REACT JS

INTERVIEW Q'S (NARESH IT)

1)What is React??

- ReactJS is the ***Client-Side JavaScript Library***.
- ReactJS given by *Facebook*.
- ReactJS used to ***simplify the complex UI***.
- By using ReactJS, we can split the complex UI to multiple executable solutions.
- Each executable solution called as ***component.***
- Components are the building blocks of any React App.ReactJS is component-based Library.
- We can ***reuse the components*** in ReactJS
- React's one-way data binding keeps everything ***modular and fast***.
- ReactJS is not a MVC Framework. ReactJS is the ***View in MVC***.

---

2) Explain Virtual DOM??

- Virtual DOM is a ***lightweight JavaScript object***.
- It is simply a copy of the real DOM
- Virtual Dom ***updates faster***.
- Virtual Dom Can't directly update HTML
- DOM manipulation is very easy
- ***No memory wastage***
- The render() function in React is responsible for creating a node tree from the React components.
- A ReactElement is a representation of a DOM element in the Virtual DOM.

---

3) What is JSX??

- JSX stands for ***JavaScript + XML***
- ***Brower can't understand XML***
- So, we ***must convert XML to JavaScript***
- We will use "***Babel***" tool for conversion.
- We will develop React Applications by using JSX.
- It makes easier to ***create templates***
- It is faster than regular JavaScript because it performs optimization while translating the code to JavaScript
- JSX is a XML-like syntax extension to ECMAScript without any defined semantics

Variable declaration:

const element =Hello, jsonworld! ;

Above tag syntax is neither a string nor HTML.

It is called JSX, and it is a syntax extension to JavaScript. JSX produces React "elements".

---

4) what is React.create Class??

- React.createClass ***allows us to generate component classes***.
- React allows us to implement component classes that use ES6 JavaScript classes.
- The end result is the same — we have a component class. But the style is different.
- And one is using a custom JavaScript class system (createClass) while the other is using a "native" JavaScript class system.
- When using React's createClass() method, we pass in an object as an argument. So, we can write a component using createClass that looks like this:

```
import React,{Component} from "react";

const Contacts = React.createClass({

render(){

return (

<div> .....</div>
);
```

```
}
});
```

export default Contacts:

- Instead of using a method from the react library, we extend an ES6 class that the library defines, Component.

```
import React,{Component} from "react";

class Contacts extends React.Component({

constructor(props){
super(props);
}
render() {
return(
<div> —– </div>
);
}
}
```

- export default Contacts.
- constructor() is a special function in a JavaScript class.
- JavaScript invokes constructor() whenever an object is created via a class.

---

5) What is React DOM??

- ***React DOM is the glue between React and the DOM***.
- When you want to ***show your react component on DOM*** you need to use this ***ReactDOM.render();*** from React Dom.
- The ***react-dom package contains ReactDOM.render***, ReactDOM.unmountComponentAtNode, and ReactDOM.findDOMNode, and in react-dom/server we have server-side rendering support with ReactDOMServer.renderToString and ReactDOMServer.renderToStaticMarkup.
- – This package serves as the entry point to the DOM and server renderers for React.

- It is intended to be paired with the generic React package, which is shipped as react to npm.

Installation:

- ***npm install react react-dom***

In the browser:

```
var React = require('react');
var ReactDOM = require('react-dom');

class MyComponent extends React.Component {
render() {
return <div>Hello World</div>;
}
}
ReactDOM.render(<MyComponent />, node);
```

ON THE SERVER :

```
var React = require('react');
var ReactDOMServer = require('react-dom/server');

class MyComponent extends React.Component {
render() {
return <div>Hello World</div>;
}
}

ReactDOMServer.renderToString(<MyComponent />);
```

API:

- react-dom
- findDOMNode
- render
- unmountComponentAtNode
- react-dom/server
- renderToString
  renderToStaticMarkupAttend

6) what are the differences between Reactjs and Angular?

REACT:

- React is open-source ***library*** for java script.
- React updates its ***virtual DOM***.
- React uses ***one-directional data flow process***.
- We can use react library with other programming libraries.
- React is good for developing web applications which contains simple mathematical operations.
- React supports ***JSX syntax***.
- React use ***.jsx*** extension.

ANGULAR:

- Angular is a open-source ***framework*** for java script.
- Angular directly updates its ***Real DOM***.
- Angular uses ***two-directional data flow***.
- But Angular is a compete solution in itself as it a complete framework we named.
- Angular is good for developing industrial applications which contains more mathematical operations.
- Angular supports ***Type script syntax***.
- Angular use ***.ts*** extension.

---

7) What are the Differences Between Angular, React, VueJS and NodeJS?

reactjs:

- React is ***open-source library*** for java script.
- React updates its ***virtual DOM***.
- React uses ***one-directional*** data flow process.

angularjs:

- Angular is open-source ***framework*** for java script.
- Angular directly updates its ***Real DOM***.
- Angular uses ***two-directional*** data flow.

Vuejs:

- Vue.js is the ***progressive framework*** for java script which builds ***user interfaces*** and ***single page applications***.
- Vue js is very easy to set-up and execute faster compare to other framework.

