            <div className="data">
              <form className=""
onSubmit={handleSubmit(submit)}>
                <div className="form-group">
                  <input
                    type="text"
                    placeholder="Phone Number:
0912004866"
                    className="form-control"
                    // id="phone"
                    maxLength="11"
                    name="phone_number"
                    {...register("phone", {
                      required: true,
                      pattern: {

```

```

value:
/ ("+"84|84|0)+([3|5|7|8|9])+([0-9]{8})\b/,
message: "Không
đúng định dạng",
},
minLength: {
value: 10,
message: "Tối
thiểu 10 số",
},
}},
/>
{errors.phone && (
<p className="text-
danger">{errors.phone.message}</p>
)}
</div>
<div className="form-group">
<input
type="text"
placeholder="Email"
className="form-control"
id="email"
name="email"
{...register("email", {
required: "Email không
được để trống",
pattern: {
value: /^[w-
.]+@([\w-]+)+[\w-]{2,4}$/,
message: "Không
đúng định dạng",
}},
/>
{errors.email && (
<p className="text-
danger">{errors.email.message}</p>
)}
</div>
<div className="form-group">
<input
type="password"
placeholder="New Password"
className="form-control"
id="password"
name="password"
{...register("password", {
required: "Mật khẩu
không được để trống",
minLength: {

```

```

        value: 6,
        message: "Tối
        thiếu 6 kí tự",
    },
    },
    maxLength: {
        value: 20,
        message: "Tối đa
        20 kí tự",
    },
    }
    }
    }
    />
    {errors.password && (
        <p className="text-
        danger">{errors.password.message}</p>
        )}
    </div>
    <div className="form-group">
        <input
            type="password"
            placeholder="New Password
            Again"
            className="form-control"
            id="confirm-password"
            {...register("confirmPassw
            ord", {
                required: "Nhập lại
                mật khẩu không được để trống",
                minLength: {
                    value: 6,
                    message: "Tối
                    thiếu 6 kí tự",
                },
                maxLength: {
                    value: 20,
                    message: "Tối đa
                    20 kí tự",
                },
            }
            }
            }
    />
    {errors.confirmPassword && (
        <p className="text-
        danger">
            {errors.confirmPasswor
            d.message}
        </p>
        )}
    </div>
    <div className="form-group">
        <button className="btn form-
        control btn-success">Sign Up</button>
    </div>

```

```
        </form>
      </div>
      <div className="sign-in">
        <Link to="/login">Sign In</Link>
      </div>
    </div>
  </div>
</div>
);
};
export default SignUp;
```

Khai báo trong App.js

