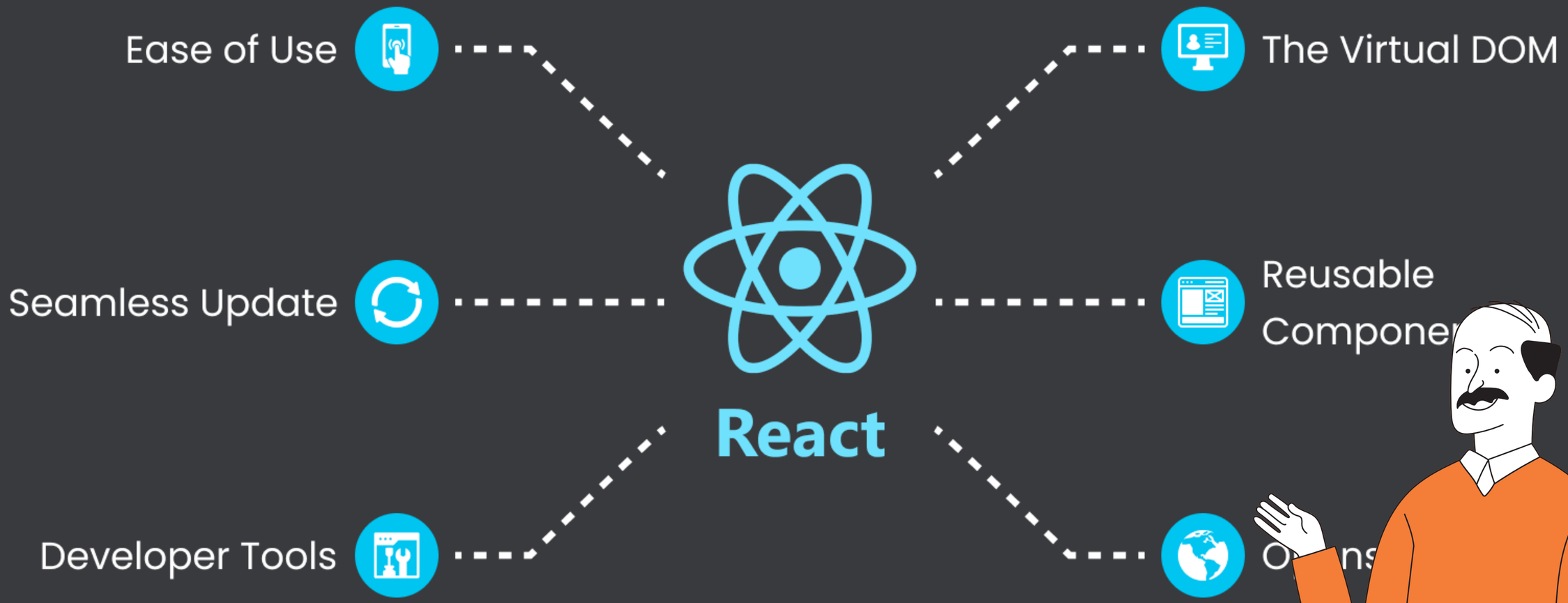


Nhập môn Reactjs

Tổng quan

- Tìm hiểu cơ bản về Reactjs và core concepts của nó.
- Xây dựng một ứng dụng ReactJS đơn giản sử dụng tsx và Vite.
- Sử dụng ReactJS components, state và props để build một ứng dụng phức tạp hơn.
- Hiểu được các cách quản trị state và sử dụng lifecycle.
- Cách làm việc với form và data trong reactjs
- Deploy Reactjs app lên host

Chuẩn bị



JS
node

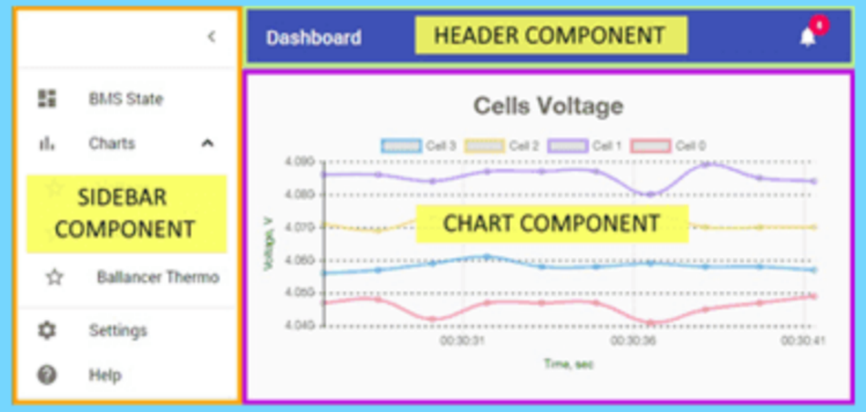

React Basic và core concepts

ReactJS là gì?

