

# Nền tảng phát triển Web

## ReactJS

[thongph88@gmail.com](mailto:thongph88@gmail.com)

# Nội dung

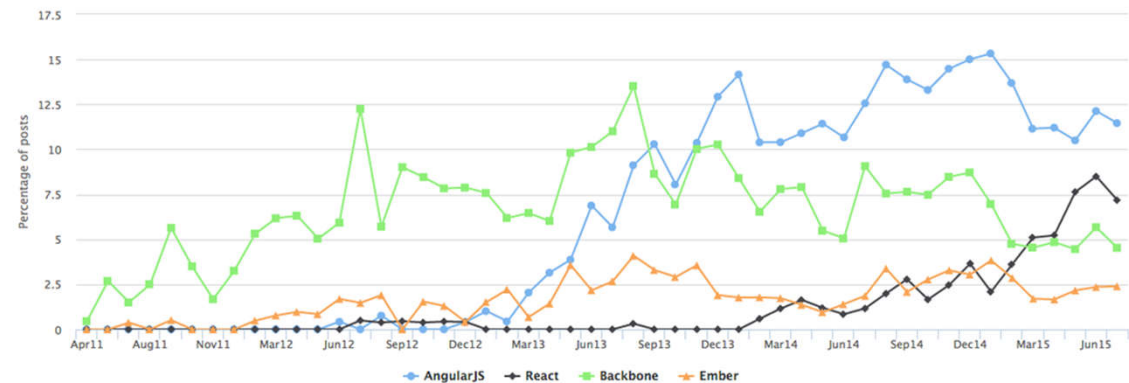


- 1) Khái niệm
- 2) Tại sao chọn học ReactJS
- 3) Làm quen với ReactJS
- 4) Cấu trúc thư mục

# Khái niệm



- ❑ Là thư viện hỗ trợ phát triển Front end nhằm mục đích:
  - Đơn giản
  - Khai báo dễ dàng
  - Xây dựng kết hợp các thành phần với nhau
- ❑ React được tạo ra bởi Facebook và phát hành vào tháng 5 năm 2013
- ❑ ReactJS được dùng phổ biến từ 2015
- ❑ Downloads : ~ 933,860/tháng



# Tại sao chọn học React



- ❑ Tốc độ cao: React sử dụng Virtual DOM (DOM ảo), cho phép chỉ cập nhật các phần của trang web đã thay đổi
- ❑ Dễ học / Dễ sử dụng: Nếu bạn đã học qua về JavaScript thì học React cũng khá đơn giản. React cho phép các lập trình viên nhóm các code liên quan lại với nhau, do đó làm cho việc xây dựng và duy trì các ứng dụng quy mô lớn dễ dàng hơn nhiều.
- ❑ Hậu thuẫn lớn / Cộng đồng mạnh: React có một cộng đồng lớn và nó là mã nguồn mở. Nó được duy trì bởi Facebook và cộng đồng.
- ❑ Việc làm nhiều: Hiện nay đang có nhu cầu rất cao về lập trình web với ReactJS.

# Tại sao chọn học React (2)



**Việc làm**  
• Gần Hà Nội

3 ngày qua


Toàn thời gian

FPT Software


Bkav


VNDIRECT

CÔNG TY CỔ PHẦN





**Reactjs Developer**  
Công ty Cổ phần Giải pháp Công nghệ TTC Việt Nam  
Hà Nội  
qua TopCV  
🕒 22 giờ trước 🏠 15 Tr đ–25 Tr đ một tháng 📅 Toàn thời gian






**Fresher Frontend ReactJS**  
Công ty Cổ Phần Thương Mại Và Dịch Vụ Công Nghệ VelaCorp  
Hà Nội, Hoàn Kiếm, Hà Nội  
qua CareerBuilder  
🕒 9 ngày trước 🏠 7 Tr đ–15 Tr đ một tháng 📅 Toàn thời gian





**Frontend Developer JavaScript/ReactJS**  
Piscada  
Hà Nội  
qua ITviec  
🕒 21 ngày trước 📅 Toàn thời gian



[→ 100+ việc làm khác](#)

# Làm quen với ReactJS



## ❑ Cài đặt:

Both React and ReactDOM are available over a CDN.

```
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>  
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
```

The versions above are only meant for development, and are not suitable for production.

Minified and optimized production versions of React are available at:

# Tạo ứng dụng React



## □ Ý nghĩa của ToolChain (create-react-app):

Use an integrated toolchain for the best user and developer experience.

This page describes a few popular React toolchains which help with tasks like:

- Scaling to many files and components.
- Using third-party libraries from npm.
- Detecting common mistakes early.
- Live-editing CSS and JS in development.
- Optimizing the output for production.

