React Lecture Notes

React là gì và dùng để làm gì?

React là một thư viện của ngôn ngữ Javascript do Facebook phát triển. Giúp lập trình viên tạo ra những giao diện cho website một cách đơn giản, nhanh chóng. Ngoài ra với React, lập trình viên có thể tạo ra những trang web có cơ chế update dữ liệu thời gian thực, tức người dùng không cần phải nhấn F5 để tải lại trang khi muốn nhận dữ liệu mới, tính năng này gọi là **real-time**. Bên cạnh đó, React còn cho phép lập trình viên làm ra những trang web mà chỉ có 1 trang duy nhất, và dùng Javascript để render thông tin của những trang khác dựa trên sự thay đổi của URL. Những website mà chỉ có 1 trang hiển thị duy nhất gọi là **Single Page Application** hay viết tắt là **SPA**.

Một tẹo lý thuyết

- React Component **là một class** có thừa kế **Component class** và có hàm **render()** giúp render ra những HTML element hoặc những component khác để hiển thị giao diện cho người dùng
- React Component có cú pháp sử dụng như một HTML element bình thường

```
<Div></Div>
```

- Để phân biệt với React Component và các HTML Element bình thường, thì các React Component đều phải viết hoa chữ cái đầu của tên Component

```
<div></div> // HTML
<Div></Div> // React
```

- Khi sử dụng React Component, hàm render sẽ được khởi chạy để render giao diện
- Để giúp một React Component có thêm thông tin từ bên ngoài. Thì hãy sử dụng **props** để truyền những thông tin đó. Khi này, bên trong một React Component, sẽ có thể truy cập những **props** thông qua field **props** kết hợp với biến **this**. -> **this.props**

```
class Greeting extends Component {
  render() {
    const { name } = this.props;
    return (
      <div>Hello {name}</div>
    )
  }
}
```

- Props của một component có thể có mọi kiểu. Từ kiểu Boolean, Number, String, Object hay thậm chí cả kiểu Function

- Để sử dụng một biến của Javascript trong code HTML. Ta dùng cú pháp `{}`

```
class Greeting extends Component {
  render() {
    const { name } = ', DMM';
    return (
      // -> Hello, DMM
      <div>Hello {name}</div>
    )
  }
}
```

- **Khi một props được thay đổi, thì Component sử dụng props đó cũng sẽ được render lại.**

Một tẹo lý thuyết - End Game

- [Spoiler Alert] Thanos đuỵt tung đúyt Iron Man và Captain Amurica. Spider Man yêu Wonder Woman. Black Widow bị les và Captain Marvel thực chất là người da màu
- Để quản lý thông tin bên trong một component và giúp component đó render lại mỗi khi thông tin thay đổi. Ta sẽ dùng một field đặc biệt nữa trong React component đó là **state**
- **Khi một state được thay đổi, thì Component chắc chắn sẽ render lại**
- **Để thay đổi một state và khiến nó render lại, bắt buộc phải thay đổi state thông qua hàm setState**

```
class StateExample extends Component {
  state = {
    name: 'Long',
  }

  changeMyName = () => {
    this.setState({
      name: ['Bảo', 'Hưng'][parseInt(Math.random() * 2)],
    })
  }

  render() {
    const { name } = this.state;
    return (
      <div>Duck you, {name}</div>
      <button onClick={this.changeMyName}>Screw it</button>
    )
  }
}
```

Cú pháp

- Cách tạo một component

```
class MyNameIsComponent extends Component {
  render() {
    return (
      <div>Hi, I am a component</div>
    )
  }
}
```

- Cách lấy props

```
class MyNameIsComponent extends Component {
  render() {
    // const propsName = this.props.propsName;
    // cú pháp ngắn hơn
    const { propsName } = this.props;
    return (
      <div>Hi, I am a component</div>
    )
  }
}
```

- Cách lấy state

```
class MyNameIsComponent extends Component {
  render() {
    // const stateName = this.state.stateName;
    // cú pháp ngắn hơn
    const { stateName } = this.state;
    return (
      <div>Hi, I am a component</div>
    )
  }
}
```

