Introduction to React

We are going to start off with React, but before we do, make sure that you know the following pre-requisites -:

1. **HTML**
2. **CSS**
3. **Javascript basics**
4. **Basics of DOM manipulation**

==========

A little bit of backdrop...

Historically speaking, websites and webpages were meant to be static in nature. That means only HTML, CSS and a little bit of Javascript was there in the webpages. But with time, websites started becoming more and more complex and dynamic in nature. This means that a lot of heavy content and data were being rendered on these web pages.

The traditional client-server architecture of server building web pages using HTML and CSS and sending these pages to the client started taking a hit on the performance. There was a dire need for something much more performant so that the load on servers could be reduced/removed all together.

This is were the concept of SPAs came into the limelight.

Single Page Application.

Take a look at the following visual diagram to understand the diff b/w MPA (Multi Page Applications) and SPA (Single Page Applications).

The major difference between the two is that with each subsequent page change, the request is not sent to the server in the case of SPAs. Rather, a 'page change' occurs on the client side only by the virtue of Javascript. That means all the heavy lifting like creating dynamic HTML and CSS webpages with JS functionality and page changes is now the headache of the browser. The server only sends a single index.html file with a bundle of Javascript to handle all the interactivity and routing on the client side.

Modern day frameworks and libraries like React, Angular, Vue etc work on the concept of SPA. SPAs provide a highly efficient way of creating extremely dynamic and versatile web applications with ease.

React

React is a Javascript based UI library to create highly customisable, modular and extensible components which combine together to create dynamic web applications.

React was the brainchild of Facebook, and was launched to the general public in the year 2011.

Statistically speaking, React is the most popular UI library for creating web applications. There are various reasons for the popularity of React. Some of the reasons are listed down below -:

1. Low learning curve.
2. Extreme flexibility in creating the logic of the web applications.
3. Fantastic online support and thousands and thousands of third party packages to help ease out of development process.

The question now arises, "Why should we use libraries like React when we can do everything with Vanilla Javascript?"

1. **Reactivity** : Modern UI libraries like React are reactive in nature. This means that the UI reacts/changes on data change automatically. This is very hard to implement in vanilla JS. The application data and the UI view are tightly coupled with each other through an internal algorithm.
2. **Performant DOM manipulation** : React is all about performant DOM manipulations. DOM manipulation directly on the DOM is very expensive and difficult if you are manipulating 100 of DOM elements at once. React is a master in doing all this efficiently, so that developers can focus on the logic of the application and build things declaratively.
3. **Less code, Achieve more** : React is also about writing less and less code and achieve high levels of fidelity.
4. **Complex build process and bundling** : Apart from the above features, React also provides a complex build process and code bundling feature which can increase the performance of the web application.

Different ways to start using React

1. CDN
2. Create-React-App (now in legacy mode)
3. Vite
4. NextJS (The de-facto standard of using React as per the React team)

The concept of modularity

The concept of modularity is an important aspect in understanding frameworks and libs. Modularity simply means breaking down your code into smaller, self contained pieces of logic which can be reused across the entire application.

In the context of React, modularity helps us in building reusable components to be used across our application.

Using the ES6 import/export feature in JS, we can create javascript modules and achieve modularity.

Component-driven architecture

React works on the concept of components. In the very core essence, components are self contained, isloated bundles of HTML, CSS and JS

which can function on it's own. We will see more about components later.