***React JS BasicConcepts:***

***CLASS Components in React:***

***SNIPPET:***

*import { Component } from 'react';*

*class App extends Component {*

*render(){*

*return (*

*<div className="App">*

*Rohit*

*</div>*

*);*

*}*

*}*

*export default App;*

***STATE in Javascript.***

***SNIPPET:***

*class App extends Component {*

*constructor(){*

*super()* ***// Calls the constructor of parent/Component class.***

*this.state={ /****/ state to be used for creating dynamic content.***

*'name':'Rohit',*

*}*

*}*

*render(){*

*return (*

*<div className="App">*

*Hello I am {this.state.name}*

*</div>*

*);*

*}*

*}*

***Changing state variable in react***

***SNIPPET:***

*class App extends Component {*

*constructor(){*

*super()*

*this.state={*

*'name':'Rohit',*

*}*

*}*

*render(){*

*return (*

*<div className="App">*

*Hello I am {this.state.name}*

*<button onClick={this.state.name='Ghost'}>Change</button>*

***// This is not going to work as we reference to different memory location here so***

***react does not re-renders its DOM here.***

*<button onClick={()=>{this.setState({'name':'Ghost'})}}>Change name</button>*

***// This is going to work because here by using setstate we are shallow merging to the***

***states memory by* *checking the same keys that state has and update it eventually.***

*</div>*

*);*

*}*

*}*

***Q. Why console.log() prints the old state even after updating the state??***

*ans.*

***#1SNIPPET:***

*<button onClick={()=>{*

*this.setState({'name':'Ghost'});*

*// console.log(this.state.name);****// Prints Rohit the old value because this setState is a***

***Asynchronous function so and console.log() is***

***synchronous by nature.So, it takes time to perform***

***operations and allows other stuffs that is synchronous***

***to execute in meantime****.*

*}*

*}>Change name</button>*

***#2SNIPPET****:* ***If we want to see the console.log correctly with change in state.***

*<button onClick={()=>{*

*this.setState(*

*()=>{*

*return{'name':'Ghost'}*

*},*

*()=>{*

*console.log(this.state.name);*

*}*

*);*

*}}>Change name</button>*

***MAPS in React:***

***Maps in Javascript iterate over each element of an array.***

***SNIPPET:***

*class App extends Component {*

*constructor(){*

*super()*

*this.state={*

*'monsters':[*

*{*

*name:'Ron',*

*id:'1'*

*},*

*{*

*name:'vlood',*

*id:'2'*

*},*

*{*

*name:'rainy',*

*id:'3'*

*}*

*]*

*}*

*}*

*render(){*

*return (*

*<div className="App">*

*{*

*this.state.monsters.map((element)=>{*

*return(*

*<div key={element.id}>*

*<h1>{element.name}</h1>*

*</div>*

*)*

*})*

*}*

*</div>*

*);*

*}*

*}*

***Key atttribute: It is needed to be added in top level html attribute as it helps the react to***

***identify,update and renrender that particular state when changes are***

***applied to it. Basically ids are only kept as keyy attributes.***

***LifeCycle methods of React: ComponentDIdMount().***

1. *When react renders a component on to a page that is called as Mounting.Happens only once in a lifecycle.*

*2. Used to make API requests.*

1. *SNIPPET:*

*componentDidMount(){*

*fetch('https://jsonplaceholder.typicode.com/users')*

*.then(response=>response.json())*

*.then((users)=>{*

*this.setState({'monsters':users});*

*})*

*}*

***Flow of control in react.***

*1. Constructor runs and sets the default state.*

*2. The the render() runs and render the component on to the page.*

*3. ComponentDidMount() runs after the component mounts/renders for first time and if it has got*

*any Api request that changes the the states it goes along and completes its task.*

*4. As soon as the state changes the component rerenders itself.*

***Filter() in react:***

*Used to filter the array of elements and create a new array.*

***SNIPPET:***

*render() {*

*const filtermonster = this.state.monsters.filter((monsterelement)=>{*

***// Filter Takes an array and filters out the elements based on the return condition to***

***create a new array.***

*return monsterelement.name.toLowerCase().includes(this.state.search\_monsters);*

