Git commands:-

1. git status
2. git add React.docx
3. git commit -m "first\_react\_app creation with component creation"
4. git push

**React**

* React is a front-end library.
* It is developed by Facebook.
* Unlike Angular JS, which is MVC(Model View Controller), it focus on View part only but not on model and controller. For model and controller, we need Node JS , Spring Boot,etc.
* It creates re-useable components.
* React is all about component. You need to think everything as a component.
* The virtual DOM makes react fast.

**Pre-requisites:-**

* **Solid understandings of HTML, CSS, Java Script is required.**

**Features:-**

1. JSX:- JavaScript syntax extension.

**Limitations:-**

It covers only View layer. So, we need to choose another technology for complete development.

**NOTE: It runs on Node, So Node JS is needed to be installed.**

**To install React App globally:**

npm install –g create-react-app –save

Where npm = node package manager

Create-react-app will create your app with all the necessary files needed.

To start React app:

We use npm start

When we create React App, By default it is created with required packages under node\_modules

And following directories to start off with.

These directories namly:

1. node\_modules
2. public: It contains index.html
3. src: It contains components

When we do **npm start**-> it will search ‘start’ in package.json .

start comes under react-scripts package.

"dependencies": {

"react": "^16.8.6",

"react-dom": "^16.8.6",

"react-scripts": "3.0.1"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test",

"eject": "react-scripts eject"

},

As we can see start comes under react-scripts.

**How to make class/Component**

Components look like HTML but it is actually **JSX.**

import React, { Component } from 'react';

class App extends Component {

render() {

return (

<div>

Hello World, Hi suryasnata

</div>

)

}

}

export default App;

**Container**:

It is a Component that can contain one or multiple Components.

When you start the app, internally it will go to index.js

**index.js**

import React from 'react';

import ReactDOM from 'react-dom';

import App from './container/App';

import \* as serviceWorker from './serviceWorker';

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

serviceWorker.unregister();

index.js will get ‘root’ from index.html and will then put <App /> Component which is our container in it.

**index.html**

<div id="root">

**App.js**

import React, { Component } from 'react';

import WelcomeMessage from '../components/WelcomeMessage';

class App extends Component {

render() {

return (

<div>

Hello World, Hi suryasnata

<WelcomeMessage />

</div>

)

}

}

export default App;

Here, App Component(In our case it is container) is using WelcomeMessage Component.

**WelcomeMessage.js**

import React, {Component} from 'react';

class WelcomeMessage extends Component {

render(){

return(

<div>

Hello

</div>

)

}

}

export default WelcomeMessage

**Starting the new Application from the beginning:-**

Every component has its state. State is the dynamic data that we render it in DOM.

This is a keyword that is used to get data from the class. This represent the class.

import React, { Component } from 'react';

class WelcomeMessage extends Component {

state = {

name: "surya",

age: 30

}

render() {

return (

<div>

Hello

<p>

My name is {this.state.name} and I am {this.state.age}

</p>

</div>

)

}

}

export default WelcomeMessage

**DAY 2: Exercise**

import React, { Component } from 'react';

class WelcomeMessage extends Component {

state = {

name: "surya",

age: 30

}

handleClick(e) {

console.log(e.target);

}

handleMouseover(event) {

console.log(event.target, event.pageX, event.pageY);

}

copy = (event) => {

console.log('dnt copy me');

}

render() {

return (

<div>

Hello <p> My name is {this.state.name} and I am {this.state.age}</p>

<button onClick={this.handleClick}>CLick Me</button>

<button onMouseOver={this.handleMouseover}>Hover me</button>

<p onCopy={this.copy}>Copy me</p>

</div>

)

}

}

export default WelcomeMessage

onClick, onMouseOver, onCopy are the predefined events.

onClick :- this event will arise when we will click on the button or tag or HTML elements.

onMouseOver:- This event will arise when we take the mouse on the HTML elements.

onCopy:- This event will arise when we try to copy any elements.

