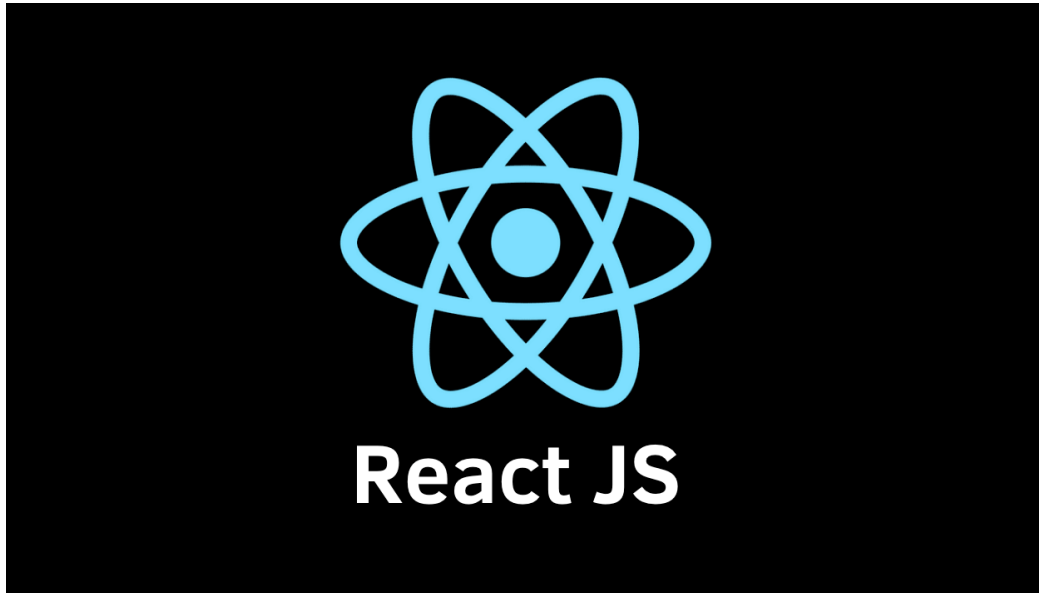
Theory 06
ReactJS Introduction, Component, Props, State

WEEK 02

Nội dung chính

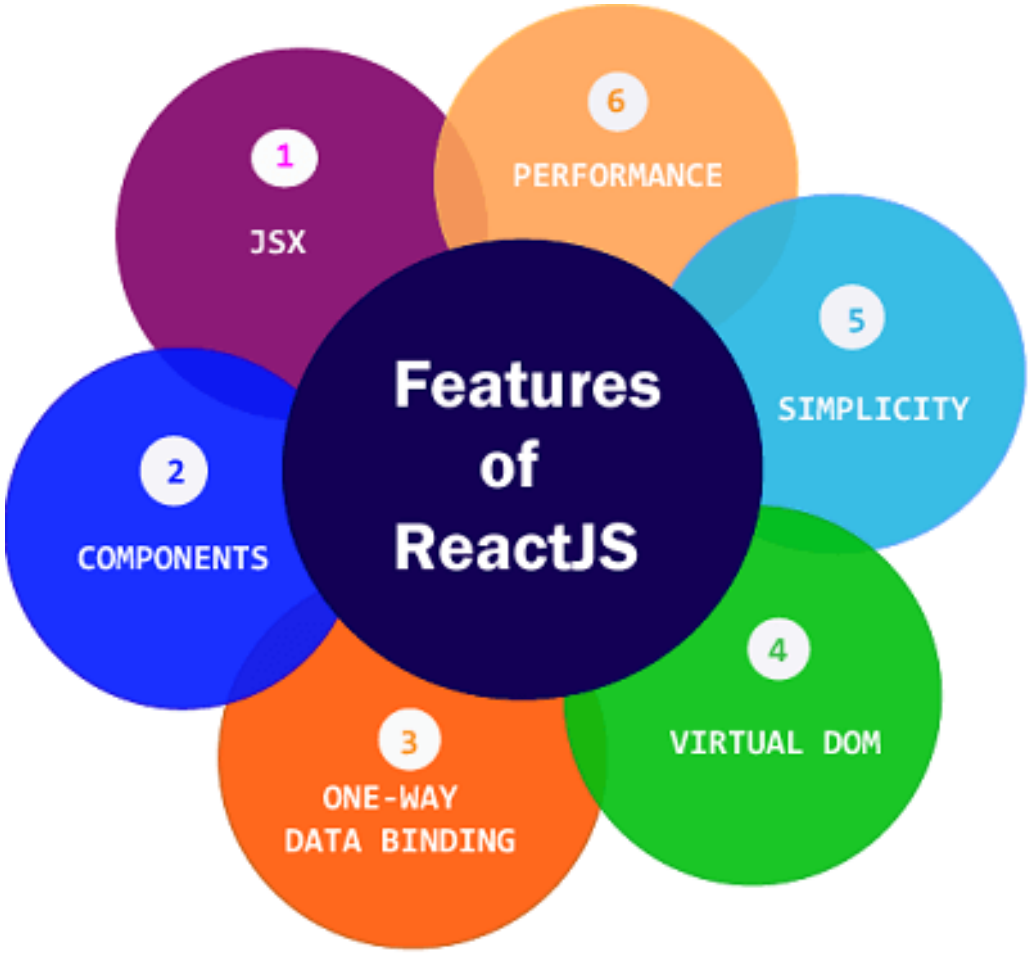
Tổng quan về ReactJS

- **ReactJS** là thư viện JS dùng để xây dựng các thành phần UI có thể tái sử dụng.
- **ReactJS** được phát triển bởi **Facebook**, thường được sử dụng để phát triển ứng dụng single-page, di động, client-side render.
- **ReactJS** trừu tượng hóa **DOM**, có mô hình lập trình đơn giản, hiệu suất tốt hơn.



