

**REACT-JS Cơ bản**

# 

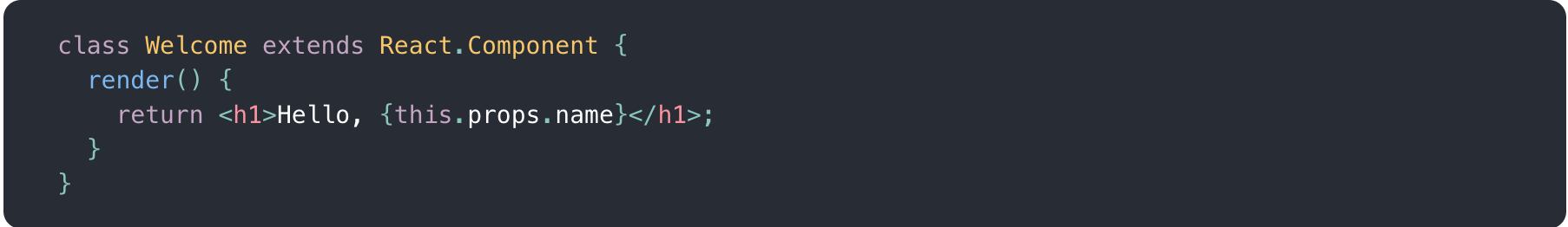
# **------------**

|  |
| --- |
| - Components and Props |
| - State and Lifecycle |
| - Handling Events |

# 

# **Components and Props**

Như đã nói ở trên props là thành phần bên trong một component, vậy thì chúng sẽ được viết như thế nào. Dưới đây là một ví dụ viết một class component:



ở code trên ta thấy class Welcome được thừa kế class React.Component. Đây là cách mà chúng ta sẽ phải tạo ra một component trong ReactJs sau này.

# **Function Components**

Các đơn giản nhất để viết component là tạo function component:



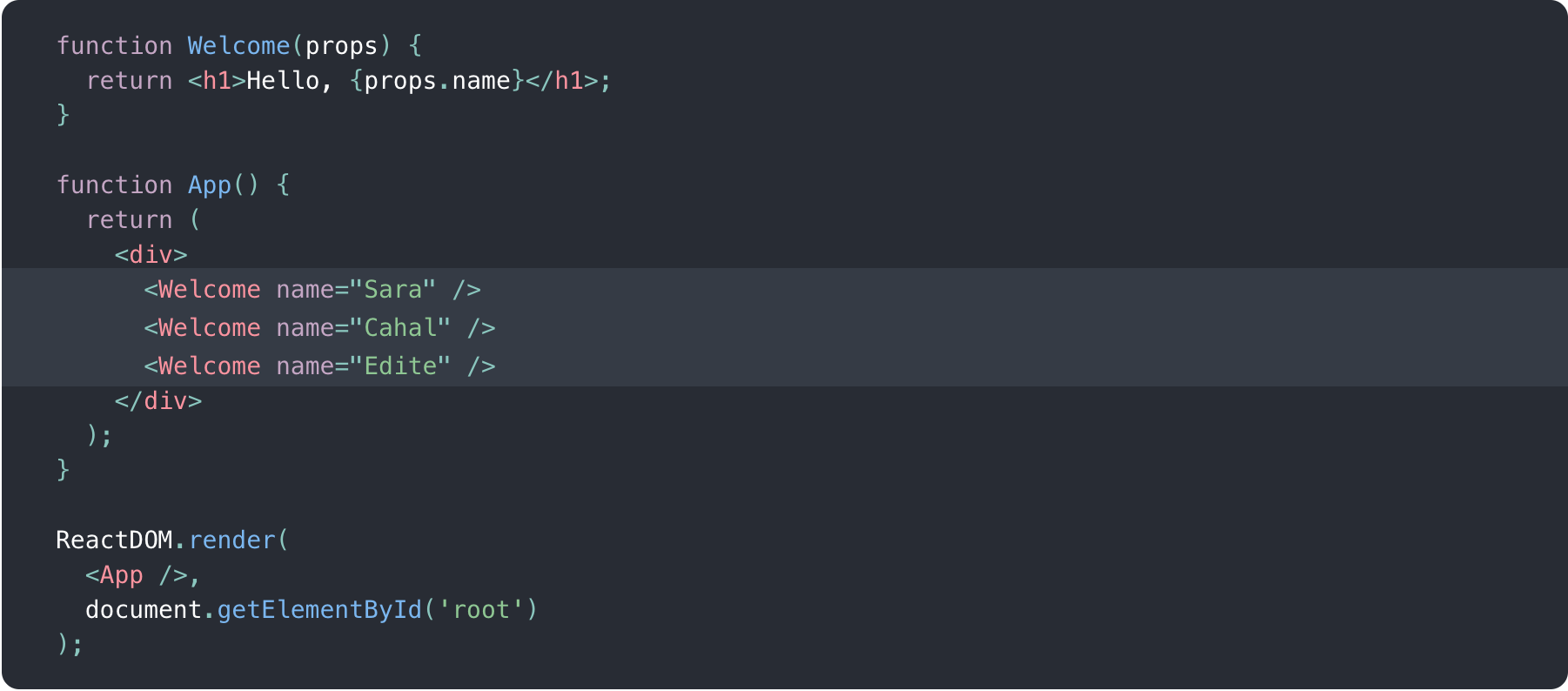
đây là cách đơn giản try nhiên nó sẽ không có được nhiều tính năng như viết một class component.

