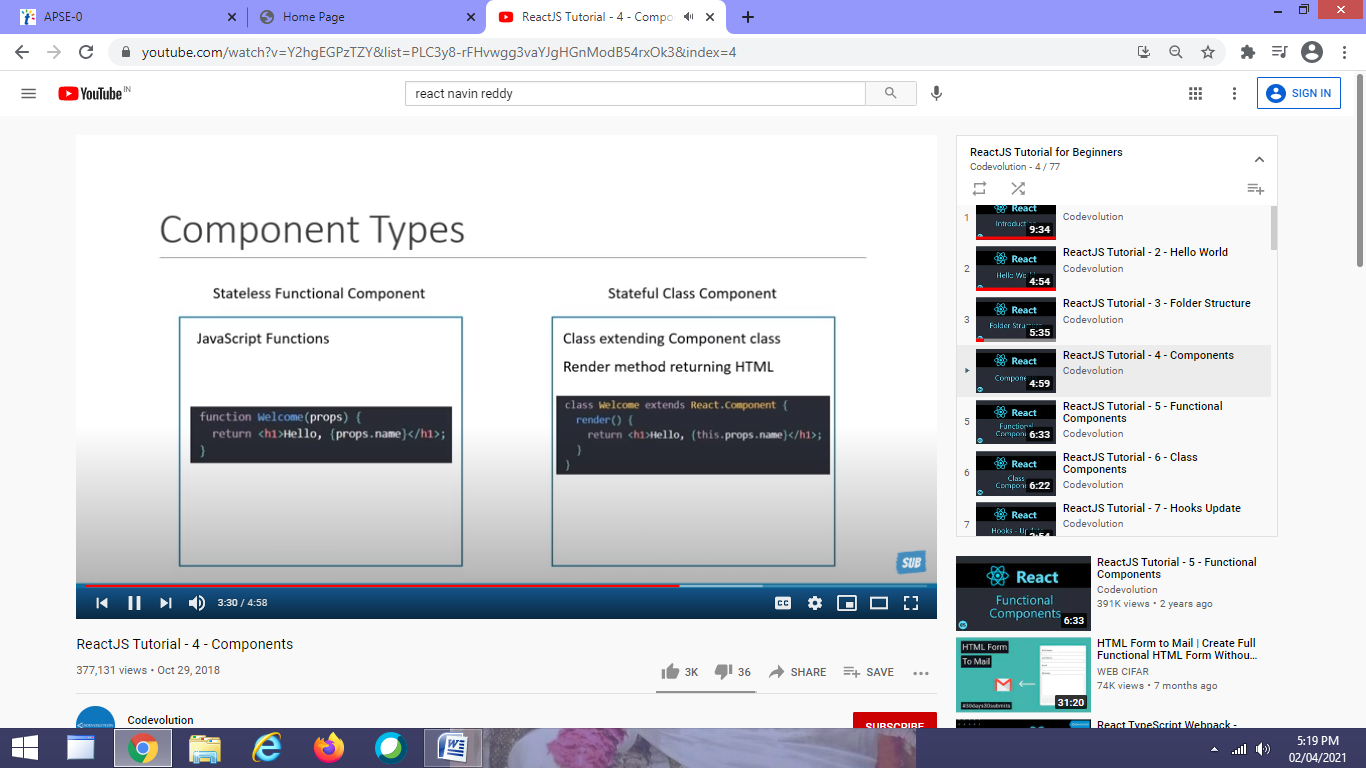
Day 2

|  |
| --- |
| * Explain React components |
| * Identify the differences between components and JavaScript functions |
| * Identify the types of components |
| * Explain class component |
| * Explain function component |
| * Define component constructor |
| * Define render() function |



