CLASSTIME | Pg. No.
Date / /

REACT JS

COMPLETE NOTES.

ATUL KUMAR (LinkedIn / Twitter)
NOTES GALLERY / CODING BUGS (Telegram)

→ **What is ReactJS ?**

→ ReactJS is an open source Javascript library used for building interfaces. It focuses on creating dynamic interactive web applications by allowing developers to create reusable UI components.

→ **Why we ReactJS ?**

→ Reacts is popular and widely used for several reasons :

Component Based Architecture

React allows you to break down your UI into reuse components. Making it easier to manage.

Efficient Updates:

It uses Virtual DOM, which optimized rendering by updating only parts of pages that change improving performance.

CLASSTIME Pg. No.
Date / /

Large Ecosystem :

React has vast ecosystem of libraries and tools.

Cross Platform:

React can be used for both web and mobile app development.

PROS AND CONS OF REACTJS

PROS :

Component - based React encourages a modular, component based approach to build user interfaces.

Easy to learn and use : React is much easier to learn and use. It comes with good supply of documentation.

Reusable Components : A ReactJS web application is made up of multiple components. each components has its own logic.

Scope for testing Codes : ReactJS applications are extremely easy to test. It offers scope where developers can test and debug their codes with help of native tool.

CONS:

Poor Documentation React Technologies updating and accelerating so fast that there is no time to make proper documentation.

View Part ReactJS covers only UI layers of app & nothing else. so you still need to choose other technologies to get complete tooling set for development.

CLASSTIME Pg. No.
Date / /

DIFFERENCE B/W REACTJS & ANGULAR

Field	ReactJS	Angular
used as	React js is a library (Javascript)	Angular is a framework.
Released	It was released in 2013.	It was released in 2010.
Written	Reactjs written in javascript.	Angular is written in microsoft's Typescript.
Routing	Routing is not easy in ReactJS.	Routing is easy compared to ReactJS.
Scalable	It is highly scalable.	It is less scalable.
DOM	It has a Virtual DOM.	It has regular DOM.
Testing	It Supports unit testing	It support unit + integration testing.
Data Bindings.	It supports uni directional	It supports bi directional.

CLASSTIME | Pg. No.
Date / /

DIFFERENCE B/W REACTJS and REACTNATIVE

REACTJS	REACTNATIVE
Installation Process React library is installed via npm package manager.	Installation Process. ReactNative is a command line interface tool requires both Node js and ReactNative CLI to be installed.
Efficiency ReactJS is more efficient interms of code reuse ability.	Efficiency. ReactNative is more efficient interms of Performance & memory usage.
Technology Base ReactJS is Javascript library used for building user interfaces.	Technology Base. ReactNative is a cross platform mobile development.
Components ReactJS Components are typically written in HTML.	Components ReactNative Components are written in JSX.

CLASSTIME Pg. No.
Date / /

REACTJS	REACTNATIVE
Storage ReactJS is a good choice for projects that require high performance	Storage It is a good choice for Projects that need to be able to scale easy.
Search engine friendly. It is more search engine friendly than React Native. ~~Platform~~.	Search engine friendly It cannot be made search engine friendly.
Navigation. ReactJS uses traditional browser based approach.	Navigation. ReactNative relies on native platform.
Platform. React is a framework for building application using javascript.	Platform It allows building native and cross platform mobile apps.
Rendering Browser code is rendered through virtual DOM.	Rendering. It uses native API to render all components.
Syntax Makes use of HTML and its syntax flow.	Syntax. ReactNative uses ReactNative syntax.

CLASSTIME Pg. No.
Date / /

Key difference between React and Vue.

The main distintion between vue and React is how they approach application design.

While React focuses on creating reusable UI components vue takes approach by providing developers with frontend tools.

let now take a look at some particular differences.

VUE	REACT
It was single file components (SFC) to build different components.	It uses JSX as a component format.
It is used to develope web-based application.	It is used to develope web as well as mobile application.
State Management library is called ~~the~~ voux	State Management library is called Redux.

CLASSTIME Pg. No.
Date / /

VUE	REACT
The Performance is slow as react.	The Performance is slow when Compared to Vue.
It is not suitable for long term support.	It is suitable for long term support.

REACT JSX

ATUL KUMAR (LinkedIn / Twitter)
NOTES GALLERY / CODING BUGS (Telegram)

What is JSX?
JSX stands for javascript XML.
JSX allows us to write HTML in React.
JSX makes it easier to write and add HTML in react.

