***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.*