**Section 1: Getting Started**

****

**Section 3: Understanding the Base Features & Syntax**

**Using create React App**

* npm i create-react-app –g [create-react-app on Github](https://github.com/facebook/create-react-app)
* create-react-app react-complete-guide --scripts-version 1.1.5

**Understanding the Folder Structures**

* **public folder:** root folder sẽ được served bởi web server in the end. Các js file được inject vào file **index.html** nằm trong folder này. Chúng ta sẽ không thêm bất cứ code HTML nào vào đây.
* <body>
* <noscript>
* You need to enable JavaScript to run this app.
* </noscript>
* <div id="root"></div>
* <!--
* This HTML file is a template.
* If you open it directly in the browser, you will see an empty page.
* You can add webfonts, meta tags, or analytics to this file.
* The build step will place the bundled scripts into the <body> tag.
* To begin the development, run `npm start` or `yarn start`.
* To create a production bundle, use `npm run build` or `yarn build`.
* -->
* </body>
* Chúng ta sẽ mount app vào div có id là root kia.
* **public\manifest.json:** nơi define some metadata cho ứng dụng

**src\index.js:** rất quen thuộc, chúng ta sẽ inject component **App** vào thẻ div có id là **root** ở file **index.html** nằm trong thư mục public

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import registerServiceWorker from './registerServiceWorker';

ReactDOM.render(<App />, document.getElementById('root'));

registerServiceWorker();

**src\App.js**

import React, { Component } from 'react';

import logo from './logo.svg';

import './App.css';

class App extends Component {

  render() {

    return (

      <div className="App">

        <h1>Hello</h1>

      </div>

    );

  }

}

export default App;

**Understanding Component Basics**

* React is all about creating **components**
  + Ở mỗi component, chúng ta luôn luôn phả return ra một HTML template để render ra screen
  + Các component sẽ kế thừa **Components** từ library của React
  + Đôi khi chúng ta sẽ thấy các file component sẽ có extension là .jsx, do method **render()** sẽ return ra code nhìn như HTMl nhưng **không phải HTML.** Đây là một syntactical sugar, là JS nhưng looking a bit different. Đoạn code ở bên trong return function trong render method sẽ được transpile thành valid javascript

**src\App.js**

**class** App **extends** Component {

  render() {

    return (

      <div className="App">

        <h1>Hello Xuân Hiếu</h1>

      </div>

    );

  }

}

**Understanding JSX**

* Chúng ta sẽ thử thay thế đoạn code bên trong render method của component để có thể render ra kết quản tương tự
* Đây là lý do vì sao chúng ta import **React** tuy nhiên không hề sử dụng nó như cách chúng ta làm ban đầu do module này sẽ được dùng khi code chúng ta được compile
* Chúng ta sẽ không sử dụng cách render này nếu như template của quá phức tạp, chỉ biết để hiểu bản chất.

**src\App.js**

**class** App **extends** Component {

  render() {

*// return (*

*//   <div className="App">*

*//     <h1>Hello Xuân Hiếu</h1>*

*//   </div>*

*// );*

    return React.createElement('div', null,

      React.createElement('h1', {

        className: 'App'

      }, 'Hello Xuân Hiếu'));

  }

}

* Các restrictions của JSX:
  + Không thể sử dụng class attribute để khai báo class cho element như bình thường, thay vào đó chúng ta sử dụng **className**
  + Chỉ được có duy nhất **một** root element.