Coding JSX

JSX allows us to write HTML elements in Javascript and place them in DOM without Creating createElement() or appendChild()

JSX Converts HTML tags into react elements.
you are not required to use JSX, but JSX makes it easier to write React application.

CLASSTIME Pg. No.
Date / /

Example 1

JSX:

```
const my element = <hi> I love JSX! </hi>;
const root = ReactDOM.createRoot(document.get
                                  Element
                                  ById ('root'));
root.render (my Element);
```

Output:-

I Love JSX!

Example 2

Without JSX:

```
Const myElement = React.CreateElement
                  ('hi', { }, 'I do not');

Const root = React DOM.(reateRoot (document.get
                  Element by Id ('root'));
root.render(myElement);
```

Output:-

I do not

In Example 1. JSX allows you to write HTML directly within Javascript.
JSX is an extension of Javascript language based on ES6.

CLASSTIME Pg. No.
Date / /

Expression in JSX

With JSX you can write expression inside curly braces { } The expression can be react variable or property. JSX will execute expression and return the result.

Example

execute the expression 5+5

```
const myElement = <h1> React is {5+5} times
                  better with JS </h1>;
```

Output

React is 10 times better with JSX

Inserting a large block of HTML

To write HTML on multiple lines, put HTML inside Paranthesis.

Example:

```
Const myElem = (

  <ul>
    <li> Apple </li>
    <li> Banana </li>
    <li> Cheries </li>
  </ul>
);
```

CLASSTIME Pg. No.
Date / /

Output :-

- Apple
- Banana
- Cherries

Example :-

```
const myElement = (
{div}    = (
 {P} I am Paragraph {/P}
 {P} Paragraph too {/P}
{/ div}
 );
```

⇦ JSX will throw error if HTML is not correct or if HTML misses a Parent element Here {div} is Parent {P} is child.

Output :-

I am Paragraph

Paragraph too

Elements must be closed in
Close empty elements with / >

Setting up a React Environment

If you have npx and node.js installed, you can create a React application by using create-react-app.

The create-react-app will set up everything you need to react application to run.

Run the React Application

```
cd my-react-app
```

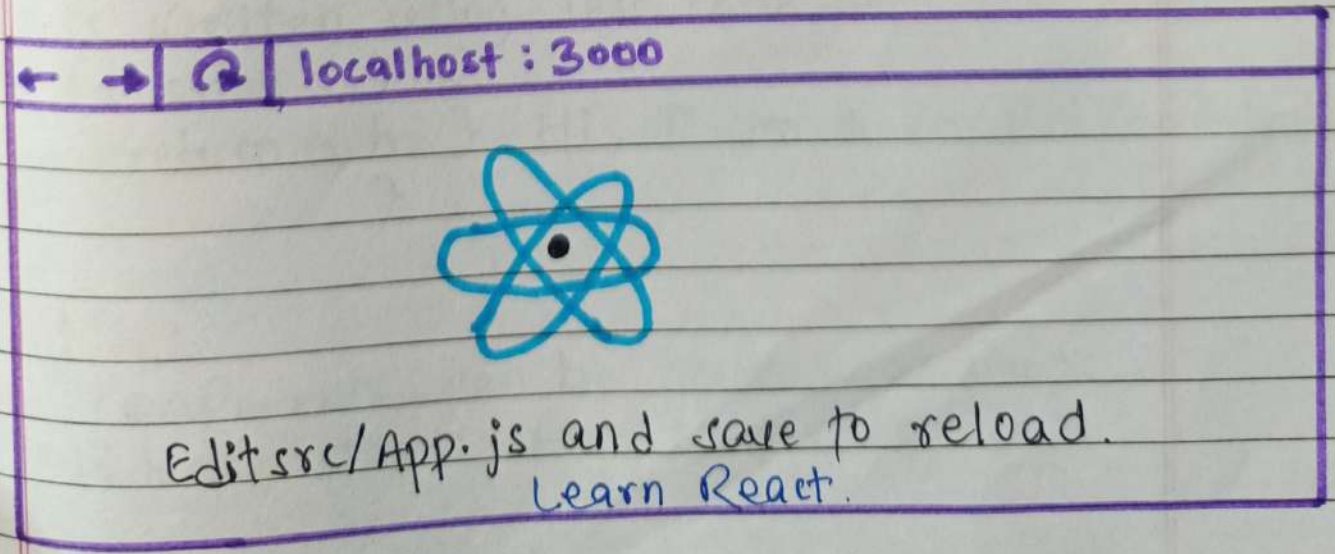
Run this command to run react application

```
npm start.
```

A new browser will pop up with your newly created ReactApp!

