ReactJS

Next gen JavaScript

1. Let & conts
2. Arrow functions
3. Export and import(modules)
4. Classes
5. Classes, properties and methods
6. Spread & Rest Operators
7. Destructuring

Before working with ReactJS, learner need to understand the above new concepts of JavaScript.

Introduction

Library to create User Interfaces

As you can look at the modern websites lots of sections exist, which also be named as components, if you leaned Angualrjs.
So react takes this components concept and build you website and splitting in to Multiple components an allow to build such components using JavaScript.

Difference
Reactjs is using to build UI views.
Angular2 is more than that, it's a framework which allow you to build single page application. It has router supports form validation and so on. Its use to controls user experience.

Environment setup

-> create-react-app

Easy reactjs environment setup

First we need to install the latest version of node.js, even though you don't know to use the node.js install it.

Fter that we open the terminal install the reactjs setup

> **Teminal**
sudo npm install -g create-react-app

> **cmd**
npm install -g create-react-app

The above command line will install create-react-app globally in your pc.
It was create by FB to make setup installation easy.

Next all you need to do is in the terminal go to the desktop.

> **Terminal**
create-react-app react_basic

It will automatically install all the files required for react app development.

https://reactjs.org/docs/add-react-to-a-new-app.html

Folder structure

node_module folder consist of all dependancy injectors.

Public folder
|__ index.html

src folder
|__ index.js(where we write js)
|__ app.js (body of app)

> Understand component [9:58]

Most of our react code comprises of components

-> Search bar components
-> Sign-up component
->To-do list Component
-->List item component
-->Add new item (form)component

App.js
First we need to import our React from react

```
import React, { Component } from 'react';
```

Now creating component
Its' really simple just clear a javascript class with a name camilcase [Header] and we need to extend to component.

```
class App extends Component {

}
```

Inside the class ver important method we need to write that is render method.
Render a React element into the DOM in the supplied container and return a reference to the component

```
class App extends Component {
  render() {

  }
}
```

This render method need to return something and it need to return, in the parentheses add a div and inside a h1 tag with a hello world text.

```
class App extends Component {
  render() {
    return(
      <div>
        <h1>Hello world</h1>
      </div>
    );
  }
}
```

? ? we are writing html in javascript ? ? its actually not HTML its JSX as the creator of react named it. It's Javascript mixed with XML.

After creating your element we need render that inside our html using the below method.

Public > index.html

```
<div id="root"></div>
```

Src > App.js

```
export default App;
```

Src > index.js

```
import App from './App';

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

Inside the JSX we can write attribute class like in html since its JSX class is reserved word in javascript we need to use camel case className.

Reference link
https://www.youtube.com/watch?v=G40iHC-h0c0&index=4&list=PL55RiY5tL51oyA8euSROLjMFZbXaV7skS

> Multiple components

We can combine components together to create an app.

Lets create a Header component in side the components folder [Header.js]

```
import React, {Component} from 'react';

class Header extends Component{
    render(){
      return(
        <nav className="navbar navbar-default">
          <div className="container">
            <div className="navbar-header">
              <ul classname="nav navbar-nav">
                <li><a href="#">Home</a></li>
              </ul>
            </div>
          </div>
        </nav>
      );
    }
}

export default Header;
```

Now go to the Home.js in components folder

```
import React, {Component} from 'react';


class Home extends Component {
  render() {
    return(
      <div>
        <p>In a new Component.</p>
      </div>
    );
  }
}

export default Home;
```

Now we got our both components, we need to import both to out App.js and use those in xml tag syntax.

```
import React, { Component } from 'react';

import Header from './components/Header';
import Home from './components/Home';


class App extends Component {
  render() {
    return(
      <div className="container">
        <div className="row">
          <div className="col-12">
            <Header/>
          </div>
        </div>
        <div className="row">
          <div className="col-12">
            <Home/>
          </div>
        </div>
      </div>
    );
  }
}

export default App;
```

> Outputting the Dynamic content

We can output dynamic content in your template in your render method using curly braces { }.

{ 2 + 2 }
{Math.floor(Math.random()*30)} // this will generate random number
{"Hello"} // we can also write strings

We can write only one line expressions only not javascript classes or function. if you want to write any conditions we can use ternary operations

{ 5 == 2 ? "Yes" : "No"}

Reference link
https://www.youtube.com/watch?v=1JZEmYwRGoU&list=PL55RiY5tL51oyA8euSROLjMFZbXaV7skS&index=6

> Working with props.

props are nothing but the attributes you write in your components tags.

we can define strings, arrays and object in side those attributes.

render(){

```
const user ={
  name:"Stephani",
  hobbies:["Sports", "Reading"]
}
```

return(

```
<div className="col-12">
  <Home name="Max" age="29" user={user}/>
</div>
```

)

}

Inside Home.js

console.log(props);

Not only just console log it we can also output it.