If you want to use `this` keyword under a method then you have to bind it.

There are two ways to bind this keyword to a function.

One of the easiest way is using new javascript syntax.

handleClick = (e) => {

console.log(e.target);

}

#Excercise 2

import React, { Component } from 'react';

class WelcomeMessage extends Component {

state = {

name: "surya",

age: 30

}

handleClick = (e) => {

this.setState({

name: 'Tanuj'

})

console.log(this.state.name);

}

handleMouseover(event) {

console.log(event.target, event.pageX, event.pageY);

}

copy(event) {

console.log('dnt copy me');

}

render() {

return (

<div>

Hello <p> My name is {this.state.name} and I am {this.state.age}</p>

<button onClick={this.handleClick}>CLick Me</button>

<button onMouseOver={this.handleMouseover}>Hover me</button>

<p onCopy={this.copy} > Copy me </p>

</div>

)

}

}

export default WelcomeMessage

If we want to change the state we use:-

this.setState({

name: 'Tanuj'

})

We don’t directly mutate state, because it can lead to unforeseen errors in our application.

this.state.name='Tanuj'

**Form creation:-**

The below is the example of form creation.

import React, { Component } from 'react';

class WelcomeMessage extends Component {

state = {

name: '',

age: 30

}

handleSubmit = (e) => {

e.preventDefault();

this.setState({

name:''

})

console.log("form Submitted")

}

handleChange = (e) => {

this.setState({

name: e.target.value

})

}

render() {

return (

<div>

<h1>My name is {this.state.name}</h1>

<form onSubmit={this.handleSubmit}>

<input type='text' onChange={this.handleChange} />

<button>Submit</button>

</form>

</div>

)

}

}

export default WelcomeMessage

**event.preventDefault()** used to prevent the default value.

handleSubmit = (e) => {

e.preventDefault();

this.setState({

name:''

})

onChange:- It is mainly used in :-

<input type='text' onChange={this.handleChange} />

<textarea type='text' onChange={this.handleChange} />

**:DAY 3:**

**#Lecture 1**

**props:**

When we want to retrive data from a parent component we can use props in other words, when we pass the data from parent component to any of the child components then props is being used.

Syntax:

**{this.props.<data>}**

**Below is the example:**

Here, **<App />** is the root Component and **<Ninjas />** is the child Component that is been used inside the **<App />** Component.

***App.js***

import React, { Component } from 'react';

import WelcomeMessage from '../components/WelcomeMessage';

import Ninjas from '../components/Ninjas';

class App extends Component {

render() {

return (

<div>

**<Ninjas name='Tanuj' age='30' />**

</div>

)

}

}

export default App;

***Ninjas.js***

import React, { Component } from 'react';

class Ninjas extends Component {

render() {

return (

<div>

<div>name: {this.props.name} </div>

<div>age: {this.props.age}</div>

</div>

)

}

}

export default Ninjas

Another way to get values from props:-

import React, { Component } from 'react';

class Ninjas extends Component {

render() {

const { name, age } = this.props;

return (

<div>

<div>name: {name} </div>

<div>age: {age}</div>

</div>

)

}

}

export default Ninjas

Iteration of Lists(1st way):

import React, { Component } from 'react';

class Ninjas extends Component {

render() {

return (

<div>

{

this.props.listOfNinjas.map(ninja => {

return (

<div key={ninja.id} >NAME: {ninja.name}

AGE: {ninja.age}

</div>

)

})

}

</div>

)

}

}

export default Ninjas

Iteration of Lists(2nd way):

import React, { Component } from 'react';

class Ninjas extends Component {

render() {

const ninjas = this.props.listOfNinjas;

return (

<div>

{

ninjas.map(ninja => {

return (

<div key={ninja.id}>

name: {ninja.name}

&nbsp;

age: {ninja.age}

</div>

)

})

}

</div>

)

}

}

export default Ninjas