**REACT NOTES**

**REACT => JavaScript library for creating front-end projects, its implementation allows a faster and more performant application load.**

**--create React App commands--**

**\* npx create-react-app “name”**

**\* npm init react-app “name”**

**D.O.M => Object interface interpreted by HTML, useful for accessing and manipulating elements.**

**Virtual D.O.M => Light rendering of D.O.M used by react for comparing and modifying specific elements if required.**

**Diffing => Comparison between real and virtual D.O.M (Document Object Model).**

**Reconciliation => Update of an specific component in the real D.O.M.**

**Component => native JavaScript reusable function executed when required to display an isolated content. We could refer to it as a piece of our entire user interface.**

**Composition => Passing elements or components as props to other components, thus giving us the ability of creating new components with the other components. (props.children)**

**In React, App is a container, or logical component, used for providing data and invoking actions; and visual components are called dumb.**

**JSX => JavaScript syntax extension that produces React elements. It is similar to HTML, but with some important differences.**

**--In JSX you shall—**

**1. <img …/>**

**2. <div>{name}</div>**

**3. USE <ul>{names.(map, filter, reduce)}</ul> NOT <ul>{names.(for, forEach)}</ul>**

**4. <div className=”…”></div>**

**5. <label htmlFor=”…”></label>**

**6. <div style={{color:”…”, border:”…”}}></div>**

**REACT ecosystem => Set of tools, extras and apps that allow React to work as a framework.**

**WEBPACK => Is a module bundler used for optimizing load times, it receives certain input files and returns other output files through components called “loaders”.**

**BABEL => Is a compiler that translates JSX into understandable JavaScript for both modern and older browsers.**

**Stateless component => They implement a return with an “HTML” like structure.**

**Props => Internal data of a component, they allow for a variable component info so we can have dynamic and reusable HTML structures.**

**--Insert props as component argument—**

**Function Table(props) {**

**}**

**Key prop => Unique identifier when iterating with map so react can follow each instance of the displayed elements.**

**Proptypes => Is a validator that checks for the correct datatype from a value passed in props.**

**defaultProps => It sets a default value in case a props attribute is undefined.**

**Stateful components => They allow for modifying their internal info from events and interactions.**

**--for using them you shall—**

**Import React, { Component } from ‘react’;**

**class Xxx extends Component {**

**constructor (props){**

**super(props);**

**this.state={**

**key: value**

**}**

**}**

**increase() {**

**This.setState({**

**key: this.state.key + 1**

**});**

**}**

**Render(){**

**Return(**

**<button onClick={ () => this.increase() }>**

**)**

**}**

**}**

**COMPONENT LYFECYCLE(Stateful components):**

**componentDidMount => Sets what to do after the first rendering of the component.**

**componentDidUpdate => Sets what to do after a change in component state.**

**componentWillUnmount => Sets what to do before destroying a component.**

**BrowserRouter => Component in charge of nesting all app to implement the router.**

**Link => Defines the route to go.**

**Route => Defines what to render in certain route.**

**Switch => It allows a dynamic rendering of components, rendering the first component that matches certain path, and correctly displaying a 404 component when there is no match.**

**HOOKS:**

**useState() => Is a function that manages the state of a component.**

**--How to use it?—**

**Const [“variable”, “setVariable”]= useState(“initState”)**

**setState({**

**…initState,**

**“newValues”**

**})**

**useEffect() => Is a function that manages the lifecycle of a component.**

**DidMount => useEffect(()=> {action code}, [])**

**DidUpdate => useEffect(()=> {action code}, [state])**

**WillUnmount => useEffect(return ()=> {action code}, [])**

**useRef() => Is a function that manages a D.O.M element.**

**--How to use it?—**

**Const variable = useRef();**

**Function xxx () {**

**variable.current.style.color = “gray”**

**}**

**<div ref={variable}></div>**

**Binding => Is to set the “this” context in a class component so we can then manipulate its state from the methods.**

**Controlled component => Is an input form element whose value is controlled by React component state.**

**Lifting state up => Sharing state of different components by moving it to their closest common ancestor.**

**Debouncing => Is a practice in which we set a specific delay in a callback, so the function doesn’t execute too often, like in the case we needed to listen the search bar typed characters.**

**React Profiler => Measures how often a React application renders and what the cost of rendering is. Its purpose is to help optimizing performance.**

**<Profiler id=”xxx” onRender={callback}>**

**<></>**

**<Profiler />**

**onRenderCallback Monitors:**

* **Profiler id.**
* **Tree’s phase.**
* **Actual duration of tree´s rendering.**
* **Base duration of each individual component most recent rendering.**
* **Start time of the current update.**
* **Commit time of the current update.**
* **Interactions of the update.**

**Memoization => Optimization technique for accelerating the app by caching the results of heavy function calls and returning them when similar inputs are encountered repeatedly.**

**Jest => Is a widely used third party framework for testing, its objective is to set different expectations to be met by our code. We may have:**

**Unit Testing => It tests a specific block of code functionality.**

**Integration Testing => It tests an entire software module or combined functions as group.**

**End to End Testing => This involves testing the entire software or major parts of it.**