ReactJs Documentation

What is ReactJs?

ReactJs is an open-source frontend Javascript Library.

Used to build 'User Interfaces' using 'UI Components' and to make it an 'Single Page Applications'.

It is maintained by Meta and a community of individual developers and companies.

React can be used as a base in the development of SPA.

MVC: Model 'View' Controller

It is used for handling the 'view' layer for 'Web' and 'Mobile' (React Native)

How React works?

React is a Declarative (<User />), Efficient and Flexible JS Library for building UI's.

It's 'V' a view layer in MVC.

Comparing to JS, how React is differing?

Js: DOM (Document Object Model)

React: VDOM (Virtual DOM)

Comparing the Library and Framework?

Inversion of Control

Packages – Modules – Functions

Setup an Env., and Creating a new project in React:

Node –version

Npm –version

Npm install -g create-react-app

Npm create-react-app <yourappName> (or)

Npx create-react-app <yourappName> (or)

Create-react-app <yourappName>

We even added the Htmls, Css (External & Inline), Declared the Variables and set the Properties into that, worked with JS's expr.,

Created the sample reactjs webpage(s) using html & css

= Created the sample reactjs webpage(s) using html & css & bootstrap

Segregating the Page & its corresponding Components

Implemented the Routing Concept

Added the Bootstrap

Started with Dynamic Page: ex., Products.,

Listing out of all products,

Added the Properties at Individual Component Level and accessed into the Def., of the Component

Later, Created the 'state' object, and maintained all the required properties and accessed into the return statement with its indexing

Event Handling:

Created the Toggling Effect,

Updated the Records through Button & ListItem

Deleted the Records

Mapping

One Way: Displaying the Value Directly from the Given Property

Two Way Property Binding: User Interaction, and that value can be manage

Forms & Validation using Regular Expression

Ref’s

RestApi

TodoApp

Cont., with TodoApp

Class Based Life Cycle Hooks – RestApi – TodoApp – Fetch, Display, Delete, Update, Add

Functional Life Cycle Hooks – RestApi – Photos

Cont., Functional Life Cycle Hooks – RestApi – Photos

Styled Components

ContextApi

What is Class Components?

```
Class className extends Component {
        // write the code
}
```

Ex., class User extends Component { // .... }

What is Functional Components?

```
Function functionName() {
        // write the code
}
```

Ex., function User() { // ... }

(or)

Arrow Functional Components

Let User = () => { // .... }

Life Cycle Hooks?

Irrespective of any application, we can have 3 stages or phases: 1. Initialize, 2. Update and 3. Destroy

What is Class Based Life Cycle Hooks?

Methods: componentDidMount(), componentDidUpdate(), componentWillUnMount()

Fetching the Data, Accessing the Data, Adding the New Item, Deleting the Existing Item, Updating the Item

What is Functional Based Life Cycle Hooks?

Methods: useState(), useEffect(), useContext(), useReducer(), useCallBack() etc.,

In Arrow Functional:

```
Let someName = (props) => {
        {props.propertName}
}
```

In Class:

{this.props.propertyName}

Concept: fetching the photos from resource url, and search functionality - Functional Life Cycle Hooks:

useState(null) or useState(“”) or useState([]) or useState({})

Styled Components: applying the predefined html selectors with styles to your custom components;

Context API: it is a way to effectively produce the global variables that can pass around globally. The alternative is ‘prop drilling’ – moving prop(s) from one node to another node or from grandparent to child(s) and so on

Material UI: CRUD Operations

Material UI: Page with Menu & Gallery ShowCase

Redux with Saga:

Redux: it is an open source javascript library for managing app., state.

Redux architecture:

1. Ui components: the ui components are pages with actions parts ex., buttons, images, etc., - where user can take an action of click, hover, drag, focus, change etc.,
2. Actions creators: are responsible for dispatching the action events
3. Reducers: receive the dispatched events and updates the state of the data store
4. Store: is responsible for storing the data associated with the app

Saga: is a redux – middleware library – which fetch the data from the network requests

We are going to the concepts of webpack & babel

Webpack: is a bundle of assets (htmls, styles,, images, files etc.,)

Babel: is a tool which compiles the .jsx code and converts into simple standard .js code format

Shopping Cart – Increment / Decrement – Add to Cart: We will plan to display the list of items with options of ‘addtocart’, when user clicks on it, the item should place in ‘cart layout’

Functional Life Cycle Hooks – Context & Redux – Shopping Cart – Adding the Product to Cart – Filters: Search, Price, etc.,

Deployment of ReactJs App ex., Github, Netlify, Heroku etc.,

Conditional Rendering: There are four main conditional statements & operators that are use:

If, if – else, inline logical && operator, inline ternary operator

Protected Routing: withRouter, redirect – react-router-dom (4v)

Add / Remove Multiple Input Fields – Dynamically

Lists – map(), filter(), -

arrayElement – it accepts the value of an array object,

arrayIndex – it accepts the index of an array object,

arrayObject – the accepts the whole array object

ex., **arrayObject.map(function(arrayElement, arrayIndex, arrayObject) {}, thisValue);**

map():

1. It is used to create a new array with each element of another array
2. Does not work for an empty array
3. Does not change the original array
4. Can handle large & complex data of an array & array object

Array.map(callback)

Callback: it is a function which is given in another function as a parameter

Example1: let developerName = [“himanshu”, “zavid”, “junhi”];

```
Function developer() {
        let developerName = [“himanshu”, “zavid”, “junhi”];
        let developerNameList = developerName.map((fullName) => {return
<li>{fullName}</li>});
        return  <ul>{developerNameList}</li>
}
```

Example2: let arr = [45, 76, 23, 45, 95, 68];

```
Function newNum() {
        let arr = [45, 76, 23, 45, 95, 68];
        let newArr = arr.map((element) => {
                return <li>{element * 10}</li>
        });
        return <ul>{newArr}</ul>
}
```

Lists of an Array:

```
Function NumList() {
        Let arr = [10, 50, 60, 234];
        Let newArr = arr.map((num) => {
                Return <li>{num}</li>
        });
        Return <ul>{num}</ul>
}
```

Lists of an Object

```
Function Developer() {
        Let developerInfo = {
                fullName: “john paul”,
                profile: “sr., dev”,
                age: 25,
                city: “Banglore”
        };

        Let developerList =
                Object.keys(developerInfo).map(key => (
                        <p>{developerInfo[key]}</p>
                ));
        Return <div>{developerList}</div>
}


Multiple Records:

Function Developer() {

        Let developerInfo = [

                { fullName: “”, profile: “”, age: },

                { fullName: “”, profile: “”, age: },

                { fullName: “”, profile: “”, age: },

                { fullName: “”, profile: “”, age: }

        ];


        Let developerList = Object.map(data => (

                <tr>

                        <td>{data.fullName}</td>

                        <td>{data.profile}</td>

                        <td>{data.age}</td>

                </tr>

        ));


        Return <table>{developerList}</table>

}
```

Create Accordion: