REACT JS

Introduction

Implementation -

create -> npx create-react-app appname

run -> npm run start OR npm start

(react app will be hoisted in local server with local ip and

port always as 3000)

Understanding the Structure of React

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Fundamentals of React js

Components

1) functional Components

2) class Components

JSX and babbel

React fragments

Formatting in react

Dynamic values

Adding styles in react

Event Handling in react js

Conditional rendering in react

Mutiple Components

Displaying Multiple data in react with the help of array map

Props

Fecthing the data in react - (with the help of promise concept)

fetch()

axios()

useEffect()

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Hooks in React

useState()

useEffect()

useRef()

useContext()

useReducer()

useCallback()

useMemo()

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Advance concepts of ReactJ

Json server (alt of Back end server)

Error handling

React Routing

Route parameters

Public Routing

Protected routing

handling 404 error page

HTTP methods

Create - POST

Read - GET

Update - PUT

Delete - DELETE

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Application Structure

App.js

index.js

index.html

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Components

In react js we split each and every structure of our web page as a component.

Components in react is a function or a class which returns JSX value.

1) functional Component

if we return a template using a function , this is the most widely used

type of component in development mode because of it's advantages.

syntax : function Componentname()

{

return (JSX);

}

2) class Components

if we return a template using a class with render , now a days class

components are being ignored because of it's disadvantages.

syntax : class Componentname extends Component

{

render()

{

return (JSX);

}

}

JSX and babbel

JSX is a combination of JS + XML , which will allows programmers to write a HTML template

inside the javascript.

-> we can write predefined tags and also custom tags.

-> JSX is a special js value but our runtime environment(node js) cannot understand these values.

-> Babbel as a transpiler which will converts the special JS values to Normal JS values.

React fragments

-> In js we know our functions can return a single value.

-> But in react js if we try to return multiple JSX template , it will throw an error.

-> We can resolve it by using fragments in react.

-> React fragments can be any scementic or JSX fragment (<> </>)

Formatting in react

-> React is a combination of HTML (jsx) and JS so we have to format the code , when it is nested.

-> <HTML> { JS CODE } <HTML>

-> JS CODE ( <HTML> ) JS CODE

Dynamic values

-> In React js we can access the js values with the help of curly braces

-> Only numbers and string data type values can be printed.

ex : <h1> {variable} </h1>

<input type="number" step={variable}/>

Adding styles in react

1) Inline - In react inline css should be added as a object properties because it is a JSX(special js value)

2) External - no changes

\*\*\*NOTE\*\*\*

HTML class attribute should be used as className to differentiate

with the js class keyword.

Event Handling in react js

Event handling in react can be done by using event listeners as an attributes.

1st case : when events are 0 aurgumanted just pass the function reference to the event listeners.

ex : <button onClick={ function reference }> click </button>

2nd case : when events are aurgumanted we take a help of call backs and then invoke our functions inside

the call back function

ex : <button onClick={ ()=>{ Fname(args) } }> click </button>

HOOKS

Hooks in react are used to hang upon some features for our react Application.

Instead of writing the code to achieve that feature , we use the hooks (inbuilt method of react )

State management in React

In react we use a dynamic value , and whenever that dynamic is updated the current component should be

rerender then only we can see that updated value in UI.

1) useState()

useState hook can be used to achieve the state management.

useState will accept a single value and return an array of two elements.

syntax : let [variable , setVariable] = useState( init value );

access a value --> <h1> {variable} </h1>

update a value --> set Variable( updating value )

Conditional rendering

we need to render some set of structure based upon some Condition.

1) && (and operator)

syntax : { Condition && <JSX/> }

if Condition is true , then <JSX/> will be displayed

if Condition is false , then <JSX/> will not be displayed

2) ? (ternary operator)

syntax : { Condition ? <JSX 1/> : <JSX 2/> }

if Condition is true , then <JSX 1/> will be displayed

if Condition is false , then <JSX 2/> will be displayed

Multiple Components

In react we divide the entire application into several different components ,

but we need to call the components what we have created inside the App component

because App component is the only component rendered to INDEX.HTML

Displaying Multiple data

In React we will fetch the multiple data from the database , but we cannot write multiple

JSX template so we take a help of array map to return the multiple JSX template based on

the number of data we have.

syntax : array.map( (value)=>{ return(<JSX template/>) })

Props

In react we can send the data from parent component to child component with the help of

object properties or \*\*\*\*Props\*\*\*

syntax : function Parent()

{

return(

<Child key={value}/>

)

}

function Child( {key} )

{

return(

<h1> {key} </h1>

)

}