**React Notes**

**Introduction:**

ReactJS is a **declarative**, **efficient**, and flexible **JavaScript library** for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.

**Why React?**

The main objective of ReactJS is to develop User Interfaces (UI) that improves the speed of the apps. It uses virtual DOM (JavaScript object), which improves the performance of the app. The JavaScript virtual DOM is faster than the regular DOM. We can use ReactJS on the client and server-side as well as with other frameworks**.**

**React Installation:**

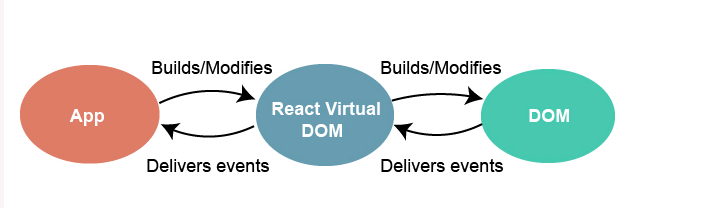
1. NodeJS and NPM
   1. Node version >= 8.10
   2. NPM version >= 5.6
2. React and React DOM
3. Webpack
4. Babel

Steps:

1. **Install NodeJS and NPM**
2. **Install React -** npm install -g create-react-app
3. **Create a new React project --** npx create-react-app app\_name
4. Running the Server --cd app\_name -🡪 npm start

**Features:**

* 1. Virtual DOM
  2. Performance
  3. Single page application
  4. Simplicity



**Components:**

1. Functional Components or stateless component
2. Class Components or statefull component

**State and Props:**

The state is an updatable structure that is used to contain data or information about the component. The state in a component can change over time. The change in state over time can happen as a response to user action or system event. A component with the state is known as stateful components.

To define a state, you have to first declare a default set of values for defining the component's initial state. To do this, add a class constructor which assigns an initial state using this.state. The '**this.state**' property can be rendered inside **render()** method.

**Props**:

Props stand for "**Properties**." They are **read-only** components. It is an object which stores the value of attributes of a tag and work similar to the HTML attributes. It gives a way to pass data from one component to other components. It is similar to function arguments. Props are passed to the component in the same way as arguments passed in a function.

Props are **immutable** so we cannot modify the props from inside the component. Inside the components, we can add attributes called props.

Props vs state?

**Component Life cycle:**

1. Mounting Phase
2. Updating Phase
3. Unmounting Phase

Mounting

1. constructor() 🡪componentWillMount
2. getDerivedStateFromProps() 🡪 ComponentWillreceiveProps
3. render()
4. componentDidMount()

Update

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

Unmounting

* componentWillUnmount()

**Refs , Fragments and Routing:**

Refs:

Refs is the shorthand used for **references** in React. It is similar to **keys** in React. It is an attribute which makes it possible to store a reference to particular DOM nodes or React elements. It provides a way to access React DOM nodes or React elements and how to interact with it. It is used when we want to change the value of a child component, without making the use of props.

Refs can be used or not used in the following cases:

* When we need DOM measurements such as managing focus, text selection, or media playback.
* It is used in triggering imperative animations.
* When integrating with third-party DOM libraries.
* It can also use as in callbacks.
* Its use should be avoided for anything that can be done **declaratively**. For example, instead of using **open()** and **close()** methods on a Dialog component, you need to pass an **isOpen** prop to it.
* You should have to avoid overuse of the Refs.

**this**.callRef = React.createRef();

Fragments:

In React, whenever you want to render something on the screen, you need to use a render method inside the component. This render method can return **single** elements or **multiple** elements. The render method will only render a single root node inside it at a time. However, if you want to return multiple elements, the render method will require a '**div**' tag and put the entire content or elements inside it. This extra node to the DOM sometimes results in the wrong formatting of your HTML output and also not loved by the many developers.

<Fragment></Fragment> or <></>

Router:

Routing is a process in which a user is directed to different pages based on their action or request. ReactJS Router is mainly used for developing Single Page Web Applications. React Router is used to define multiple routes in the application. When a user types a specific URL into the browser, and if this URL path matches any 'route' inside the router file, the user will be redirected to that particular route.

React Router is a standard library system built on top of the React and used to create routing in the React application using React Router Package. It provides the synchronous URL on the browser with data that will be displayed on the web page. It maintains the standard structure and behavior of the application and mainly used for developing single page web applications.

React Router plays an important role to display multiple views in a single page application. Without React Router, it is not possible to display multiple views in React applications. Most of the social media websites like Facebook, Instagram uses React Router for rendering multiple views.

* **<BrowserRouter>:** It is used for handling the dynamic URL.
* **<HashRouter>:** It is used for handling the static request.

Route is used to define and render component based on the specified path. It will accept components and render to define what should be rendered.

What is < Link> component?

This component is used to create links which allow to **navigate** on different **URLs** and render its content without reloading the webpage.

React Router switch

The <**Switch**> component is used to render components only when the path will be **matched**. Otherwise, it returns to the **not found** component.

Redirect:

A <Redirect> component is used to redirect to another route in our application to maintain the old URLs. It can be placed anywhere in the route hierarchy.

Why Router:

* In this, it is not necessary to set the browser history manually.
* Link uses to navigate the internal links in the application. It is similar to the anchor tag.
* It uses Switch feature for rendering.
* The Router needs only a Single Child element.

**React styling:**

1. Inline Styling
2. CSS Stylesheet
3. CSS Module
4. Styled Components

React Hooks:

Hooks allow us to "hook" into React features such as state and lifecycle methods.

* Hooks can only be called inside React function components.
* Hooks can only be called at the top level of a component.
* Hooks cannot be conditional
  1. **Usestate**:

To use the useState Hook, we first need to import it into our component.

We initialize our state by calling useState in our function component.

useState accepts an initial state and returns two values:

* The current state.
* A function that updates the state.
  1. **UseEffect**:
* The useEffect Hook allows you to perform side effects in your components.
* Some examples of side effects are: fetching data, directly updating the DOM, and timers.
* useEffect accepts two arguments. The second argument is optional.

1. UseSontext:
   1. State should be held by the highest parent component in the stack that requires access to the state.
   2. To illustrate, we have many nested components. The component at the top and bottom of the stack need access to the state.
   3. To do this without Context, we will need to pass the state as "props" through each nested component. This is called "prop drilling".
   4. React Context is a way to manage state globally.
2. UseRefs:
   1. The useRef Hook allows you to persist values between renders.
   2. It can be used to store a mutable value that does not cause a re-render when updated.
   3. It can be used to access a DOM element directly.