- ReactJS là một thư viện JavaScript mã nguồn mở được sử dụng để xây dựng giao diện người dùng.
- ReactJS được tạo ra bởi Facebook và hiện đang được sử dụng rộng rãi trong cộng đồng phát triển web.

Core concepts của ReactJS

- JSX: đây là một cú pháp mở rộng của JavaScript cho phép viết HTML trong JavaScript.
- Components: là các khối xây dựng giao diện độc lập được tái sử dụng trong ứng dụng.



Components

A React component is an individual item that can contain other items. It's like a piece of a puzzle that when combined with other pieces represents your application.

JSX

JSX (JavaScript XML) allows us to write XML-like code for simplicity and elegance, and then it is transpiled into pure JavaScript function calls with React.createElement()

```
class HelloOwls extends React.Component {
  render() {
    return (
      <h1 className='hoot'>Hello {this.name}!</
    );
  }
}
```

your JSX is here

State: đại diện cho trạng thái hiện tại của ứng dụng và có thể được thay đổi bởi các sự kiện.

Props: là các giá trị được truyền vào các component để thay đổi hoạt động của chúng.

```
class Dashboard extends React.Component {
  render() {
    return <Chart title='Cell Voltage' value={50}/>
  }
}
                        passing props

class Chart extends React.Component {
  render() {
    return <LineChart
                title={this.props.title}
                data={this.props.value}  />
  }
}
                 using props
```

Props

A prop in JSX is the same thing as an attribute in HTML, except that you can pass it any data type that JavaScript supports. Once you've passed the data into it, you can easily access your data on this.props.

State

It's a place where you can store component data. State is going to be an object with any key-value pairs that we want to store there. Every component can have state.

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }
             initial state

  render() {
    return (
      <div>
        <h1>It is
          {this.state.date.toLocaleTimeStrin
        </h1>
      </div>          using state
    );
  }
}
```

Lifecycle: là các phương thức được gọi tại các thời điểm khác nhau trong quá trình sống của một component

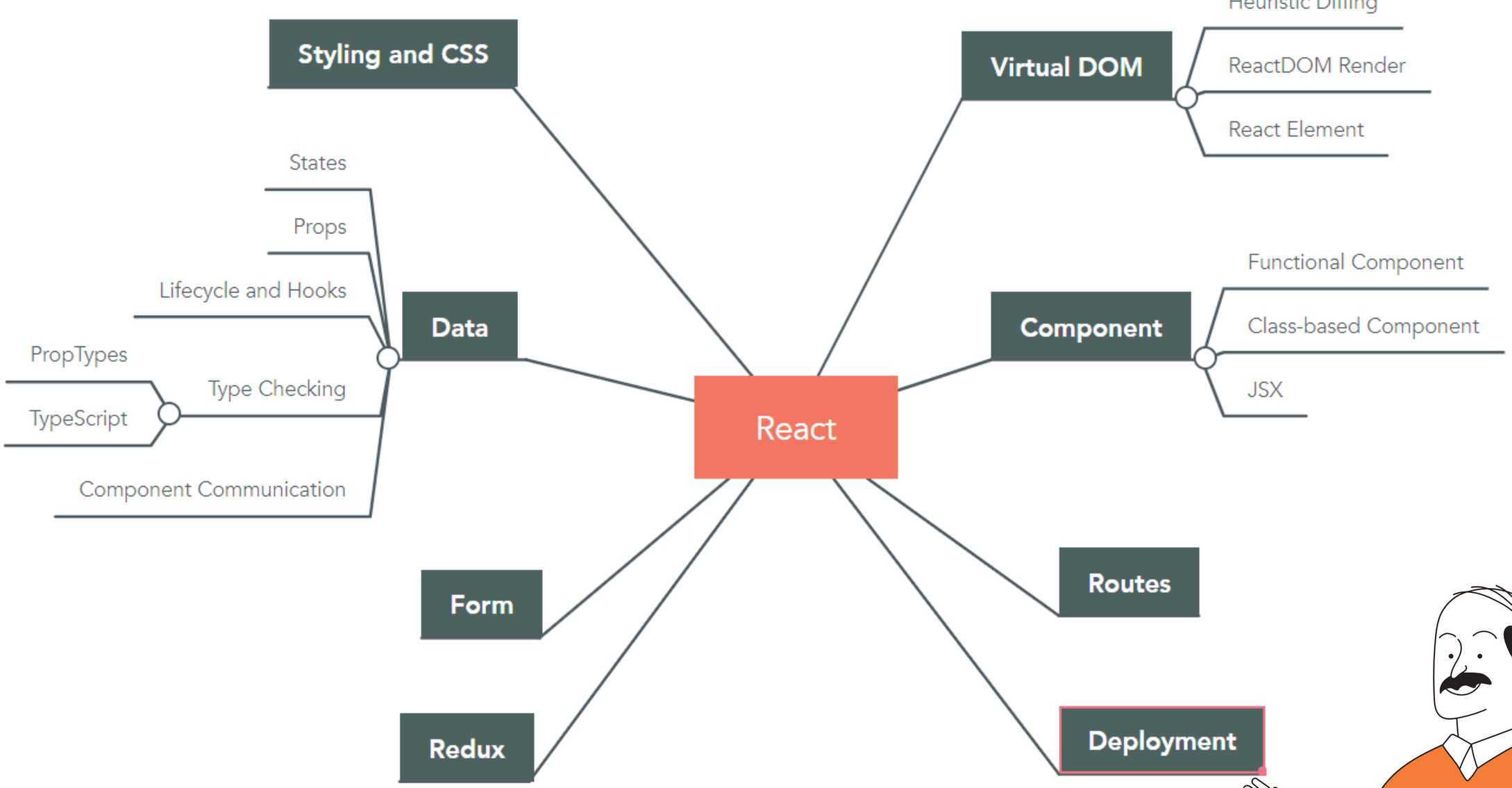
```
                    handler method
<button onClick={this.handleClick}>Click me!</button>
 inline event handler
```

Events in React work pretty much the same as in JavaScript, except that they're wrapped in something called a synthetic Event, which is basically a way to ensure that they work the same across all other browsers and devices.

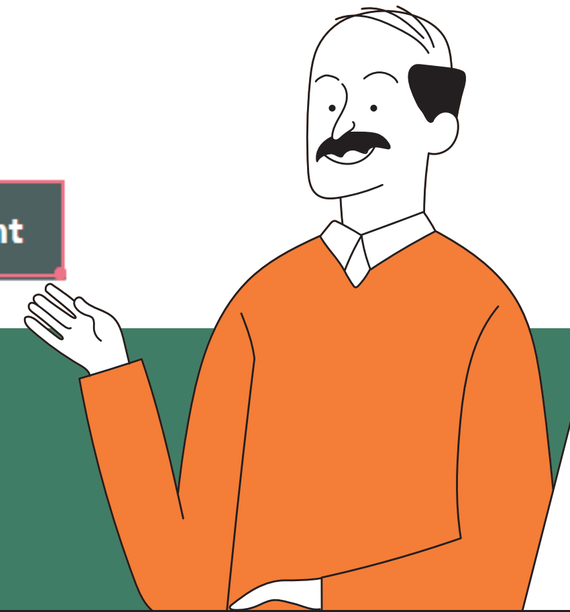
Lifecycle

Every component in React goes through a set of stages that are called lifecycle methods. There's a specific order in which these things run. Some of the widely used methods are: componentDid-Mount(), componentDidUpdate()

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }
            initial state
  render() {
    return (
      <div>
        <h1>It is
          {this.state.date.toLocaleTimeString()}.
        </h1>
      </div>       using state
    );
  }
}
```



Xây dựng một ứng dụng Reactjs



Thực hành sử dụng Components, State và Props

Requirement:

- **Xây dựng SideBar Components.**

- Được ghép lại từ nhiều components nhỏ hơn.

- Có thể render động dựa vào data truyền vào.

Sử dụng các component để tạo các khối giao diện độc lập và tái sử dụng chúng trong ứng dụng.

Rules

Sử dụng các phương thức lifecycle để tương tác với các API hoặc thay đổi DOM.

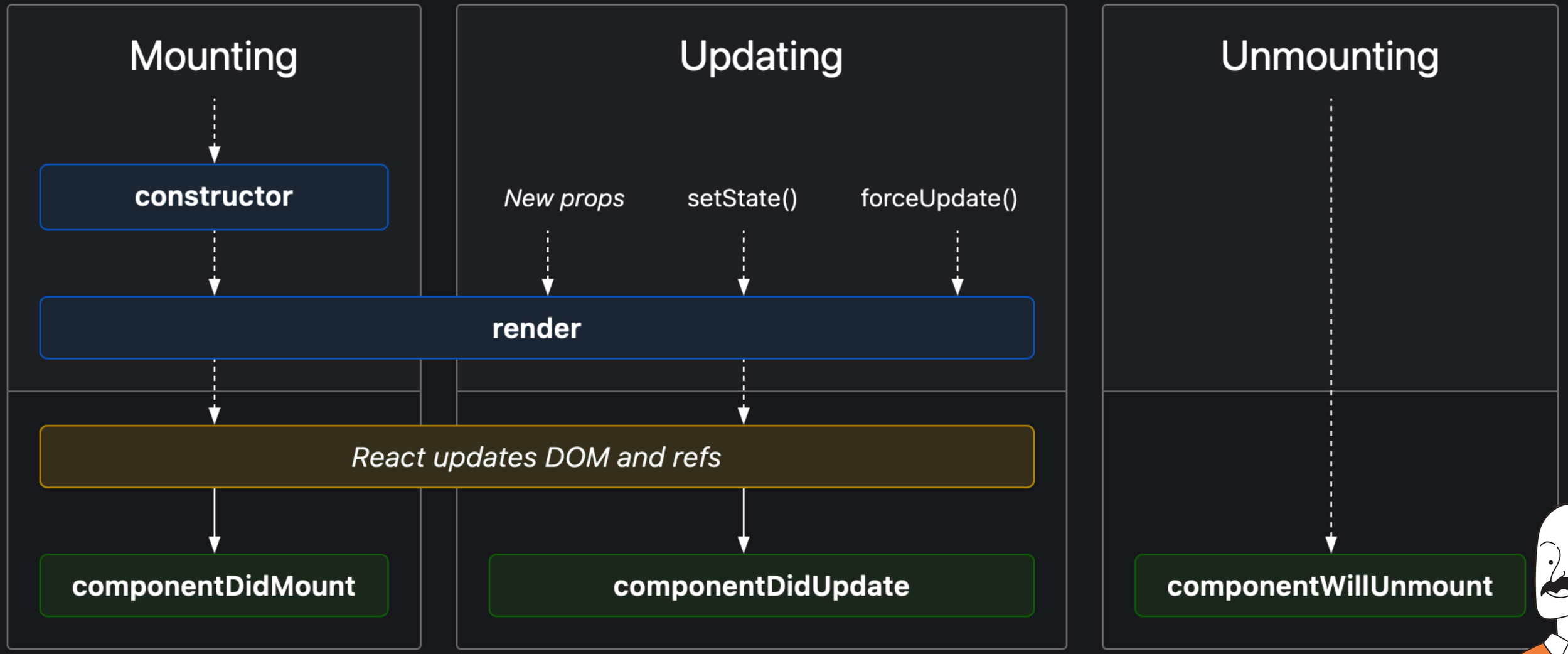
Sử dụng state để lưu trữ trạng thái của ứng dụng.

Sử dụng các phương thức lifecycle để quản lý và thay đổi state của component.

Sử dụng props để truyền các giá trị giữa các component.

Quản lý state và lifecycle

State là một đối tượng JavaScript chứa các thông tin quan trọng và có khả năng thay đổi của component. Khi state thay đổi, React sẽ tự động render lại component. Bằng cách cập nhật state.


Mounting
constructor
render
React updates DOM and refs
componentDidMount
Updating
New props
setState()
forceUpdate()
componentDidUpdate
Unmounting
componentWillUnmount

Lifecycle

Mounting: các phương thức được gọi khi một component được tạo ra và đưa vào DOM.

constructor(): được gọi trước khi component được render.

render(): được gọi để render component.