**Creating a Functional Component**

* Đầu tiên chúng ta sẽ tạo ra một component

**src\Person\Person.js**

import React, { Component } from 'react';

**const** person = () **=>** {

*// đây là jsx syntax, chúng ta cần import React module để sử dụng*

  return <p>I'm a person</p>

}

export default person;

* Sử dụng component này trong **App** component như sau:

**src\App.js**

import React, { Component } from 'react';

import './App.css';

import Person from './Person/Person';

**class** App **extends** Component {

  render() {

    return (

      <div className="App">

        <h1>Hello Xuân Hiếu</h1>

        <Person />

        <Person></Person>

      </div>

    );

  }

}

export default App;

**kết quả**



**Working with components and re-use them**

**-** Sử dụng components giúp chúng ta focus vào một file và dễ dàng hơn trong việc maintain

- Không cần đưa tất cả mọi thứ vào **app.js** file

- Có thể dễ dàng re-use và configurable

* Dynamic content trong React components:
  + Chúng ta có thể sử dụng { } để bọc những đoạn code js của chúng ta trong JSX, khi đó React app sẽ hiểu đây không phải là text mà là những đoạn code JS.
  + Đây là có thể là những single line of code hoặc những function call

**src\Person\Person.js**

**const** person = () **=>** {

*// đây là jsx syntax, chúng ta cần import React module để sử dụng*

  return <p>I'm a Xuân Hiếu and I'm {(new Date()).getFullYear() - 1994} years old</p>

}

export default person;

* Working with Props

**src\Person\Person.js**

import React, { Component } from 'react';

**const** person = (props) **=>** { *// tên param ở đây là up to you, tuy nhiên nên để là props, đây chính là object có các attribute tương ứng là name của các attribute chúng ta truyền vào lúc gọi components*

*// đây là jsx syntax, chúng ta cần import React module để sử dụng*

  return <p>I'm a {props.name} and I'm {props.age} years old</p>

*// đối với class-based components, chúng ta sẽ dùng this.props trong JSX code để get value của các attribute này.*

}

export default person;

**src\App.js**

**class** App **extends** Component {

  render() {

    return (

      <div className="App">

        <h1>Hello Xuân Hiếu</h1>

        <Person name="Xuân Hiếu" age="26"/>

        <Person name="Max" age="30">My hobbies: Racing</Person>

        <Person name="Quân" age="31"></Person>

      </div>

    );

  }

}



* **Children** props:
  + Chúng ta có thể sử dụng props.children để refer đến các elements nằm giữa opening và closing tag của component

**src\Person\Person.js**

**const** person = (props) **=>** { *// tên param ở đây là up to you, tuy nhiên nên để là props, đây chính là object có các attribute tương ứng là name của các attribute chúng ta truyền vào lúc gọi components*

*// đây là jsx syntax, chúng ta cần import React module để sử dụng*

*// props.children ở đây sẽ là bất cứ elements nào nằm giữa opening và closing tag, không chỉ là text mà có thể là các complex HTML*

  return (

    <div>

      <p>I'm a {props.name} and I'm {props.age} years old</p>

      <p>{props.children}</p>

    </div>

  )

*// đối với class-based components, chúng ta sẽ dùng this.props trong JSX code để get value của các attribute này.*

}

**Understanding & Using State**

* Đối với các component được kế thừa từ **Component,** chúng ta có một property là **state,** là nơi mà props vdc set và passed vào từ outside.
* Hiện giờ chúng ta đang hardcode các giá trị như **name** hay **age,** tuy nhiên sử dụng state chúng ta có thể thay đổi các giá trị này.
* Khi sử dụng property **state,** để gọi property này chúng ta phải dùng **this** keyword (ES6 syntax - **class**).
* Khi **state** thay đổi, React sẽ re-render DOM để update theo giá trị mới này của **state.**

**src\App.js**

**class** App **extends** Component {

  state = {

    persons: [

      { name: 'Xuân Hiếu', age: 26 },

      { name: 'Max', age: 30 },

      { name: 'Quân', age: 31 },

    ]

  }

  render() {

    return (

      <div className="App">

        <h1>Hello Xuân Hiếu</h1>

        <button>Switch Name</button>

        <Person name={this.state.persons[0].name} age={this.state.persons[0].age} />

        <Person name={this.state.persons[1].name} age={this.state.persons[1].age} >My hobbies: Racing</Person>

        <Person name={this.state.persons[2].name} age={this.state.persons[2].age} ></Person>

      </div>

    );

  }

}