Nodejs:

- Node.js is an open-source ***runtime environment*** for developing ***server side*** and ***networking application***.
- Node.js uses an ***event-based approach*** to address implicit ***scalability***.

---

8) what are the differences between framework and library?

Framework:

- A framework is a skeleton where ***application defines its own features*** to fill out the skeleton.
- But in the frame work the control is vice versa of library where ***framework calls or control.***

Library:

- A library is a ***collection of class definitions*** for which ***code reusability is high***.
- when we call a method from a library at that time we are in control over the library

---

9) Will react follows MVC?

- No, React ***won't follow MVC*** because it follows ***unidirectional*** data flow wherever ***MVC follows two-directional*** data flow.

---

10) In how many ways we can create component?

- In react.js we can create component in two ways, that given below.

1>Class component:

Ex.

```
import React, {Component} from "react";
Class App extends React.Component{
Render() {
Return(
<div>
<b>HELLO WORLD</b>
</div>
)
}
Export default App;
```

2>Function component:

EX.

```
Import React from "react";
const App = () = > {
return <h1>Hello Rajesh!!!!</h1>
}
Export default App;
```

11) what is state??

- State is predefined object in react js. It is
  z.
- We can create state only inside the class component. If we want, we can update the state value within the component itself.
- There are two ways to initialize the state in React component.
  i) Directly inside the class

  ii) Inside the constructor.

Ex: State Inside class:

```
Class Test extends Component {
state={
Name:" Siva"
}
render (){ return(
<div>{ this.state.Name}</div>
)}
}
}
```

Ex:State inside constructor:

```
class App extends React.Component{

constructor () {

super();

this. state= {Name:"Sai" }
}
render (){
return(

<div>{ this.state.Name}</div>
)}
}
```

12) what are props??

- React props are like function arguments in JavaScript and attributes in HTML.
- React allows us to pass information to a component using something called props.
- To pass the data form parent component to child component we can use props.
- Props are immutable (can't change the value of props).

Ex:

```
import React, {Component} from 'react'
export class App extends Component {
render() {
return (
<div>
<Greet wish="welcome"></Greet>
</div>
)
}
}
```

```
import React, {Component } from 'react'
export class Greet extends Component {
render() {
return (
<div>
<h2>{this.props.wish}</h2>/*output is welcome*/
</div>
)
}
}
```

Note: The value of wish can't be change in Greet Component. Because props are immutable.

13) what are the differences between state and props ?

state:

- state is managed with in the component
- state is mutable (can change). The value of state properties can be change by using setState.
- variables declared inside the function body
- In class component state can be accessed using this keyword

Ex: this.state{ name:"rahul"}

- In functional component state can be accessed using "useState"hook

import React, { useState } from 'react';

function Example () {

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

}

props:

- props get passed through parent to child component
- props are immutable (can't change)
- props are function parameters
- In class component props can be accessed using
- "this.props".
- In Functional component props can be accessed by using "props".
- function Welcome(props) {
- return <h1>Hello, {props.name}</h1>;

---

14) how to change the state in ReactJS??

- By using setState() we can change the state.**Ex.**

```
import React, { Component } from 'react'
export class StateChange extends Component {
constructor(){
super();
this.state={
age:20
}
}
// without setState
/* increment=()=>{
this.state.age=this.state.age+1;// age value won't be incremented
console.log(this.state.age);//incremented
} */
increment=()=>{
this.setState({
age:this.state.age+1 //whenever we click on button age value will change
})
console.log(this.state.age);
}
render() {
return (
<div>
<h3 align='center'>Age:{this.state.age}</h3>
<button onClick={this.increment >Increment</button>
</div>
)
}
}
export default StateChange
```

15) explain map() function in reactjs

- Map() function used to render array of data.

- import React,{Component} from ‘react’;

```
export default class State extends Component{
constructor(){
super();
this.state={
products:[
{‘pid’:111,’pname’:’pone’,’pcost’:10000},
{‘pid’:222,’pname’:’ptwo’,’pcost’:20000},
{‘pid’:333,’pname’:’pthree’,’pcost’:30000},
{‘pid’:444,’pname’:’pfour’,’pcost’:40000}],
sub:’ReactJs’
}
}
render(){
return(
<div align=’center’>
<h1>{this.state.sub}</h1>
{this.state.products.map((element,index)=>(
<p>pid:{element.pid},pname:{element.pname}</p>))}
</div>
)
```

}

---

16) How to apply the styles in reactJS?

Four ways to style react components.

CSS Stylesheet:

- Simply import css file import ‘./DottedBox.css’ so you can have a separate css file for each component.

Inline styling:

- In React, inline styles are not specified as a string. Instead they are specified with an object whose key is the camelCased version of the style name, and whose value is the style’s value, usually a string.
- We can create a variable that stores style properties and then pass it to the element like style={nameOfvariable}
- We can also pass the styling directly style={{color: ‘pink’}}

CSS Modules:

- A CSS Module is a CSS file in which all class names and animation names are scoped locally by default.

Styled-components:

- Styled-components is a library for React and React Native that allows you to use component-level styles in your
- application that are written with a mixture of JavaScript and CSS

---

17) how to link the external templtes to Components ??

- Every component has a separate html file. However, in React, I see that render function itself includes the html template.

```
import React, { Component } from 'react';

class HelloWorld extends Component {
render() {
return (
<h2> Hello World </h2>
);
}
}

export default HelloWorld;
```

Well I want to take

```
<h2> Hello World </h2>
```

outside the js file and put it in a separate html and import the html file to render function, for example

```
render() {
return (
import content of helloworld.html
);
}
```

---

18) how to provide communication between Components ??

At the moment I'm using three components: <list />, < Filters /> and <TopBar />,
now obviously when I change settings in < Filters />
I want to trigger some method in <list /> to update my view.

<Filters /> is a child component of <List />
Both <Filters /> and <List /> are children of a parent component
<Filters /> and <List /> live in separate root components entirely.

Scenario – 1:

- You could pass a handler from <List /> to <Filters />, which could then be called on the onChange event to filter the list with the current value.

Scenario 2:

- Similar to scenario #1, but the parent component will be the one passing down the handler function to <Filters />, and will pass the filtered list to <List />. I like this method better since it decouples the <List /> from the <Filters />..

---

19) what is redux??

- Redux is an open-source JavaScript library for managing application state.
- It is most commonly used with libraries such as React or Angular for building user interfaces.
- Redux is a predictable state container for JavaScript applications. It helps you write applications
- that behave consistently, run in different environments (client, server, and native), and are easy to test.
- it helps you manage the data you display and how you respond to user actions

---

20) what is flux ??

- Flux is a JavaScript architecture or pattern for UI which runs on a unidirectional data flow and has a centralized dispatcher.
- It was created by Facebook, and complements React as view. Flux is a pattern and Redux is a library.
- In Flux, an action is a simple JavaScript object, and that's the default case in Redux too, but when using Redux middleware, actions can also be functions and promises.
- Flux is a pattern for managing how data flows through a React application.
- It is a method of working with React components is through passing data from one parent component to it's children components..

---

21) what is store ??

- A store is basically just a plain JavaScript object that allows components to share state.
- A store holds the whole state tree of your application.
- The only way to change the state inside it is to dispatch an action on it.

---

22) what is reducer ??

- The reducer is a pure function that takes the previous state and an action, and returns the next state.
- newState =(initialState,action)
- It “reduce” a collection of actions and an initial state (of the store) on which to perform these actions to get the resulting final state.

---

23) what is dispatch??

- dispatch is a function of the Redux store.
- dispatch() is the method used to dispatch actions and trigger state changes to the store.
- You call store.dispatch to dispatch an action. This is the only way to trigger a state change. By default, a connected component receives props.dispatch and can dispatch actions itself

---

24) what is subscribe ??

- It is a function of the Redux store.
- Every time the store changes.
- so it watches for changes and then tells react to redraw.
- subscribe() just lets you know every time the store changes..

---

25) explain state management in ReactJS ??

- React includes several ways of managing state in an application. State is a interface between your data and react.
- React projects break data up into two categories:
- State :read-write data that lives within a component
- Props :read-only data sent to a component, may be updated later by the sending component (a parent)
- The state can be changed by the component itself by calling set state which will trigger a re-render of the component. The state is managed in a container that also includes methods to work with a state object. A container looks and feels like any React component without the UI-part. Also, the setState function follows React's setState the only difference is, that unstated setState returns a Promise you can await.

---

26) how to maintain the immutablity in ReactJS ??

- We can use this.setState to maintain the immutability
- Immutable data management ultimately makes data handling safer.
- An immutable value or object cannot be changed, so every update creates new value, leaving the old one untouched.
- Debugging requires immutability in react.
- React component's API provides a setState method to make changes to the component internal state — but as the documentation makes clear, we have to be careful to always use the setState method and never manipulate this.state directly.

---

27) what is middleware in ReactJS ??

- Middleware is software that lies between an operating system and the applications running on it.
- In ReactJS the middleware sits in between the dispatch and reducers. We can alter our dispatched actions before they get to the reducers or execute some code during the dispatch.

---

28) what are the middlewares are available in ReactJS ??

- The famous middleware's in ReactJs are React-Thunk and React-Saga. The middleware sits in between the dispatch and reducers.

---

29) what is thunk ??

- Thunk is a middleware library in react. It supports for asynchronous data flow. Asynchronous Messaging means that, it is a one way communication and the flow of communication is one way only.
- Thunk created by using "promises" functions. It allows to write action creators that return a function instead of an action.

---

30) what is saga ??

- Thunk is a middleware library in react. It supports for asynchronous data flow. The reason that we need to use a middleware such as Redux Thunk is because the Redux store only supports synchronous data flow. Sagas created by using "generator" functions.

---

31) what are the differences between Saga and Thunk ??

- Thunk and Saga are middle ware of redux.
- Without middleware redux store supports only synchronous data flow.
- Sagas and Thunk responsible for asynchronous, especially for AJAX calls.

Redux saga :

- Sagas created by using "generator" functions.
- Sagas takes advantage of the "yield" keyword to halt execution with in a function.

- Easy to test
- It is useful to express complex application logic.

Redux thunk:

- Thunk created by using "promises" funtions.
- Promises simple to use.
- Thunk is a function that already has everything it needs to execute.
- Difficult to test.
- It is good for small use cases and for beginners.
- Thunk logic is all contained inside the function.