If not Open browser and type localhost :3000

The result:



CLASSTIME Pg. No.
Date / /

REACT Components

When creating a React component, the component name must start with uppercase letter.

Class component must include extends. React. Component statement.

The component also requires render(). this method return HTML.

Example :-

```
Class Car extends React.Component {
 render () {
  return {h2} Hi, I am a Car! {/h2}
      }
 }
```

Function Component

Same as react component, only difference is written using less code.

```
function () {
  return {h2} Hi, I am a Car! {/h2};
        }
```

Props

Components can be passed as Props, which stands for Properties.

Component in files

React is all about re-using code and it is recommended to split your components into separate files.

To do that Create a new file with .js files.

React class Component State

React class components have built in state object.

Creating the State Object

```
Class Car extends React.Components {
    constructor (props) {
    super (props);
   this.state = { brand: "ford" };
    }
     render () {
      return (
      {div}
       {h1} My Car {/h1}
       {/div}
        );
     }
     }
```

Using the State Object

Refer to the state object anywhere in the component using this.state.propertyname syntax.

Example :-

```
Class Car extends React.Component {
    constructor(props) {
    Super(props);
    this.state = {
     brand : "Ford"
     Model : Mustang"
     color : "red"
     year : 1964
    };
}

 render() {
     return (
     { div }
    {h}My {this.state.brand}</h1>
    {p}
     It is a {this.state.color}
     {this.state.model}
     from {this.state.year}
    </p}
     </div>
     );
 }
}
```

Output:-

MY FORD
It is a red Mustang from 1964

Changing the State Object

To change a value in state object, use the this.setState()

Example:-

```
Class Car extends React.Component {
Constructor ( Props )
Super ( props );
 this. state = {
      brand = "Ford",
      Model = "Mustang",
      color = "red",
      year = 1960
    };
 }
  changeColor = ( ) => {
     this. setstate ( { color: "blue" } );
  }
  render ( ) {
   return (
 < div >
```

```
<h1> My {this.state.} </h1>
<p>
  It is a {this.state.color}
  {this.state.model}
  from {this.state.year}
</p>

<button
    type = "button"
    onClick = {this.changeColor}
    > change color </button>
  </div>
  );
 }
}
```

Output :-

My Ford
It is a red Mustang from 1964

change color (When you click)

My Ford
It is a blue Mustang from 1960

Change color

LIFECYCLE OF COMPONENTS

Each Component in React has a lifecycle which you can monitor and manipulate during three main phases.

The three phases are:

- Mounting
- Updating
- Unmounting

Mounting

Mounting means putting elements in DOM.
React has four built in methods that gets called, when mounting a component:

1. Constructor ()
2. getDerivedStateFromProps ()
3. render ()
4. componentDidMount ()

Constructor

The Constructor Method is called by React, every time you make a Component.

Example :-

```
class Header extends React.Component {
 constructor (props) {
 super (props);
 this.state = { favourite color : "red" };
  }
 render () {
  return {

< h1 > My favourite color is { this.state.
                    favouritecolor } < /h1 > );
  }
 }

const root = React DOM.createRoot (document.
                    getElement ById ('root'));
 root.render (< Header / >);
```

Output :-

My favourite color is red

getDerivedStateFromProps

The getDerivedStateFromProps() is called right before element(s) in the DOM.

The example below starts with favourite color being "red", but the getDerivedStatefromprops() updates the favourite color based on favcol attribute.

Example:-

```
class Header extends React.Component {
 constructor (props) {
 super (props } ;
 this. state = { favouritecolor : "red " } ;
 }
 static getDerivedstateFromProps (Props, state ) {
    return {favouritecolor : props. favcol } ;
 }
  render () {
     return (
 < h1 > My Favourite Color is { this. state.
                    favourite color </ h1 >

     ) ;
  }
 }
```

```
const root = ReactDom.createRoot(document.
                    getElementById('root'));
root.render(<Header.favcol = "yellow" />);
```

Output:-

My Favourite Color is yellow

Render

ATUL KUMAR (LinkedIn/Twitter)
NOTES GALLERY/CODING BUGS (Telegram)

render() is required and is the method that actually outputs HTML to the DOM.

Example:-

```
class Header extends React.Component {
render() {
 return (
 <h1> This is the Component of Header </h1>
  );
 }
}
const root = ReactDOM.createRoot(document.
                    getElement ById('root'));

root.render(<Header />);
```

Output :-

This is the Component of Header

Component DidMount

ComponentDidMount() method is called after the component is rendered.