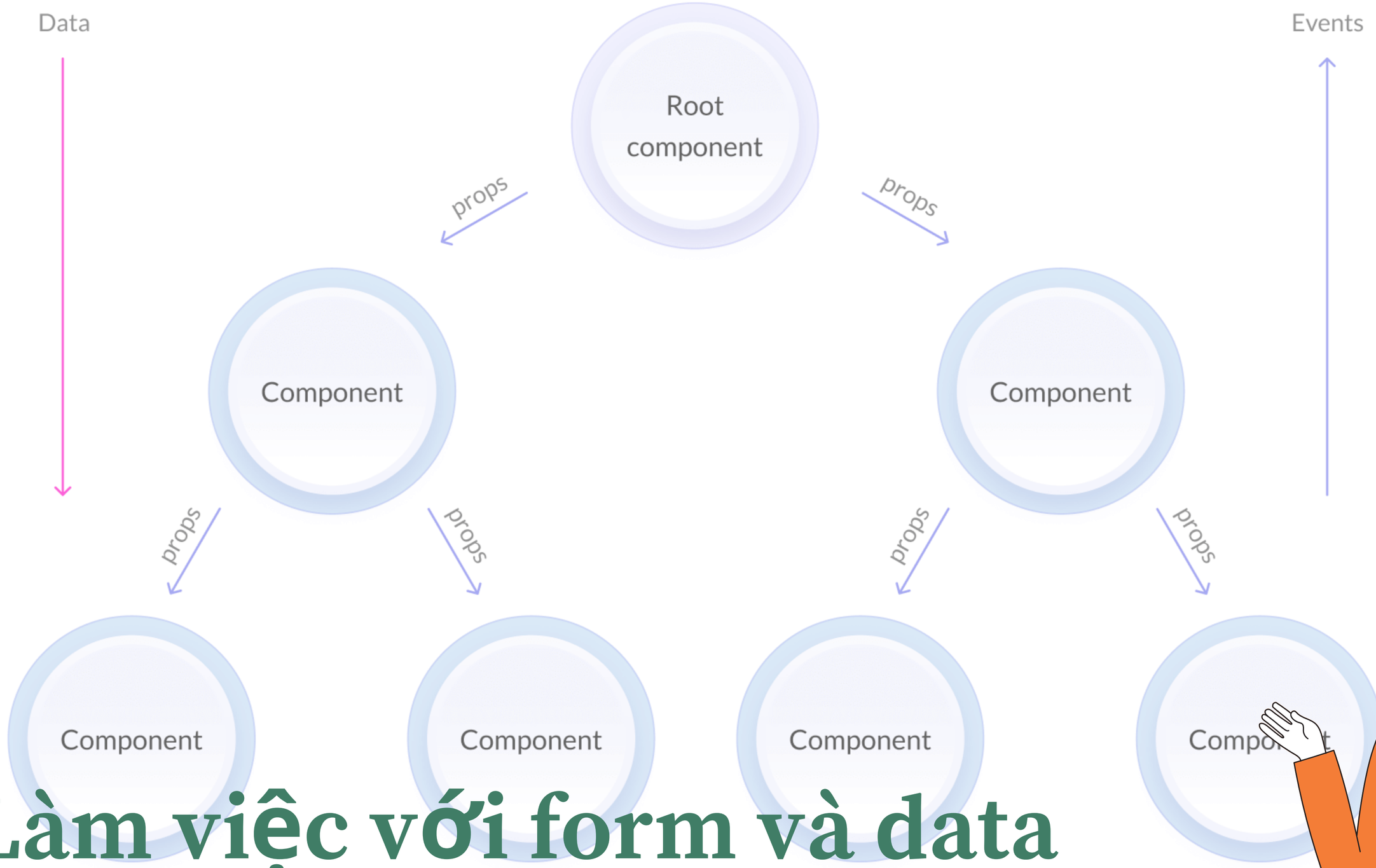
componentDidMount(): được gọi sau khi component được render lần đầu tiên.

Updating: các phương thức được gọi khi state hoặc props của một component thay đổi.

- shouldComponentUpdate(): được gọi để xác định xem component có cần được cập nhật hay không.
- render(): được gọi để render component.
- componentDidUpdate(): được gọi sau khi component đã được cập nhật.

Unmounting: các phương thức được gọi khi một component bị xoá khỏi DOM.

componentWillUnmount(): được gọi trước khi component bị xoá.



Làm việc với form và data

Deploy

Tạo production build

- Sử dụng lệnh ` npm run build ` hoặc ` yarn build ` để tạo production build của ứng dụng.
- Production build sẽ được tạo ra trong thư mục dist.

Deploy lên host

- Sử dụng các dịch vụ hosting như Netlify, vercel hoặc Firebase để deploy ứng dụng ReactJS.
- Tạo một tài khoản và tạo một project mới.
- Tải lên production build của ứng dụng lên host.
- Cấu hình các thiết lập cần thiết để ứng dụng chạy đúng trên host.
- Kiểm tra ứng dụng đã được deploy thành công trên host.

Bài tập

lammas-laptrinh/**React-tutorial**

1 Contributor

0 Issues

0 Stars

0 Forks

lammas-laptrinh/React-tutorial

Contribute to lammas-laptrinh/React-tutorial development by creating an account on GitHub.

GitHub