ReactJS

Bài 2: Component, Prop - state

React.js Component

- Như đã giới thiệu ở bài React.js là gì?, React chia nhỏ các phần của trang thành từng phần riêng biệt, gọi là component, ví dụ như các phần: header, footer, sidebar, navigation, itemList,...
- Tính chất của component giống như một hàm (function) Javascript, có thể tái sử dụng ở nhiều nơi khác nhau.

Cách viết một component

- Viết dưới dạng function (hoặc Class - nói ở bài sau), và function luôn được return.
- Bên trong return luôn tồn tại tag bao ngoài tất cả (tag wrap).
- Sử dụng thư viện React.DOM để render một component.
- Function render phải có cấu trúc XML, Để hiểu hơn về component ta tạo một file index.html với nội dung sau:

Cách viết một component

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>React Component</title>
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>
    <div id="member"></div>

    <script type="text/babel">
      function Member(){
        return(
          <div className="member">
            <h2>Nguyễn Văn A</h2>
            <p>Tuổi: 25</p>
          </div>
        )
      };

      ReactDOM.render(
        <Member />,
        document.getElementById('member')
      );
    </script>
  </body>
</html>
```

- Function Member() đóng vai trò là một component, nhiệm vụ là tạo một đối tượng member với tên và tuổi.
- ReactDOM.render giữ vai trò render nội dung ra trình duyệt, dựa vào id="member".

Cách tạo và include component

- Giả sử ta có nhiều component khác nhau trong một trang, mỗi component có một nhiệm vụ riêng, Ví dụ:
  - Component avatar dành để xử lý hình ảnh member.
  - Component memberInfo dành để hiển thị thông tin member như tên, tuổi.
  - Component comment dành để quản ý bình luận của member.
  - Nhiệm vụ của ta là làm sao viết kết hợp các component này lại để hiển thị thông tin cho member, ta làm như sau:

Cách tạo và include component

```
<script type="text/babel">
 function Avatar(){
  return( <div className="avatar">
     <img src="img_avatar.jpg" alt="" />
    </div> )
 };

 function MemberInfo(){
  return(  <div className="info">
     <h2>Nguyễn Văn A</h2>
     <p>Tuổi: 25</p>
    </div> )
 };

 function Comment(){
  return(
    <div className="comment">
     Lorem ipsum dolor sit amet, consectetur adipiscing elit proin sit amet neque.
    </div> )
 };

 function Member(){
  return(
    <div className="member">
     <Avatar />
     <MemberInfo />
     <Comment />
    </div> )
 };
 ReactDOM.render(<Member />,  document.getElementById('member') );
</script>
```

- Với ví dụ trên, các bạn sẽ thấy việc tạo component rất đơn giản, chỉ cần tách các phần riêng biệt cho từng chức năng khác nhau, là ta đã có được các component, việc này thuận lợi trong việc sử lý các phần riêng mà không bị ảnh hưởng hay bị phân tâm với các component khác.
- Khi làm thực tế, ta cần tách từng component ra riêng một file Javascript để tiện sử dụng, cách làm như thế nào sẽ được giới thiệu sau nhé.
- Một vấn đề khác: Như đã nói component có thể tái sử dụng được, vậy với nội dung cố định như trên thì làm sao ta có thể tạo thêm được một member khác được?, không lẽ cần phải viết thêm một component mới? rất may là không cần, React có một khái niệm gọi là Props, giúp component truyền tham số dễ dàng cho các đối tượng

React.js Props

- **"props"** là một từ viết ngắn gọn của **"properties"**, nhưng nó là một khái niệm trong **ReactJS**. Về cơ bản **props** là một đối tượng, nó lưu trữ các giá trị của các **attribute** (thuộc tính) của một thẻ (Tag).

- Ta có thể xem props như là một biến giá trị thuộc tính riêng của đối tượng Component mà chúng ta tạo ra

Cách viết một React.js Props

- Thay vì viết cố định tên và tuổi bên trong component Member, thì ta có thể viết như sau:

```
function Member(props){
    return(
    <div className="member">
    <h2>{ props.name }</h2>
    <p>Tuổi: { props.age }</p>
    </div> )
};
```

- Ở component Member ta đã sử dụng props, giống như cách báo cho hệ thống React biết nơi nào xuất hiện props thì nơi đó sẽ được liên kết với các biến (name, age), khi render thì các tham số (Nguyễn Văn A, 30) sẽ được truyền vào các biến này.

Props - arrow function

- Hàm mũi tên (arrow function) đã được giới thiệu trong bài ES6 - arrow function, cấu trúc có những vượt trội hơn so với cách sử dụng cũ, sử dụng tham số cũng linh hoạt hơn.
- Bài học này sẽ giúp các bạn chuyển đổi cách viết một props theo function thông thường sang arrow function.

Cấu trúc function thông thường	Cấu trúc arrow function

function Member(props){
return(
 	
 { props.name }
 Tuổi: { props.age }
 	
)
};
	var Member = (props) => {
	return(
	
	 { props.name }
	 Tuổi: { props.age }
	
	)
	};

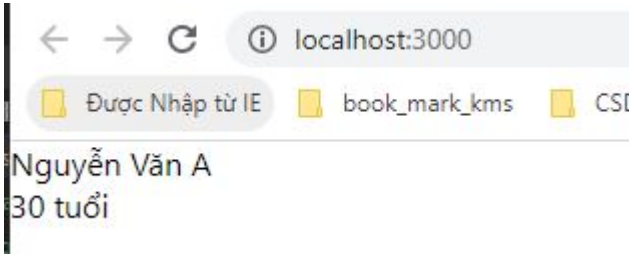
- Nhìn vào so sánh cấu trúc bên trên, ta thấy cấu trúc bên trong function không có gì thay đổi, chỉ thay đổi cách khai báo function.
- Từ giờ về sau, nếu viết một function, ta sẽ sử dụng cách viết arrow function thay cho cách viết cũ.

Cách viết một React.js Props

- Truyền giá trị props vào bên trong Component

```
function App() {
  return (
    <Member name="Nguyễn Văn A" tuoi="30" />
  );
}
```

- Kết quả



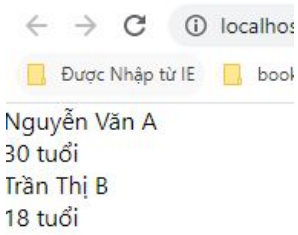
Tái sử dụng Component trong cùng khu vực (danh sách, nhóm, ...)

- Giả sử ta cần render thêm một member với thông tin "Trần Thị B", "18" tuổi, thay vì viết thêm một component mới, thì ta có thể sử dụng lại component đã có như sau:

```
function App() {
  return (
    <div>
      <Member name="Nguyễn Văn A" tuoi="30" />
      <Member name="Trần Thị B" tuoi="18" />
    </div>
  );
}
```

- Với cách khai báo một biến mới memberlist như trên, ta có thể sử dụng nhiều component cùng lúc, việc này sẽ giúp giải quyết nhiều cấu trúc sau này.

- Kết quả:



Tái sử dụng Component ở nhiều nơi khác nhau

- Thay Member Nguyễn Văn A thành một component GiangVien mới:

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
import Member from './member/member';
import GiaoVien from './giaovien/giaovien';

function App() {
  return (
    <div>
      <GiaoVien name="Nguyễn Văn A" tuoi="30" />
      <Member name="Trần Thị B" tuoi="18" />
    </div>
  );
}

export default App;
```

Tái sử dụng Component ở nhiều nơi khác nhau

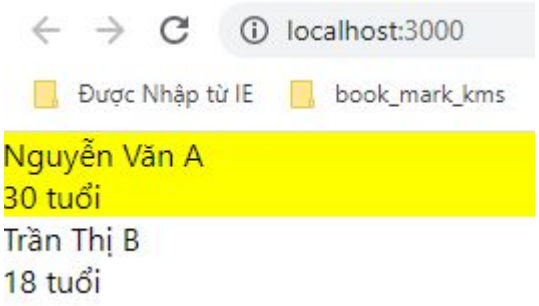
- Và tạo ra Component GiaoVien như sau:

```
class GiaoVien extends Component {

    render() {
        return (
            <div className="giao_vien">
                <Member name="Nguyễn Văn A" tuoi="30" />
            </div>
        );
    }

}
```

- Kết quả:



React.js props xử lý data

- Phần học này sẽ giúp bạn xử lý dữ liệu props truyền vào cho một component
- Thay vì viết dữ liệu trực tiếp bên trong phần render, ta sẽ tách riêng phần dữ liệu ra một dữ liệu của một đối tượng (data Object). Khi này ta cần xử lý lại phần render sao cho có thể lấy được dữ liệu, thao tác rất đơn giản như sau:

```
const info = {
  name: "Nguyễn Văn A",
  tuoi: 30
}

function App() {
  return (
    <div>
      <GiaoVien />
      <Member name={info.name} tuoi={info.tuoi} />
    </div>
  );
}
```

- const info khai báo giá trị cho một object.
- info.name lấy giá trị name của object.
- info.tuoi lấy giá trị tuoi của object.

React.js component xử lý data

- Phần học trước ta đã biết các xử lý data đơn giản cho một component, bài học này sẽ giúp các bạn hình dung được các dữ liệu được truyền giữa các component như thế nào.

- Tạo một data object như sau:

```
const member = {
  text: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit proin sit amet neque.',
  info: {
    path: 'img_avatar.jpg',
    name: 'Nguyễn Văn A',
    age: 25
  }
}
```

React.js component xử lý data

- Lần lượt tiến hành xử lý dữ liệu trong các function như sau:

```
function App() {
  return (
    <div>
      <Member name={member.info.name} tuoi={member.info.tuoi} />
    </div>
  );
}
```

- {props.info.name} props sẽ lấy giá trị path từ info của dữ liệu, tương tự cho tuoi

Xử lý dữ liệu khi các component lồng vào nhau (tái sử dụng component)

- Để hiểu rõ hơn về vấn đề này, Chúng ta sẽ dựa vào Component GiaoVien và Member có sẵn đang được lồng vào nhau để dễ hiểu hơn.

- Ta sẽ thay toàn bộ chỗ chuỗi truyền trực tiếp trong Component GiaoVien thành như sau:

```
class GiaoVien extends Component {

    render() {
        return (
            <div className="giao_vien">
                <Member name={this.props.name} tuoi={this.props.tuoi} />
            </div>
        );
    }

}
```

React.js component xử lý data

- Sau đó bên ngoài chỗ gọi Component GiaoVien sử dụng sẽ thay thành:

```
function App() {
  return (
    <div>
      <GiaoVien name={member.info.name} tuoi={member.info.tuoi} />
      <Member name={info.name} tuoi={info.tuoi} />
    </div>
  );
}
```

- Lúc này ta thấy mọi dữ liệu vẫn chạy đúng và các giá trị được truyền từ Component App bên ngoài vào bên trong Component GiaoVien và xử lý dữ liệu hiển thị vẫn đúng như ban đầu

React.js State

- Như đã đề cập ở bài trước giá trị của Props không thay đổi, chỉ truyền từ component này sang component khác.
- Tuy nhiên trong thực tế, bản thân component đôi lúc cũng cần xử lý dữ liệu riêng của nó, ví dụ thay đổi thông tin dữ liệu như name, ... do đó React đã cho ra đời khái niệm State để giải quyết bài toán về dữ liệu bên trong component.

Cấu trúc chung của State

- Cấu trúc chung của một component sử dụng với State được viết dưới dạng Classes (hoặc arrow function) như bên dưới.

```
class ClassName extends React.Component {
 constructor(props) {
  super(props);
  this.state = {};
 }

 render() {
  return (
    ...
  );
 }
}
```

Cấu trúc chung của State

- ClassName tên Class.
- extends React.Component phương thức kế thừa component (thực ra là kế thừa class đã tồn tại).
- constructor(props) hàm khởi tạo đối tượng cho một class, mỗi class chỉ chứa một hàm khởi tạo duy nhất.
- super(props) gọi lại constructor trong React.Component, khi này ta mới có thể sử dụng phương thức this được.
- this.state phải là một Object.
- render giống như cách dùng của props

Cấu trúc chung của State

- Để nắm vấn đề chúng ta thử làm ví dụ sau:

```
<script type="text/babel">
  class Member extends React.Component {
    constructor(props) {
      super(props);
      this.state = {
        name: "Bùi Văn Tèo",
        age: 25
      };
    };
    render() {
      return (
        <div>
          <h2>{this.state.name}</h2>
          <p>Tuổi: {this.state.age}</p>
        </div>
      );
    };
  };
  ReactDOM.render(<Member />,document.getElementById('member'));
  </script>
```

- Nội dung trên không có gì đặc biệt, chỉ là khai báo State với constructor, gán giá trị ban đầu cho State là this.state, sau đó render nội dung đã được khai báo.

Cấu trúc đơn giản của State

- Chúng ta cũng có thể viết một State đơn giản như sau:

```
class ClassName extends React.Component {
 state = {};

 render() {
  return (
    ...
  );
 }
}
```

- Ví dụ:

```
class Member extends React.Component {
  state = {
    name: "Bùi Văn Tèo",
    age: 25
  };
  render() {
  return (
    <div>
    <h2>{this.state.name}</h2>
    <p>Tuổi: {this.state.age}</p>
    </div>);
  };
};
```

Thay đổi dữ liệu bên trong component với State

- Phần trước ta đã biết cách viết một State là như thế nào, bài này sẽ giúp các bạn cách cập nhật dữ liệu của component bằng các sử dụng setState.
- Muốn thay đổi hay cập nhật nội dung của một State, chúng ta sử dụng this.setState, khi này dữ liệu thay đổi sẽ được cập nhật tới State hiện tại. Để hiểu rõ hơn về this.setState, chúng ta xem ví dụ sau:

```
class Member extends React.Component {
    constructor(props) {
      super(props);
      this.state = {
      name: "Bùi Văn Tèo", age: 25 };
    };
    changeName = () => {
      this.setState({
        name: "Bùi Văn Tý", age: 27
      })
    };
    render() {
      return (
      <div>
      <h2>{this.state.name}</h2>
      <p>Tuổi: {this.state.age}</p>
      <button type="button" onClick={this.changeName}>Change name</button>
      </div> );
    };
};
ReactDOM.render(<Member />, document.getElementById('member'));
```

Thay đổi dữ liệu bên trong component với State

- Kết quả:

- Nội dung trên ta đã dùng this.setState để thay đổi name của State.
- Khi click vào button sẽ gọi function changeName, khi này this.setState sẽ cập nhật lại dữ liệu mới cho State.

Question Time