# Tạo ứng dụng React



## ❑ *Các lệnh cần nhớ:*

`npm install create-react-app`

`npm create-react-app my-app`

`cd my-app`

`npm start`

**\*\*** Cài đặt ứng dụng toolchain create-react-app

**\*\*** Chạy toolchain để tạo dự án có tên my-app

**\*\*** Di chuyển vào thư mục dự án my-app

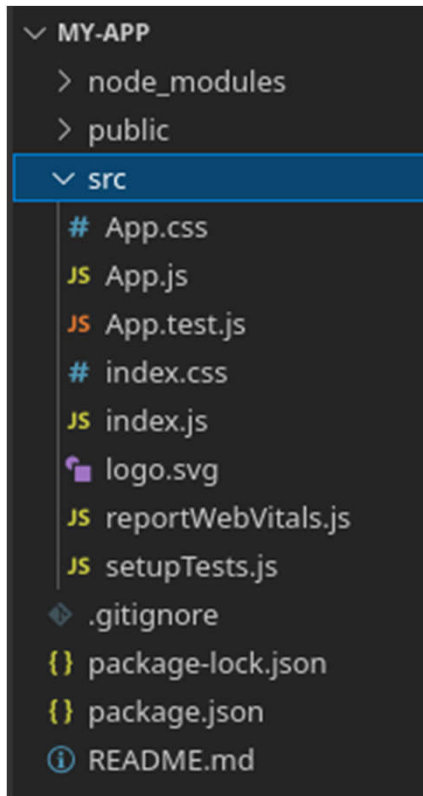
**\*\*** Tạo máy chủ chạy dự án my-app

## ❑ *Nếu muốn gỡ bỏ:*

*`npm uninstall -g create-react-app` hoặc `yarn global remove create-react-app`*



# Cấu trúc một dự án React



- ❑ **Thư mục *public*:** chứa các tệp liên quan đến cách ứng dụng sẽ hiển thị trên phía client, tệp quan trọng nhất trong số đó là `index.html`
- ❑ **Thư mục *src*:** chứa tất cả các file JavaScript, CSS và hình ảnh, cấu trúc trang sẽ được biên dịch từ đây

Khi build `webpack` được sử dụng và gói nhiều file thành một file duy nhất. Từ đó giảm thời gian tải

# Cấu trúc một dự án React(tiếp)



❑ *index.js*: Sử dụng phương thức

ReactDOM.createRoot để truyền nội dung được tạo ra và phần tử HTML có id = 'root'

```
JS index.js  ×
src > JS index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import './index.css';
4  import App from './App';
5  import reportWebVitals from './reportWebVitals';
6
7  const root = ReactDOM.createRoot(document.getElementById('root'));
8  root.render(
9    <React.StrictMode>
10     <App />
11   </React.StrictMode>
12 );
```

# Cấu trúc một dự án React(tiếp)



- ❑ *App.js*: Tập này là thành phần chính (main components) sẽ được hiển thị cho DOM, bao gồm hình ảnh, icon

```
JS index.js JS App.js X
src > JS App.js > ...
1  import logo from './logo.svg';
2  import './App.css';
3
4  function App() {
5    return (
6      <div className="App">
7        <header className="App-header">
8          <img src={logo} className="App-logo" alt="logo" />
9          <p>Edit <code>src/App.js</code> and save to reload.</p>
10         <a className="App-link" href="https://reactjs.org" target="_blank" rel="noopener noreferrer">
11           Learn React
12         </a>
13       </header>
14     </div>
15   );
16 }
17
18 export default App;
```

# Các khái niệm trong React



- ❑ **JSX:** Là cú pháp mở rộng của js cho phép xây dựng các phần tử UI bằng js
- ❑ **VirtualDom:** Khi react biên dịch sẽ xây dựng một DOM ảo và từ DOM ảo sẽ tạo ra cấu trúc trang HTML (DOM thật) tương ứng trên browser. DOM được xây dựng nhanh chóng vì được bằng code do React quản lý. Khi có bất kỳ thay đổi state, React tạo lại DOM ảo và chỉ cập nhật những phần bị thay đổi trên browser
- ❑ **Components:** Components đề cập đến việc giúp bạn chia các thành phần của trang thành những phần có thể tái sử dụng.

# Các khái niệm trong React(tiếp)



- ❑ **Class components:** Class components thường được sử dụng khi cần tạo thành phần có tương tác người dùng cao hơn như Forms và Animation,... Các class component có thể tái sử dụng và quản lý props và states bên trong nó

```
// Class Components
class Hello extends React.Component {
  render() {
    return <h2>Hello React!</h2>;
  }
}

const el = <Hello />;
```

# Các khái niệm trong React(tiếp)



- ❑ **Functional components:** Tên của Function component bắt đầu bằng một chữ cái viết HOA. Đây là quy tắc. Tương tự Class Component nhưng được viết dưới dạng function

```
function Hello() {  
  return <h2>Hello React!</h2>;  
}
```

# Các khái niệm trong React(tiếp)



- ❑ **Props:** Các thuộc tính được truyền từ components cha sang components con. Có thể truyền một components khác làm prop Chỉ có thể đọc không thể sửa.

```
// Function Components
function Hello(props) {
  return <h2>Hello {props.name}!</h2>
}
```

```
// Class Components
class Hello extends React.Component {
  render() {
    return <h2>Hello {this.props.name}!</h2>;
  }
}
```

```
const el = <Hello name = "Ngọc Anh" />;
```

# Các khái niệm trong React(tiếp)



**States:** Nhiều ứng dụng web cần các component của chúng để thay đổi dữ liệu của chúng.

- React cung cấp một tính năng gọi là state (trạng thái). State chỉ là một đối tượng đơn giản, chứa các cặp key:value.
- States Cho phép các component quản lý và thay đổi dữ liệu
- Tương tự như props, các giá trị có thể được truy cập bằng cách gọi this.state
- React cung cấp phương thức setState() . Khi muốn thay đổi states phải dùng state



# Các khái niệm trong React(tiếp)



Ví dụ về state:

```
class MouseClick extends React.Component {  
  // State  
  state = {  
    click: 0  
  }  
  
  // Phương thức thay đổi state  
  incermentClick = () => {  
    this.setState({  
      click: this.state.click + 1  
    });  
  }  
  
  render() {  
    return <div className = "mouseClick">  
      <h1>{this.state.click}</h1>  
      <button onClick={this.incermentClick}>CLICK VÀO ĐÂY!</button>  
    </div>;  
  }  
}
```

# Các khái niệm trong React(tiếp)



## Tổng kết states và props:

- ☐ Chúng ta sử dụng props để truyền dữ liệu cho các component.
- ☐ Các component sử dụng state để quản lý dữ liệu của chúng.
- ☐ props ở chế độ chỉ đọc và không thể sửa đổi.
- ☐ State thì có thể được sửa đổi bởi component của nó bằng phương thức `setState()`.
- ☐ Phương thức `setState()` giúp ta thay đổi state và kích hoạt hành động render lại nội dung đã thay đổi.

# Các khái niệm trong React(tiếp)



**Hook:** Hooks được sử dụng để quản lý state bên trong Functional components.  
Ví dụ:

```
function MouseClick() {  
  // State  
  const [click, setClick] = useState(0);  
  
  // Thay đổi state  
  function incrementClick() {  
    setClick(click + 1);  
  }  
  
  return <div className = "mouseClick">  
    <h2>{click}</h2>  
    <button onClick={incrementClick}>CLICK VÀO ĐÂY!</button>  
  </div>  
}
```

# Các khái niệm trong React(tiếp)



**Lifecycle Method:** React cung cấp các life cycle method sử dụng trong **ClassComponents** khi Component được render trên trang (Mounting) và bị xóa khỏi trang (Unmounting)

- ❑ **componentDidMount():** Được gọi khi components render trên trang.
- ❑ **componentWillUnmount():** Được gọi ngay trước khi components bị xóa khỏi trang
- ❑ **componentDidUpdate():** Được gọi khi component được thay đổi

Đối với function components sử dụng `useEffect()` hooks để quản lý vòng đời

## Ví dụ Lifecycle với ClassComponents



```
class MouseClick extends React.Component {
  state = {
    click: 0
  }

  incrementClick = () => {
    this.setState({
      click: this.state.click + 1
    });
  }

  // Phương thức được gọi khi
  // nội dung được render trên trang
  componentDidUpdate() {
    console.log("Click: " + this.state.click);
  }

  render() {
    return <div className="mouseClick">
      <h2>{this.state.click}</h2>
      <button onClick={this.incrementClick}>CLICK VÀO ĐÂY!</button>
    </div>;
  }
}
```

## Ví dụ Lifecycle với FunctionComponents



```
function MouseClick() {
  const [click, setClick] = useState(0);

  // Gọi khi nội dung được render
  useEffect(() => {
    console.log("Click: " + click);
  });

  // Hàm xử lý đếm số click
  function incrementClick() {
    setClick(click + 1);
  }

  return <div className = "mouseClick">
    <h2>{click}</h2>
    <button onClick={incrementClick}>CLICK VÀO ĐÂY!</button>
  </div>
}
```

# Handling Events trong React



Việc xử lý các sự kiện trong React rất giống với việc xử lý các sự kiện trong DOM.