```
<p>Your name is {this.props.name} and your age is{this.props.age} </p>
<p>User Object => Name:{this.props.user.name}</p>
<div>
  <h4>Hobbies</h4>
  <ul>
    {this.props.user.hobbies.map((hobby)=> <li>{hobby}</li>)}
  </ul>
</div>
```

We can also validate our props valuse by using PropTypes

Home.js

```
Home.PropTypes={
  name:React.PropTypes.string,
  age:React.PropTypes.number,
  user:React.PropTypes.object
}
```

We can also pass some text or html with out props like below

App.js

```
<Home name="Max" age="29" user={user}>
  <p>This is a paragraph</p>
</Home>
```

Home.js

```
<div>
  {this.props.children}
</div>
```

Children is reserved word which means whatever content passed between opening, closing tag.

Reference link
https://www.youtube.com/watch?v=GIU8ekYndKw&list=PL55RiY5tL51oyA8euSROLjMFZbXaV7skS&index=7

> Handling events and functions

At first we need to create button to trigger our event in Home.js.

<button className="btn btn-primary">Make me older!</button>

Now we can create a mthod in the Home.js outside the render.

onMakeOlder(){

}

how change the age first wen need to store the age in our Home class.

```
constructor(props){
  super();
  this.age = parseInt(props.age);
}
```

Now we can increase age by writing the blow syntax

```
onMakeOlder(){

  this.age +=3;

}
```

Since we stored the age in the constructor we don't need to use this.props.age in the p tag, instead we can write this.age because we can change the value onclick.

Now, we created out button, event and where to output our increased value. all we need to is use event handler attribute in the button tag.

```
<button onClick={this.onMakeOlder.bind(this)} className="btn btn-primary">Make age change</button>
```

Now if you click age will get update but it's not gone show in the view instead lets check in console.

```
onMakeOlder(){
  this.age +=3;
  console.log(this.age);
}
```

Why it is not changing because state is not changing. Let see in the next slide what is state ?

Reference link
https://www.youtube.com/watch?v=OcM__8q6p4c&index=8&list=PL55RiY5tL51oyA8euSROLjMFZbXaV7skS

> Understanding the state

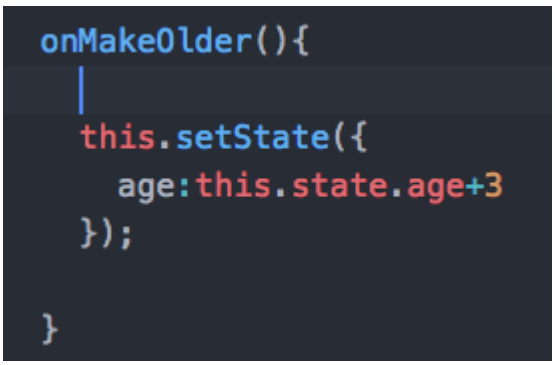
State is an object that determines how the component renders and behave. It is place where the data comes from.

Now we are going to update out syntax in Home.js

this.age = props.age;

```
constructor(props){
  super();
  this.state = {
    age:parseInt(props.age),
  };
}
```

After updating syntax we need to call a setState method in the onMakeOlder method to change the state of or data.



In my out put we will writing {this.state.age}.

```
<p>Your name is {this.props.name} and your age is {this.state.age} </p>
```

Reference link
https://www.youtube.com/watch?v=e5n9j9n83OM&t=0s&index=9&list=PL55RiY5tL51oyA8euSROLjMFZbXaV7skS

> Stateless Components

Components not having state ans uses only props is called stateless component.

Why we have this because when you app becomes more complex it becomes hard to mage state and your app will be unpredictable and therefore stateless more predictable.

Header.js

```
const Header =()=>{
    return(
    JSX
);
}
```

Reference link
https://www.youtube.com/watch?v=SEkfzqIgvTo&t=0s&index=11&list=PL55RiY5tL51oyA8euSROLjMFZbXaV7skS

> Communicating Between parent and children component

Now we can communicate from a child to parent component can pass data to the child with props but what's the equivalent for passing parent to child.

App.js
We will create a method onGreet out side render method and pass from parent to child by writing props greet={this.onGreet}

```
onGreet() {
    alert("Hello");
}
```

```
<Home name="Max" age="29" user={user} greet={this.onGreet}>
```

Let go to **Home.js** and add a button and event handler to call that event.

