
REACT JS 26th November

Promise static method :

1.Promise.all(): it will take one argument ie array of promises and this method will return new promise of resolved state if all the promises are resolved.
If any one promise also in rejected state then it will return promise with rejected state.
If promise is resolved then promise result will be the array of all the promises and if the promise is in the rejected state then promise result will be first rejected promise value

2. Promise.any(): This also will take one argument ie array of promises.
This method will return promise with resolved state if any promise is also resolved.
This method promise with rejected state if all the promises are rejected.
if resolved then promise result will be first resolved promise value.
if rejected then promise result will be aggregate error ,all promises are promises are rejected

3. Promise.allSettled():

This also will take one argument ie array of promises.
it will always return promise with resolved state
promise result will be the array of the objects and it holds status and value or reason for all the promises

4. Promise.race():
This also will take one argument ie array of promises.
it will the return resolved promise or rejected promise based on which function is calling first ie resolved or reject function

#Arguments object :

argument is a keyword which is used to fetch all the arguments values inside the function without passing the parameter
it will return the argument object which is index data structure
we could not able to use arguments keyword as a identifier when use strict" statement at the beginning of a JavaScript file

==========
27th Nov
==========
Single Page Web Application :

1.In single page application the reloading will not happening for every functionality we are performing
2.In SPA if we send the request to the server we will get the response and again if we are sending any new request to the server we will get the new response and old response will store into the browser cache.
3.SPA are much faster performance than MPA.

Multi Page Web Application :

1. MPA the reloading will happen for every functionality we are performing.
2. In MPA every time both old and new request will go to the server and we will get the new response.
3. MPA are better in search engine optimization. than SPA.
4. To Create MPA we can use the technology like JQUERY (JS Library)
5. To create SPA we can use the technology like React , Angular , Vue
6. Mostly we can create E-commerce website or any traditional website like banking or Educational website a MPA.
7. We can create social media application or any streaming application as SPA. eg Facebook , Instagram

#Library :
Library is collection of pre-written code that developer use to perform to specific task.
The developer calls library functions or methods if needed.
The developer will control the flow of the application.
Libraries usually focus on the particular functionality or set of related functionality.
Libraries are the tool which we can use for your application.
ex. JQUERY , REACT

#Framework:
Framework provides structured foundation or architecture for your application.
The framework controls the flow of the program.
Developer can use that component if needed.
Framework is set of libraries or packages and it include tools for multiple aspect of development. (eg Database ,UI ,routing)
ex. Angular ,

#React Specification:

1.React is the java script library which is used to create the user interface.

2.Component based Architecture :
-React is used to create single page application and application are built using reusable independent bits of code called as Component.
-Each component can manage its own state and logic and we can use the component multiple time.
3. Declarative in Nature :
React allows developer to describe what the UI should like based on the application states.

4.Virtual DOM:
1.React application are faster in performance once because it follows the virtual dom.

2.When the webpage is loaded both actual dom and virtual dom is created for the webpage.

3.If we make any changes in the component UI first it will create the new virtual dom then it compare both actual dom and virtual dom
and make the necessary changes and update the actual dom.

#What is the virtual DOM:
It is the light weight copy of the actual dom which is used to optimize the update in actual dom

#What is Reconciliation
Reconciliation is the process of comparing the virtual dom and actual dom when the dom is updated or state of the component is changed.
It will add only the changes which is made in virtual dom to actual dom

#Diffing Algorithm :
When diffing two trees , React first compares the two root element. The behaviour is different depending on the type of the root element.

#NPM and NPX :
1.Npm is the package manager for javascript that comes with node.js and it is used to install and uninstall , update the javascript packages
and libraries which are required for the development.

2. Manage Dependencies in project using packages.json file

-How to install the package locally and globally :
npm install package-name -g [globally]
npm install package-name [locally]

-Uninstall the package
npm uninstall package-name -g [globally]

npm uninstall package-name [locally]

#NPX [Node package Execute]
npx is a tool bundled with npm version 5.2 which is used to execute the packages directly without globally install them.
to execute the package we can use the command
npx package-name

#Create react-app package
it is present inside the npm and it is used to create new react application which involves basic project structure and necessary
dependencies only.
To install the package we can use the command
npm install create-react-app -g

To create the react application we can use the command
create-react-app my-app
app_name and app name should be in lower case otherwise it will throws warning.

