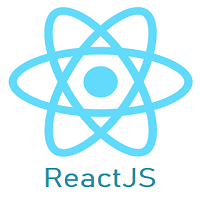
REACT JS

**AGANDA :**

* React JS
* Introduction to React JS
* Virtual Dom
* Jsx
* Component
* Functional & Class Components
* Component Lifecycle
* Conditional Rendering
* ES6 Basics For ReactJs
* Higher-Order Components
* React Context
* Next JS
* Introduction to Next JS
* Pre-rendering
* Static Generation
* Server-side Rendering
* Routing
* Redux
* Introduction to Redux

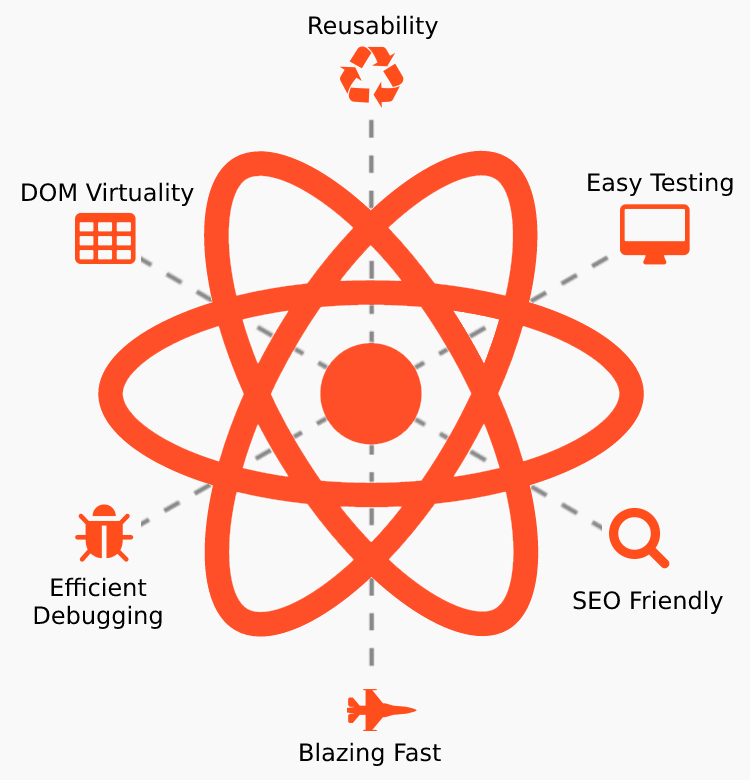
**WHAT IS REACT.JS?**

* React.js is a JavaScript library user interfaces.
* React.js can be used to create SPA and Mobile applications.
* Not a Framework it is just a library.
* React is often used in conjunction with other libraries such as Redux or Flex.



**WHY REACT.JS?**

* Component-based Reusable component.
* Unidirectional Data flow.
* Not a templates to create.
* Native.
* Easy to learn.



**REACT VIRTUAL DOM**

* Virtual DOM is in-memory representation of Real DOM it is light weight java Script object which is copy of Real DOM.
* Updating the virtual DOM in Reactjs is faster because Reactjs uses
* Virtual DOM gets updated, react compares the two virtual DOMs and gets to know about which virtual DOM objects were updated.
* React renders only those objects inside the real DOM instead of rendering the complete real DOM.

**JSX**

* JSX stands for JavaScript XML
* It allows us to write HTML inside JavaScript and place them in the DOM without using functions like appendChild( ) or createElement( )
* JSX provides syntactic sugar for React.createElement( ) function.

