**Tìm hiểu Component Life Cycle trong ReactJS**

Trong bài viết này chúng ta sẽ cùng nhau đi **tìm hiểu về component life cycle trong React**, đây là một phần quan trọng trong React, nó cho phép bạn hiểu được cách mà một component họat động ra sao ?

**Mục lục**

* [Component Life Cycle là gì ?](https://freetuts.net/component-life-cycle-trong-reactjs-2387.html#goto-h2-0)
* [Component Life Cycle](https://freetuts.net/component-life-cycle-trong-reactjs-2387.html#goto-h2-1)
  + [Initialization](https://freetuts.net/component-life-cycle-trong-reactjs-2387.html#goto-h3-0)
  + [Mounting](https://freetuts.net/component-life-cycle-trong-reactjs-2387.html#goto-h3-1)
  + [Updating](https://freetuts.net/component-life-cycle-trong-reactjs-2387.html#goto-h3-2)
  + [Unmounting](https://freetuts.net/component-life-cycle-trong-reactjs-2387.html#goto-h3-3)

**Component Life Cycle là gì ?**

Chúng ta có thể thấy được mọi thứ trong thế giới đều hoạt động theo một chu kì (ví dụ như con người và cây cối). Cây được mọc lên, sẽ phát triển rồi đến một khoảng thời gian nào đó là sẽ chết đi. Trong React Component cũng vậy, một chu kì cũng xuất hiện, components được khởi tạo (hiển thị ra DOM), update, và kết thúc (unmount),..đó được gọi là một component life cycle.

React cho phép chúng ta tham gia vào các giai đoạn của mỗi component bằng cách sử dụng các phương thức được xây dựng sẵn trong mỗi giai đoạn đó. Khi một components được khởi chạy nó sẽ phải trải qua 4 giai đoạn chính:

* initialization
* mounting
* updating
* unmounting

Phần tiếp theo chúng ta sẽ đi vào cách mà bạn có thể tham gia vào giai đoạn trong một components.

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

**Component Life Cycle**

Chúng ta sẽ tìm hiểu về các lifcecycle methods có trong mỗi giai đoạn.

**Initialization**

Đây là giai đoạn mà thành phần sẽ bắt đầu hành trình của mình bằng cách *khởi tạo state và props*. Điều này thường được thực hiện bên trong phương thức constructor. Ở đây mình có ví dụ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | class App extends React.Component {    constructor(props) {      super(props);      this.state = {        website: 'Học ReactJS cùng Freetuts.net'      };    }  } |

Ở giai đoạn này, React Component sẽ tiến hành khởi tạo các state, props hay các câu lệnh được khởi tạo trong constructor(),...

**Mounting**

Giai đoạn này được thực hiện sau khi quá trình initialization(khởi tạo) được hoàn thành. Nó thực hiện nhiệm vụ chuyển **virtual DOM (DOM ảo)** trong React thành **DOM** và hiển thị trên trình duyệt. Component sẽ được render lần đầu tiên, ở đây chúng ta có 3 phương thức để tham gia vào giai đoạn này.

**componentWillMount()**

Được khởi chạy khi một component chuẩn bị được mount (tức là trước khi thực hiện render), sau khi thực hiện xong componentWillMount() thì component mới có thể được mount.

**Lưu ý:** Chúng ta **không** nên thực hiện bất cứ thay đổi nào liên quan đến state, props hay call API ở trong hàm này, bởi thời gian chuẩn bị mount -> mount rất ngắn nên các tác vụ đó không thể hoàn thành kịp. Khiến cho component render ra kết quả không như bạn mong muốn.

**componentDidMount()**

Được gọi khi component đã được mount (render thành công ), quá trình này xảy ra sau khi componentWillMount() thực hiện xong. Trong phương thức này bạn có thể gọi API, thay đổi *state, props*.

Ở đây mình có ví dụ về 2 phương thức mà mình vừa để cập :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | class LifeCycle extends React.Component {    componentWillMount() {        console.log('Component will mount!')     }    componentDidMount() {        console.log('Component did mount!')        this.getList();     }    getList=()=>{     /\*\*\* Gọi API, update state,vv...\*\*\*/    }    render() {        return (           <div>              <h3>Mouting Method</h3>           </div>        );     }  } |

**Updating**

Đây là giai đoạn thứ ba mà các component phải thực hiện, sau giai đoạn *initialization (khởi tạo )* , *mount (render lần đầu)*,... . Trong giai đoạn này, dữ liệu của các phần (props & state) sẽ được cập nhật để đáp ứng với các sự kiện của người dùng như click, gõ, v.v. Điều này dẫn đến việc re-render lại component, ở trong giai đoạn này chúng ta sẽ có 4 phương thức chính:

**shouldComponentUpdate()**

Phương thức này xác định xem component có nên được render lại hay không ? Theo mặc định, nó trả về true. Nhưng bạn có thể thay đổi giá trị trả về của nó theo từng trường hợp.

Nó sẽ nhận về 2 tham số truyền vào là nextState và nextProps.

**componentWillUpdate()**

Phương thức này được gọi trước khi tiến hành re-render, bạn có thể thực hiện các hành động như update state, props,...trong phương thức này trước khi tiến hành re-render. Giống như shouldComponentUpdate(), componentWillUpdate() sẽ nhận vào 2 tham số đó là nextState và nextProps

**ComponentDidUpdate()**

Phương thức này được gọi khi component đã re-render xong. Chúng ta có ví dụ về cả 3 phương thức về đề cập ở trên.

|  |  |
| --- | --- |
| 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 | class LifeCycle extends React.Component {    constructor(props)    {      super(props);       this.state = {         date : new Date(),         clickedStatus: false,         list:[]       };    }    componentWillMount() {        console.log('Component will mount!')     }    componentDidMount() {        console.log('Component did mount!')        this.getList();     }    getList=()=>{     /\*\*\* method to make api call\*\*\*/     fetch('[https://api.mydomain.com](https://api.mydomain.com/)')        .then(response => response.json())        .then(data => this.setState({ list:data }));    }     shouldComponentUpdate(nextProps, nextState){       return this.state.list!==nextState.list      }     componentWillUpdate(nextProps, nextState) {        console.log('Component will update!');     }     componentDidUpdate(prevProps, prevState) {        console.log('Component did update!')     }    render() {        return (           <div>              <h3>Mounting Lifecycle Methods</h3>           </div>        );     }  } |

**Unmounting**

Đây là bước cuối cùng trong mỗi component, khi tất cả các tác vụ hoàn thành và bạn tiến hành unmount DOM. Quá trình này chỉ có duy nhất 1 phương thức đó là componentWillUnmount() :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | class App extends React.Component {    constructor(props) {      super(props);      this.state = {        website: 'Học ReactJS'      };    }    componentWillUnmount() {      console.log('component will unmount')    }    render() {      return (        <div>          <p>Component LifeCycle</p>        </div>      );    }  } |

Đây là bước cuối cùng và sẽ kết thúc một vòng đời của components.

Trên đây chúng ta đã cùng nhau đi tìm hiểu về Component Life Cycle 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.