**Tìm hiểu về Context trong ReactJS**

Trong bài viết này chúng ta sẽ cùng nhau đi**tìm hiểu về Context trong ReactJS**. Trong quá trình làm việc với ReactJS, các dữ liệu trong các component phải được chia sẻ với nhau.

Chúng ta thực hiện điều này bằng cách đưa các dữ liệu này lên một nơi có tên là local state. Local state chịu trách nhiệm phân phối dữ liệu tới các component. **Context** hỗ trợ chúng ta thực hiện điều này một cách đơn giản.

**Mục lục**

* [1. Context trong ReactJS là gì ?](https://freetuts.net/context-trong-reactjs-2431.html#goto-h2-0)
* [2. Context API trong ReactJS](https://freetuts.net/context-trong-reactjs-2431.html#goto-h2-1)
  + [React.createContext](https://freetuts.net/context-trong-reactjs-2431.html#goto-h3-0)
  + [Context.Provider](https://freetuts.net/context-trong-reactjs-2431.html#goto-h3-1)
  + [Context.Consumer](https://freetuts.net/context-trong-reactjs-2431.html#goto-h3-2)
  + [Class.contextType](https://freetuts.net/context-trong-reactjs-2431.html#goto-h3-3)
* [3. Sử dụng Context trong ReactJS](https://freetuts.net/context-trong-reactjs-2431.html#goto-h2-2)

**1. Context trong ReactJS là gì ?**

Theo tài liệu chính thức của ReactJS định nghĩa **Context** là :

Context provides a way to pass data through the component tree without having to pass props down manually at every level.

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

Chúng ta có thể hiểu:

Context cung cấp cho chúng ta cách để thực hiện chia sẻ dữ liệu tới các component trong cây mà không cần truyền dữ liệu qua props theo từng cấp bậc.

Khi truyền dữ liệu tới các component bằng props thì bạn chỉ có thể truyền từ component cha sang component con. Nếu bạn muốn truyền sang component cháu hoặc sang component họ hàng xa thì điều này rất phức tạp. Bởi vậy **context** sẽ là kênh giao tiếp cho các component, cho phép bạn truyền dữ liệu một cách đơn giản hơn rất nhiều.

Ở đây mình có một ví dụ, mình muốn chuyển lời nhắn từ ComponetÔng sang Componet Cháu :

Khi sử dụng truyền dữ liệu qua props thì mình bắt buộc phải gửi lời nhắn qua (*Ông -> Cha -> Cháu* )

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import React from "react";  const ComponentChau = (props) => {    return <h1>Ông bảo là "{props.message}"</h1>;  };  const ComponentCha = (props) => {    return <ComponentChau {...props} />;  };    const ComponentOng = () => {    const message = "Vào freetuts.net học lập trình";    return <ComponentCha message={message} />;  };  export default ComponentOng; |

Còn nếu sử dụng **Context** thì chúng ta sẽ gửi trực tiếp từ *Ông* đến *Cháu* luôn :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | import React from "react";    const MessageContext = React.createContext();    class ComponentChau extends React.Component {    render() {      return <h1>Ông bảo là : "{this.context}"</h1>;    }  }  ComponentChau.contextType = MessageContext;    const ComponentOng = () => {    return (      <MessageContext.Provider value="Vào freetuts.net học lập trình">        <ComponentChau />      </MessageContext.Provider>    );  };  export default ComponentOng; |

Kết quả của 2 cách này vẫn tương tự nhưng việc sử dụng **Context** sẽ đơn giản hơn nhiều.

**2. Context API trong ReactJS**

Trước tiên, chúng ta sẽ đi tìm hiểu về **Context API**, sau đó sẽ đi vào ví dụ cụ thể nhé. Ở đây chúng ta có một vài API mà React cung cấp.

**React.createContext**

|  |  |
| --- | --- |
| 1 | const MyContext = React.createContext(defaultValue); |

Khởi tạo một **Context Object,** giá trị của defaultValue là giá trị mặc định của props value trong Provider.

**Context.Provider**

|  |  |
| --- | --- |
| 1 | <MyContext.Provider value={/\* some value \*/}> |

Mỗi **Context Object**phải đi kèm với một Provider, nó cho phép bạn nhận về sự thay đổi của *context*.

**Context.Consumer**

|  |  |
| --- | --- |
| 1  2  3 | <MyContext.Consumer>    {value => /\* render something based on the context value \*/}  </MyContext.Consumer> |

Một React component được khởi chạy mỗi khi gía trị của context thay đổi, và nhận về giá trị của context đó.

Context.displayName

|  |  |
| --- | --- |
| 1  2  3  4  5 | const MyContext = React.createContext(/\* some value \*/);  MyContext.displayName = 'MyDisplayName';    <MyContext.Provider> // "MyDisplayName.Provider" in DevTools  <MyContext.Consumer> // "MyDisplayName.Consumer" in DevTools |

Đặt tên cho Context, tên này sẽ được hiện thị trong DevTools.