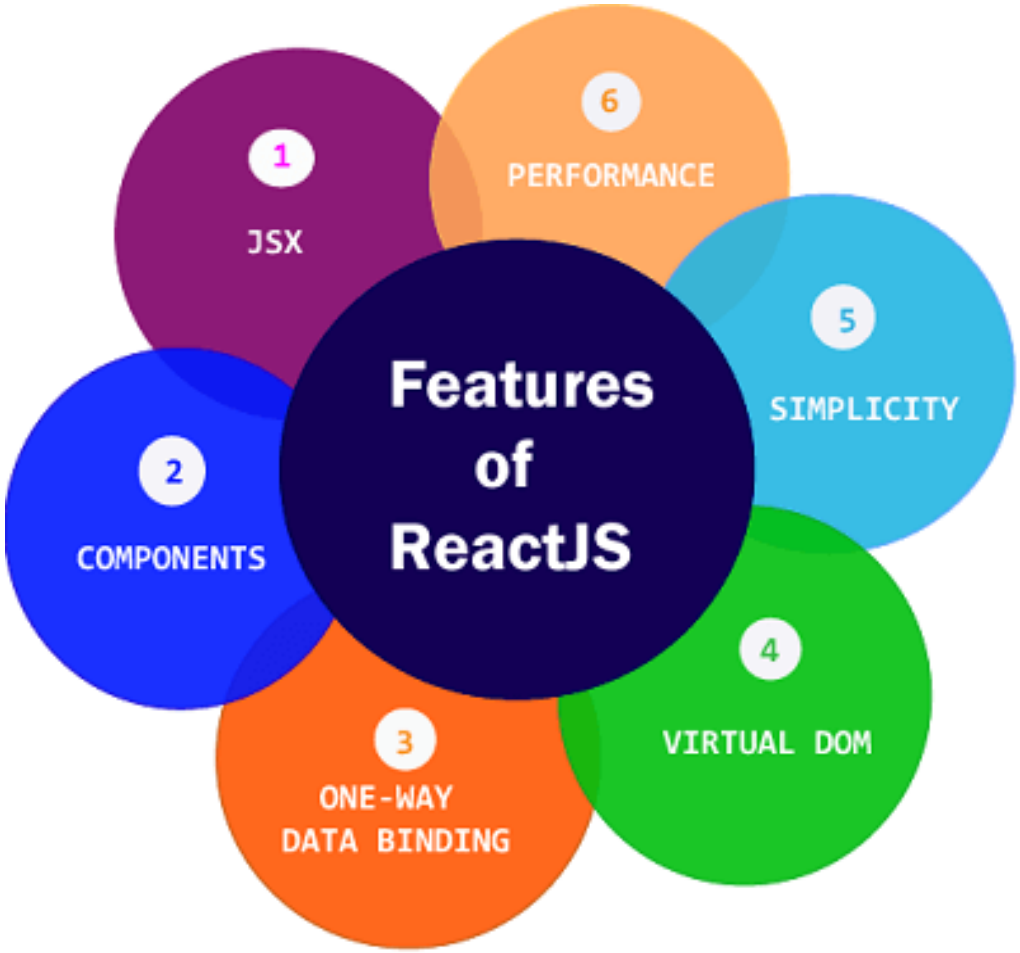
Tính năng của ReactJS

- **JSX** – là sự kết hợp HTML và JS, nhúng JS vào HTML, cú pháp thân thiện, dễ viết, dễ hiểu khi đã biết HTML và JS. Không nhất thiết phải sử dụng JSX trong ReactJS, nhưng được khuyến khích xài.
- **Component** – Làm việc với React là làm việc với Component, mỗi component tự xử lý được logic và các tùy chỉnh. ReactJS giúp dễ dàng chỉnh sửa, tái sử dụng mã nguồn khi làm việc trên các dự án quy mô lớn.