Execution :
npm start: To start the development server

npm run build : bundles the app into static files for the production

npm test: Starts the test runner

npm run eject : Removes this tool and copies build dependencies configuration files and scripts into the app directory. if you do
this you cant go back.

after creating project to navigate app directory we can use the command
cd app_name

#Folder and file details of react-app when we created with the help of create-react-app

1.node modules :
It contains all the dependencies and libraries
We generally dont modify anything in node modules fold and it handle by npm and yarn.

2.Public folder :
It contains static files and asset that will not be processed by web pack and these static files can we use our web application.
Inside the public folder we have index.html which is rendering on the browser first when we start the development server and it contains one
element called div with id root for rendering the react component.

3. Src folder :
It is a main directory where we can write react code [component ,style etc]
With in the src mandatory to create one index.js file which is entering point for react app.
These file renders the react app into the dom by targeting the div tag with id root in the html file.

4.Package.json file : -
It contains metadata about the project which includes project name , version and description , dependencies and devt dependencies.

#React package :
It includes in-built methods and properties which we can use for react application.
eg: createRef()

createRoot():

It is used to create the root element for the react application so that we can make every component as a child of these root element using render()
It is present inside the ReactDOM package.

React.createElement():
It is present with in the react package and it is used to create new react element and it will take three arguments
1. element type
2.Attribute in the form object or null
3.content /children

==========
2nd Dec
==========

JSX (Javascript)
JSX stands for java script xml and it is used to write html code in react.
JSX allows html elements in javascript and place into the dom without any create element and append child method
JSX convert html tags into react element

Rules of JSX:
1. if we define multiple react elements mandatory it should be wrapped into parent elements ie one top level element should be present. alternatively we can use fragment {<></>}

2. each every tag must be closed
3. if multiple elements are there it should be wrapped with in the ()

Changes in the html attribute
In react class attribute should be named className property because class is the keyword JS

For attribute in label tag should be named htmlFor property in react because for is also a keyword

EXPRESSION IN JSX
With JSX you can write the expression with in the {}
The expression can be the react variable or property any validation or any other JS expression.
JSX will execute the expression and it will return the result

FRAGMENT
the fragment looks like empty html tag (<> </>)
instead of taking one top level element for JSX alternatively we can use fragment to wrap multiple lines this will prevent unnecessary adding extra node to the dom

Conditions in JSX:
react supports if statements but not inside the JSX we can use if statement outside the JSX and take the result inside the JSX or we can use ternary expression.

Components in React :
Components are independent reusable bits of code and it returns JSX.
There are two types of components
1. Class Based { stateful} ->
2. Function Based {Stateless }

==========
3rd Dec
==========

```
// function component 3rd dec
function App() {
  return (
   <>
```

```
   <h1>hello world!</h1>
   </>
  );
}

// class based Component
class App extends React.Component
{
  render()
  {
    return(<h1>Welcome to </h1>);
  }
}

export default App;
```

State : hold the information about the component

Without Using Constructor ->

```
class App extends React.Component {

  state = {
    name: "abc",
    id: 123,
    sub: ["js", "java"],
  };

  render() {
    return (
      <div>
        <h1>{this.state.name}</h1>
        <h1>{this.state.id}</h1>
```

```
        <ul>{this.state.sub.map((x) => {

          return <li>{x}</li>;
        })}</ul>
      </div>
    );
  }
}

export default App;

----------------
Using Constructor ->
----------------


class App extends React.Component {

constructor()
{
  super();
  this.state = {
    name: "xyz",
    id: 123,
    sub: ["js", "java"],
  };
console.log(this)  //app component
}
  render() {
    return (
      <div>
        <h1>{this.state.name}</h1>
        <h1>{this.state.id}</h1>
        <ul>{this.state.sub.map((x) => {
```

```
          return <li>{x}</li>;
        })}</ul>
      </div>
    );
  }
}

export default App;
```

SetState : to update the state

```
class App extends React.Component {
  constructor() {
    super();
    this.state = {
      name: "xyz",
      id: 123,
      sub: ["js", "java"],
    };
  }

  render() {
    setTimeout(() => {
      this.setState({ name: "krishna",id:400 });
    }, 3000);
    return (
      <div>
        <h1>{this.state.name}</h1>
        <h1>{this.state.id}</h1>
        {this.state.sub.map((x) => {
          return <h1>{x}</h1>;
```

```
        })}
      </div>
    );
  }
}
export default App;
```

Notes :

Class based Component :
This components are simple java script classes made of multiple functions that add functionality to the application
Every class component are stateful ie in class based component we can define state with the help of state property {in build property for every class based component}
Every class component must be inherit React.Component class and it should render method which returns JSX.

