**useContext trong React Hook**

Trong bài viết này chúng ta sẽ cùng nhau **đi tìm hiểu về useContext trong React Hooks**. Đây là bài viết tiếp nối các bài viết về React Hooks trước đó. Để hiểu rõ hơn về các khái niệm trong bài viêt này chúng ta có thể tham khảo thêm về bài viết [React Context API](https://freetuts.net/context-trong-reactjs-2431.html).

**Mục lục**

* [1. useContext là gì ?](https://freetuts.net/usecontext-trong-react-hook-2611.html#goto-h2-0)
* [2. Ví dụ về useContext trong React](https://freetuts.net/usecontext-trong-react-hook-2611.html#goto-h2-1)

**1. useContext là gì ?**

**useContext** là một hooks trong React Hooks cho phép chúng ta có thể làm việc với React Context trong một functional component. Giả sử khi chúng ta muốn lấy giá trị của context trong class component:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | class MyClass extends React.Component {    render() {      //Lấy giá trị của context      const value = this.context;    }  }  MyClass.contextType = MyContext; |

Bạn cũng có thể lấy giá trị của context trong funcational componetn bằng cách sử dụng useContext.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | import AppContext from './appContext.js';    const MyClass = () => {    //Lấy giá trị của context    const value = useContext(AppContext);    return (      ...    );  } |

Để sử dụng useContext chúng ta sẽ truyền vào hooks này một tham số duy nhất đó là *Context Object*(được tạo bởi React.createContext).

*Bài viết này được đăng tại [free tuts .net]*

|  |  |
| --- | --- |
| 1 | const AppContext = React.createContext({ foo: 'bar' }); |

khi cần lấy giá trị của context trong funtional component chúng ta sẽ sử dụng:

|  |  |
| --- | --- |
| 1 | const context = useContext(AppContext); |

Phần tiếp theo chúng ta sẽ đi vào xây dựng các ví dụ liên quan đến useContext dựa vào các kiến thức đã học ở các phần trước.

**2. Ví dụ về useContext trong React**

Ở đây chúng ta sẽ đi xây dựng một bộ đếm sử dụng React Context, sử dụng hook useContext để lấy giá trị của context.

Ví dụ này chúng ta sẽ viết trực tiếp vào file src/App.js, ở đây chúng ta sẽ tiến hành khởi tạo một Context.

|  |  |
| --- | --- |
| 1 | const CounterContext = React.createContext(); |

Tiếp theo, chúng ta sẽ bọc các component cần chia sẻ state bằng Provider, và truyền giá vào props value giá trị của context.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | export default class App extends Component {    constructor(props) {      super(props);      this.state = {        count: 1,      };    }    //Cập nhật giá trị của state count - Tăng    countUp() {      this.setState({        count: this.state.count + 1,      });    }      //Cập nhật giá trị của state count - Giảm    countDown() {      this.setState({        count: this.state.count - 1,      });    }    render() {      return (        <CounterContext.Provider          value={{            count: this.state.count,            countUp: this.countUp.bind(this),            countDown: this.countDown.bind(this),          }}        >          <Counter />        </CounterContext.Provider>      );    }  } |

Bên trên chúng ta đã gán giá trị của *Context* là một object bao gồm count, và 2 hàm cho phép tăng giảm giá trị của count. Đồng thời, bên trong Provider chúng ta đã gọi component Counter.

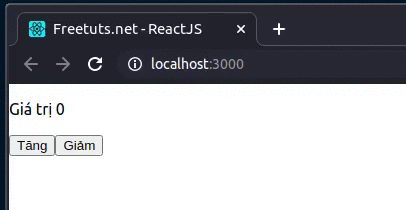
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | const Counter = () => {    // Sử dụng useContext để lấy gía trị của context    // Tham số truyền vào là object CounterContext.    const counter = useContext(CounterContext);      return (      <div>        <h1>{counter.count}</h1>        <button          onClick={() => {            counter.countUp();          }}        >          Tăng        </button>        <button          onClick={() => {            counter.countDown();          }}        >          Giảm        </button>      </div>    );  }; |

Componet Counter có nhiệm vụ hiển thị giá trị của context, và gọi các hành động liên quan như tăng giảm. Ở trên, chúng ta đã sử dụng useContext để lấy giá trị của CouterContext. Khi một hành động tương ứng được thực thi thì giá trị của context sẽ được thay đổi.

Cuối cùng, chúng ta sẽ có file App.js hoàn chỉnh như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62 | import React, { Component, useContext } from "react";  // Khởi tạo context mới.  const CounterContext = React.createContext();  const Counter = () => {    // Sử dụng useContext để lấy gía trị của context    // Tham số truyền vào là object CounterContext.    const counter = useContext(CounterContext);      return (      <div>        <code>freetuts.net</code>        <h1>{counter.count}</h1>        <button          onClick={() => {            counter.countUp();          }}        >          Tăng        </button>        <button          onClick={() => {            counter.countDown();          }}        >          Giảm        </button>      </div>    );  };  export default class App extends Component {    constructor(props) {      super(props);      this.state = {        count: 1,      };    }    //Cập nhật giá trị của state count - Tăng    countUp() {      this.setState({        count: this.state.count + 1,      });    }    //Cập nhật giá trị của state count - Giảm    countDown() {      this.setState({        count: this.state.count - 1,      });    }    render() {      return (        <CounterContext.Provider          value={{            count: this.state.count,            countUp: this.countUp.bind(this),            countDown: this.countDown.bind(this),          }}        >          <Counter />        </CounterContext.Provider>      );    }  } |

Khởi chạy project chúng ta sẽ nhận được kết quả như hình.



Chúng ta có thể tách các đoạn code thành các file riêng bíệt để dễ dàng quản lí hơn sau này. Ví dụ bên trên khá quen thuộc và đơn giản. Chủ yếu dựa trên kiến thức về React Context, bạn nên xem lại bài viết về Context API trong ReactJS để có thể hiểu rõ hơn về phần này.

Trên đây chúng ta đã cùng nhau đi tìm hiểu về useContext trong React Hooks. Đây là kiến thức rất cơ bản về nó nhưng cũng hết sức quan trọng trong quá trình làm việc với ReactJS sau này. Mong rằng bài viết sẽ giúp ích cho bạn.