---

32) (how to provide the communication between reducers ??)
How to Combine reducers?

- combineRedcers useful for provide communication between reducers.

Ex: if we are taken 2 reducers

- reducerA and reducerB
- we can combine the two reducers in index.js like :
- const rootReducer = combineReducers({reducerA,reducerB});

---

33) what is combineReducer ??

- The combineRedcers helper functions. It is useful for combine reducers in various places to create root reducer.
- This function helps you organize your reducers to manage their own slices of state.
- In redux, there is just one store, but combineReducers helps you keep the same logic between reducers.

Ex: If your app grows more complex, if you want to split your reducing function into separate functions

- The combineReducers helper functions turns an object whose values are different reducing functions into a single reducing function you can pass to createStore.

---

34) what is connect in Redux ??

- This function connects a react component to redux store.
- Connect() is used to interact with the store.
- We can dispatch and subscribe the data by connect() function.

---

35) How to make store available globally ??

- By using provider in index.js we can make store available globally.
  The <Provider/> makes the redux store globally

ReactDOM.render(<Provider store =
{store}><App/></Provider>,document.getElementById('root'));

---

36) explain HOC in ReactJS ??

- "Higher Oreder Component" (HOC) is an Advanced technique in React for using Component logic.
- "Higher Order Component" is a function that takes a Component and returns a new Component.
- "Higher Order Component" are common in third-party React libraries,such as Redux.
- "HOC" is powerful tool based on which many libraries are gettting developed.
- "HOC" can help in simplifying and abstracting repeated logic in a React Applications.
- "HOCs" are functional implementations of javascript.
- "HOCs" has an advanced react pattern allows us the capability of reusing component logic.
- "HOC" is a pure function,it has no side effects.It only returns a new Component.

- In “Higher Order Component” we dont’t modify or mutate the component.we create new ones.
- “HOC” is an abstraction or middle layer gives us the freedom to make any kind of changes in the core logic of container components without affecting the UI Layer.
- “HOCs” is the Redux framework which is generally used for the state management in complex ReactJS projects.

---

(37) Explain life cycle hooks in Reactjs...?

(1) GETDEFAULTPROPS():

- used as defaults props to the component,if the component is not passed with props and is used with this.props

Syntax:

```
getDefaultProps:function(){

return{ Hello };

};
```

GETINITIALSTATE():

- thegetInitialState method enables to set the initialState value,that is accessible inside the component via this.state

Syntax:

```
getInitialState:function(){

return{ Hello } ;

};
```

COMPONENTWILLMOUNT():

- This hook has the ability to access the props(this.props) and state(this.state).But not the DOM.

componentWillMount:function(){

};

RENDER():

- When we call render evertime the state changes its value

i.e.,when the state value is modified by

this.setState({

counter:

})

syntax:

render:function(){

return JSX DOM

}

- Neither props nor state should be modified inside this function.
- render function has to be pure,meaning that the same results is returned every time the method is invoked.

COMPONENTDIDMOUNT():

- Called once, after trhe render method is called.
- This hook has the ability to aaccess the props(this.props) and state(this.state) and the DOM too.

Syntax:

componentDidMount:function(){

};

COMPONENTWILLRECEIVEPROPS():

- It will not be called on the initial render,but called in each subsequent renders.

Syntax:

componentWillReceiveProps:function(newProps){

};

SHOULDCOMPONENTUPDATE():

- This is always called before the render method and enables to define if a re-rendering is needed or can be skipped based on the boolean value it returns.

Syntax:

- shouldComponentUpdate:function(nextProps,nextState){

return true;

};

COMPONENTWILLUPDATE():

- ComponentWillUpdate gets called as soon as the shouldComponentUpdate returned true.
- Any state changes via this.setState are not allowed.
- Strictly used to prepare for an upcoming update,not trigger an update itself.

Syntax:

- componentWillUpdate:function(nextProps,nextState){

};

COMPONENTDIDUPDATE():

- componentDIDUpdate gets called as soon as the ShouldComponentUpdate returned true.
- Any State changes via this.setState are not allowed.
- Strictly used to prepare for an upcoming update not trigger an update itself.

Syntax:

componentDidUpdate:function(prevProps,prevState){

};

COMPONENTWILLUNMOUNT():

- called once. after the render method is called and before the component is removed from the DOM.
- used when needing to perform cleanup operations,i.e.,removing any timers defined in componentDidMount.

syntax:

componentWillUnMount:function(){

};

---

(38) Is render manditory in ReactJs....?

- Yes, ***render() function is manditory in ReactJs***.
- The render() method is the only required method in a class Component.
- The render() function ***should be pure***.
- Render() function does not modify component state.
- Render() function ***returns the same result*** each time it is invoked.
- render() function ***does not directly intearct with the browser***.

---

(39) What are pure Components in Reactjs...?

- Pure Components compare all the properties of the ***current state with the next state***,
  and ***current props with the next props***.
- Pure Components helps in reducing unnecessary render() method calls.
- A specific thing about pure Components is the ***shallow comparison***.

