React ⚛ 💗

☑ Also known as React.js/ React.JS

Free & Open-source JavaScript library for building user interfaces (UIs).

Created by Meta (Facebook) & is now maintained by Meta & community of developers.

> HTML/CSS/JS code refreshes the whole page unexpectedly upon interaction (like clicking a button, submitting a form, etc.)

- React is used to build *SPAs (Single Page Applications)*
- Creates *reusable UI components*
- Develops dynamic, fast & scalable front end applications.

👉 **ReactJS uses a Virtual DOM mechanism to efficiently update the HTML DOM.**

Instead of reloading the entire DOM, the virtual DOM updates only the specific elements that have changed, resulting in faster performance.

🧑‍💻 React follows a Component-Driven Architecture

The user interface is built by composing independent, reusable pieces called components. Each component manages its own logic and rendering, making development more modular, scalable, and easier to maintain.

A Component is a reusable UI element.

React Element : an object that describes the properties of an actual DOM node which will be created by React.

Simple Structure of a React Elements:

```
{
    type: "button"
    props: {
        classname: "btn"
    }
}
```

✸ SPAs(Single Page Application)

A web application that loads a single HTML page and dynamically updates content as users interact with it is known as a Single Page Application (SPA).

In an SPA, the entire page doesn't reload with every interaction; only the necessary parts of the page are updated, providing a smoother and faster user experience.

- It uses client side rendering (CSR) [Javascript Updates the DOM]
- Communicates to server via APIs (e.g. REST, GraphQL etc.)

Example: Facebook, Gmail, Twitter

☑ **Pros**

- Faster Navigation (No full page reloads)
- Smoother UX
- Easier to build complex, interactive UIs
- Good for real time updates

✕ **Cons**

- Slower initial loads (Heavy JS Bundles)
- Search engines historically struggled with Javascript heavy SPAs. But solutions like Next.JS do help.
- Browser history management needs extra efforts (e.g React Router)

MPAs(Multi Page Application)

A traditional web app where each page is a seperate HTML document loaded from the server.

- It uses Server side rendering (SSR) [Server generats HTML for each request]

Example: Amazon, Wordpress, Old school sites

☑ **Pros**

- Better SEO out of the box as each page is a seperate HTML page.
- faster initial loads (Less Javascript code, server handles rendering)
- Easier to scale as pages can be cached independently.

✕ **Cons**

- Slower navigation (full page reloads upon each interaction)
- Less interactivity (more clunky UX)
- Harder to maintain state (e.g Shopping Cart across pages)

Where to use SPA & MPA**

- If you need a dynamic, app alike experience you need to use SPA (e.g SaaS Tool, Social Network Site)
- If you prioritize SEO, Simplicity, Server-Driven content you need to use MPA (e.g Blogs, News etc.)

Hybrid Approach

Modern frameworks like **Next.JS (React), Nuxt.JS (Vue)** allows **SSR + SPA** hybrid models for a better SEO & Optimization.

This approach is best for both cases.

👨‍💻 React application is made up of Multiple Components

- Each of them are **responsible for outputting a small, reusable piece of HTML**.
- **Components can be nested within other components to allow complex applications** to be built out of simple building blocks.

👉 Key Features of React

1. **Component Based Architecture :** React apps are built using reusable components, making code easier to manage & scale

2. **Virtual DOM :** React uses a Virtual DOM to efficiently update only the parts of the page that change, improving performance.

3. **Declare Syntax :** Instead of manually updating the DOM like Vanilla JS, you describe how the UI should look & React handle updates automatically.

4. **Strong Ecosystem :** React has a massive community, tons of libraries (like Redux, Next.JS & Extensive documentation)

5. **Works with other frameworks :** React can be integrated with backend/mobile apps. (Node.JS, Django) [React Native]

📚 Library

A collection of pre-written functions/modules that you can call as needed. It only provides the boilerplate for the project.

Key Idea : You can control the application's flow & structure. The library will provide you utility.

Analogy : Like a toolbox, you'll pick which tools (functions) to use & when to use.

Example: React (UI Library), Loadash (Utility Function), JQuery (DOM Manipulation)

Charecteristics :

1. Flexibility : Uses only what I need
2. Less Opinionated : No Strict rules on application structure
3. More Responsibility : You decide how to integrate everything together.

🖼 **Framework**

A fully featured structure that dictates how to build an application (includes library, tools, rules)

Key Idea : It controls the application's flow. The library will provide you utility.

Analogy : Like a blueprint, you build within it's rules.

Example: Angular, Django, Ruby on Rails

Charecteristics :

1. Built in tools like tools for routing, state management etc. is already included.
2. Scalability : Enforces best practices for large teams.
3. Less flexibility : Must follow the framework's convention.