**REACT COMPONENT**

* Component let you split the UI into independent, Reusable pieces and think about each piece in isolation .
* JavaScript Function.
* Accept any types and any numbers of properties “props”
* React element describing what should appear on the screen.
* Two way of define a component Functional Component Class component

**FUNCTIONAL AND CLASS COMPONENTS**

**FUNCTIONAL COMPONENTS**

* Receive parameter (props) -Optional.
* Stateless or dumb Component.
* Just plain old javaScript function.
* Shorter to write.
* For UI components

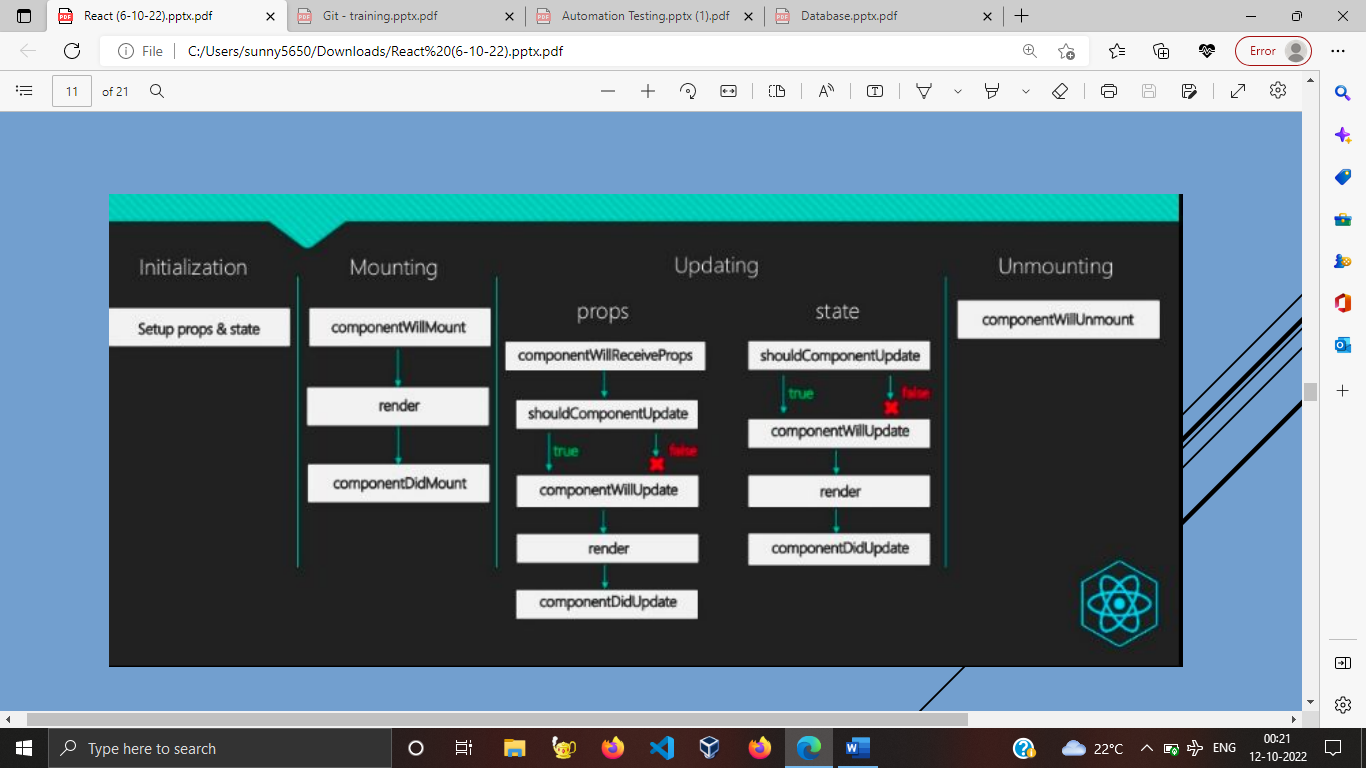
**CLASS COMPONENT**

* Has local state.
* Receive parameter (props) -Optional..
* Statefull or smart Component
* Has Lifecycle Hooks.
* Can handle fetching data via ajax calls.

**REACT COMPONENT LIFECYCLE**

* The Component Lifecycle.
* Each component have several “Lifecycle Methods” that you can override to run code at particular times in the process.
* Method prefixed with will are called before some times happens, method prefixed with did are called right after something happens.