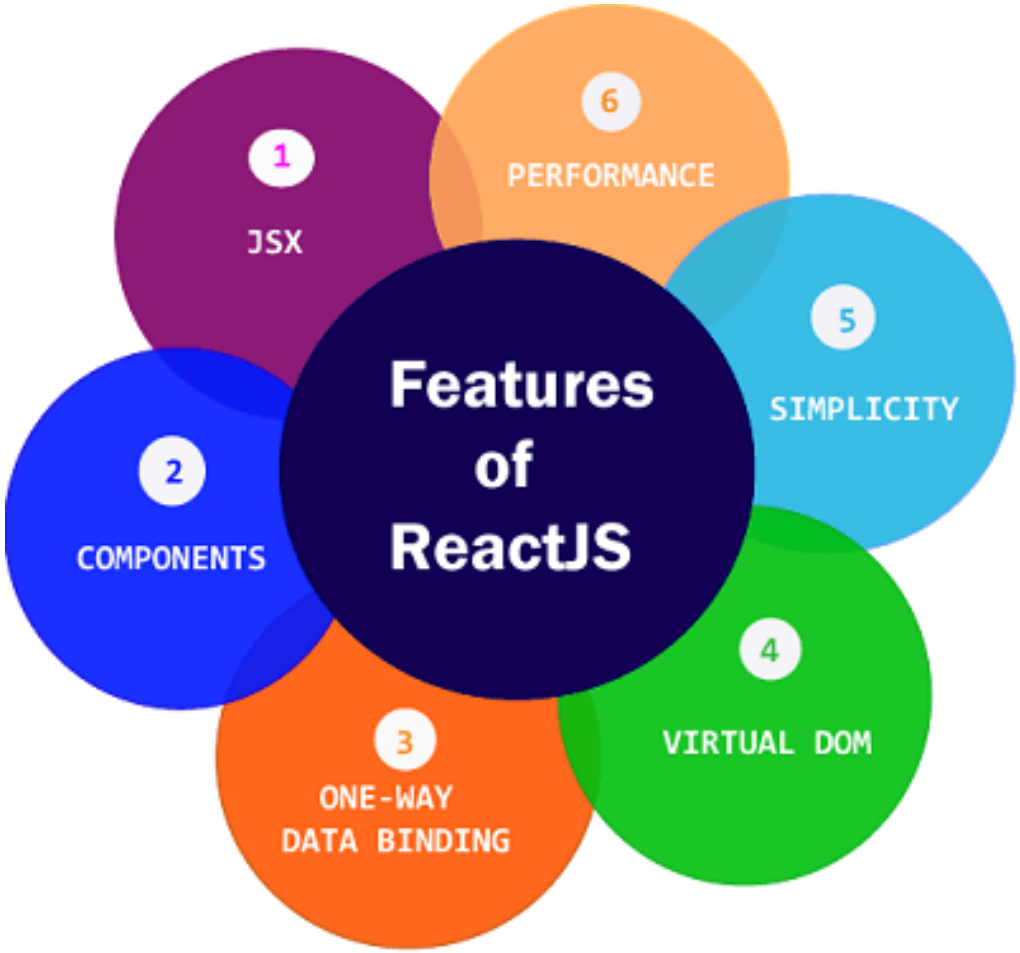
Tính năng của ReactJS

- **One-way Data Binding** – triển khai luồng dữ liệu một chiều giúp bạn dễ dàng suy luận về ứng dụng của mình. Flux là một mẫu giúp dữ liệu truyền theo một chiều nhất định.
- **Virtual DOM** – DOM ảo giúp ReactJS xây dựng bộ cấu trúc đệm cho dữ liệu ở bộ nhớ để tính toán các thay đổi, sau đó cập nhật trình duyệt tại cụ thể các component thực sự thay đổi, không cập nhật lại toàn bộ trang.



Tính năng của ReactJS

- **Simplicity** – ReactJS sử dụng các file JSX làm cho ứng dụng đơn giản, dễ code, đọc dễ hiểu. Với định hướng phát triển dựa trên component giúp dễ phát triển, dễ chỉnh sửa và dễ tái sử dụng mã nguồn.
- **Performance** – Vì sử dụng Virtual DOM, ReactJS chỉ cập nhật phần Component đã thay đổi mà không cập nhật toàn bộ trang, điều này giúp DOM chạy nhanh hơn.



Cài đặt dự án ReactJS

- **Tải và cài đặt:**
  - NodeJS: https://nodejs.org
  - Visual Studio Code: https://code.visualstudio.com
  - Khởi tạo dự án ReactJS theo 2 cách:
    - Sử dụng **create-react-app**
    - Sử dụng **Vite + TypeScript**

Cú pháp React Element, JSX

❑ React Element là gì?

❍ Trong React, **element** là những mảnh ghép nhỏ nhất của các ứng dụng React

❍ Không giống những element DOM của trình duyệt, React **element** là những "đối tượng đơn giản" (plain object) và rất dễ để tạo ra.

Cú pháp React Element, JSX

- **React Element là gì?**
  - React element là những gì được trả về từ các component.
  - Với functional component, element là đối tượng mà hàm trả về.
  - Với class component, element là đối tượng mà hàm render trả về.

Cú pháp React Element, JSX

❑ React Element là gì?

❍ React element không phải là những gì chúng ta thấy trong trình duyệt. Chúng chỉ là những đối tượng trong bộ nhớ và chúng ta không thể thay đổi.

❍ React element có thể có các thuộc tính kiểu khác với các phần tử HTML gốc.