Example :-

At first my favourite color is red, but give me a second, and it is yellow instead.

```
class Header extends React.Component {

 constructor (props) {
   super (props);
  this. state = { favouritecolor : "red " };
 }
   componentDidMount () {
     setTimeout (() => {
 this. setstate ( { favouritecolor: "yellow" } )
   } , 1000 )
  }
```

```
render() {
  return (
<h1> My favourite color is { this.state.
                    favourite color } </h1>
  );
  }
}
const root = ReactDOM.createRoot(document.
                getElementById('root'));
```

Output :-

My favourite color is yellow

Updating

The nextphase in lifecycle is when component is updated.

React has five built in methods that gets called when component is updated.

CLASSTIME Pg. No.
Date / /

React Props

Props are arguments passed into React components.
props stands for properties.

Example :-

Add a brand name attribute to car element

```
const MyElement = < Car brand = "Ford" /> ;
```

Example :-

use brand attribute in the example

```
function Car (props)
return <h2> I am a { props.brand } ! </h2>;
const MyElement = < Car brand = "ford" /> ;

const root = ReactDOM.createRoot ( document.
                    getElementById ('root'));
root.render (myElement );
```

Output :-

I am a Ford

React Events

React has same events as HTML : click, change, mouseover etc.

Adding Events

React events are written in camelcase syntax on click instead onclick

React event handler are written inside curly braces

onclick = { shoot } instead of
onclick = "shoot ()"

React

```
< button onclick = { shoot } > Take shot !
                                < / button >
```

HTML

```
< button on click = "shoot ()" > Takeshot !
                                < / button >
```

Passing Arguments

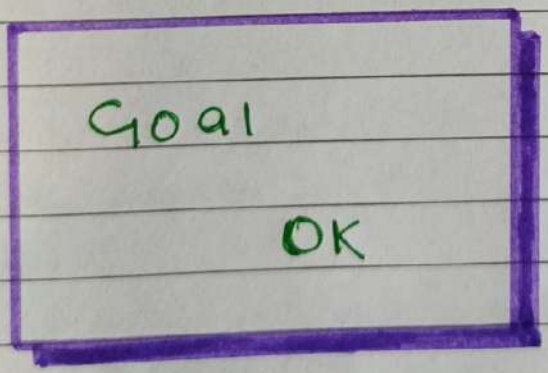
To pass an argument to event handler use an arrow function.

Example :-

Send "Goal" as parameter to shoot function

```
function Football () {
 const shoot = (a) => {
  alert (a) ;
 }
 return (
   < button on click = { () ⇒ shoot ("Goal")
  } > Take shot !
    < / button >
   ) ;
 }
Const root = ReactDOM. createRoot (
root. render ( < Football / > ) ;
```

Output :-



CLASSTIME Pg. No.
Date / /

React Conditional Rendering

If Statement

Example

```
function MissedGoal () {
  return <h1> MISSED! </h1> ;
}
```

Logical && Operator

We can embed javascript expression in Jsx by using curly braces.

```
Function Garage (props) {
 const cars = props. cars ;
return (
< >
 <h1> Garage </h1>
{ cars. length > 0 &&
 <h2>
   you have { cars. length } cars in your garage.
    </h2>
  }
</ >
 );
 }
```

```
const Cars = ['Ford', 'BMW', 'Audi'];
const root = ReactDOM.CreateRoot(document.
                getElementById('root');
root.render(<Garage cars = {cars} />);
```

Output :-

```
Garage

You have 3 Cars in your garage
```

Ternary Operator


```
Condition ? true : false
```

Example :-

```
function Goal(props) {
 const isGoal = props.isGoal;
 return (
  <>
{isGoal ? <MadeGoal/> : <MissedGoal/>}
  </>
 );
 }
```

```
Const root = ReactDOM.CreateRoot( document.
                getElementById ('root'));
root.render (< Goal is Goal = {false} />);
```

Output :-

```
MISSED
```

React List

List are used to display data in Ordered format & mainly used to display menus on websites.

Example.

```
  Var numbers = [1, 2, 3, 4, 5];
Const multiplyNum = numbers.map ((number)=>
{
 return (number * 5)
});
 console.log (multiplyNum);
```

Output :-

```
[5, 10, 15, 20, 25]
```

CLASSTIME Pg. No.
Date / /

React Keys

A key is a unique identifier. In react, it is used to identify which items have changed, updated or deleted from the list.