***// includes function here checks whether the given string/character is a substring of***

***string on which we have applied our include().***

*})*

*return (*

*<div className="App">*

*<input*

*type="text"*

*placeholder="search monsters"*

*onChange={(event) => {*

***//Here OnChange returns the value wriiten inside the input box by***

***event.target.value.***

*this.setState(()=>{*

*return {search\_monsters:event.target.value.toLowerCase()};*

***//toLowerCase() changes all characters of a string to lowercase****.*

*})*

*}}*

***//Note: Whenever we need to update an array we need to update the entire array***

***else the changes won't get rendered on our page.***

*/>*

*{filtermonster.map((element) => {*

*return (*

*<div key={element.id}>*

*<h1>{element.name}</h1>*

*</div>*

*);*

*})}*

*</div>*

*);*

*}*

***Performance Optimisation in React :***

***SNIPPET:***

*class App extends Component {*

*constructor() {*

*super();*

*this.state = {*

*monsters: [],*

*search\_monsters:''*

*};*

*}*

*componentDidMount() {*

*fetch("https://jsonplaceholder.typicode.com/users")*

*.then((response) => response.json())*

*.then((users) => {*

*this.setState({ monsters: users });*

*});*

*}*

*onSearchChange=(event) => {*

***// SearchChange() we have removed out from the htmp templates and placed it inside the class in order to optimise our code, as with the re-rendering of the components this function was getting initialised again again which reduces the performance of our code but, by putting this inside the class this function only gets intialised once, thereby incresing the performance of our app.***

*this.setState(()=>{*

*return {search\_monsters:event.target.value.toLowerCase()};*

*})*

*}*

*render() {*

***//Destructing of the class state and function is done here in order to increase the readibility of our code.***

***const {monsters,search\_monsters} = this.state;***

***const {onSearchChange} = this;***

*const filtermonster = monsters.filter((monsterelement)=>{*

*return monsterelement.name.toLowerCase().includes(search\_monsters);*

*})*

*return (*

*<div className="App">*

*<input*

*type="text"*

*placeholder="search monsters"*

*onChange={(event)=>{onSearchChange(event)}}*

*/>*

*{filtermonster.map((element) => {*

*return (*

*<div key={element.id}>*

*<h1>{element.name}</h1>*

*</div>*

*);*

*})}*

*</div>*

*);*

*}*

*}*

***Q. How Components Lie inside React Project:***

* ***Inside Src ->*** *create a components folder****.***
* ***Inside components folder ->*** *Make the component folders and keep the code of each component inside their respective folders.*

***Props in React:***

*Props is an object of properties/values passed from parent component to the child*

*component.*

***SNIPPET:***

***Parent Component***

*class App extends Component {*

*constructor() {*

*super();*

*this.state = {*

*monsters: [],*

*search\_monsters:''*

*};*

*}*

*componentDidMount() {*

*fetch("https://jsonplaceholder.typicode.com/users")*

*.then((response) => response.json())*

*.then((users) => {*

*this.setState({ monsters: users });*

*});*

*}*

*onSearchChange=(event) => {*

*this.setState(()=>{*

*return {search\_monsters:event.target.value.toLowerCase()};*

*})*

*}*

*render() {*

*const {monsters,search\_monsters} = this.state;*

*const {onSearchChange} = this;*

*const filtermonster = monsters.filter((monsterelement)=>{*

*return monsterelement.name.toLowerCase().includes(search\_monsters);*

*})*

*return (*

*<div className="App">*

*<input*

*type="text"*

*placeholder="search monsters"*

*onChange={(event)=>{onSearchChange(event)}}*

*/>*

*<CardList monsters={filtermonster}/>*

***{/\* monster here is passed to the component as a Prop. Props is an object of***

***properties/values passed from parent component to the child***

***component.  \*/}***

*</div>*

*);*

*}*

*}*

*export default App;*

***Child component:***

*class CardList extends Component{*

***// Always keep the Customized components name in Upper Camelcase convention. It becomes easier for react to identify that the very component is made by the developer itself.***