# **Vòng hoạt động của một component:**

* Mounting
  + [constructor()](https://reactjs.org/docs/react-component.html#constructor)
  + [static getDerivedStateFromProps()](https://reactjs.org/docs/react-component.html#static-getderivedstatefromprops)
  + componentWillMount()
  + [render()](https://reactjs.org/docs/react-component.html#render)
  + [componentDidMount()](https://reactjs.org/docs/react-component.html#componentdidmount)
* Updating
  + [static getDerivedStateFromProps()](https://reactjs.org/docs/react-component.html#static-getderivedstatefromprops)
  + [shouldComponentUpdate()](https://reactjs.org/docs/react-component.html#shouldcomponentupdate)
  + [render()](https://reactjs.org/docs/react-component.html#render)
  + [getSnapshotBeforeUpdate()](https://reactjs.org/docs/react-component.html#getsnapshotbeforeupdate)
  + [componentDidUpdate()](https://reactjs.org/docs/react-component.html#componentdidupdate)
* Unmounting
  + [componentWillUnmount()](https://reactjs.org/docs/react-component.html#componentwillunmount)

## **Composing Components**

Chúng ta có thể viết các component lồng nhau như viết html vậy:

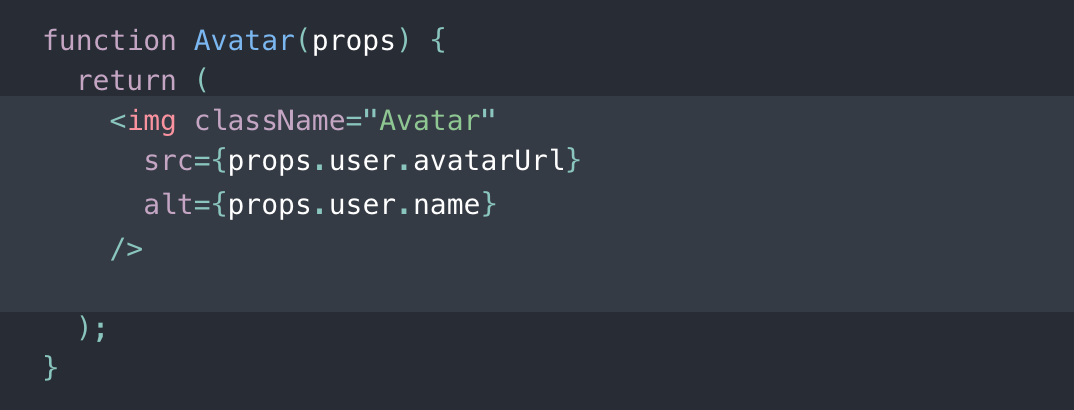


## **Extracting Components**

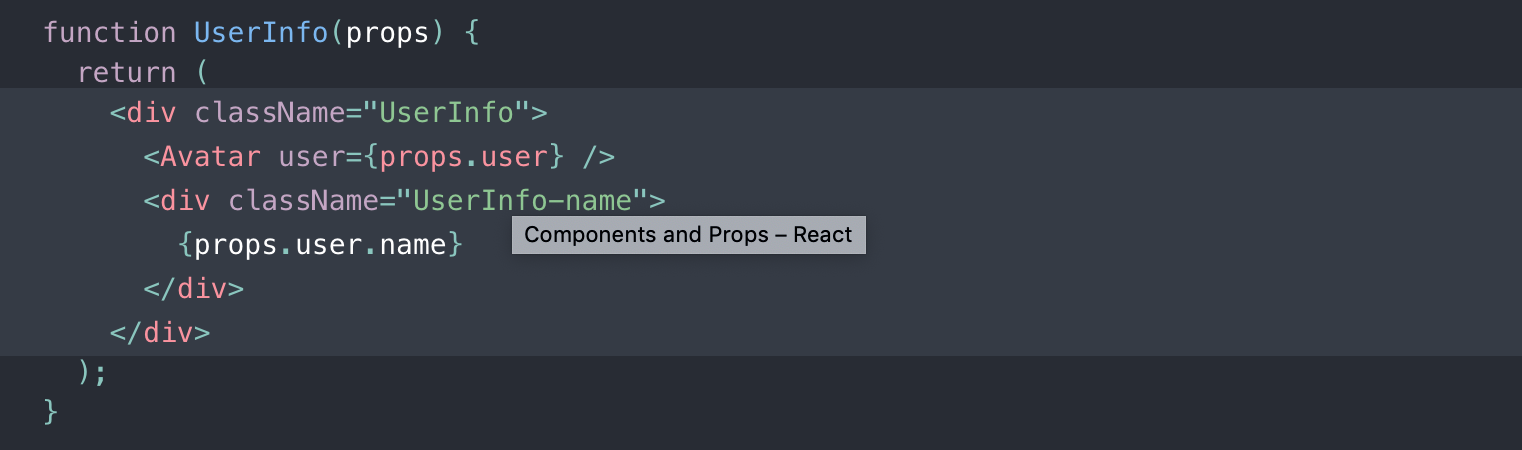
chúng ta có thể tách các thành phần bên trong code dưới đây thành nhiều thành phần component con khác nhau để có thể tái sử dụng chúng ở những đoạn code khác mà không cần phải viết đi viết lại nhiều lần.



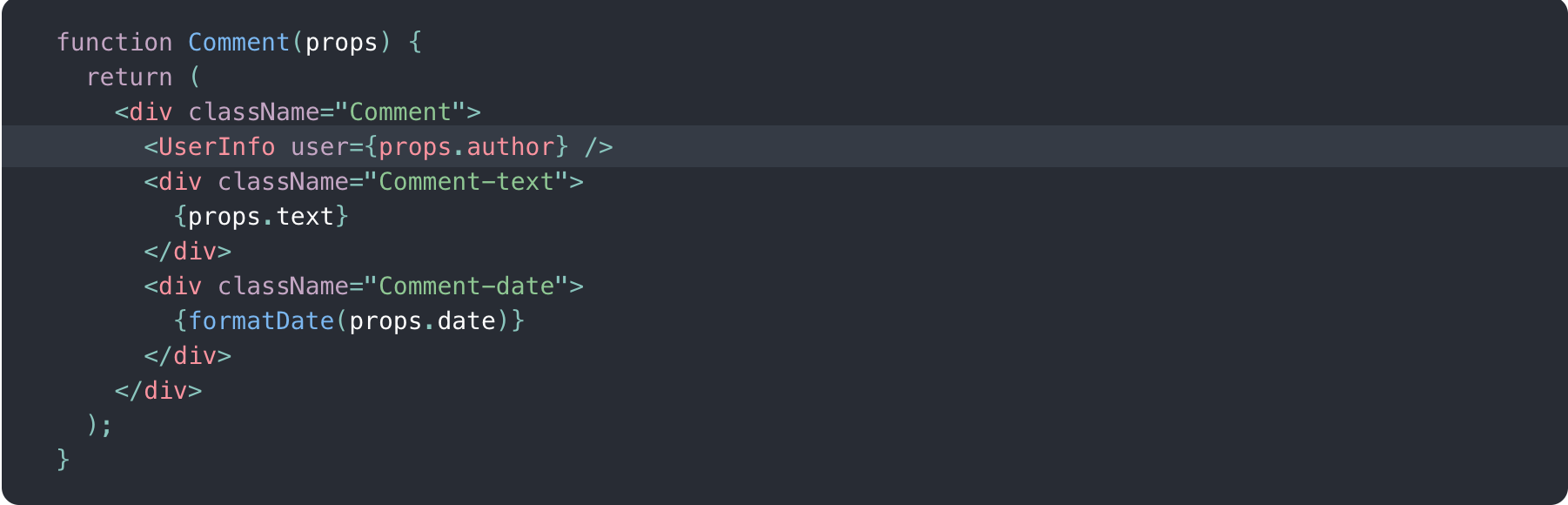
đầu tiên chúng ta tách Avatar



sau đó chúng ta lại tách UserInfo



và giờ thì lồng chúng lại với nhau

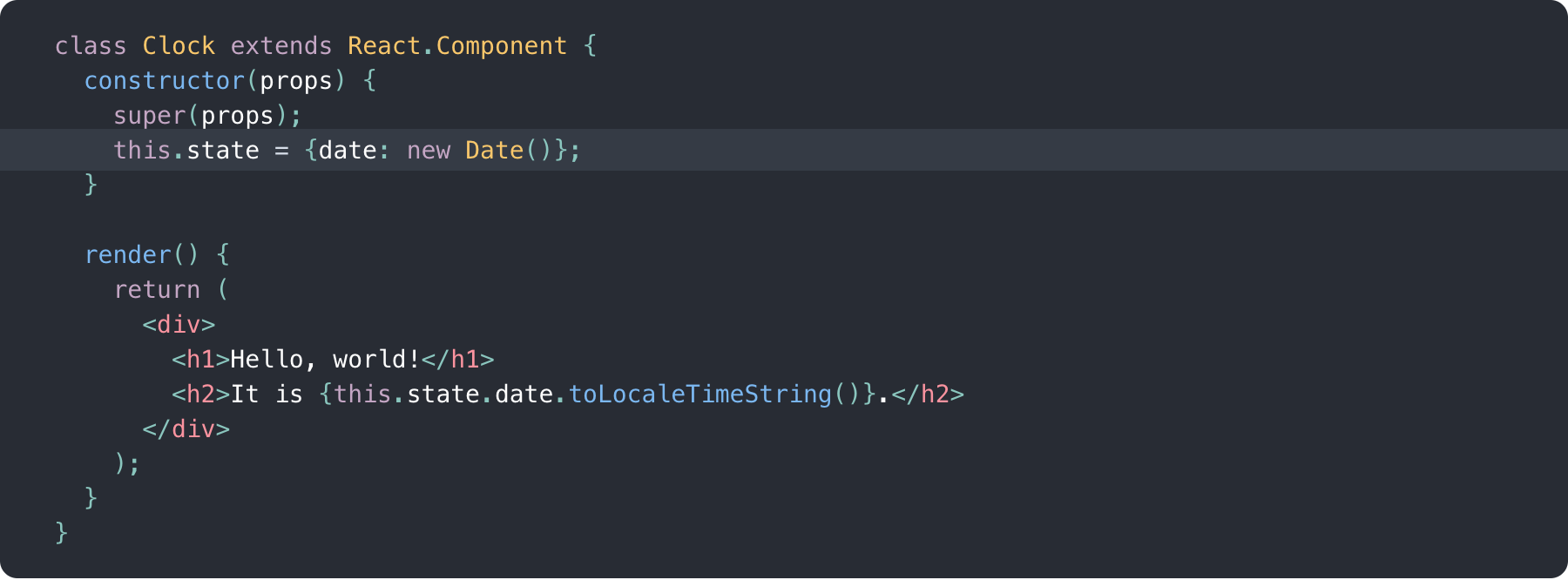


# **State and Lifecycle**

Trước đó chúng ta chỉ có cách tạo function component và thay đổi nội dung hoặc giá trị bên trong bằng cách chạy lại hàm render với giá trị hoặc nội dung mới, ví dụ:

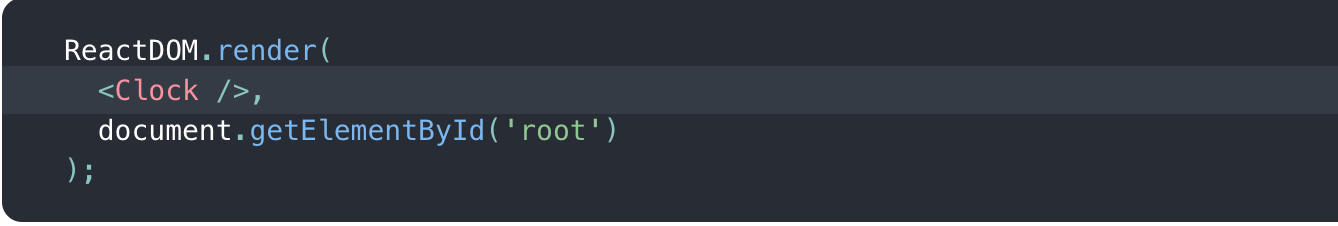


vì cách viết function component này do không có state nên chúng ta khó quản lý sự thay đổi dữ liệu bên trong component. Giờ thì chúng ta hay thử xem cách viết component với class component như sau, ví dụ:



với cách viết nay ta thấy có thêm thuộc tính this.state đây là thuộc tính state của một component, mà mỗi khi state thay đổi giá trị thì component sẽ phải render lại nội dung.