**Class.contextType**

class MyClass extends React.Component {

render() {

let value = this.context;

}

}

MyClass.contextType = MyContext;

contextType là một thuộc tính của class được tạo bởi React.createContext() được dùng để lấy giá trị của context.

**3. Sử dụng Context trong ReactJS**

Chúng ta sẽ sử dụng context theo 3 bước cụ thể:

1. Khởi tạo object context bằng phương thức React.createContext(), sau đó chúng ta sẽ nhận được 1 object bao gồm các thuộc tính quan trọng như Provider và Consumer.
2. Sử dụng Provider bọc quanh các component, và truyền giá trị vào props value
3. Thêm Consumer vào bất cứ đâu mà bạn muốn chia sẻ context miễn là ở bên trong Provider, bạn có thể lấy gía trị của context thông qua props.chirlden.

Bây giờ chúng ta sẽ đi vào ví dụ cụ thể. Mình sẽ xây dựng 1 ứng dụng cho phép random các số và hiển thị. Trước tiên, mình sẽ tiến hành khởi tạo một object context.

|  |  |
| --- | --- |
| 1  2  3 | import React from "react";  //Khởi tạo môt  const NumberContext = React.createContext(); |

Sau đó, mình sẽ bọc quanh các component cần chia sẻ bằng Provider, và Consumber :

|  |  |
| --- | --- |
| 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 | export default class ContextComponent extends React.Component {    constructor(props) {      super(props);      this.state = {        number: 0,      };    }    updateNumber = () => {      this.setState({        number: Math.random(),      });    };    render() {      return (        <NumberContext.Provider          value={{            number: this.state.number,            update: this.updateNumber.bind(this),          }}        >          <NumberContext.Consumer>              { () =>  (                <>                  <ShowNumber />                  <UpdateNumber />                </>              )}          </NumberContext.Consumer>        </NumberContext.Provider>      );    }  } |

bên trên mình truyền vào props value trong provider giá trị là một object có các thuộc tính như:

* number: Chứa giá trị của số.
* update: hàm thực hiện update số.

Consumer sẽ có nhiệm vụ xem xét sự thay đổi và trả về giá trị của context trong props chirlden, mình sẽ chia sẻ props cho 2 component đó là ShowNumber và UpdateNumber.

Tiếp theo, mình sẽ đi xây dựng 2 component là ShowNumber và UpdateNumber mà mình đã gọi ở Consumer.

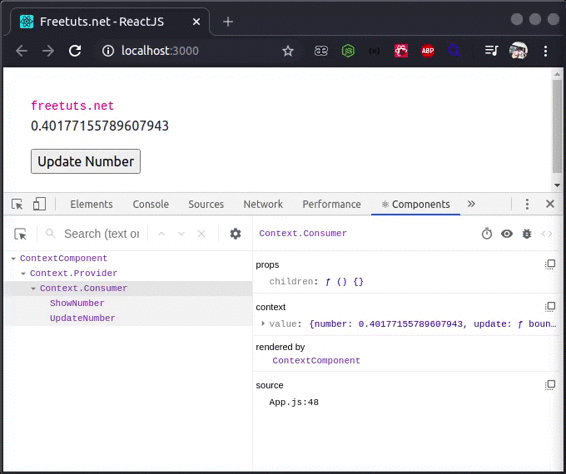
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | class UpdateNumber extends React.Component {    render() {      return (        <button onClick={() => {          //Gọi hàm update để thực hiện update số.          console.log(this.context.update())        }}>Update Number</button>      );    }  }  UpdateNumber.contextType = NumberContext;    class ShowNumber extends React.Component {    render() {      //Hiển thị ra số.      return (        <p>{this.context.number}</p>      );    }  }  ShowNumber.contextType = NumberContext; |

Ở đây mình sử dụng phương thức contextType để gán giá trị của context, lúc này thuộc tính this.context trong class sẽ chứa giá trị của context.

Cuối cùng, chúng ta sẽ có file đầy đủ:

|  |  |
| --- | --- |
| 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 | import React from "react";  //Khởi tạo môt  const NumberContext = React.createContext();    class UpdateNumber extends React.Component {    render() {      return (        <button onClick={() => {          console.log(this.context.update())        }}>Update Number</button>      );    }  }  UpdateNumber.contextType = NumberContext;    class ShowNumber extends React.Component {    render() {      return (        <p>{this.context.number}</p>      );    }  }  ShowNumber.contextType = NumberContext;      export default class ContextComponent extends React.Component {    constructor(props) {      super(props);      this.state = {        number: 0,      };    }    updateNumber = () => {      this.setState({        number: Math.random(),      });    };    render() {      return (        <NumberContext.Provider          value={{            number: this.state.number,            update: this.updateNumber.bind(this),          }}        >          <NumberContext.Consumer>              { () =>  (                <>                  <ShowNumber />                  <UpdateNumber />                </>              )}          </NumberContext.Consumer>        </NumberContext.Provider>      );    }  } |

khởi chạy ứng dụng và xem kết quả:



Trên đây chúng ta đã cùng nhau đi tìm hiểu về Context trong ReactJS. 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.