Example :-

```
const stringLists = ['Peter', 'sachin', 'Kelvin',
                     'Dhoni' ]
const updatedLists = stringList.map((strList)=>{
<li key={strList.id}>{strList}</li>;
});
```

React Refs

Refs is shorthand used for reference. It is an attribute which makes it possible to store a reference to particular DOM nodes or React elements.

When to use Refs

When we need DOM measurements such as managing fails, text selection or media playback.
It can also use as in call backs.

When not to use Refs

Instead of using open( ) and close( ) methods on a Dialog components, you need to pass an is Open drop it.

How to access Refs.

```
const node = this.callRef.current
```

Callback refs

Another way to using refs is callback ref and it gives more control when refs are set and unset.

React Forms

Example:-

Add a form that allows user to enter name:

```
Function MyForm ( ) {
  return (
  <form>
    <label> enter name :
      <input type = "text" />
    </label>
    </form>
  )
}
```

CLASSTIME Pg. No.
Date / /

Output:-

Enter your name: [ ]

React Router

Add react router in your application, run this in terminal from root directory of application.

```
npm i -D react-router-dom
```

Basic usage

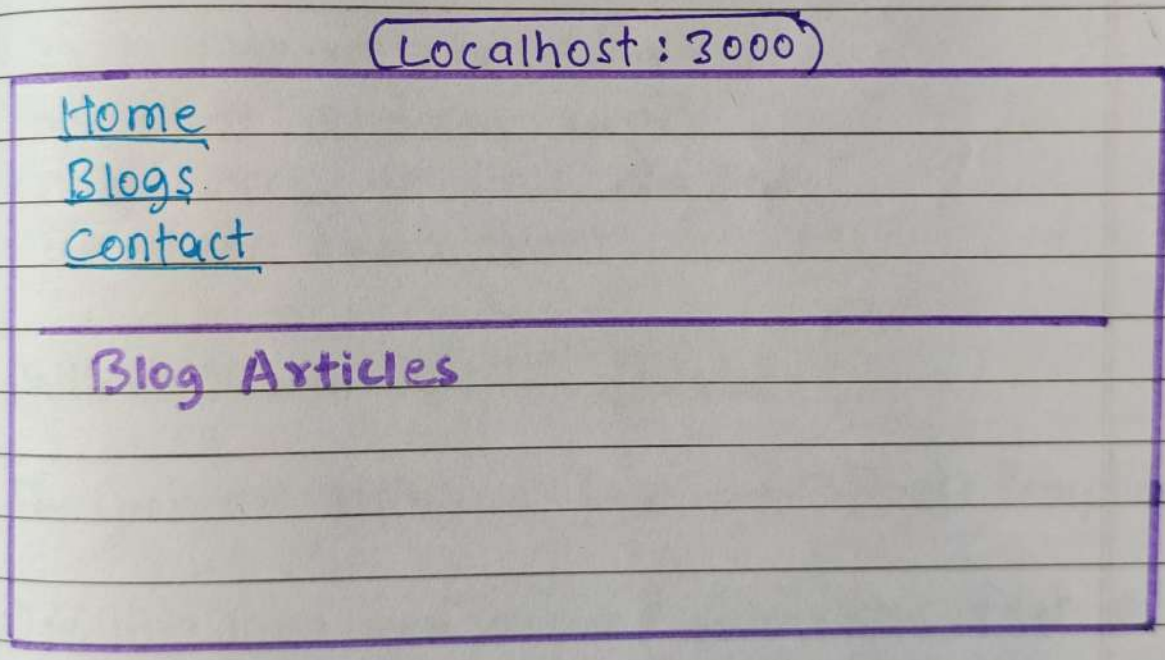
Now use router index.js file.

Example:-

```
export default function App () {
 return (
  < BrowserRouter >
   <Route path = "/" element = {< layout /> } >
  <Route index element = {< Home /> } />
  <Route path = "blogs" element = {< Blogs /> } />
  <Route path = "contact" element = {< contact />
                                    } />
  <Route path = "*" element = {< Nopage /> } />
    < Route >
```

```
</Router>
</BrowserRouter>
);
}
const root = ReactDOM.CreateRoot(document.
                getElementById('root'));
root.render(<App/>);
```

Output :-



Benefits of React Router

- It is not necessary to set browser history manually link uses a navigate internal links in application.

→ It uses switch features for rendering.

→ If the router need only single child element.