```
<Route exact path="/signup" element={(<SignUp />)} />
```

Chạy app:

Home

Create New Account

Phone Number: 0912004866

Email

New Password

New Password Again

Sign Up

Sign In

Home

Create New Account

Phone Number: 0912004866

Email

Email không được để trống

New Password

Mật khẩu không được để trống

New Password Again

Nhập lại mật khẩu không được để trống

Sign Up

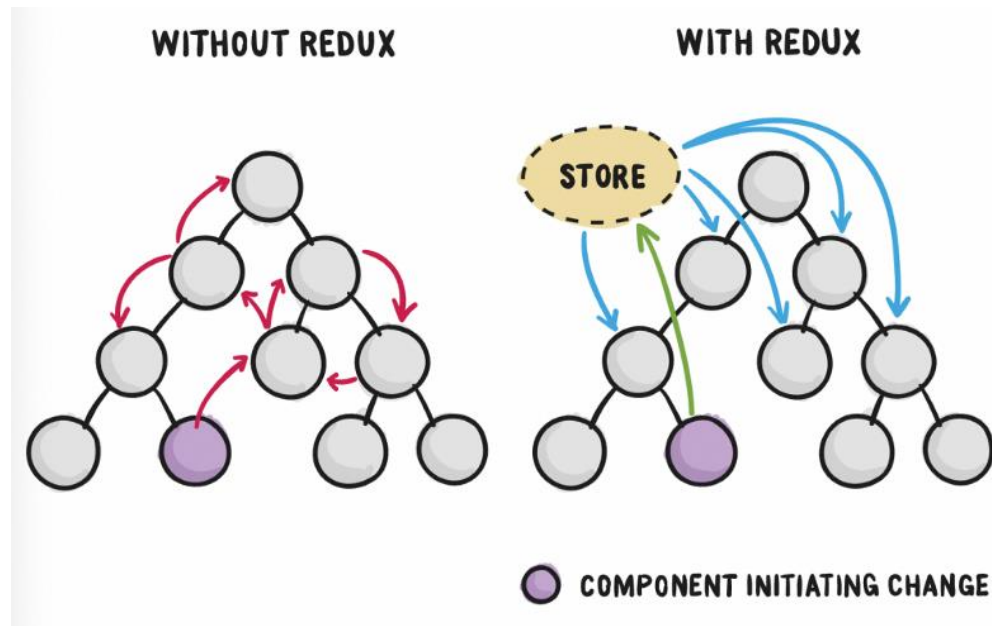
Sign In

13.Redux



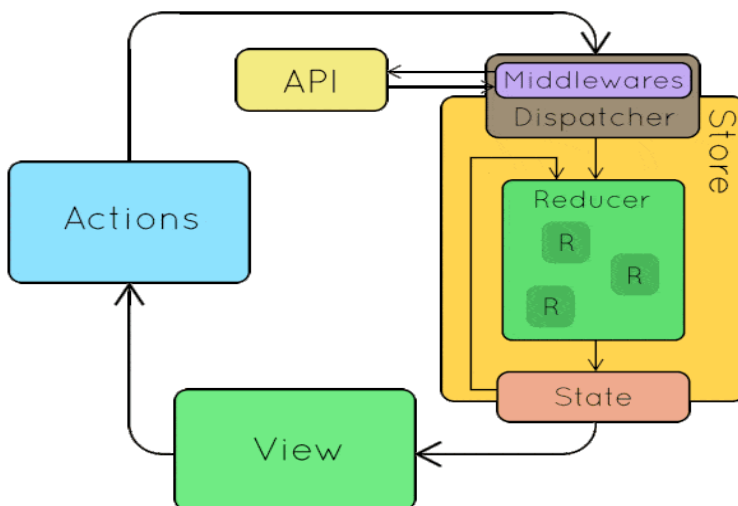
Redux là một **predictable state container** cho các ứng dụng JavaScript.

Nó giúp bạn viết các ứng dụng hoạt động một cách nhất quán, chạy trong các môi trường khác nhau (client, server and native) và dễ dàng kiểm tra. Trên hết, nó cung cấp trải nghiệm phát triển tuyệt vời, chẳng hạn như [live code editing combined with a time traveling debugger](#).



Nguyên lý vận hành

- **Nguồn dữ liệu tin cậy duy nhất:** *State* của toàn bộ ứng dụng được chứa trong một object tree nằm trong *Store* duy nhất
- **Trạng thái chỉ được phép đọc:** Cách duy nhất để người dùng có thể thay đổi *State* của ứng dụng là phát một *Action*, tức một object mô tả tất cả những gì xảy ra. Trạng thái của *Redux* chỉ là một đối tượng và nó chỉ có thể thay đổi chỉ khi xuất hiện một sự kiện. Ngoài ra thì không được phép thay đổi **trực tiếp**.
- **Chỉ thay đổi bằng hàm thuần túy:** Để chỉ ra cách mà *State* được biến đổi bởi *Action*, người dùng sử dụng các *pure function* được gọi là *Reducer*. Thông qua hàm thuần túy, bạn có thể thực hiện việc thay đổi trạng thái của ứng dụng. Cụ thể, dữ liệu của các sự kiện và trạng thái hiện tại đưa vào sẽ được hàm xử lý và trả về trạng thái tiếp theo.



Cài đặt và ví dụ

- Cài đặt thư viện redux và react-redux

```
- yarn add redux
- yarn add react-redux
```

Khởi tạo file actionTypes.js đây là file xác định tất các loại *actions* mà app sẽ sử dụng:

```
// actionTypes.js

export const FETCH_USERS_REQUEST = "FETCH_USERS_REQUEST";
export const FETCH_USERS_SUCCESS = "FETCH_USERS_SUCCESS";
export const FETCH_USERS_FAILURE = "FETCH_USERS_FAILURE";
```

- Tiếp theo, khởi tạo file actions.js đây là file xác định tất cả các *actions*. Chúng ta sẽ sử dụng các loại *actions* được định nghĩa bên trong actionTypes.js để khởi tạo:

```
// actions.js

import {
  FETCH_USERS_REQUEST,
  FETCH_USERS_SUCCESS,
  FETCH_USERS_FAILURE,
} from "../actionTypes";
const fetchUsersRequest = () => {
  return {
    type: FETCH_USERS_REQUEST,
  };
};
const fetchUsersSuccess = (users) => {
  return {
    type: FETCH_USERS_SUCCESS,
    payload: users,
  };
};
const fetchUsersFailure = (error) => {
  return {
    type: FETCH_USERS_FAILURE,
    payload: error,
  };
};
export const fetchUsers = () => {
  return (dispatch, getState) => {
    dispatch(fetchUsersRequest());
    const { users } = getState();
    fetch(`https://jsonplaceholder.typicode.com/users?since=${users.length}`)
      .then((response) => response.json())
      .then((data) => {
```