Consider the following code:

import React from ‘react’;

class Catalog extends React.Component{

render(){

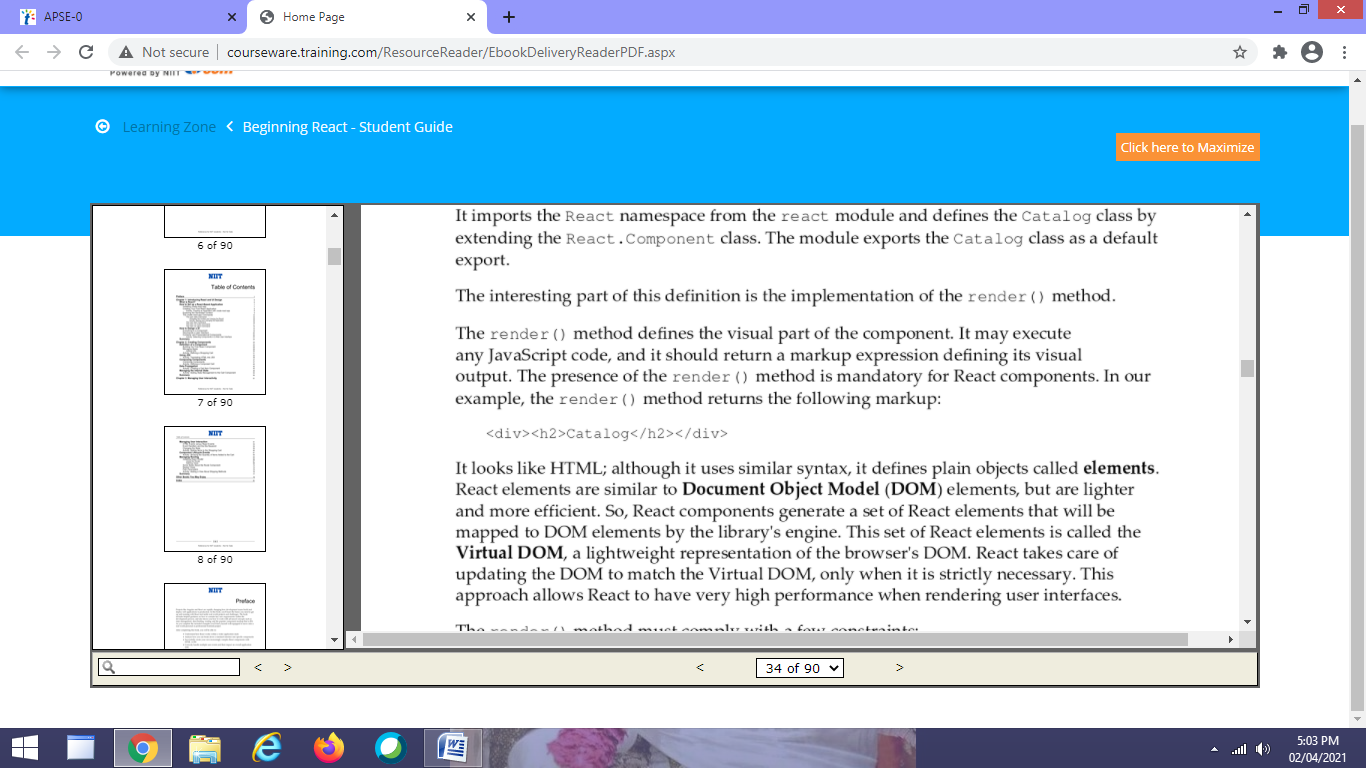
return <div> <h2> Catalog </h> </div>;