React Memo

React Memo is Higher Order Component which itself wraps around component to rendered Output

It is higher Order Component
uses last rendered result
only check for prop changes
Effect on React Hooks

Where to use React Memo

If you are using a pure functional Component:

Use it when you know before hand that a component will render quite often.

Use it if re-rendering is done using same Props.

Use it if your Component is big enough to have Props equality check done by React with decent number of UI elements.

CLASSTIME Pg. No.
Date / /

Where NOT to use React Memo ?

You should not use React Memo in all React component that implements the should Component Update ( ) method.
This is because it return true by default.

The render change that occurs by using React Memo is same implementation of should ComponentUpdate ( ) which does shallow comparison.
Other than this

- Don't use React Memo if Component is'nt heavy.
- Wrapping a class based Component in React Memo is undesirable.

React Fragments

ATUL KUMAR (LinkedIn / Twitter)
NOTES GALLERY / CODING BUGS (Telegram

When you want to render something, you need to use render ( ) inside Component.

This render method can return single or multiple elements.

If you want to return multiple elements. The render method requires 'div' tag and put entire content inside it.

This extra node to DOM sometimes results in wrong formatting of your HTML output.

Example :-

```
// Rendering with div tag
class App extends React.Component {
  render () {
  return (
    // div element
  < div >
   <h2 > Hello world </h2>
   </ div >
  );
  }
}
```

To solve the above problem React introduced Fragments.
Fragments allow you to group list of childern without adding extra nodes to DOM.

Syntax :-

```
< React.Fragment >
  <h2 > child1 </h2 >
  <p > child2 </p >
</ React.Fragment >
```

Example:-

```
class App extends React.Component {
 render() {
 return (
  <React.Fragment>
  <h2> Hello World </h2>
 <p> welcome to TopperWorld </p>
 </React.Fragment>
 );
 }
}
```

Why we use Fragments

The main reason to use fragments is
It makes execution of code faster as compared to div tag.
It takes less memory.

Fragment short Syntax

There is also another shorthand exists for declaring fragments. It looks like empty tag in which we can use '< >' and " instead of 'React.fragment'.

Example :-

```
class Columns extends React.Component {
 render () {
 return (
 < >
 <h2> Hello </h2>
 <p> Welcome to Topperworld </p>
 </>
  );
  }
 }
```


Keyed Fragments

The shorthand syntax does not accept Key attributes Key is the Key Only attribute that can be passed with the fragments.

You need a Key for mapping a Collection to an array of fragments such as to create a description list.

If you need to Provide Keys you have to declare fragments with explicit syntax.

```
Function = (Props) {
 return (
< Fragment >
{Props-items.data.map (item = > (
// without 'Key' React will give warning
< React.Fragment Key = {item.id} >
 <h2> {item.name} </h2>
 <p> {item.url} </p>
 <p> {item.description} </p>
   </ React.Fragment >
  )) }
  </ Fragment >
   )
  }
```

Difference between State and Props

Props	State
The data is passed from one component to another.	The data is passed within the Component.
Props can be used within state and functional Components.	The state can be used only with state Component / Class Component.

Props are read only.	The state is both read and write.
It is immutable.	It is mutable.
Props are controlled by whoever renders the component.	State is controlled by react Component.
Props can be accessed in functional component using props parameter and in-class component props can accessed using this Props.	State can be accessed using the state hooks in functional Components and in class components can be accessed.
Props are read only.	State changes can be asynchronous.
Props Communicate between Components.	State display changes with component.

CLASSTIME | Pg. No.
Date / /

Controlled Component vs Uncontrolled Component

Controlled	Uncontrolled
It accepts current value as Props	It uses ref for current values.
It has better Control over form elements and data.	It has limited Control over form elements.
It does not maintain internal state.	It maintain internal state.
Data is Controlled by Parent Component.	Data is Controlled by DOM.
It allows validation Control.	It does not allow validation Control.

Styling using CSS (React)

There are many ways to style React with CSS.
Common ways are :-

- → Inline Styling
- → CSS stylesheets
- → CSS Modules.

ATUL KUMAR (LinkedIn/Twitter)
NOTES GALLERY (CODING BUGS (Telegram)

Inline Styling :-

To style an element with Inline styling attribute.
Value must be javascript Object.

Example:-

```
Const Header = ( ) => {
  return (
  <>
<h1 style = <{ color: "red" }}> Hello Topper
                    </h1>
<p> welcome to Topperworld ! <p>
  </>
  );
}
```

```
const root = ReactDOM.createRoot(document.
                    getElementById('root'));
root.render(<Header/>);
```

Output :-

```
Hello TOPPer
Welcome to Topperworld !
```

Note :-

In JSX, javascript expression are written inside curly braces, and since javascript objects also uses curly braces, styling above written inside two set of curly braces.