- Shallow comparison means that you compare the immediate contents of the objects instead of recursively comparing all the key/value pairs of the object.
- In pure Components javaScript completely based on Objects.
- Pure Components are ideal for classes with minimal and immutable properties.
- A component is said to be pure if it is guaranteed to return the same result given the same props and state.
- Class Component can be pure too as long as their props and state are immutable.
- Fuctional Component is a good example of a pure Component.
- ***React.PureComponent*** is used for ***optimizing performances***.

---

(40) what are the differences between **stateful** and **stateless** components in Reactjs....?

Stateful Components:

- Stateful Components are always ***class components***.
- Stateful components have a state that gets initialised in the constructor.
- Stateful Component ***can render both props and state***
- In stateful Components the props and state are rendered like **{this.probs.name}** and **{this.props.state}** respectively.
- In stateful Component render depends upon the value of the state.
- In Stateful class Component can be stateful or Stateless component
- Stateful presentational Component responsible for rendering form,gathering user input.
- stateful container component responsible for storing application data.

Stateless Components:

- we can use either function or class for creating ***stateless Components***.
- stateless Components are those components ***which dont have any state at all***.
- A Stateless Components ***can render props***.
- In Stateless Components the props are displayed like **{props.name}**
- Stateless function looks like a normal function with no render method.
- A stateless Component renders output which depends upon ***props value***.
- In Stateless Component Functional Component is always a stateless component.

- stateless presentational component responsible for iterating through the data to be rendered.

---

41) what is **functional component** in ReactJS ??

- it is ***simple javascript function*** and ***pass data by using props***
- it is ***use for ajax and API request.***

Collection of navigational component:

- And Enable navigation among view
- Router Componet(<BrowserRouter>)
- Routing matching component(<Route/> and <Switch/>)
- Navigational Component(<Link>) Collection of navigational component
- And Enable navigation among view
- Router Componet(<BrowserRouter>)
- Routing matching component(<Route/> and <Switch/>)
- Navigational Component(<Link>)

---

42) explain "**axios**" module in ReactJS ??

- Axios is a ***promise-based HTTP client*** that ***works in both the browser and in a node.js*** environment.
- Axios is a ***lightweight*** HTTP client based like ***a Fetch API***.
- Axios is ***promise-based async/await library*** for the readable asynchronous code.
- We can easily integrate with React.js, and it is effortless to use in any frontend framework

---

43) explain **Routing** in ReactJS ??

- React router is ***a routing library*** built on top of the react which is ***used to create the routing in react apps***.

- we are using the ***create-react-app*** to create the app.
  npx create-react-app routing
  cd routing
- To install the react router you need to download the react-router-dom package by running the following commands.
  npm install react-router-dom
  npm start //to run dev serverIf you navigate to the public/index.html you will see a single html file which looks like this.

```
<!DOCTYPE html>
<html lang=”en”>
<head>
<meta charset=”utf-8">
<meta name=”viewport” content=”width=device-width, initial-scale=1, shrink-to-fit=no”>
<meta name=”theme-color” content=”#000000">
<link rel=”manifest” href=”%PUBLIC_URL%/manifest.json”>
<link rel=”shortcut icon” href=”%PUBLIC_URL%/favicon.ico”>
<title>React App</title>
</head>
<body>
<div id=”root”></div>
</body>
</html>
```

Currently, in our app, there is only a single App component.

users.js:

```
import React from ‘react’
class Users extends React.Component {
render() {
return <h1>Users</h1>
}
}
export default Users
```

contact.js:

```
import React from 'react'
class Contact extends React.Component {
render() {
return <h1>Contact</h1>
}
}
export default Contact
```

app.js:

```
import React from 'react'
class App extends React.Component {
render() {
return (
<div>
<h1>Home</h1>
</div>
)
}
}
export default App
```

- Now our app has three components one is App and the other two are Users and Contact.

Routing:

- open the index.js file and import the three components (App,Users,Contact)

index.js:

- import React from 'react'
- import ReactDOM from 'react-dom'
- import './index.css'
- import App from './App'
- import Users from './users'
- import Contact from './contact'
- ReactDOM.render(<App />, document.getElementById('root'))

- React router gives us three components [Route,Link,BrowserRouter] which help us to implement the routing.

index.js:

- import React from ‘react’
- import ReactDOM from ‘react-dom’
- import { Route, Link, BrowserRouter as Router } from ‘react-router-dom’
- import ‘./index.css’
- import App from ‘./App’
- import Users from ‘./users’
- import Contact from ‘./contact’
- ReactDOM.render(<App />, document.getElementById(‘root’))
- In the Route component, we need to pass the two props
- path: it means we need to specify the path.
- component: which component user needs to see when they will navigate to that path..
- import React from ‘react’
- import ReactDOM from ‘react-dom’
- import ‘./index.css’
- import { Route, Link, BrowserRouter as Router } from ‘react-router-dom’
- import App from ‘./App’
- import Users from ‘./users’
- import Contact from ‘./contact’
- const routing = (
- <Router>
-
- <Route path=”/” component={App} />
- <Route path=”/users” component={Users} />
- <Route path=”/contact” component={Contact} />
-
- </Router>
-)
- ReactDOM.render(routing, document.getElementById(‘root’))
-
- Now if you enter manually localhost:3000/users you will see Users component is rendered.
-

- But still, Home component is also rendered in the screen this happens because of our home path is '/' and users path is '/users' slash is same in both paths so that it renders both components to stop this behavior we need to use the exact prop.