**COMPONENT LIFECYCLE FOUR PHASES**

****

**REACT CONDITIONAL RENDERING**

* React can create distinct components encapsulate behavior you need then you can render only some of them depending on the state of the application.
* It is all about if,else,switch statements and conditional(ternary) - ? :- operator.
* Logical && operator.
* Logical || operator
* To prevent a component from rendering just return null.

**ES6 – ECMA SCRIPT 2015**

* Modules, Import and export.
* Arrow functions.
* Constructor functions.
* Bind, apply, call.
* Var,let and const.
* Template literals -Back- tick.
* Object Destructuring.

**HIGHER – ORDER COMPONENTS**

* Higher-order component is a function that takes a component and returns a new component.
* component Higher-order component (HOC) is an advanced technique in React for reusing logic. HOCs are not part of the React API. They are a pattern that emerges from React’s compositional nature
* That component wrapped inside another React component, so we can modify it. Thus, it is the primary application of the Higher-Order Components..

**REACT – CONTEXT**

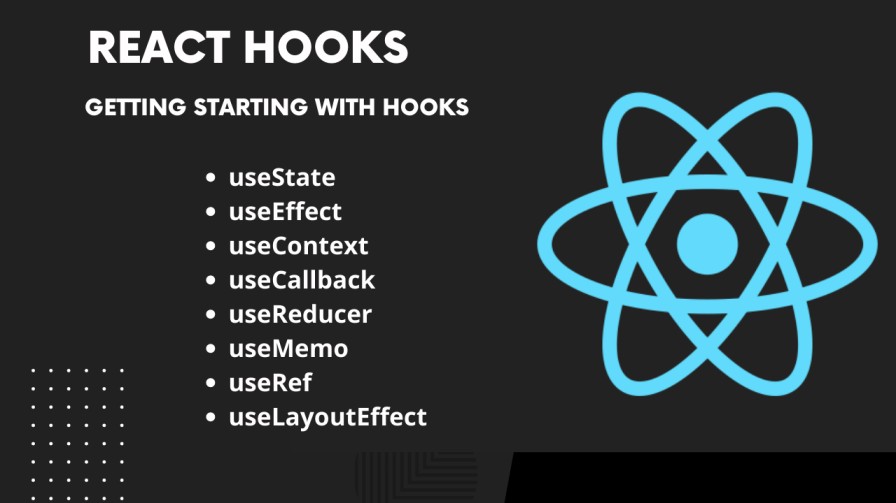
* Context provides way to pass data through the component tree without having to pass props down manually at every level.
* Context is designed to share data that can be considered “global” for a tree of React components, such as the current authenticated user, theme, or preferred language.
* Context overview

React.createContex

Context.Provider

Context.Consumer

**REACT HOOKS**

Hooks are new addition in React 16.8.0 they let you use state and other react feature without writing a class.

* Basic Hooks.
* UseState
* useEffect
* useContext
* Additional Hooks.
* UseReducer
* UseCallback
* useMemo
* useRef

**NEXT.JS**

* Next.js is a JavaScript framework created by Zeit.
* React framework for developing single page JavaScript applications. The benefits of this framework are numerous, both for our clients’ applications
* Improved SEO → more indexable, SEO friendly applications.