```

                dispatch(fetchUsersSuccess(data));
            })
            .catch((error) => {
                dispatch(fetchUsersFailure(error.message));
            });
    };
};

```

Khởi tạo file reducer.js để xác định các *initial state* và *reducers* cho từng loại *actions type* đã được định nghĩa:

```

// reducer.js
import {
    FETCH_USERS_REQUEST,
    FETCH_USERS_SUCCESS,
    FETCH_USERS_FAILURE,
} from "../actionTypes";
const initialState = {
    loading: false,
    users: [],
    error: "",
};
const userReducer = (state = initialState, action) => {
    switch (action.type) {
        case FETCH_USERS_REQUEST:
            return {
                ...state,
                loading: true,
            };
        case FETCH_USERS_SUCCESS:
            return {
                ...state,
                loading: false,
                users: action.payload,
                error: "",
            };
        case FETCH_USERS_FAILURE:
            return {
                ...state,
                loading: false,
                users: [],
                error: action.payload,
            };
        default:
            return state;
    }
};
export default userReducer;

```

Trong file reducer.js, **userReducer** được tạo ra để xử lý các **actions** được định nghĩa trong file actionTypes.js. Nó sẽ lấy **state** hiện tại và **actions** làm đối số và trả về một **state** mới dựa trên các loại **action**.

Cuối cùng, khởi tạo file index.js đây là file để kết hợp các **reducers** và khởi tạo **Redux store**

```
// index.js
import { createStore, combineReducers } from "redux";
import { reducer as formReducer } from 'redux-form'
import userReducer from "../reducer";
import middleware from "../middleware";
const rootReducer = combineReducers({
  users: userReducer,
  form: formReducer
});
const store = createStore(rootReducer, middleware);
export default store;
```

Trong file index.js, chúng ta import các chức năng **createStore** và **combinereducers** từ **Redux** cũng như **UserReducer**.

Hàm **combinereducers** biến một đối tượng có các giá trị và các **reducers** khác nhau thành một hàm **reducer** duy nhất sau đó khởi tạo thành **Redux Store** thông qua sử dụng hàm **createStore**.

Áp dụng Redux vào bên trong project ReactJS

- Đầu tiên hãy tạo một component sẽ sử dụng **Redux Store**. Trong ví dụ này, chúng ta sẽ tạo **Users** component để fetch danh sách người dùng và hiển thị chúng lên:

```
// Users.js

import React, { useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { fetchUsers } from "../redux/actions";
const Users = () => {
  const dispatch = useDispatch();
  const { loading, users, error } = useSelector((state) =>
state.users);
  useEffect(() => {
    dispatch(fetchUsers());
  }, []);
  if (loading) {
    return <div>Loading...</div>;
  }
  if (error) {
    return <div>Error: {error}</div>;
  }
  return (
```

```

        <div>
          <h1>Users</h1>
          <ul>
            {users.map((user) => (
              <li key={user.id}>{user.name}</li>
            ))}
          </ul>
        </div>
      );
    };
    export default Users;

```

- Trong **Users** component, chúng ta import **useDispatch** và **useSelector** hooks từ thư viện **React Redux**:

useSelector hooks cho phép chúng ta lấy **state** từ **Redux store** bằng cách sử dụng một **selector function** làm tham số đầu vào.

useDispatch hooks return về một tham chiếu đến **dispatch function** từ **Redux store** và được sử dụng để **dispatch** các **action**.

Khi **dispatch** một **action**, **useSelector** sẽ thực hiện so sánh tham chiếu với giá trị được return trước đó và giá trị hiện tại. Nếu chúng khác nhau, component sẽ bị re-render. Nếu chúng giống nhau, component sẽ không re-render

- Các **actions** được xác định trong actions.js cũng được import vào. Trong Hook UseEffect, chúng ta **dispatch action** fetchUsersRequest và sau đó thực hiện yêu cầu fetch API đến JsonPlaceholder để có được danh sách Users. Nếu requests thành công, chúng ta **dispatch action** fetchUsersSuccess với dữ liệu đã nhận được lên **Redux Store**. Nếu requests không thành công, chúng ta sẽ **dispatch action** fetchUsersFailure với thông báo lỗi lên **Redux Store**.
- Sau đó, chúng ta sử dụng các biến loading, users<, error từ **Redux store** để hiển thị nội dung thích hợp.
- Tiếp theo, khởi tạo file App.js:

```

import React from "react";
import { Provider } from "react-redux";
import store from "../redux";
import Users from "../components/Users";
import ContactForm from "../components/ReduxForm/ContactForm";
import showResults from "../components/ReduxForm/showResults";
function App() {
  return (
    <Provider store={store}>
      <div>
        <Users />
      </div>
      <div style={{ padding: 15 }}>

```

```
        <h2>Simple Form Redux form</h2>
        <ContactForm onSubmit={showResults} />
      </div>
    </Provider>
  );
}
export default App;
```

Trong file App.js, chúng ta import **Provider** components từ **React Redux** và **Redux store** từ file index.js. Sau đó, gói **Users** component vào trong **Provider** components để có thể truyền được dữ liệu trong **Redux store** như prop.

14.Redux form

Redux-form là một thư viện hỗ trợ trong việc quản lý React form state.

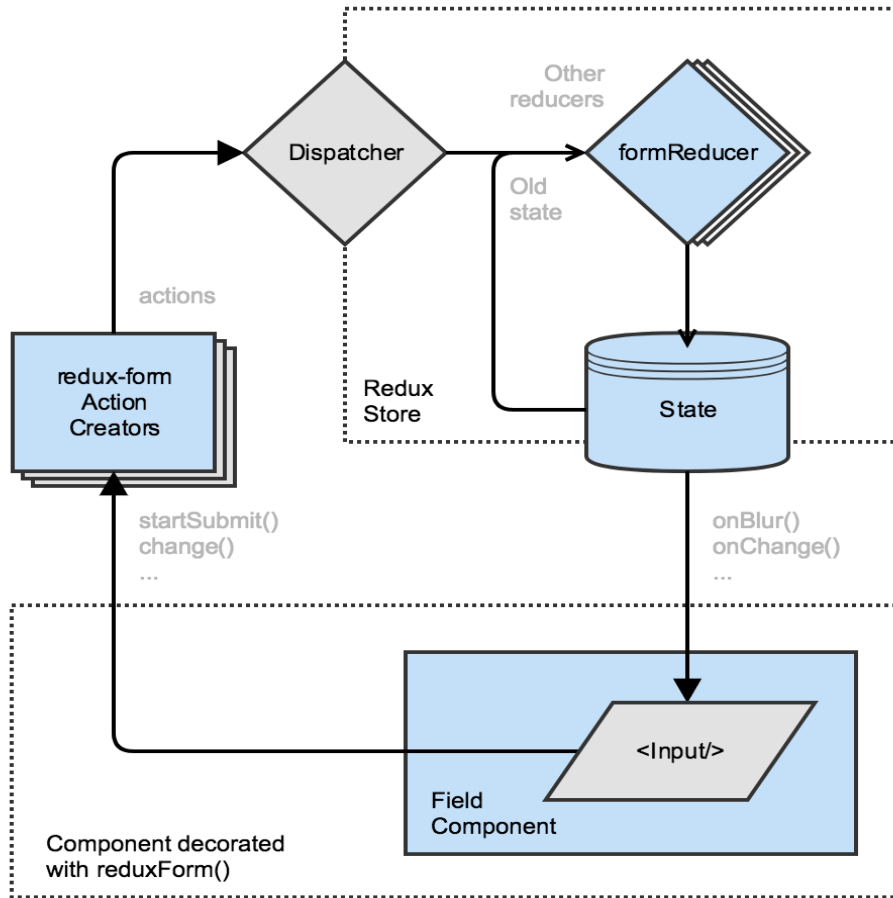
Để kết nối giữa React form component và Redux store chúng ta cần những thành phần sau từ **redux-form**

- **formReducer** : function chịu trách nhiệm cập nhật Redux store dựa trên những thay đổi từ app.
- **reduxForm()** : là một **HOC** với đầu vào là một object và đầu ra là một function mới. Sử dụng hàm này để bao lấy form component và bind các hàm xử lý tương tác người dùng và dispatch action tương ứng.
- **<Field/>** : components bên trong form component, dùng để kết nối input component với redux-form logic.

Redux-form theo dõi tất cả các trạng thái của ứng state như:

- Định dạng của các trường.
- Các giá trị của từng trường.
- Các trường được tập trung xử lý.
- Các giá trị hợp lệ
- Các trường người dùng đã tương tác
- Các form đang gửi.

Trường hợp xảy ra bất kì xác thực không đồng bộ nào.



Cài đặt và sử dụng **Redux form**

- Cài đặt thư viện **Redux form**

```
- yarn add redux-form
```

Đầu tiên ta truyền **formReducer** đến **store**. Reducer này sẽ phục vụ tất cả form components trong app của mình.