Sự khác biệt duy nhất là tên sự kiện trong React sử dụng cú pháp camelCase và trình xử lý sự kiện cần được truyền trong dấu ngoặc nhọn { }. Ví dụ:

```
<button onClick={incrementClick}>  
CLICK VÀO ĐÂY!  
</button>
```

# Handling Events trong React (tiếp)



Ví dụ xử lý form:

```
function AddForm() {  
  const [num, setNum] = useState(0);  
  const [sum, setSum] = useState(0);  
  
  // Hàm xử lý khi người dùng  
  // nhập vào input  
  function handleChange(e) {  
    setNum(e.target.value);  
  }  
  
  // Hàm xử lý khi người dùng  
  // nhấn nút Cộng (submit)  
  function handleSubmit(e) {  
    setSum(sum + Number(num));  
    e.preventDefault();  
  }  
  
  return <form onSubmit={handleSubmit} className="formSubmit">  
    <input type="number" value={num} onChange={handleChange} />  
    <input type="submit" value="Cộng" />  
    <h3> Tổng là {sum} </h3>  
  </form>;  
}
```



# Dự án Danh sách sinh viên



Yêu cầu:

Tên sinh viên	Thêm sinh viên
---------------	----------------

- Ngọc Anh
- Vũ Hà
- Thu Hương

# Dự án Danh sách sinh viên



```
function SinhViens(props) {  
  // Lấy dữ liệu ban đầu  
  const danhSachSV = props.data;  
  
  // Đặt tên từng sinh viên vào trong thẻ li  
  const sinhViens = danhSachSV.map((sv, index) =>  
    <li key={index}>{sv}</li>  
  );  
  
  return <ul>{sinhViens}</ul>;  
}
```

```
const arrSV = ["Ngọc Anh", "Vũ Hà", "Thu Hương"];  
  
const el = (  
  <div>  
    <ThemSinhVien />  
    <SinhViens data={arrSV} />  
  </div>  
>);
```



```
function ThemSinhVien(props) {
  const [ ten, setTen ] = useState("");

  // Hàm xử lý khi
  // người dùng điền tên sinh viên
  function handleChange(e) {
    setTen(e.target.value);
  }

  // Hàm xử lý khi
  // người dùng nhấn Thêm sinh viên (submit)
  function handleSubmit(e) {
    props.handleSubmit(ten);
    setTen('');
    e.preventDefault();
  }

  return (
    <form onSubmit={handleSubmit}>
      <input type="text"
        placeholder="Tên sinh viên"
        onChange={handleChange}
        value={ten} />
      <button type="submit">Thêm sinh viên</button>
    </form>
  );
}
```



```
function DSSinhVien(props) {  
  const [duLieu, setDuLieu] = useState(props.data);  
  
  function themSVMoi(ten) {  
    setDuLieu([...duLieu, ten]);  
  }  
  
  return (  
    <div>  
      <ThemSinhVien handleSubmit={themSVMoi}/>  
      <SinhViens data={duLieu} />  
    </div>  
  );  
}
```

# Dự án Danh sách sinh viên



Source code:

```
import logo from './logo.svg';
import './App.css';
import { useState } from 'react';

function ThemSinhVien(props) {
  const [ ten, setTen ] = useState("");

  // Hàm xử lý khi người dùng đã điền tên sinh viên
  function handleChange(e) {
    setTen(e.target.value);
  }
}
```

# Dự án Danh sách sinh viên



Source code:

```
// Hàm xử lý khi người dùng nhấn Thêm sinh viên (submit)
function handleSubmit(e) {
  if (ten !== "") {
    props.handleSubmit(ten);
    setTen("");
  }
  e.preventDefault();
}

return (
  <form onSubmit={handleSubmit}>
    <input type="text"
      placeholder="Tên sinh viên"
      onChange={handleChange}
      value={ten} />
    <button type="submit">Thêm sinh viên</button>
  </form>
)
```

# Dự án Danh sách sinh viên



Source code:

```
function SinhViens(props) {  
  // Lấy dữ liệu u ban đầu u  
  const danhSachSV = props.data;  
  
  // Đặt tên từng sinh viên vào trong thẻ li  
  const sinhViens = danhSachSV.map((sv, index) =>  
    <li key={index}>{sv}</li>  
  );  
  
  return <ul>{sinhViens}</ul>;  
}
```

# Dự án Danh sách sinh viên



Source code:

```
function DSSinhVien(props) {  
  const [duLieu, setDuLieu] = useState(props.data);  
  
  function themSVMoi(ten) {  
    setDuLieu([...duLieu, ten]);  
  }  
  
  return (  
    <div>  
      <ThemSinhVien handleSubmit={themSVMoi}/>  
      <SinhViens data={duLieu} />  
    </div>  
  );  
}  
  
export default DSSinhVien
```

