**Sử dụng Ref , useRef và forwardRef trong React**

Trong bài viết này mình sẽ cùng các bạn tìm hiểu về một thành phần khá quan trọng và được rất nhiều trong các dự án React , đó chính là Refs

**1. Vậy Refs là gì ?**

Ref được viết tắt và có ý nghĩa là reference , là một thuộc tính của 1 thẻ Jsx và tham chiếu tới chính nó . Hay nói cách khác là ref cho phép chúng ta có thể truy cập trực tiếp đến element và sửa đổi nó ngay lập tức mà không cần đến props hay state để component bị re-render lại . Nó khá giống việc chúng ta làm việc với DOM element thông qua việc gọi “document.getElementById()”

**2 . Cách sử dụng Ref với React element**

Hãy cùng xem ví dụ sau : **Đối với Class Component**

import "./App.css";

import React, { Component } from "react";

class App extends Component {

constructor(props) {

super(props);

this.textInput = React.createRef();

}

focusInput = () => this.textInput.current.focus();

render() {

return (

<div className="App">

<input type="text" ref={this.textInput} />

<button onClick={this.focusInput}>Focus input di nao</button>

</div>

);

}

}

Ở ví dụ trên mình đã tạo ra 1 biến ref thông qua hàm React.createRef() và assign cho nó với thuộc tính textInput cuả Component App . Sau đó chúng ta chỉ cần truyền biến ref textInput vào thuộc tính ref của element input . Sau đó element input có thể đưọc truy cập sửa đổi thông qua ref . Cụ thể với ví dụ ở trên , mình tạo ra một 1 button ,mỗi lần click button, event focusInput sẽ thông qua biến refs và thực hiện method focus() (đây là 1 method được định nghĩa sẵn trong input element)

Tương tự với function component :

const App = ()=> {

const textInput = React.useRef()

const focusInput = () => textInput.current.focus();

return (

<div className="App">

<input type="text" ref={textInput} />

<button onClick={focusInput}>Focus input di nao</button>

</div>

);

}

Ở trong function component chúng ta sẽ không sử dụng React.createRef() mà sẽ sử dụng React.useRef() . Đây là 1 tính năng mới của Hooks được phát triển để sử dụng trong function component . Ngoài ra có một lý do nữa là nếu sử dụng React.createRef() , mỗi lần component re-render , hàm này sẽ tạo ra 1 instance ref thay vì giữ nguyên ref instance ban đầu . Điều này sẽ ảnh hưởng performance .

**3. Cách sử dụng ref ,forwardRef với React component**

Như chúng ta đã tìm hiểu bên trên ref có thể giúp truy cập đến 1 element vậy nó có khả năng giúp chúng ta truy cập đến 1 component hay không ?

Khi các bạn lập trình với React , các bạn sẽ có xu hướng tách nhỏ Component của mình ra thành các modules ( sub component) . Điều này giúp clean code và sẽ giúp cho quá trình maintain sau này trở lên dễ dàng hơn . Lúc mới bắt đầu làm quen với React , mình chỉ biết mỗi truyền dữ liệu , event từ component Cha xuống các Component con thông qua Props .

Mình đã tự hỏi là “vậy có cách nào get được data ,thao tác event của component Con khi đứng tại component Cha ?. Thôi thôi lằng nhằng, viết chung 1 component cho nó lành” .Đấy chính là lý do 1 trang code của mình khi có đến 5 6 trăm dòng code và bây giờ nhìn lại nó trông không khác gì một mớ hỗn độn .Sau này khi biết đến ref và mới nhất là forwardRef, đây chính là 1 trong các chìa khóa giúp mình cải thiện chất lượng code React.

Bài toán : làm thế nào để có thể lấy được text từ input của component Con và hiển thị được nó ở component Cha và đứng từ component Cha gọi được hàm increaseNumber từ component Con .Hãy cùng xem demo sau :

class Child extends React.Component {

constructor(props) {

super(props);

this.state = {

text: "",

number: 1,

};

}

handleChange = (event) => {

this.setState({ text: event.target.value });

};

increaseNumber = () => this.setState({ number: this.state.number + 1 });

returnText = () => this.state.text

render() {

return (

<div>

<input type="text" onChange={this.handleChange} />

<p>{this.state.number}</p>

</div>

);

}

}

class Father extends React.Component {

constructor(props) {

super(props);

this.childRef = React.createRef();

}

showTextInChild = ()=>{

alert(‘Text in Child : ' + this.childRef.current.returnText())

}

increaseNumberInChild = ()=>{

this.childRef.current.increaseNumber()

}

render() {

return (

<div className="App">

<Child ref={this.childRef} />

<button onClick={this.showTextInChild}>Show Text in Child compoent</button>

<p></p>

<button onClick={this.increaseNumberInChild}>Increase number in Child compoent</button>

</div>

);

}

}

Đối với các class component chúng sẽ mặc định có thuộc tính ref như mình trình bày với component Child ở trên . Tuy nhiên nếu các component Child là function component , bạn sẽ phải sử dụng thêm forwardRef để bọc Child component như 1 HOC (higher order component) để có thể sử dụng biến ref được truyền vào component Con thông qua props và sử dụng React.useImperativeHandle như sau đây để có thể gọi được các method thông qua thuộc tính ref của được gán giá trị từ component Cha .

import React, { Component, useImperativeHandle, useRef } from "react";

const Child = React.forwardRef((props, ref) => {

const [text, setText] = React.useState("");

const [number, setNumber] = React.useState(1);

const handleChange = (event) => {

setText(event.target.value);

};

useImperativeHandle(ref, () => ({

incrseaseNumber: () => setNumber(number + 1),

returnText: () => text,

}));

return (

<div>

<input type="text" onChange={handleChange} />

<p>{number}</p>

</div>

);

});

class Father extends React.Component {

constructor(props) {

super(props);

this.childRef = React.createRef();

}

showTextInChild = () => {

const txt = this.childRef.current.returnText();

alert(txt);

};

increaseNumberInChild = () => {

this.childRef.current.incrseaseNumber();

};

render() {

return (

<div className="App">

<Child ref={this.childRef} />

<button onClick={this.showTextInChild}>

Show Text in Child compoent

</button>

<p></p>

<button onClick={this.increaseNumberInChild}>

Increase number in Child compoent

</button>

</div>

);

}

}

Tóm lại đối với function component để có thể truyền biến ref thông qua component thì bắt buộc phải bọc component được truyền bên trong forwardRef .Còn Class component sẽ không cần điều đó

**Tổng kết**

Thông qua bài viết này , hy vọng mọi người có thể hiểu được ref, useRef và cách sử dụng chúng một cách hiệu quả !!