Function based component:
This components are simple java script functions. {()=> ,name function , anonymous()} that returns JSX.
Function based component stateless ie we dont have any inbuild property called as state to define the data for that particular component
but after introduction of react hooks we can able to define the state in function based component also. We can use useState() hook.

State in React :
state hold the information or data about that particular component.
We could not able to transfer the state from one component to another component.
states are mutable { we can change }
Whenever there is any change into the state value the component will re render [imp]

How to define the state in class based component?
We can define the state in two ways in class based component
1. Using state property inside the class and outside the constructor
2. Using this.state inside the constructor.
3. In class based component state value should be object or null otherwise it will throws warning App.state : must be set to an object or null
4. Inside the constructor first statement should be super() which calls the parent class constructor
5. In class based component first constructor will execute and after that render() will execute

How to update the state value in class based component?
Every class component have one inbuild method ie setState() which is used to update the state value in class based component and it will take one argument ie value we are going to update {object or null}

==========
3rd Dec
==========

React hook

React hooks were introduced in the version 16.8
hook allows function component to have access to state and other react features so that class component are generally not needed
hook allows us to use life cycle method and state

Rules of hooks
There are three rules :
hooks can only we can call inside the function component
hooks can only call at the top level of the component
hooks can not be conditional

Note :

hooks will not work at class based component
we can also create custom hooks but it should start with use
hooks are simple java script function which is used to perform some functionality in function based component.

How to define the state function based component
Function based component are stateless ie we dont have any inbuild property to define the state but we can use useState hook to define state value
useState hook will take only argument ie state value which can be any data type and it will return one array with two elements first one is state value and second is updated function
This updated function is used to update state value and it will take one argument ie value which are we going to update

Events in react

Events are react are triggered action by user such as clicking button and handled by event handlers
we have to pass event handler property such onClick , onSubmit in he opening tag of the react element
this property will take one function as a value and this function will execute when user triggered the event

==========
4th Dec
==========

const [count ,setCount]=useState(0);

Handling events in React :

We can handle the events by passing the event handle property in the opening tag of the react element and the every event handle property will take function as a value and that function execute if user triggers the event
We can pass arrow function or named function as a value but incase of class based component when we are passing named function or anonymous function
this keyword refers to undefined we have to explicitly set the value for this keyword as the current component (same component)
We can set the value for this keyword inside the constructor using bind()
syntax : this.handleMouseOver.bind(this) [handlMouseOver is function]

Passing function as an argument for the updated function in Class based and function based component :
To update the state value we can also pass function as an argument that function will take one parameter which hold the previous value
In the return statement we have to pass value which we are going update
We can use this both class based and function based component.

==========
6th Dec
==========

Props :

Props are property which is used to transfer the data from one component to another
Props are immutable ie we can not change the props value.
Props follow uni-directional data flow ie from parent to child
Props can pass any type of data such as string , number , object , array , function

Passing the data through props :
The syntax for passing the data from one component to another is same in class based component and function based component

```
<Child
        name={"abc"}
        id={124}
        age={16}
        demo={this.demo()}
        skills={["java", "react", "spring"]}
/>
```

When we are passing the string data no need to write with in the expression directly we can pass in single code or double

Reciving the data in Child component
In class based component to receive the props we should use this.props property and it will return each and every data in the form of object
In function based component we can receive the props by passing the parameter and this also will return data in the form of the object

==========
9th Dec
==========

Aeum santacruz

==========
10th Dec
==========

How to set Default props :
In class based component we can set the default props using
1. Using static property defaultProps inside the component

2. Using OR operator
3. Using defaultProps property outside the component -> componentName.defaultProps={props: defaultvalue}

In function based component we can set the default props in two way
1. Using OR operator
2. Using defaultProps property outside the component -> componentName.defaultProps={props: defaultvalue}

How to set Datatype of the props ie we are receiving in the child component
Step 1 : Install the package ie prop-types
Step 2: import {PropTYpes }from props-type
Step 3: We can use PropTypes property outside the component and set the data type
syntax
ComponentName.PropsType =
{
prop1: proptypes.string,
prop2:proptypes.Number,
prop3: proptypes.func
}

Note :-React follows uni directional data flow ie through props we can pass the data from parent to child component
Through props we could not able to transfer the data from child to parent component we can use callback function in order to send the data from child to parent component

ref in React :

In react ref is an attribute or property ie provides a way to directly access or interact with dom element or react component
How to create the ref

In class based component we should use the method React.createRef() and it will return ref object with the key current and value is null
we can create ref object inside the constructor in class based component
In function based component to create the ref object we can use hook ie useRef() and it will return object with the key current and value is undefined
to pass the ref for the react element we should use ref property in both class based and function based component.

