REACT

**TOPICS:**

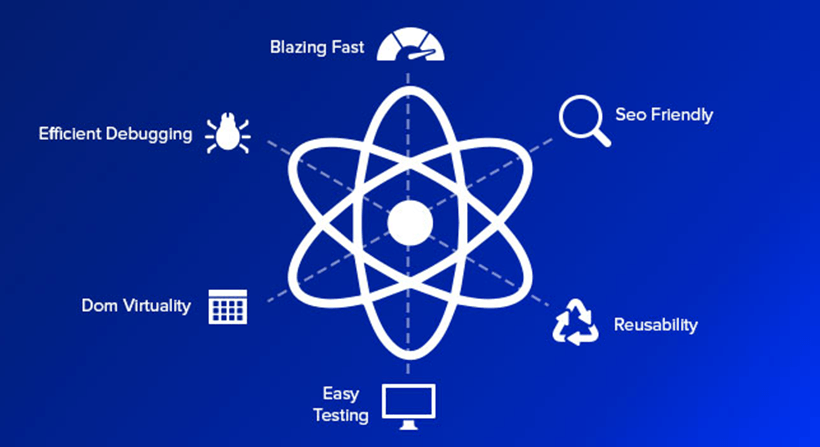
* React JS
* Introduction to React JS
* Virtual Dom
* Jsx
* Component
* Functional & Class Components
* Component Lifecycle
* Conditional Rendering
* ES6 Basics For ReactJs
* Higher-Order Components
* React Context
* Next JS
* Introduction to Next JS
* Pre-rendering
* Static Generation
* Server-side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

**WHAT IS REACT.JS?**

* React is a declarative, efficient, and flexible JavaScript library for building user interfaces
* It lets you compose complex UIs from small and isolated pieces of code called “components”.
* It’s an open-source and component-based framework responsible for creating the application’s view layer.
* ReactJs follows the Model View Controller (MVC) architecture, and the view layer is accountable for handling mobile and web apps.
* React is famous for building single-page applications and mobile apps.

**WHY REACT.JS?**

* **Reusable UI components** - React improves development and debugging processes
* **Fast** - Feel quick and responsive through the Apps made in React can handle complex updates
* **Modular** - Allow you to write many smaller, reusable files instead of writing large, dense files of code. The modularity of React is an attractive solution for JavaScript's visibility issues.
* **Easy to learn** - Since it requires minimal understanding of HTML and JavaScript, the learning curve is low.
* **Server-side rendering and SEO friendly** - ReactJS websites are famous for their server-side rendering feature

****

**REACT VIRTUAL DOM**

* React contains a lightweight representation of real DOM in the memory called Virtual DOM.
* Manipulating real DOM is much slower compared to VDOM as nothing gets drawn on the screen.
* When any object’s state changes, VDOM modifies only that object in real DOM instead of updating whole objects.
* That makes things move fast, particularly compared with other front-end technologies that have to update each object even if only a single object changes in the web application

**JSX**

* JSX stands for JavaScript XML
* JSX is a preferable choice for many web developers.
* It isn't necessary to use JSX in React development, but there is a massive difference between writing react.js documents in JSX and JavaScript.
* JSX is a syntax extension to JavaScript. By using that, we can write HTML structures in the same file that contains JavaScript code.

**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

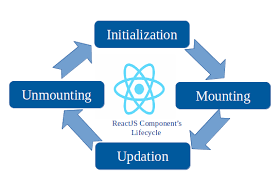
**CLASS COMPONENT**

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

**REACT COMPONENT LIFECYCLE**

* Each component in React has a lifecycle which you can monitor and manipulate during its three main phases.
* The three phases are: **Mounting, Updating,** and**Unmounting.**
* Component can be defined as the series of methods that are invoked in different stages of the component's existence.

**COMPONENT LIFECYCLE FOUR PHASES**



**REACT CONDITIONAL RENDERING**

* Conditional rendering in React works the same way conditions work in JavaScript.
* Use JavaScript operators like [if](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/if...else) or the [conditional operator](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Operators/Conditional_Operator) to create elements representing the current state, and let React update the UI to match them.
* Logical && operator.
* Logical || operator
* To prevent a component from rendering just return null.

**ES6 – ECMA SCRIPT 2015**

* ES6 stands for ECMAScript 6
* It is the sixth edition of the ECMAScript language specification standard
* It is used to create the standards for JavaScript language such that it can bring multiple independent implementations

**HIGHER – ORDER COMPONENTS**

* Higher-order components (HOCs) in React were inspired by higher-order functions in JavaScript.
* A HOC is an advanced technique for reusing logic in React components. It is a pattern created out of React’s compositional nature.
* HOCs basically incorporate the don’t-repeat-yourself (DRY) principle of programming, which you’ve most likely come across at some point in your career as a software developer

**React Context**

* React Context is a way to manage state globally.
* Context provides a 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.

**React Hooks**

* React Hooks are simple JavaScript functions that we can use to isolate the reusable part from a functional component.
* Hooks can be stateful and can manage side-effects.
* A Hook is a special function that lets you “hook into” React features. For example, useState is a Hook that lets you add React state to function components.

**Conclusion**

Learning “ReactJS” is the need of the hour and makes sense as it is providing much-needed ease to developers in building highly engaging web applications and user interfaces in a very lesser time, where they create large-scale apps with frequently changing data. React’s benefits of being robust, advanced, responsive, non-risky, user-friendly far exceed its disadvantages, and developers and organizations understand React’s relevance in the market, so they are promoting its learning and deployment wholeheartedly.