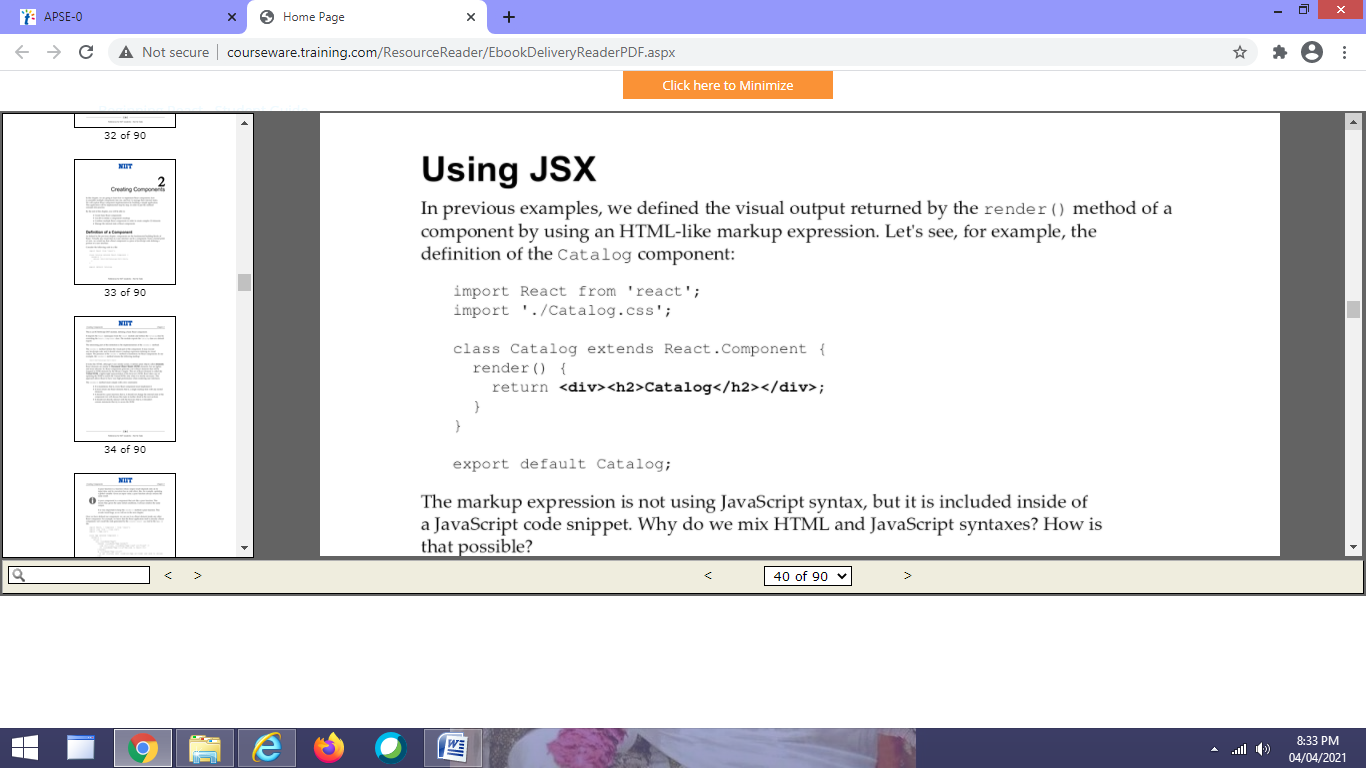
}

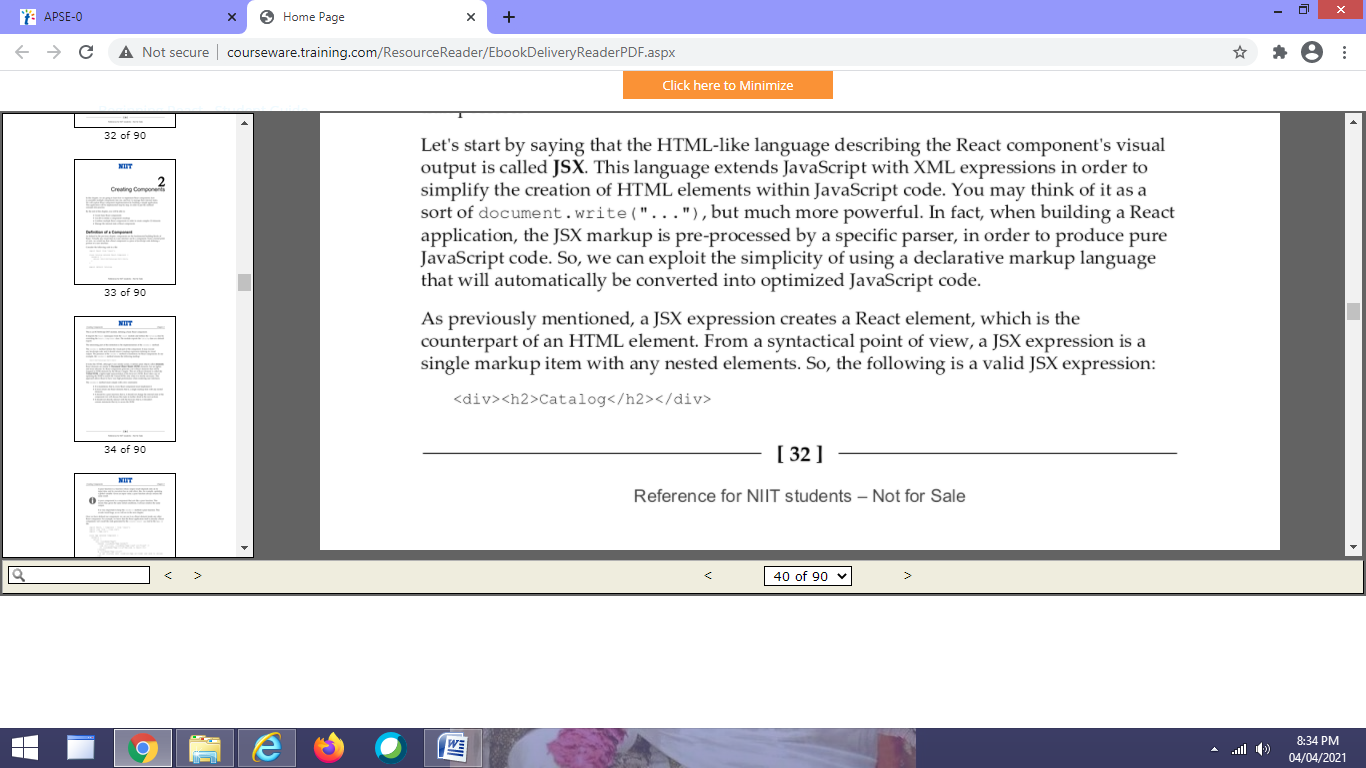
}

export default Catalog;

This is an ECMAScript2015 module, defining a basic React component.







The following is not a valid JSX expression, since it contains two markup items.

<div> <h2> Catalog </h2> </div>

<div> <img src=”” /> </div>

<div>

<h2> </h2>

<img src=”” width=”” height=”” />

</div>

We can assign JSX expression to a variable

import React from ‘react’;

class Catalog extends React.Component{

render(){

let output=<div> <h2> Catalog </h> </div>;

return output;

}

}

export default Catalog;

we can also embed any JavaScript expression inside of a JSX expression by wrapping it in curly braces

import React from ‘react’;

class Catalog extends React.Component{

render(){

**let title=”Catalog”;**

let output=<div> <h2> {title} </h> </div>;

return output;

}

}

export default Catalog;

Ofcourse, JavaScript expression can be as complex as we need it to be,

**let title=”The Catalog of today ”+ new Date().toDateString();**

**return <div> <h2> {title} </h2> </div>;**

**A common use of a combination of JavaScript and JSX expression is called conditional rendering; that is , a technique that allows you to generate a JSX expression based on some Boolean condtions.**

**class Message extends React.Component{**

**render(){**

**let message;**

**let today=new Date().getDay();**

**if(today==0){**

**message=<div className=”sorry”> We are closed on Sunday... </div>;**

**}**

**else{**

**message=<div className=”happy”> How can we help you? </div>;**

**}**

**return message;**

**}**

**}**

**You can put a JSX expression in multiple lines**

**return <div>**

**<h2> {title} </h2>**

**</div>;**

**================ if you want to start expression on new line**

**return (**

**<div>**

**<h2> {title} </h2>**

**</div>);**

**JSX tags match HTML tags, which is why we can use the whole HTML syntax to define JSX elements. However, there are few restrictions.**