Note : - Dont overuse ref only use ref if needed otherwise it will affect the performance
Common use cases for UseRef()
Managing focus , text selection , controlling the media player , triggering the animation , integrating third party libraries that request direct dom manipulation

==========
12th Dec
==========

Conditional rendering :

We can render the component based on the some condition.
Conditional rendering have been done in four ways
if else
ternary operator
switch case
&& operator

Note : if else statement we could not able to pass inside the JSX
but we can pass ternary operator and && operator. We can also use nested component in order to perform conditional rendering.

Controlled Component :
In controlled component the form data is handled by react state.

The value of form element like input ,text area are set by the component state and any changes to the form element that we can update using event handlers like onChange().

Uncontrolled Component :
In Uncontrolled component the form data is handled by dom itself. The ref attribute is used to access the dom node to get or set the value.
React does not manage the state of the input instead browser does.

==========
24th Dec
==========

Component Life Cycle->

In react the component life cycle refers to the sequence of events that happens during the component the existed.
There are three phases
1. Mounting - Mounting phase is occurs when component is created and placed in DOM
2. Updating -> Updating phase happens if any change in the component state or if component is receiving the any props
3. Unmounting :Unmounting phase happens if the component is removed from DOM

Mounting phase methods :
1. Constructor :
It is best place to the initialize the state and also used to bind event handler and it is not suitable to perform side effect

2. static getDerivedStateFromProp(props,state) :
This method is used to update the state value that depends on props.
The value we are going to update should pass in the return statement otherwise null
.

it will take two parameters first is props and second state
it is also not suitable to perform side effect.

3. render() :
must use in class based component and return JSX
it is also not suitable to perform side effect.

4. componentDidMount():
it is used to perform the side effect like data fetching , manually changing the dom and setting up the subscription

side effect in react :
Side effect rfers to any operations or behaviours that occurs in component after rendering and that dont directly impact the current component render cycle.

React Life Cycle Method-->
Mounting->{ 1.Constructor 2. getDerivedStateFromProps 3.render 4. componentdidMount }
Updating->{ 1. getDerivedStateFromProps 3.shouldComponentUpdate ->[4.render 5.getSnapshotBeforeUpdate 6.componentDidUpdate] }
UnMounting->{compoentWillUnMount}

==========
25th Dec
==========

updating phase methods
1. static getDerivedStateFromProps

2. shouldComponentUpdate():
This method defines whether component should re render or not

it will return Boolean value if return value is true then component will re render and if false component will not re render .

3. Render():

4. getSnapShotBeforeUpdate():
This method is used to store the previous state and previous prop before it is getting updated. The value which we are going to store should pass in the return statement.
This method should use along with componentDidUpdate otherwise it will throw warning.
This snapshot value we can fetch inside the componentDidUpdate by passing the third parameter.

5. componentDidUpdate():
This method is invoked immediately after component has updated
it is commonly used for performing the side effects such as data fetching and manually updating the dom
it will take three parameters 1. previous props 2. previous state 3. snapShotValue

Unmounting phase method:

1. componentWillUnMount():
This method is called just before component is unmounted or destroyed.
it is used for cleanup task such as cancelling network request and removing event listeners

Note : ->
If we have parent and child component and whenever the child component is un mounting phase before execution of componentWillUnMount() in child component getSnapShotBeforeUpdate() in parent Component will execute.
This is because to store the previous props and previous state before the child component is removed from DOM tree.

---
27 December 2024

#useEffect: It is inbuild hook which present inside the react package. It is used to perform side effect in the function based component.
Common use Cases for use Effect includes Data fetching , Manually dom manipulation ,setting up the subscription

Syntax :->
#useEffect(()=>{},[dependency list])
it will take two arguments first one is call back function ie effect callback and second one is dependency list { []}
If we are passing empty array as second argument effect call back function will not execute in the updating face.
if we are not passing second argument then effect call back function will execute for every state change or when it receives any props.

if we are passing array of dependencies then effect call back function will execute only when the dependencies changed.

Note : We can pass multiple use Effects hook also and it will execute in sequence.
The effect call back function will always execute once in the mounting phase.

Unmounting phase in useEffect :
The effect call back function will return another function ie Destructor function which is used to clean the things like cancelling networking request and

useEffect will execute after the render
The destructor function will execute in the updating phase also (before making any changes in state or props)to cleanup the any task.