```
<button onClick={this.props.greet}>Greet</button>
```

Reference link
https://www.youtube.com/watch?v=2teBTI0_eRw&index=12&list=PL55RiY5tL51oyA8euSROLjMFZbXaV7skS

> Passing Data between parent and child components

We can also send data to our child from the parent example input field.

App.js

/// Storing our home link in the state "homeLink"

```
constructor(){
  super();
  this.state= {
    homeLink: "Home"
    };
}
```

```
onchangeLinkName(newName){
  this.setState({
    homeLink:newName
  })
}
```

///Output the initial state homelink

```
<Header homeLink={this.state.homeLink}/>
```

<Home changeLink={this.onchnageLinkName.bind(this)}/>

Home.js

Setting the new value for the homeLink in the state

```
constructor(props){
  super();
  this.state = {
    age:parseInt(props.age),
    homeLink:"Chnage Link"
  };
}
```

Creating the event method to change the Home link for header though props

```
 onChangeLink(){
   this.props.changeLink(this.state.homeLink);
 }
```

Event trigger for change for homelink

```
<button className="btn btn-warning" onClick={this.onChangeLink.bind(this)}>Change the link</button>
```

Reference link
https://www.youtube.com/watch?v=5Xew--ycx0o&index=13&list=PL55RiY5tL51oyA8euSROLjMFZbXaV7skS

> Two way data binding

To do the two way data binding we need a text input field on our Home.js.

<input type="text"/>

Lets now state what we intended to do with out two way data binding concept , at first i want to show my initial value later change to what ever value I enter in the input field.

App.js

<Home initialLinkName={this.state.homeLink}/>

Home.js

<input type="text" value={this.props.initialLinkName}/>

```
onHandleChange(event){
  this.setState({
    homeLink:event.target.value
  });
}
```

<input type="text" onChange={(event)=>this.onHandleChange(event)}/>
We have to use the onChange handler to do two way data binding inside it using ternurey syntax

```
<input type="text"
       className="form-control"
       value={this.state.homeLink}
       onChange={(event)=>this.onHandleChange(event)}/>
```

<input type="text" value={this.state.homeLink}/>

Home.js
Getting the initial value from the parent

```
constructor(props){
  super();
  this.state = {
    age:parseInt(props.age),
    homeLink:props.initialLinkName
  };
}
```

Reference link
https://www.youtube.com/watch?v=IK9k9hSuYeA&index=14&list=PL55RiY5tL51oyA8euSROLjMFZbXaV7skS

> Component Lifecycle

First method

ComponentWillMount ---> Immediately before initial rendering
ComponentDidMount ---> Immediately after initial rendering
ComponentWillReceiveProps ---> When Component receives new props
ShouldComponentUpdate ---> Before rendering, after receiving new props or state
ComponentWillUpdate ---> Before rendering, after receiving new props or state
ComponentDidUpdate ---> After component's update are flushed to DOM
ComponentWillUnmount ---> Immediately before removing component from DOM

Lets see all the above lifecycle in action

Home.js

Adding console in the constructor
console.log("Constructor")

outside the constructor

```
componentWillMount(){
console.log("Component will mount");
}
componentDidMount(){
console.log("Component did mount");
}

componentWillReceiveProps(){
console.log("Component will receive props", nextProps);
}

shouldComponentUpdate(nextProps, nextState){
    console.log("should component update ", nextProps, nextState);
return true;
}

componentWillUpdate(nextProps, nextState){
console.log("component will update")
}

componentDidUpdate(previous )
```

Reference link
https://www.youtube.com/watch?v=Oioo0IdoEls&list=PL55RiY5tL51oyA8euSROLjMFZbXaV7skS&index=15

> React Router

To setup react route in your project all you need to do is type below commands in the terminal where your folder is located.

npm install react-router-dom

It will automatically install the packages required for routing. After the installation we need to use router and enclose your entire app content.

App.js

```
import {BrowserRouter as Router, Route} from 'react-router-dom';



return (
    <Router>
        <div>
            <Route exact path="/" component={home}/>
            <Route path="/about" component={about}/>
        </div>
    </Router>
);
```

Nav.js

import {Link} from 'react-router-dom';

Update all the <a> tag to <Link> and href - > to ="/" , to ="/home" , to ="/about"

and import all the pages to required path

Reference link

https://www.youtube.com/watch?v=rALJykvIM-M