**PRE-RENDERING**

* Static Generation
* Server-side Rendering

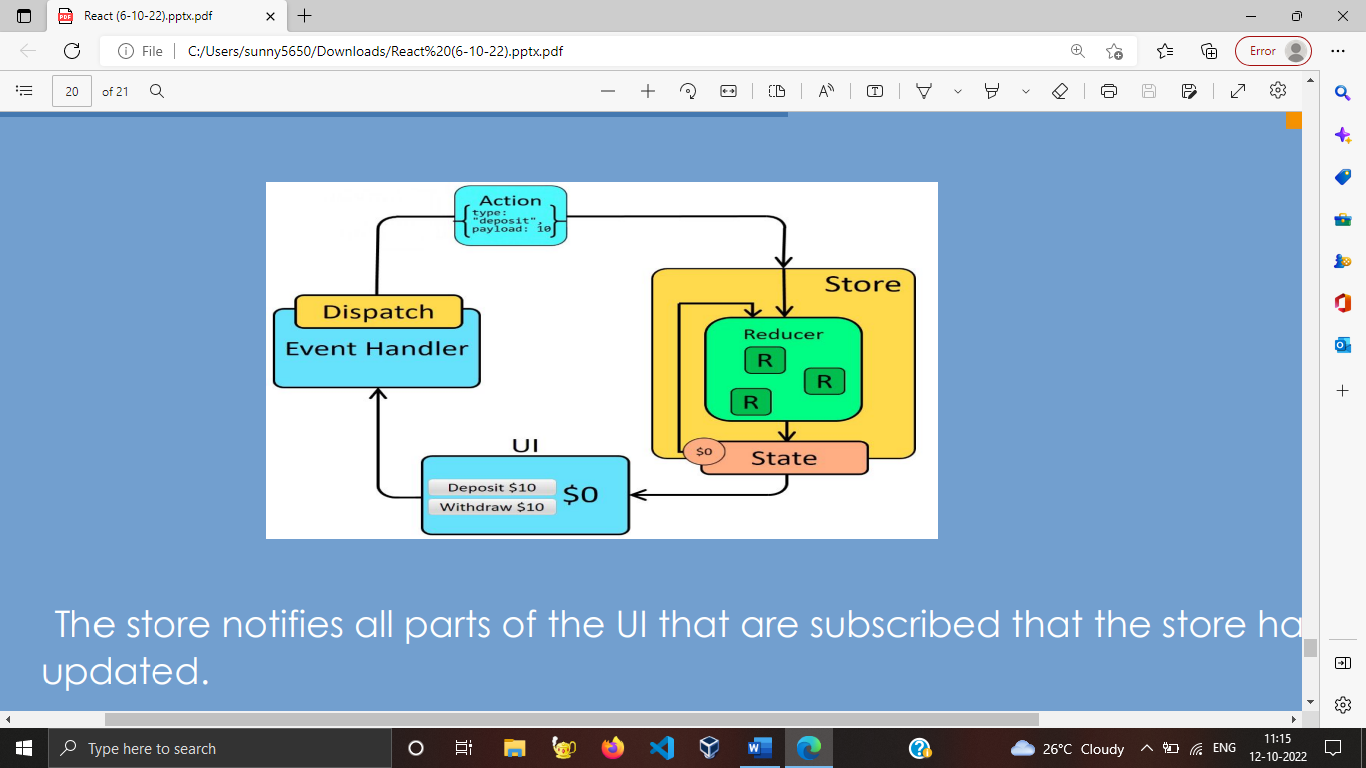
**NEXT.JS ROUTING**

* Next.js has a file-system based router built on the concept of pages.
* When a file is added to the pages directory, it's automatically available as a route.
* Index routes.
* The router will automatically route files named index to the root of the directory.
* Nested routes
* The router supports nested files. If you create a nested folder structure, files will automatically be routed in the same way still. - ”pages/blog/first-post.js → /blog/first-post”
* Dynamic route segments
* To match a dynamic segment, you can use the bracket syntax. This allows you to match named parameters. -pages/blog/[slug].js → /blog/:slug (/blog/hello-world)

**REDUX**

* Redux is a predictable state container for JavaScript apps.
* Redux helps you manage "global" state - state that is needed across many parts of your application.
* Redux can integrate with any UI framework, and is most frequently used with React. React-Redux is our official package that lets your React components interact with a Redux store by reading pieces of state and dispatching actions to update the store
* Actions :- Redux is a pattern and library for managing and updating application state, using events called "actions"
* Dispatch(action) :- Dispatches an action. This is the only way to trigger a state change.

**REDUX DATA FLOW**



* The store notifies all parts of the UI that are subscribed that the store has been updated.
* Each UI component that needs data from the store checks to see if the parts of the state they need have changed.
* Each component that sees its data has changed forces a re-render with the new data, so it can update what's shown on the screen.