- import React from 'react'
- import ReactDOM from 'react-dom'
- import './index.css'
- import { Route, Link, BrowserRouter as Router } from 'react-router-dom'
- import App from './App'
- import Users from './users'
- import Contact from './contact'
- const routing = (
- <Router>
-
- <Route exact path="/" component={App} />
- <Route path="/users" component={Users} />
- <Route path="/contact" component={Contact} />
-
- </Router>
-)
- ReactDOM.render(routing, document.getElementById('root'))
- Adding Navigation using Link component

index.js:

- import React from 'react'
- import ReactDOM from 'react-dom'
- import './index.css'
- import { Route, Link, BrowserRouter as Router } from 'react-router-dom'
- import App from './App'
- import Users from './users'
- import Contact from './contact'

const routing = (

<Router>

```
<ul>
<li>
<Link to="/">Home</Link>
</li>
<li>
<Link to="/users">Users</Link>
</li>
<li>
<Link to="/contact">Contact</Link>
</li>
</ul>
<Route exact path="/" component={App} />
<Route path="/users" component={Users} />
<Route path="/contact" component={Contact} />
</div>
</Router>
)
```

ReactDOM.render(routing, document.getElementById('root'))

- After adding navigation, you will see the routes are rendered on the screen. if you click on the users, you will see url is changing and Users component is rendered.

---

44) how to ***read*** the ***Routing Parameters*** in Single Page Applications ??

- ***Route.match.param.variableName***
  (`https://api.github.com/users/${props.match.params.id}`)

---

45) Explain **MERN Development** ??

- MERN Means ***Combination of four Technology***

  client side ,server-side ,database and UI

- ***MondoDB***(database)
- ***Express***(For Api development)
- ***ReactJs*** (For UI)
- ***Nodejs***(For Server side)

---

46) How to Handle the **Form Validations** in ReactJS ??

Explore It's Proper Answer

- We will use create-react-app to get up and running quickly with a simple React app.

Install the package from npm and create a new app:

- $ npm install -g create-react-app
- $ create-react-app react-form-validation-demo

Now let us run the app:

- $ cd react-form-validation-demo/
- $ npm start

That opens http://localhost:3000/ where our new app is running.

Next, let us add bootstrap so that we can style our form easily:

- $ npm install react-bootstrap — save
- $ npm install bootstrap@3 — save

Import Bootstrap CSS and optionally Bootstrap theme CSS in the beginning of the src/index.js file:

- import 'bootstrap/dist/css/bootstrap.css.'
- import 'bootstrap/dist/css/bootstrap-theme.css'.
- Ok, now let us build the core of our demo app. Let us add a Form component.
- In src/App.js, let us replace the default intro text markup with a Form component that we're going to build. We also need to import it:

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
import Form from './Form.js';class App extends Component {
render() {
return (
<div className="App">
<div className="App-header">
<img src={logo} className="App-logo" alt="logo" />
<h2>React Form Validation Demo</h2>
</div>
<Form />
</div>
);
}
}

export default App;
```

Now, let's create that Form component in src/Form.js.

---

47) Explain **Yarn tool** importance in Reactjs ?

yarn tool is the ***Native tool given by facebook*** .

yarn tool used to ***download the libraries in faster manner in React***.

yarn is ***extremely fast*** when installing dependencies in project and NPM installs packages sequentially.

slows down the performance significantly yarn solves this problems installing these packages in parallel .

---

48)How to Create React Application?

Ans.Create React Application ***Mainly 8 Steps*** are there.

STEP 1:

- **Download And Install Node Js** .

STEP 2:

- **Download And Install Git** .

STEP 3:

- Install **yarn tool** by following Command
- **npm install -g yarn@latest**

STEP 4:

- Install "**create-react-app**" tool by following command
- **npm install -g create-react-app@latest**

STEP 5:

- Create the React Application by following Command
- **create-react-app filename**

STEP 6:

- Switch to React Application by following Command
- **cd filename**

STEP 7:

- Exicute the React Application Following Command
- **npm start (or) yarn start**

---

49) What is **NPX** ?

- NPX means ***Node Package eXecute***.
- NPX is a ***NPM package runner*** . its helps to execute packages without installing explicity. ***NPX makes it easy to install*** and ***mange dependencies*** hosted in NPM registry and its simplifies the process and provides a better for executables .

---

50) what is **create-react-app** ?

- “create-react-app” is the tool ***given by facebook developers***. and this ***tool used to create the React Applications*** and its **saves** you from **time** consuming setup configuration.
- We will install ” create-react-app ” by using following command
- ***npm install -g create-react-app@latest***

---

51) what is **babel** ??

- “Babel” is a ***free*** and ***open-source JavaScript transpiler***.
- “Babel” is popular tool for ***using the newest features of the JavaScript***.
- “Babel” can convert ***JSX syntax***.
- We can install “babel” using following command:

npm install –save-dev ‘babel-cli’.

- Babel enables us to write modern JavaScript that will be “transpiled” to supported ES5 JavaScript. We call this process transpiling.
- There are a few ways to use Babel in projects.
- The simple and fast way is to use the package “***babel-standalone***”.
- We can include it in a script tag using “***cdn***” like this:
- <script src=”https://cdnjs.cloudflare.com/ajax/libs/babel-standalone/6.18.1/babel.min.js”> </script>
- Babel will automatically transpile any script tags with type “text/babel” or “text/jsx”.

- **Example:**

