**React Revision**

* **Origins and Founder**
* React was created by Jordan Walke, a software engineer at Facebook, in 2011. The initial prototype, called "FaxJS," was inspired by Facebook’s earlier work on XHP, an HTML component system for PHP.
* React was first used internally at Facebook for the News Feed and then on Instagram in 2012. It was open-sourced to the public at JSConf US in May 2013, which led to rapid adoption by developers worldwide.
* Jordan Walke’s innovation was later developed and maintained by the broader team at Facebook (now Meta) and an active open-source community.
* **Why Was React Created? (When JavaScript Already Existed)**
* While anything you can do in React *could* be achieved with vanilla JavaScript, React arose out of a need to better handle the *growing complexity* and *performance bottlenecks* Facebook engineers faced in large-scale apps:
* **Complex UI State:**As Facebook's interfaces grew (like News Feed and Ads), managing the interaction logic and state changes directly with JavaScript and libraries like jQuery became increasingly messy, error-prone, and hard to maintain.
* **Performance Issues:**Manual DOM manipulation (as with traditional JavaScript or jQuery) slowed down apps, especially when many parts of the page updated frequently or in unpredictable ways.
* **Cascading Updates:**Developers often struggled to keep the UI consistent as changes in one part could cause unintended side effects elsewhere. The codebase became challenging to update and scale.
* **How React Solves These Problems**
* **Component-Based Architecture:**React encourages breaking the UI into reusable, independent pieces (components), making interfaces easier to reason about, maintain, and scale.
* **Virtual DOM:**Instead of updating the browser DOM directly every time something changes, React first updates a lightweight, in-memory representation (the Virtual DOM). It then calculates the minimal real DOM changes needed for better performance.
* **Declarative Style:**Rather than imperatively telling the browser *how* to update the UI, you declare *what* you want the UI to look like for a given application state, and React takes care of efficiently applying the changes.
* **Improved State Management**: React and its ecosystem (like Redux) make it much easier to predict and manage UI changes as data evolves, which was a major pain point for developers using only JavaScript.
* **Why to learn React?**
* Makes easy to manage & build complex frontend.
* React is the JavaScript Library.
* **Topics in react?**
* Core of React (state or UI manipulation, JSX)
* Component Reusability
* Reusing of Component (Props)
* How to propagate change (Hooks)
* Router
* State management – Redux, Redux toolkit, Zu stand, Context Api
* Class based component (old concept)
* BAAS Apps (backend as a service) eg: firebase
* **Road Map :   
  HTML – CSS – JAVASCRIPT – REACTJS - NEXTJS - ANGULAR-JS**

**How to install react ?**

* npx create-react-app appname
* npm vite@latest

**npx create-react-app appname**

Bootstraps a React app with Webpack as the build tool.

Hides most configuration unless you "eject".

Widely used and reliable, with many tutorials and resources.

Development server and build times are slower, especially for larger apps.

Limited to React only.

**npm create vite@latest (or npm init vite@latest)**

Bootstraps a frontend app with Vite, which is framework-agnostic (supports React, Vue, Svelte, etc.).

Uses native ES modules and esbuild for rapid cold starts and Hot Module Replacement (HMR), resulting in much faster dev experience.

Offers an open and easily editable configuration from the start.

The build process for production is highly optimized with Rollup.

Designed for modern browser and JavaScript standards, making it more "future-proof".

- [Note : Install node module folder if you are using vite – npm install ]