Cú pháp React Element, JSX

- **Cú pháp React Element:**
  - Cú pháp: **React.createElement(type, [props], [...children])**
  - Ví dụ:

```
const hello = React.createElement(
                "H1",
                {id: "msg", className: "title"},
                "Hello React Element"
);
ReactDOM.render(
  hello,
  document.getElementById('root')
);
```

  - Kết quả:

```
{
  type: 'H1',
  props: {
        id: 'msg',
        class:'title'
        children: 'HelloReactElement '
  }
}
```

Cú pháp React Element, JSX

- **Render Element trong DOM**
  - Để render một phần tử React vào root DOM, ta dùng phương thức **ReactDOM.render()**
  - **Cú pháp**: ReactDOM.render(**element**, container[**callback**]);

Cú pháp React Element, JSX

❑ Render Element trong DOM

❍ Ví dụ:

```
const items = ["C++","PHP","JAVA"];
const programming = React.createElement("section", {id: "coding"},
      React.createElement("h1", null, "Programming Languages"),
      React.createElement("ul", {className: "coding"},
                                    items.map((coding, i) =>
                                          React.createElement("li", { key: i }, coding),
                                    );
     ReactDOM.render(programming, document.getElementById('root')
);
```

Cú pháp React Element, JSX

❑ JSX là gì?

❍ **JSX** là viết tắt của **JavaScript XML**, một cú pháp mở rộng cho phép lập trình viết HTML trong React một cách dễ dàng.

❍ React sử dụng **JSX** để tạo khuôn mẫu thay vì JavaScript thông thường.

❍ Bạn không bắt buộc phải sử dụng **JSX** nhưng phần mở rộng này sẽ giúp bạn viết ựng dụng dễ dàng hơn.

Cú pháp React Element, JSX

❑ **Ưu điểm của JSX:**

❍ Mang lại tốc độ nhanh hơn vì nó thực hiện tối ưu hóa khi biên dịch mã sang JS.

❍ Hầu hết các lỗi đều có thể được phát hiện ngay trong quá trình biên dịch mã.

❍ Nếu đã quen thuộc với HTML, JSX sẽ giúp bạn viết các mẫu (templates) nhanh và dễ dàng hơn.

Cú pháp React Element, JSX

❑ Sử dụng JSX trong ReactJS

❍ JSX cho phép chúng ta viết các phần tử HTML bằng JavaScript và đặt chúng trong DOM mà không cần bất kỳ phương thức **createElement**(), **appendChild**().

❍ **Ví dụ**:

➢ Có JSX, đoạn code sẽ như sau:

```
const myElement = <h1>Hello JSX!</h1>;
ReactDOM.render(myElement, document.getElementById('root'));
```

➢ Không có JSX, đoạn code sẽ như sau:

```
const myElement = React.createElement('h1', {}, 'I do not use JSX);
ReactDOM.render(myElement, document.getElementById('root'));
```

Cú pháp React Element, JSX

❑ **Biểu thức JSX trong ReactJS**

❍ Với **JSX** bạn có thể viết các biểu thức bên trong dấu ngoặc nhọn **{}**. Biểu thức có thể là một biến hoặc thuộc tính hoặc bất kỳ biểu thức JavaScript nào đó khác hợp lệ

❍ **Ví dụ**:

➢ Thực hiện biểu thức 1 + 1:

```
const myElement = <h1>React is {1 + 1} times better with JSX </h1>;
```

➢ Kết quả hiển thị trên trình duyệt:

React is 2 times better with JSX

Cú pháp React Element, JSX

- **Chèn khối HTML trong ReactJS**
  - Để viết HTML trên nhiều dòng, bạn có thể đặt HTML bên trong dấu ngoặc đơn
  - **Ví dụ**:
    - Tạo một danh sách với ba danh mục con như sau:

```
const myElement = (
  <ul>
    <li>JavaScript</li>
    <li>ECMAScript 6</li>
    <li>TypeScript</li>
    <li>ReactJS</li>
    <li>NextJS</li>
  </ul>
);
```

Cú pháp React Element, JSX

❑ **Chèn khối HTML trong ReactJS**

❍ Bạn có thể sử dụng các thuộc tính tùy chỉnh của riêng mình ngoài các thuộc tính HTML thông thường.

❍ Khi muốn thêm thuộc tính tùy chỉnh của **JSX** trong ReactJS, bạn cần sử dụng tiền tố **data-**

❍ Trong ví dụ bên phải, ta có **data-myattribute** được thêm làm thuộc tính của phần tử p.

```
import React from 'react';

class App extends React.Component {
  render() {
    return (
      <div>
        <h1>Header</h1>
        <h2>Content</h2>
        <p data-myattribute = "somevalue">This is the
content!!!</p>
      </div>
    );
  }
}

export default App;
```

Cú pháp React Element, JSX

❑ **Chèn khối HTML trong ReactJS**

❍ Khi muốn thiết lập các định kiểu CSS, bạn cần phải sử dụng cú pháp **camelCase**.