```
<div id=”output”></div>

<script type=”text/babel”>
const getMessage = ()=> “Hello World”;
document.getElementById(‘output’).innerHTML = getMessage();
</script>
```

52: Is it possible to develop **React Applications by using TypeScript** ??

- **Yes**, it is possible to develop React Applications by using TypeScript.
- ***Typescript is used*** for making React apps ***more stable, readable and manageable***.
- To create a new app with Typescript use the following command:

yarn create react-app app_name –typescript (or) npx create-react-app app_name –typescript

- ***But TypeScript, requires much more time to compile as compare with JavaScript***.

---

53) what is **super()** ??

- “super” keyword is ***used as*** a “***function***” which ***calls the parent class***.

- The super() in javascript is used to ***call the methods of the parent class***.
- It is used within a constructor function to call the parent constructor function.

---

54) **Advantages and disadvantages** of reactjs ??

Advantages:

- ***Virtual DOM*** in ReactJS ***makes user experience better*** and ***developer's work faster***.
- React ***components*** can be ***re-used easily throughout the application***.
- ***JSX makes components/blocks code read-able.***
- ReactJS ***improves performance*** due to ***virtual DOM***.
- ReactJS applications are ***easy to test.***
- Benefit of Having JavaScript Library because ***JavaScript library provides more flexibility***.
- JSX is an optional syntax extension to JavaScript that makes writing our own components much easier.
- It is SEO(Search Engine Optimization) friendly.

Disadvantages :

- ***React technologies updating so fast*** that there is ***no time to make proper documentation.***
- ReactJS ***Covers only the UI Layers*** of the app and ***nothing else***.
- ***JSX***(JavaScript extension) ***as a barrier*** : React uses JSX, HTML with JavaScript mixed together. ***For new Developers, JSX is a barrier***.
- ***ReactJS is not for small scale projects.***
- The ***frequent updates*** of react js is ***a major cause of frustration*** among ***developer***.

Q 55) what are **uses** of reactJS ??

- ReactJS is an ***open-source JavaScript library*** that is used ***for building UI(User Interface)*** specifically for ***single-page applications***.

S**implicity:**

- ReactJS is very ***simple to learn***.

- React having ***component-based approach***,
- Well-defined lifecycle and React ***uses plain JavaScript that makes React very simple to learn and build a professional web page.***
- React can be ***used to create mobile applications*** with the help of ***"React Native"***.

Testability:

ReactJS ***applications are easy to test***.

- ***Increases productivity*** and ***helps in maintenance***.
- ReactJS **uses** ***Downward Data Flow***. If we made any small changes in child structure it won't affect the parents. Technically we call it "***Code stability***".
- Another important uses of React JS is a ***user-friendly development platform***.

---

56) what is **DOM** ??

- DOM stands for "***Document Object Model".***
- DOM ***represents*** the ***documents(HTML or XML***) and it can be ***modified with a scripting language*** such as "***javascript***".
- DOM is a ***programming interface*** for "**HTML**" and "**XML**" ***documents***.
- In simple words we can say that DOM is a ***programming API for documents***.
- DOM defines a ***standard for accessing documents***.
- It represents the ***web pages in a structured hierarchical***.
- With the help of DOM, a ***programmer can create and bulid documents***, navigate their structure and add, modify, or delete elements and content.

---

57) what are the ***differences*** between **DOM** & **Virtual DOM** ??

DOM	**Virtual DOM**
• It updates ***slow***.	1. It updates ***faster***.
• DOM ***can directly update HTML***.	2. ***Can't directly update HTML***.
• ***Creates a new DOM*** if element updates.	3. ***Updates the JSX if element***
updates.	

- DOM manipulation is ***very expensive***.
- ***Too much of memory wastage***.

4. DOM manipulation is ***very easy***.
5. ***No memory wastage***.

---

58) what is **shadow DOM** ??

- Way of encapsulating the implementation of web components is known as "Shadow DOM".
- ***We can hide the implementation details of web component*** by using "Shadow DOM".

Example:

- HTML5 slider input, while the regular DOM recognizes this as a simple <input/> tag, there is underlying HTML and CSS that make slide feature.
- The sub-tree of DOM nodes is hidden from the main DOM to encapsulate the implementation of HTML5 slider.
- Shadow DOM provides isolated scope for web components.
- "Isolate scope" make the components reusable and permit to control the binding.

59) explain **events in ReactJS** ??

- React events are named using camelCase, rather than lowercase.

Example:

```
<button onClick={activateLasers}>
Activate Lasers
</button>
```

- **Component method is called when an event occurs**.

---

60) explain **ref** ??

- The “ref” is used to ***return a reference to the element***.
- “ref” is used ***when we want to add methods to the components***.
- “ref” ***act as an id***.

Example:

```
<input value = {this.state.data}

onChange = {this.updateState}

ref = “myInput”>

</input>
```

- If “ref” points to a “standard component” (DOM node, such as input, select, div etc) then to retrieve the element; we just need to call “this.refs.ref”.
- If “ref” points to a “composite component” (a custom component we have created ourself) we need to use the new ReactDOM module like “ReactDOM.findDOMNode(this.refs.ref)”.

---

61) how to **pass the data between components with out props** ?? without Props ?

- **No**, ***we can not send data*** between two components without using props.

---

62) **selector in redux** ??

- “**Selector**” is ***simply a function*** that ***accept Redux “state***” as an argument and return that data which is derived from “state”.
- “Selector” is very ***efficient***.
- Selector is ***a middleman*** between ***components and state***.
- This middleman will ***be a function that access state directly***.

Example:
Suppose we save a property called firstName and lastName in our state. We want to display the full name in our component.

- So instead of pulling “firstName” and “lastName” to our component with the help of connect, we can just create a “selector”.
- This “selector” will just join firstName and lastName and return the full name.

---

63) explain **closures** ??

- ***Any inner function that accesses the outer function*** members as well as “Lexical scope” members called as “***closures***”.
- In JavaScript, closures are ***created every time a function is created***, at function creation time.
- closures are the ***primary mechanism used to enable data privacy***.
- In functional programming, closures are frequently ***used*** for ***“partial application” & “currying”.***

Example:

```
<script>

let fun_one = function(){

let i = 10;

let j = 20;

return function(){

console.log(i);

console.log(j);

}

};
```

consol.dir(fun_one()); //closure(fun_one){i:10, j:20}
</script>

The closure has access to the variable in three scopes:

- Variable declared in his ***own scope***
- Variable declared in ***parent function scope***
- Variable declared in the ***global namespace***.

---

64) In how many **ways can we create functions** ??

In JavaScript, a function can be declared using several ways:

Function declaration:

A function declaration is made of function keyword, followed by function name.

Example: function isEven(num) {

return num % 2 === 0;
}

Function expression: A function expression is determined by a function keyword, followed by an optional function name.

const myFunctionVar = function(variable) {

return typeof variable;
}

Shorthand method definition: Shorthand method definition can be used in a method declaration.

Example:

const collection = {

```
items: [],

add(...items) {

this.items.push(...items);
},

get(index) {

return this.items[index];
}
};

collection.add('C', 'Java', 'PHP');

collection.get(1); //'Java'
```

Arrow function: An arrow function is defined using parenthesis that contains the list of parameters followed by arrow symbol(=>).

– When arrow function has only "one parameter" parenthesis can be omitted.

– When it contains a single statement, the curly{} braces can be omitted too.

Example:

```
const absValue = (number) => {

if (number < 0) {

return -number;
}

return number;
};

absValue(-10); //10

absValue(5); //5
```

Generator function:

The generator function in JavaScript returns a Generator object.

– The generator function can be declared in the following forms:

a.function* <name>():

Example:

```
function* indexGenerator(){

var index = 0;

while(true) {

yield index++;
}
}

const g = indexGenerator();

console.log(g.next().value); //0

console.log(g.next().value); //1
```

b. function* ():

Example:

```
const indexGenerator = function* () {

let index = 0;

while(true) {

yield index++;
}
};

const g = indexGenerator();

console.log(g.next().value); //0
```

```
console.log(g.next().value); //1
```

c. *<name>():

Example:

```
const obj = {

*indexGenerator() {

var index = 0;

while(true) {

yield index++;
}
}
}

const g = obj.indexGenerator();

console.log(g.next().value); //0

console.log(g.next().value); //1
```

---

65) what are the **advantages of arrow function**??

- Arrow functions were ***introduced in ES6***.
- Arrow functions allow us to ***write shorter function syntax***.
- If function has only one statement and the statement returns a value, we can remove the brackets and the return keyword.
- ***If we have only one parameter***, ***we can skip the parentheses as well***.
- No ***binding of this keyword***.
- Arrow functions are ***more secured as compared to normal function***.
- Arrow function ***utilizes the heap memory***.
- We can use arrow function as a ***"call back"*** function.

---

66) what is **memo** ??

- "memo" is ***like pure components***.
- It will help ***us to control when our components re-render***.
- ***React.memo()*** works with ***functional components***.
- ***React.memo()*** is used to ***wrap a functional component***.

To import "memo" use following code:

```
import { memo } from "react";
```

React.memo() improves the performance.

Example:

```
export function Movie({ title, releaseDate }) {
return(
<div>
<div> Movie Title:{title} </div>
<div> Release Date:{releaseDate} </div>
</div>);
}
export const MemoizedMovie = React.memo(Movie);
```

---

67) Explain **Hooks** in ReactJS ??

1. React "Hooks" are a ***way to add React.Component features to functional components***. Features like:

- ***State***
- ***Lifecycle***

2. Hooks use ***React's features without classes***.

3. React ***provides a few build-in Hooks*** like ***useState, useEffect*** etc.

4. Hooks are a new addition in ***React 16.8.***

68) What is **Synthetic event** in ReactJS ??

- “Synthetic Events” is a ***cross-browser wrapper arround the browser’s native event.***
- React ***implements a “synthetic event” system*** that brings consistency and high performance to react applications and interfaces.