*render(){*

*const {monsters} = this.props* ***//Destructuring of PROPs.***

*return(*

*<div>*

*{*

*monsters.map((element)=>{*

*return(*

*<div key={element.id}>*

*<h1>{element.name}</h1>*

*</div>*

*)*

*})*

*}*

*</div>*

*)*

*}*

*}*

*export default CardList;* ***//Used to export our components.***

***Q. How Components re-renders in react:***

***It is based on 2 conditions:***

* ***Either our state changes***
* ***Or, our props changes.***

***Flow of Components in React :*** *TOP-Down -> From Parent to Child component.*

***CSS in React:***

*Can be used by creating a .css file and writing css by targeting classnames or ID.*

***NOTE: CSS file we import in 1 component of our react is accessible to all the components present inside our Tree of components. This sometimes leads to overlapping of styles because of same classname.***

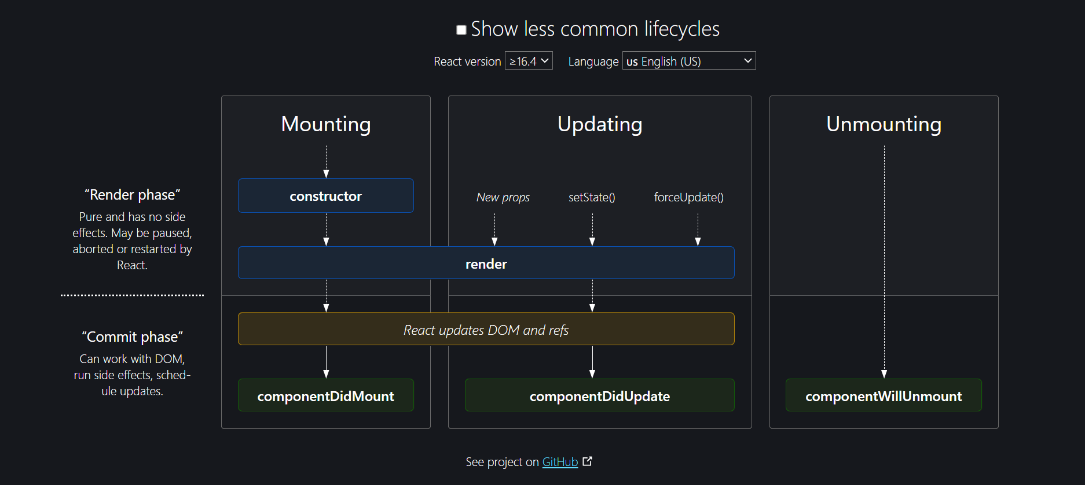
***String Interpolation:***

***Example: `Rohit is a ${this.rohit.personality}`***

*In**String interpolation we can write dynamic variables along with string using*

*backticks.*

***Functional VS Class Based Components:***



***Functional component donot have this lifecycle methods Like componentDidMount(), ComponentDidUpdate() and ComponentWillUnmount() as class components .****But it follows this 3 lifecycles of React given in diagram, i.e.*

***1. Mounting***

***2. Updating***

***3. Unmounting***

***NOTE:***

***ComponentDidUpdate():*** *This method runs when our react component re-renders/updates*

*due to change in state/props/use of forceUpdate() method.*

***ComponentWillUnmount():*** *This method runs at end(i.e. before the component is destroyed or*

*unmounted from the DOM tree.) to perform clean-ups which prevents any*

*type of memory leaks.*

***Functional Components in React:***

* *It does not have any lifecyle like class components.*
* *React display the component as it is from Top to Bottom.*
* ***SNIPPET:***

*const Text = ()=>{*

*return(*

*<h1 className="app-title">Monsters Rolodex</h1>*

*)*

*}*

***Pure and Impure functions in React:***

* ***Pure functions:*** *It return the same output if we use the same input parameters as it only uses the*

*arguments that lies within its scope.*

* ***Impure functions:*** *It give different outcomes when we pass the same arguments multiple times as*

*they use some parameters/variables which lies outside their scope.*