❍ React sẽ tự động thêm px sau giá trị số trên các phần tử cụ thể

❍ Ví dụ bên đây sẽ cho bạn thấy cách để thêm CSS **myStyle** vào phần tử h1

```
import React from 'react';

class App extends React.Component {

  render() {
    var myStyle = {
      fontSize: 100,
      color: '#FF0000'
    }

    return (
      <div>
        <h1 style = {myStyle}>Header</h1>
      </div>
    );
  }
}

export default App;
```

Cú pháp React Element, JSX

❑ **Quy tắc đặt tên JSX trong ReactJS**

❍ Các thẻ HTML luôn sử dụng tên thẻ viết thường, trong khi các thành phần **React** bắt đầu bằng chữ hoa.

❍ **Lưu ý**: Bạn nên sử dụng **className** và **htmlFor** làm tên thuộc tính **XML** thay vì class và for

❍ Trên trang web chính thức của React, điều này được giải thích là do JSX là JavaScript nên các định danh như **class** và **for** không được khuyến khích làm tên thuộc tính XML. Thay vào đó, các thành phần React DOM yêu cầu các tên thuộc tính DOM tương ứng như **className** và **htmlFor**

Công dụng React Component

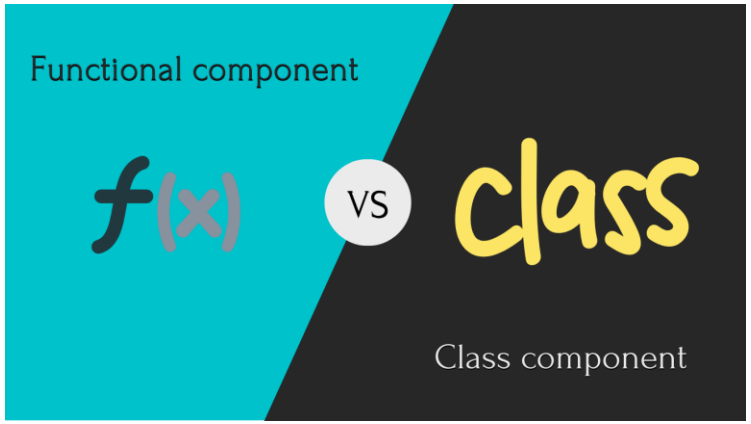
❑ **Component là gì?**

❍ **Component** là một block code độc lập, có thể tái sử dụng.

❍ Khi ứng dụng phình to có hàng ngàn dòng code, việc phân tách thành component là việc cần thiết và hữu ích

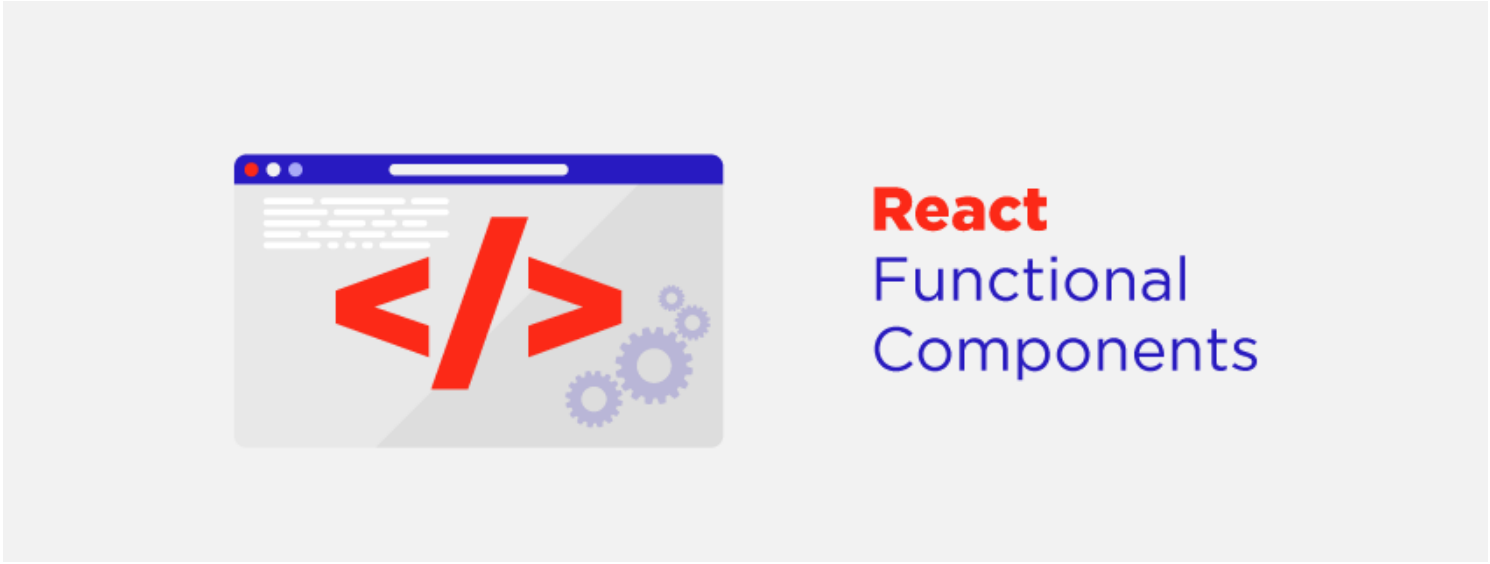
❍ Mỗi component có code JS, CSS riêng, có thể tái sử dụng, dễ đọc, viết, dễ test

❍ Có 2 loại: **Functional Component** (**Stateless**) và **Class Component** (**Stateful**)



Functional/Class Component

- **Functional Component**
  - Functional Component là một **JavaScript/ES6 function**.
  - Functional Component trả về một **React Element**
  - Functional Component có thể nhận **props** làm tham số nếu cần



Functional/Class Component

❑ Functional Component

❍ Ví dụ:

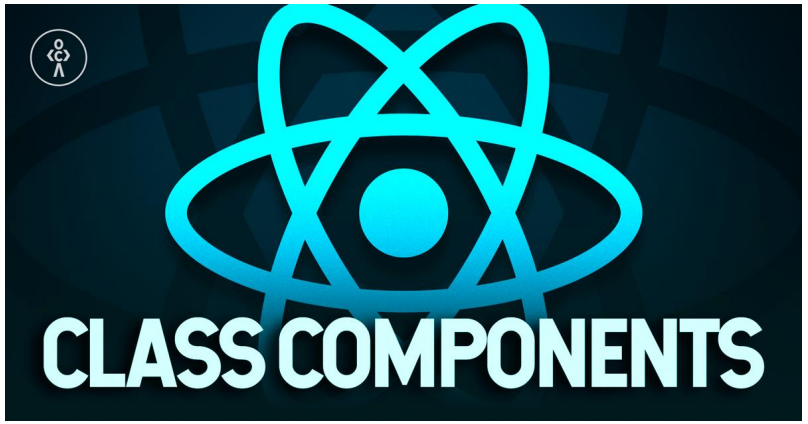
```
import React from 'react';

const App =()=>{
   const greeting = 'Hello Functional Component!';
   return <Headline value={greeting} />;
}
const Headline = ({ value }) => {
   return <h1>{value}</h1>;
};

export default App;
```

Functional/Class Component

- **Class Component**
  - Class Component là một **JavaScript/ES6 class**.
  - Class Component là một component kế thừa từ **React component**
  - Class Component có thể nhận **props** (trong hàm khởi tạo) nếu cần
  - Class Component có thể khởi tạo và cập nhật **state**.
  - Phải có một method **render()**, trả về 1 **React Element** (**JSX**) hoặc null



Functional/Class Component

- **Class Component**
  - **Ví dụ**:

```
class App extends React.Component{
    greeting = “Hello Class Component!”;
    render(){
        return <Headline value={greeting} />
    }
}

class Headline extends React.Component {
    render() {
        return <hi>{this.props.value}</hl>;
    }
}
```

Props và State trong ReactJS

❑ Props là gì?

❍ **Props** là viết tắt của Properties (thuộc tính), được sử dụng để chứa đựng dữ liệu hiển thị trong **Component**, hoạt động tương tự một thuộc tính của HTML.

❍ Trong ReactJS, **props** được **sử dụng để gửi dữ liệu đến các component**

❍ Mọi **component** trong ReactJS đều được coi là một hàm JavaScript thuần tùy, **props** cũng **đóng vai trò tương đương với các tham số của hàm** này

❍ Giá trị của Props có thể được thay đổi bằng cách truyền dữ liệu từ ngoài vào.

❍ Vì **props** là hàm thuần túy nên một khi **props** đã được truyền vào **component**, **giá trị của nó sẽ không thể thay đổi được**.

Props và State trong ReactJS

❑ **Ví dụ Props:**

```
class MyComponent extends React.Component{
        render(){
           return (
                  <div>
                    <h1>Hello</h1>
                    <Header name="Dao" id="123" />
                  </div>
           );
      }
 }
```

Props và State trong ReactJS

❑ Ví dụ Props

```
function Header(props) {
        return (
            <div>
                <Footer name={props.name} id={props.id} />
            </div>
        );
 }
```

Props và State trong ReactJS

❑ Ví dụ Props

```
function Footer(props) {
        return (
            <div>
                <h1> Welcome: {props.name} <h1/>
                <h1> Id is: {props.id} <h1/>
            </div>
        );
    }

    ReactDOM.render(
        <MyComponent/>, document.getElementById('content')
    );
```

Props và State trong ReactJS

❑ **State là gì?**

❍ **State** là **thành phần của component** và đóng vai trò như **một kho lưu trữ dữ liệu cho các component** của **ReactJS**.

❍ **State** thường được sử dụng **để cập nhật các component** khi **người dùng thực hiện một số hành động** như **clicking button**, **typing some text**, **pressing some key**

❍ **React.Component** là lớp cơ sở cho tất cả các thành phần ReactJS dựa trên lớp. Bất cứ khi nào có một lớp kế thừa lớp **React.Component** đó, hàm tạo của nó sẽ tự động gán thuộc tính state cho lớp với giá trị nguyên bản được đặt thành **null**.

❍ Giá trị của Props có thể được thay đổi bằng cách truyền dữ liệu từ ngoài vào.

❍ Chúng ta có thể thay đổi nó bằng cách ghi đè phương thức **constructor**

❍ Khi cần thay đổi/cập nhật **state**, nên hạn chế sử dụng **this.state** = {'key': 'value'} mà nên sử dụng phương pháp **this.setState** để cập nhập các đối tượng.