* **All HTML tags are in lowercase.**
* **You need to use className instead of the class attribute.**
* **You need to htmlFor instead of the for attribute.**

export default Message;

In src folder – create new folder components

Inside this create a file Header.js

Import React from ‘react’;

function Header(){ //add object as a parameter Header({name,title})

return(

<div style={{background:”yellow”, padding :20}}>

<h1> {title} </h1>

<h1> Learning to create a new component with {name} </h1>

</div>

);

}

Export default Header;

=====================================================

In App.js

import Header from “.components/Header”;

<div>

<Header name=”Shreehari” title=”First Name”/> // this is called as props

<hr>

<Header name=”Jitendra” title=”Second Name”/>

...

...

<Header name=”Ayesha” title=”Third Name”/>

In this example App component is used to pass

State :- It is basically to store data of the specific component, in our example Header component contains state which can be changed

Property :- It used to pass data to one component. Like in App.js we are using property for communication. Here we passing data “Shreehari” to Header component for communication.

Here we understood the working of React component and now will understand how to work with states.

This was to just practice

Summary of Components

Components are the basic building blocks of any React app.

Components let you split the UI into independent, resuable pieces, and think about each piece in isolation

Components are like Javascript functions. They accept arbitrary inputs (called “props”) and return React elements describing what should appear on the screen.

Always start component names with a capital letter.

React treats components starting with lowercase letters as DOM tags. For example, <div/> represents an HTML div tag, but <App/> represents a component requires App to be in scope.

We can create Components in two ways

Function Components

It is a JavaScript function which accepts a single “props” object argument with data and returns a React element.

Syntax:-

Without Props

function function\_name(){

return ReactElement;

}

Example:- Create new Student.js

import React from “react”;

function Student(){

return <h1> Hello Jitendra </h1>

}

OR

This is the old version, now lets see using arrow function ie using ES6 format

import React from “react”;

const Student=()=>{

return <h1> Hello Jitendra </h1>

}

export default Student;

With props :- Recommended to use keyword props

function functionname(props){ // props is nothing but object containing data

return reactelement;

}

Example:

function Student(props){

return <h1> Hello Jitendra </h1>

}

function Student(props){

return <h1> Hello {props.name} </h1>

}

OR ES6 using arrow function

const Student=(props)=>{

return <h1> Hello {props.name} </h1>

}

Class Component

A class component requires you to extend from React.Component. the class must implement a render() member function which returns a React component to be rendered., similar to a return value of a functional component. In a class-based component, props are accessible via this.props.

Syntax:

class App extends Component {

render(){

return react element

}

}

Example:

class Student extends Component{

render(){

return <h1> hello Jitendra </h1>

}

}

class Student extends Component{

render(){

return <h1> Hello {this.props.name}</h1>

}

}

Rendering a component

In index.js

Import Student from “./Student”;

ReactDOM.render(<Student/>,document.getElementById(“root”));

ReactDOM.render(<Student name=”Shreehari”/>,document.getElementById(“root”));

Or using ES6

function Student(props){

return <h1> Hello {props.name} </h1>

}

ReactDOM.render(<Student name=”Shreehari”/>,document.getElementById(“root”));

When React sees an element representing a user-defined component, it passes JSX attributes to this component as a single object. We call this object “props”.

Component Constructor

Test.js

Import React from ‘react’;

class Test extends React.Component{

constructor(props){

super(props);

this.state=

{

data:”Madhuri”;

}

}

render(){

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

}

}

Export default Test;

========================================App.js

Import Test from ‘./Test’;

function App(){

return <Test/>;

}

Props :- React Function Component’s parameter

import React from ‘./react’;

function Testing(props){

return <h1> {props.designation} </h1>;

}

export default Testing;

or

const Testing=({designation})=>{

return <h1> {designation} </h1>

===================================================

function App(){

return <Testing designation=”Developer”/>;

}

export default App;