```
import { createStore, combineReducers } from "redux";
import { reducer as formReducer } from 'redux-form'
const rootReducer = combineReducers({
  form: formReducer
  // other reducers
  ...
});
const store = createStore(rootReducer);
export default store;
```

Như vậy **store** sẽ biết phải handle **actions** đến từ các form component.

- Khởi tạo **Form component**.

Để **Form component** được kết nối đến **store**, ta cần bao component đó bởi **reduxForm()**. Nó sẽ cung cấp các props liên quan đến state và function để handle việc submit form. Khởi tạo file ContactForm.js

```
// ContactForm.js
import React from 'react'
import { Field, reduxForm } from 'redux-form'
let ContactForm = props => {
  const { handleSubmit } = props
  return <form onSubmit={handleSubmit}>{/* form body*/}</form>
}
ContactForm = reduxForm({
  // ten của mọi form là duy nhất
  form: 'contact'
})(ContactForm)
export default ContactForm
```

- Khởi tạo Component **<Field/>**

Component **<Field/>** làm nhiệm vụ kết nối input đến **store** **<Field name="inputName" component="input" type="text" />**. Nó sẽ tạo phần tử HTML **<input/>** với type text và truyền vào những props như value, onChange, onBlur ,... giúp cho việc theo dõi và cập nhật **state**.

```
import React from 'react';
import { Field, reduxForm } from 'redux-form';
const ContactForm = props => {
  const { handleSubmit } = props;
  return (
    <form onSubmit={handleSubmit}>
      <div>
        <label htmlFor="firstName">First Name</label>
        <Field name="firstName" component="input" type="text" />
      </div>
      <div>
        <label htmlFor="lastName">Last Name</label>
        <Field name="lastName" component="input" type="text" />
      </div>
      <div>
        <label htmlFor="email">Email</label>
        <Field name="email" component="input" type="email" />
      </div>
      <button type="submit">Submit</button>
    </form>
  )
};
export default reduxForm({
  form: 'contact', // a unique identifier for this form
})(ContactForm);
```

- Handle submit form

Dữ liệu submit được truyền dưới dạng object JSON đến function onSubmit. Khởi tạo file showResult.js:

```
const sleep = ms => new Promise(resolve => setTimeout(resolve, ms));
export default (async function showResults(values) {
  await sleep(500); // simulate server latency
  window.alert(`You submitted:\n\n${JSON.stringify(values, null, 2)}`);
})();
```

Cuối cùng, thêm components ContactForm vào bên trong App.js

```
import React from "react";
import { Provider } from "react-redux";
import store from "../redux";
import ContactForm from "../components/ReduxForm/ContactForm";
import showResults from "../components/ReduxForm/showResults";
function App() {
  return (
    <Provider store={store}>
      <div style={{ padding: 15 }}>
        <h2>Simple Form Redux form</h2>
        <ContactForm onSubmit={showResults} />
      </div>
    </Provider>
  );
}
export default App;
```

Các ví dụ khác về việc sử dụng *Redux Form*

- [Simple Form](#)
- [Synchronous Validation](#)
- [Field-Level Validation](#)
- [Submit Validation](#)
- [Asynchronous Blur Validation](#)
- [Initializing From State](#)
- [Field Arrays](#)
- [Remote Submit](#)
- [Normalizing](#)
- [Immutable JS](#)

- [Selecting Form Values](#)
- [Wizard Form](#)

15.JSON Server

Là một thư viện giúp chúng ta có thể tạo một máy chủ giả (fake) cung cấp RESTful API một cách đầy đủ trong một khoảng thời gian ngắn mà không cần code.

Cài JSON Server

```
npm i --dev-save json-server
```

hoặc nếu dùng *yarn*:

```
yarn add json-server -D json-server
```

Site tham khảo

```
https://xerosource.com/how-to-manage-login-session-in-react-js/  
https://hackernoon.com/understanding-session-management-using-react-router-v6
```

Thêm một đoạn script cho JSON Server trong khối *scripts* trong file *package.json*:

```
"scripts": {  
  ...  
  "start-server": "json-server --watch db.json --port 9999",  
  ...  
}
```

Bắt đầu chạy JSON Server

```
npm run start-server  
or  
yarn start-server
```

Khi đã chạy server xong, nếu bạn call đến api <http://localhost:9999/posts/1>, bạn sẽ nhận được:

```
{  
  "id": 1,  
  "title": "The first post",  
  "body": "This is the first post",  
  "author": { "username": "user1", "fullname": "User 1" }  
}
```

<https://jsonplaceholder.typicode.com/>

16.React Animation