Props và State trong ReactJS

❑ **Ví dụ State:**

```
class MyComponent extends React.Component{
    render(){
        return (
            <div>
              <h1>Hello</h1>
              <Header name="Dao" id="123" />
            </div>
        );
    }
}
```

Props và State trong ReactJS

❑ **Ví dụ State:**

```
class MyComponent extends React.Component{
        constructor(){}
        render() {
            setTimeout(() => {this.setState({name: 'Teo', id: 456'})}, 3000)
            return(
                <div>
                    <h1>Hello {this.state.name}</h1>
                    <h2>Your id is {this.state.id}</h2>
                </div>
            );
        }
    }
```

Props và State trong ReactJS

❑ **Ví dụ State:**

```
class MyComponent extends React.Component{
    constructor(){}
    render() {
        //...
    }
}

ReactDOM.render(
    <MyComponent/>, document.getElementById('content')
);
```

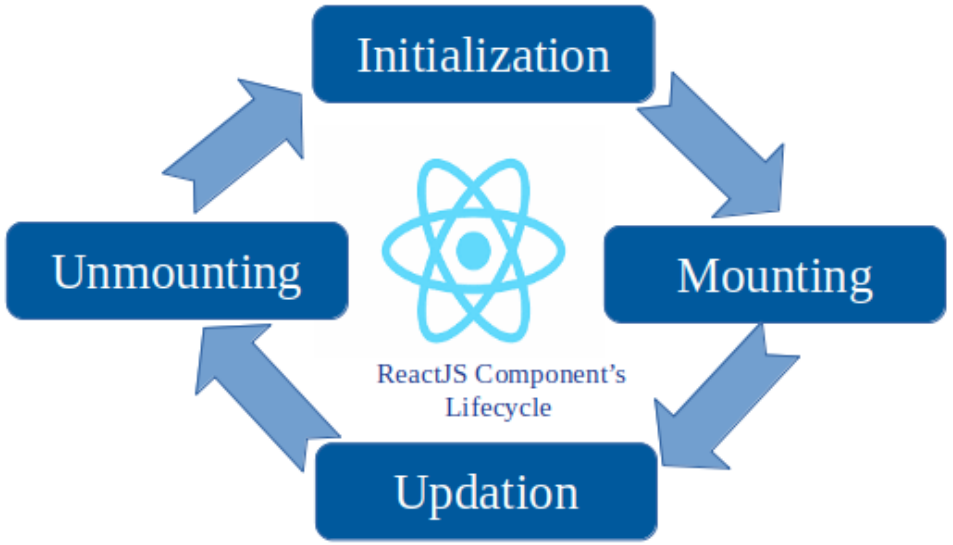
Props và State trong ReactJS

❑ So sánh Props và State:

Điều kiện	State	Props
Nhận các giá trị ban đầu từ các Component cha	V	V
Component cha có thể thay đổi được giá trị	X	V
Đặt các giá trị mặc định bên trong Component	V	V
Có thể thay đổi được khi ở bên trong Component	V	X
Đặt giá trị ban đầu cho các Component con	V	V
Thay đổi bên trong Component con	X	V