Do đó chúng ta chỉ cần gọi ReactDom.render() một lần thôi:



Code hoàn thiện sẽ như sau:

|  |
| --- |
| class Clock extends React.Component {  constructor(props) {  super(props);  this.state = {date: new Date()};  }    componentDidMount() {  this.timerID = setInterval(  () => this.tick(),  1000  );  }    componentWillUnmount() {  clearInterval(this.timerID);  }    tick() {  this.setState({  date: new Date()  });  }  render() {  return (  <div>  <h1>Hello, world!</h1>  <h2>It is {this.state.date.toLocaleTimeString()}.</h2>  </div>  );  }  } |

# **Sử dụng State đúng cách:**

* Không được set state trực tiếp:

// Wrong

this.state.comment = 'Hello';

* Sử dụng phương thức setState():

// Correct

this.setState({comment: 'Hello'});

State update có thể bất đồng bộ:

Ví dụ sai:

// Wrong

this.setState({

counter: this.state.counter + this.props.increment,

});

Cách đúng:

// Correct

this.setState((state, props) => ({

counter: state.counter + props.increment

}));

Khi set state thì giá trị của state được gán theo kiểu merge. Ví dụ trong component có thể có các giá trị state như sau:

// Correct

this.setState({comment: 'Hello'});

constructor(props) {

super(props);

this.state = {

posts: [],

comments: []

};

this.setState({posts: [“abc”]});

}

sau đó có thể set giá trị cho từng biến trong đó:

componentDidMount() {

fetchPosts().then(response => {

this.setState({

posts: response.posts

});

});

fetchComments().then(response => {

this.setState({

comments: response.comments

});

});

}

Cả component cha và con đều không thể xác định là stateful hay stateless, chúng không cần quan tâm được định nghĩa theo kiểu class hay function.

State của một component không thể truy cập bởi một component khác trừ chính nó, và chỉ được set bởi chính nó.

Một component có thể truyền giá trị từ state xuống component con thông qua props, ví dụ:

<FormattedDate date={this.state.date} />

Ví dụ component sau được lấy data từ props và không cần quan tâm dữ liệu đến từ props do nguồn nào truyền đến:

function FormattedDate(props) {

return <h2>It is {props.date.toLocaleTimeString()}.</h2>;

}

Trong react các dữ liệu data được truyền xuống dưới theo dạng nhánh cây và dữ liệu của mỗi nhánh là độc lập. Để chứng minh điều đó thì chúng ta có thể thử kiểm chứng bằng cách tạo nhiều <Clock> component như sau:

function App() {

return (

<div>

<Clock />

<Clock />

<Clock />

</div>

);

}

ReactDOM.render(

<App />,

document.getElementById('root')

);

# 

# **Handling Events**

Bắt các event (sự kiện) trong React element cũng giống như bắt các event trong DOM HTML.

HTML:

<button onclick="activateLasers()">

Activate Lasers

</button>

Trong React:

<button onClick={activateLasers}>

Activate Lasers

</button>

Một sự khác biệt nữa là không thể return false trong event để dừng event:

<a href="#" onclick="console.log('The link was clicked.'); return false">

Click me

</a>

Phải sử dụng preventDefault() để dừng event trong React:

function ActionLink() {

function handleClick(e) {

e.preventDefault();

console.log('The link was clicked.');

}

return (

<a href="#" onClick={handleClick}>

Click me

</a>

);

}

Ví dụ event trong component:

class Toggle extends React.Component {

constructor(props) {

super(props);

this.state = {isToggleOn: true};

// This binding is necessary to make `this` work in the callback

this.handleClick = this.handleClick.bind(this);

}

handleClick() {

this.setState(state => ({

isToggleOn: !state.isToggleOn

}));

}

render() {

return (

<button onClick={this.handleClick}>

{this.state.isToggleOn ? 'ON' : 'OFF'}

</button>

);

}

}

ReactDOM.render(

<Toggle />,

document.getElementById('root')

);

Truyền tham số vào event:

<button onClick={(e) => this.deleteRow(id, e)}>Delete Row</button>

<button onClick={this.deleteRow.bind(this, id)}>Delete Row</button>

\* *Bài thực hành tại lớp yêu cầu làm một chương trình ToDo list*