Camel Case Property Names.

Example :-

Use backgroundColor instead of background-color :-

```
Const Header = ( ) => {
  return (
    <>
```

```
<h1 style = {{ backgroundColor: "lightblue"
        }} >
        Hello </h1>
  <p> Add style </p>
    </>
    );
  }
const root = ReactDOM.createRoot ( document.
          getElementById ('root'));
root.render (< Header />);
```

Output:-

Hello
Add style

Javascript Object

You can also create object with styling.

Example:-

Create style object named mystyle:-

```
Const Header = () => {
    const mystyle = {
```

```
    color : " white "
    font-family : " Sans-Serif "
    background color : " red "
  } ;
  return (
    <>
<h1> style = { mystyle } > Hello </h1>
<p> Welcome to React </p>
  </>
  );
 }
const root = ReactDOM.CreateRoot( document
                .getElementById ('root'));
root.render (< Header/ >);
```

Output :-

Hello
Welcome to React

Css Stylesheet

Write your c.ss styling in .c.ss file extension and import it in your application.

Create App.Css and insert some css code inside it.

App.css
```
body {
background-color : #28a034;
color : white;
font family : sans-serif ;
}
```

CSS Modules

ATUL KUMAR (LinkedIn / Twitter)
NOTES GALLERY / CODING BUGS (Telegram)

Another way of adding styles to your application is using css modules.

Css modules are convenient for components that are placed in separate files.

Create css module with .module.css extension example:- my-style.module.css.

My-style.module.css
```
.bigblue {
  color : DogerBlue ;
  font-family : sans-serif ;
  text-align : center ;
  padding : 40px ;
}
```

Styling React Using Sass

→ What is Sass?
→ Sass is a css - Processor
Sass files are executed on server and sends css to browser.

Install sass by running this Command on your terminal

```
> npm : sass
```

Create a sass file with .sass.

Example :-

My - sass. scss

Create a variable to define color of text

```
$ myColor : red ;
 h1 {
   color : $mycolor ;
 }
```

React Hooks

What are react hooks ?

React Hooks are a new feature of react-js using this it is possible to use state and other react features without writing class.

Rules to use books

Hooks should be used in topmost scope of code and never be used within loops, conditions or nested functions.

Hooks should only be used by react function Components.

Don't use Ordinary javascript methods to call hooks.

These rules are also applicable for custom hooks.

React has also built in hooks. i.e useEffect and Usestate.

CLASSTIME | Pg. No.
Date / /

Use State Hook

The use state Hook is used for storing a state within a Component.

The Usestate hook allows you to store and access state inside a component without using this.state or this.setState().

UseEffect Hook

It gives function components the ability to perform side effect, resulting in accomplishing the same thing that ComponentDidMount, ComponentDidUpdate and ComponentWillUnmount do in React Classes, but with single API.

Custom HOOKS.

Custom hooks is an effective option in case where we want to implement derived functionality of both usestate and useEffect.

CLASSTIME Pg. No.
Date / /

Benefits of React Hooks

Easy to Understand Complex components.
Reduced Complexity without classes.
Easy to reuse Stateful Logic.

React Flux vs MVC

MVC stands for Model View Controller. It is an architectural pattern used for developing the user interface.