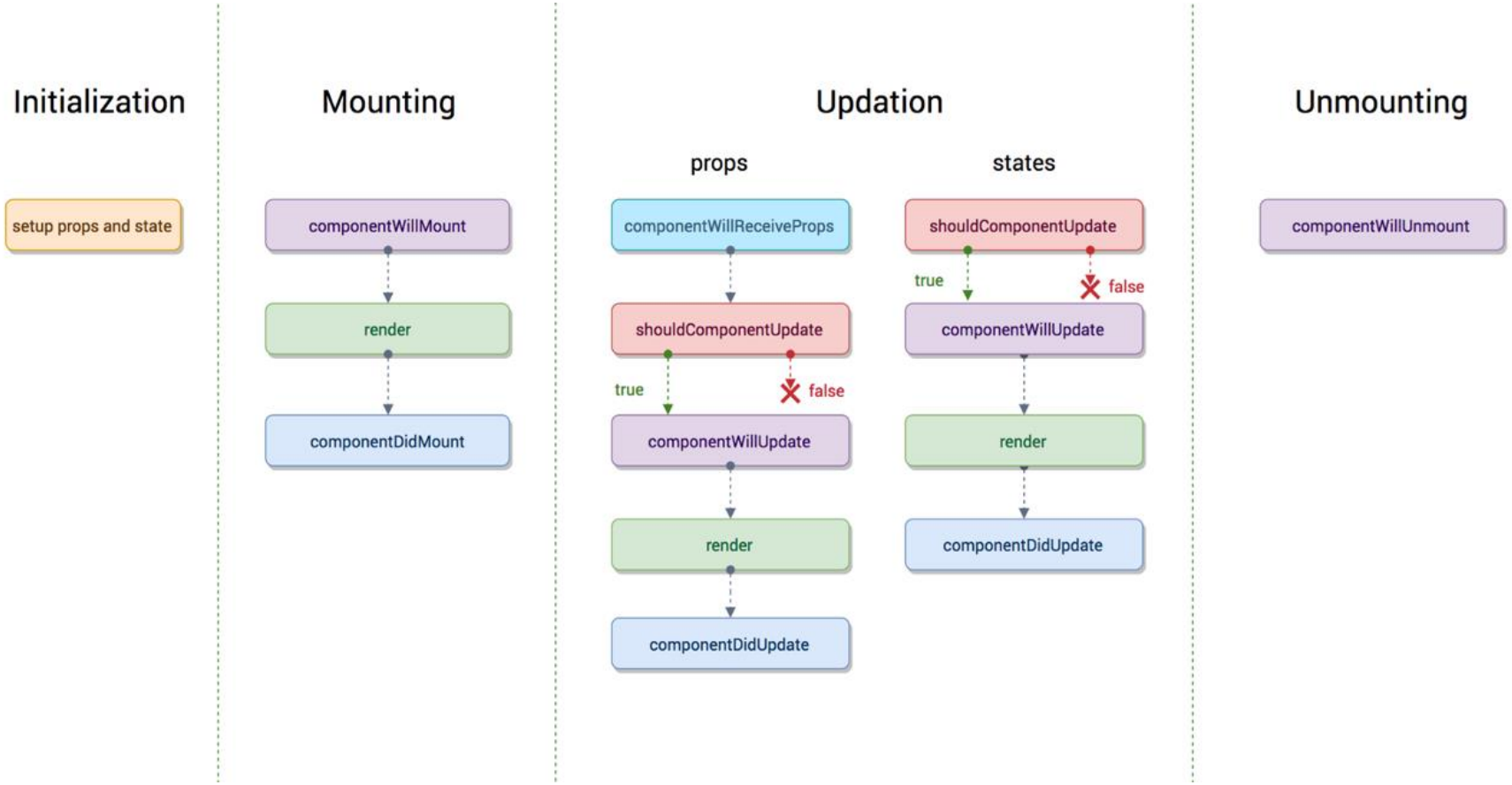
React Component Life-Cycle

- Mỗi **Component** trong React **đều có một vòng đời của riêng nó**
- Vòng đời của một **Component** có thể được định nghĩa là **một loạt các phương thức được gọi trong các giai đoạn khác nhau của sự tồn tại của Component** đó



React Component Life-Cycle

❑ Mỗi **Component** trong React có thể trải qua 4 giai đoạn trong vòng đời của nó

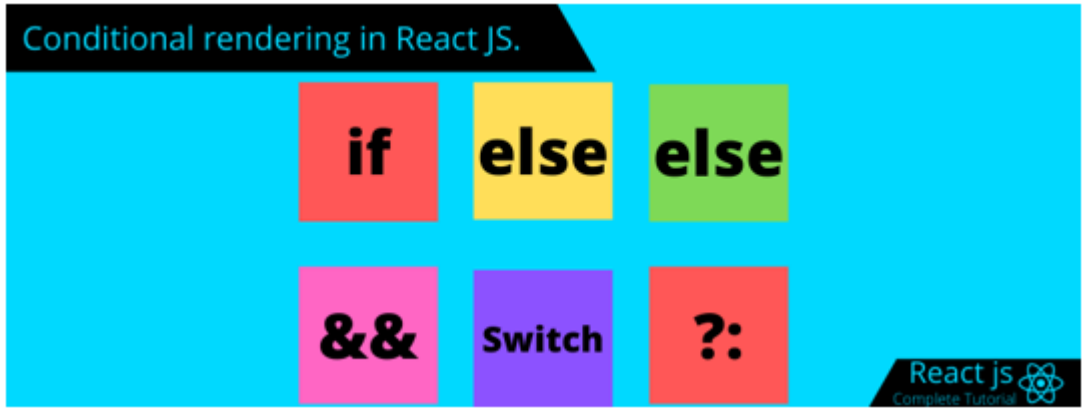


Sử dụng Conditional Rendering

- Trong ReactJS, đôi khi có một số **Component** và **tùy thuộc** vào **từng điều kiện cụ thể** nào đó của **Component** **mà bạn muốn hiển thị** **Component đó hoặc không**
- Khi đó, bạn có thể sử dụng **render có điều kiện** (**conditional rendering**) để **render** ra **component** mà bạn mong muốn

Sử dụng Conditional Rendering

- Có nhiều cách để sử dụng **render** có điều kiện, có thể sử dụng hầu hết các kiến thức trong lập trình để hiện thực **render** có điều kiện, tùy vấn đề cần giải quyết:
  - Sử dụng câu lệnh **if..else** hoặc **switch**
  - Sử dụng **null** để ẩn một **Component**
  - Sử dụng **ternary operator** (toán tử 3 ngôi)
  - Sử dụng **AND operator** (toán tử AND)



Tóm tắt bài học

- ❑ Tổng quan về ReactJS
- ❑ Tính năng của ReactJS
- ❑ Cài đặt dự án ReactJS
- ❑ Cú pháp React Element, JSX
- ❑ Công dụng React Component
- ❑ Functional/Class Components
- ❑ Props và State trong ReactJS
- ❑ React Component Life-Cycle
- ❑ Sử dụng Conditional Rendering