- Cách dùng props / state hay variable bất kỳ trong HTML dùng cú pháp `{variable}`

```
class MyNameIsComponent extends Component {
  render() {
    const { propsName } = this.props;
    const { stateName } = this.state;
    const name = 'Thanos';

    return (
      <div>My name is {name} doggy</div>
      <div>My name is {stateName} kitty</div>
      <div>My name is {propsName} puppy</div>
    )
  }
}
```

- Cách tạo một state

```
class MyNameIsComponent extends Component {
  state = {
    name: 'Hưng',
  }
  render() {
    const { name } = this.state;

    return (
      <div>My name is {name} meo meo</div>
    )
  }
}
```

- Cách tạo một state thông qua constructor

```
class MyNameIsComponent extends Component {
  constructor(props) {
    super(props);

    this.state = {
      name: 'Bảo',
    }
  }
  render() {
    const { name } = this.state;

    return (
      <div>My name is {name} gâu gâu</div>
    )
  }
}
```

- Cách update state

```
class StateExample extends Component {
  state = {
    name: 'Long',
  }

  changeMyName = () => {
    // luôn luôn dùng setState
    this.setState({
      name: ['Bảo', 'Hưng'][parseInt(Math.random() * 2)],
    })
  }

  render() {
    const { name } = this.state;
    return (
      <div>Duck you, {name}</div>
      <button onClick={this.changeMyName}>Screw it</button>
    )
  }
}
```

- Cách truyền props cho một component

```
class MyNameIsComponent extends Component {
  render() {
    const { name } = this.props;

    return (
      <div>My name is {name} gâu gâu</div>
    )
  }
}

class App extends Component {
  render() {
    return (
      <MyNameIsComponent
        name="Long"
      />
    )
  }
}
```

- Cách truyền props cho một component dùng cú pháp `{ }`

```
class MyNameIsComponent extends Component {
  render() {
    const { name } = this.props;

    return (
      <div>My name is {name} gâu gâu</div>
    )
  }
}

class App extends Component {
  render() {
    const name = "Long";
    return (
      <MyNameIsComponent
        name={name}
      />
    )
  }
}
```

- Cách truyền một hàm cho một component dùng cú pháp `{}`

```
class MyNameIsComponent extends Component {
  render() {
    const { sayHello } = this.props;

    return (
      // In ra: Boring
      <div>{sayHello()}</div>
    )
  }
}

class App extends Component {
  sayHello = () => {
    return 'Boring';
  }
  render() {
    const name = "Long";
    return (
      <MyNameIsComponent
        // lưu ý là this.sayHello
        // không phải this.sayHello()
        sayHello={this.sayHello}
      />
    )
  }
}
```

- Cách update state của component cha thông qua component con

```
class ChildComponent extends Component {
  render() {
    const { name, changeMyName } = this.props;
    return {
      <div>Duck you, {name}</div>
      <button onClick={changeMyName}>Screw it</button>
    }
  }
}

class ParentComponent extends Component {
  state = {
    name: 'Long',
  }

  changeMyName = () => {
    this.setState({
      name: ['Bảo', 'Hưng'][parseInt(Math.random() * 2)],
    })
  }

  render() {
    const { name } = this.state;
    return (
      <ChildComponent
        name={name}
        changeMyName={this.changeMyName}
      />
    )
  }
}
```

- Cách render nhiều component dùng Array.map và obj để lưu trữ props

```
class PlayerInfo extends Component {
  render() {
    const { name, age, score } = this.props;

    return (
      <div>Name: {name}</div>
      <div>Age: {age}</div>
      <div>Score: {score}</div>
    )
  }
}

class GameSet extends Component {
  // để ý cú pháp và nơi đặt cái này
  playerList = [
    {
      name: 'Bảo'
      age: 20,
      score: 0,
    },
    {
      name: 'Hưng'
      age: 20,
      score: 0,
    }
  ]

  render() {
    //. hmmm, hơi lạ nghen
    const { playerList } = this;
    const playerInfoComponentList = playerList.map(
      function(playerInfo, index) {
        return (
          <PlayerInfo
            key={index} // key luôn luôn phải có khi xài trong map
            name={playerInfo.name}
            age={playerInfo.age}
            score={playerInfo.score}
          />
        )
      }
    )
    return (
      <div>
        {playerInfo}
      </div>
    )
  }
}
```