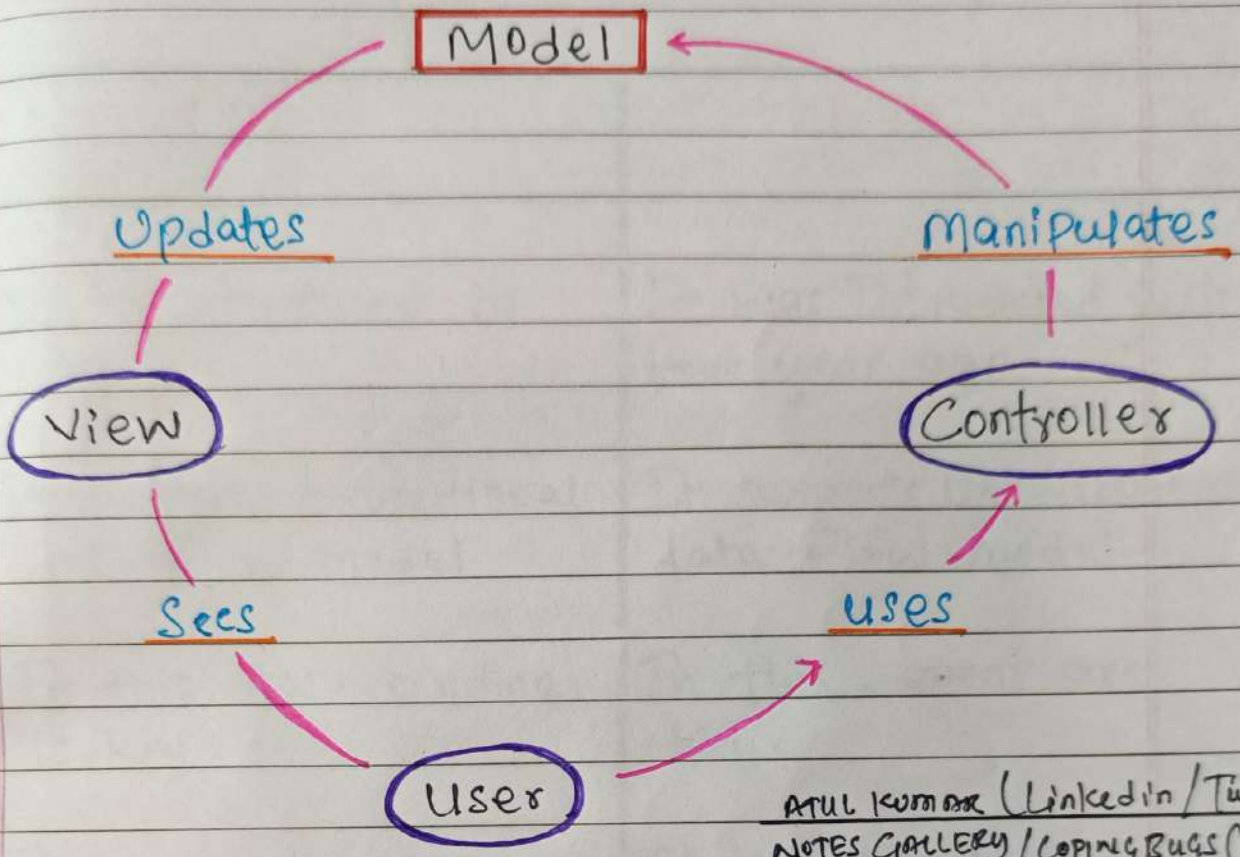
MVC Architecture

Model :- It is responsible for maintaining the behaviour and data of an application.

View :- It is used to display Model in user interface.

Controller :- It acts an interface between Model and view components.

It takes user input, manipulates the data and causes view.



ATUL KUMAR (LinkedIn / Twitter)
NOTES GALLERY / COPING BUGS (Telegram)

MVC architecture.

Flux:

Flux is an application architecture that facebook uses for building client-side web application.

Flux architecture has 3 major rules in dealing with data.

- Dispacher
- Store
- Views

CLASSTIME Pg. No.
Date / /

Difference between MVC and Flux

MVC	FLUX
It is introduced in 1976.	It was Introduced just few year ago.
It supports bidirectional data flow model.	It supports Unidirectional data flow model.
In this, data binding is key.	In this, event are Key.
It is synchronous	It is asynchronous
It is hard to debug.	It is easy to debug.
It is difficult to Understand as Project size Increases.	It is easy to Understand.
Testing of application is difficult.	Testing of application is easy.
Scalability is Complex	Scalability is easy.

CLASSTIME Pg. No.
Date / /

React Redux

React Redux is a predictable state container for javascript application.

It helps you write apps that behave consistently run in different environments (client, server, native) and are easy to test.

Redux is a state management tool

Redux can be used with any javascript framework.

Redux stores the state of application, and the components can access the state from state store.

Principle of Redux.

The three most important redux Principles are:-

→ Single source of Truth.
→ State is Read Only.
→ Changes are Made with Pure functions.

CLASSTIME | Pg. No.
Date / /

Single Source of Truth.

The state of your whole application is stored in an object tree within Single-store.

A Single state tree makes it easier to debug or inspect an application.

It gives you faster development of cycle by enabling you to persist in you app's navigation state.

State is read Only.

The Only way to change the state to initiate an action an Object describing.
This feature ensures that no events like network callback or views can change state.
Actions are just plain Objects, they can be logged.

Changes are made with Pure functions.

To Specify how actions transform state tree.
The user can return new state objects instead of Mutating Previous state.