---
30 December 2024

#Props.children :
It is special property that contains children pass to the react component
If we want to transfer the children which is present between the opening and closing tag. we can use props.children property

#Props drilling :
It is the process of passing the data through props from parent component to deeply nested child component.
We can avoid prop drilling using contextapi , redux customHooks with the help of this directly we can pass the data from one component to another without prop drilling.

#useReducer():
-This hooks is present inside the react package.
-It is used to define the state value for the particular the function component, When the state is having complex logic
-It will take two argument 1. reducer function 2. initial state value
-useReducer hook will return array of two elements 1. state value 2. dispatch function

--Dispatch():
When we call dispatch function the reducer function will get execute.
It will take one argument ie action which we want to perform (mostly object)

--Reducer():
It will take two parameters 1. state 2. action .
With in the reducer function we have to write complex state logic to update the state value.
The state value which we are going to update it should pass in the return statement.

---
31 December 2024

#ContextAPI in React
It is a React feature that allow us to manage the state and data otherwise transfer the data across the your component tree without passing the props manually at every level. It is particularly useful for managing the state at global level in react

When we need to share the state between the component without prop drilling
Manage the global State :

Note --> ContextAPI is used for smaller application otherwise it is very hard to debug. for larger applications we can use state management libraries such as Redux.

There are three steps in contextAPI
Step 1:
Creating the context :
To create the context we use a method React.createContext(). it will create one global repository.

Step2:
Provide the context:
We can provide the data inside the gloabal repository by using Provider Component. Provider is the component that allow consuming component that are children. it takes value prop where we should pass the data in the form of object.

Step3:
Use or Consume the context:
We can access or consume the data that is present inside the global repository with the help of useContext()

it will take one argument that Context Reference and it will return the data in the form of object.

2nd Jan 2025

#React router Dom:

It is a library for managing client side routing in react application. It enables you to define and handle navigation between different components In react aap without page reload.
It is mostly used for single page application.
To perform routing we should install the package react-router-dom.{ npm install react-router-dom}
first step is to make the browserRouter as a parent of App component
it wraps your application and providing routing context.

Routes and Route is used to define the page and corresponding component
route will take two property 1. path 2. element

Link and NavLink provides navigation links
both will take one property called to and pass path as a value. { to={"/about"}}

Note: Navlink component will add active className for the link which are active.

#useNavigate()
It is used to programmatically navigate to between the routes in react
It returns one function ie navigateFunction() and that navigate function it take one argument ie path
We can also pass -1 as an argument to go back one step in the history . 1 for go forward in the history

Nested Route: Route inside another route is known as Nested route. To perform the nested routing we have to set the path and element for nested component with in the opening and closing tag of the parent route.
In nested routing while setting the path dont use / and same for to property also

<Outlet/> : The outlet component is used in nested route to render child routes . it present inside react router dom package.

Dynamic Routing : It involves creating routes that can handle dynamically by using parameters or path. Dynamic routes are defines using : followed by a parameter name in the route path.

useParams():
It is used to fetch the parameter or id that we are passing dynamically.
It returns object and value will params

#useRoute():

It is present inside the react router dom
It allows us to define and use routes programmatically and providing and alternative to the traditional
<Routes> jsx structure
This hook will take one argument ie []
path and the element we should pass as object
Nested routing can be performed by children property and it takes one [] as value.
index routing can be performed by using index property and pass the value as true.

#Redux :(IMP)

Redux is the popular state management library for JavaScript application it is used with react to manage the global state in a predictable way.
Redux uses a central store the to hold the application state which can be updated by dispatching actions that are handled by reducers.

-->Store: It is the centralized container for the application state. it holds every data ie state data which want to transfer to the different components

Slice : it is the collection of redux logic ie state and reducer for the single future.
Slice is created by using createSlice() which is present inside the redux-js-toolkit package

#Use Selector() :
This hook is used to fetch the state value from the slices
It is present react redux package
It will take one argument ie call back function (state)=>{}
and that () take one parameter which is state eg let users =useSelector((state)=> return state.userInfo.users)
the state value which want fetch it should pass in the return statement

#useDispathch:
This hooks returns dispatch function from the redux store and dispatch() is used to dispatch the action
dispatch(action:(payload))
dispatch() will take one argument ie action and we should call the action by passing the payload as an argument.

#useMemo()[imp]

It is used to optimize the performance by memo zing the result of the slow function or any costly operations it will take two arguments first is

1.call back function 2. dependency list

call back function that returns the value which want to memoize
this call back function will execute if any changes in the dependency list
useMemo() returns memoize value of the function