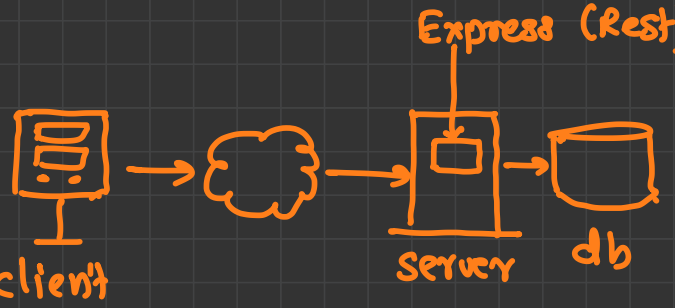
React

Web development

- Frontend
  - browser
  - website
    - HTML
    - CSS
    - JS
- backend
  - server



stack →	platform	web server	frontend	db
LAMP	Linux	Apache	PHP, Perl, Python	MySQL
MEAN	Node JS	Express	Angular	Mongo
MERN	Node JS	Express	React	Mongo
WISA	Windows	IIS	Aspx	SQL server

Communication standards

- REST → Representational state transfer
  - → Design pattern (HTML method)

  } uses JSON
- GraphQL → Graph Query Language
  - → design pattern (implemented over REST)
- SOAP → Simple object Access Protocol
  - → protocol (XML was used to exchange data)
  - → deprecated
- gRPC → Remote Procedure Call
  - → protocol
  - → Fastest & complex
  - → used in IoT devices

API = Application Programming Interface

→ functions

types

→ framework

→ consists of multiple components (languages, modules, libraries)

→ larger scope than libraries

→ eg. Angular [TS, html, css, JS]

→ may have a custom architecture → pattern

→ libraries

→ consists of single component (language, module)

→ eg. React (JS)

→ most of the times, does have any architecture (pattern)

Contents

Comparison, why, features

- Introduction
- Why React?

JSX = JS + XML (HTML)

- Using JSX

component based development

- Class based Components

↳ deprecated

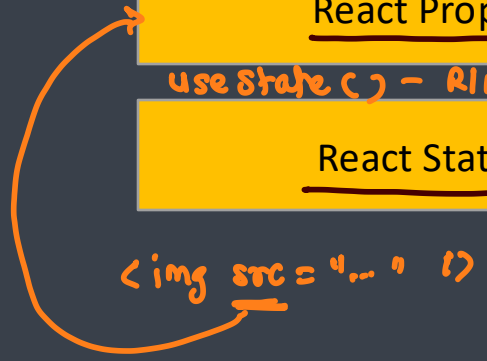
- Functional Components

properties - readonly - immutable

- React Props

useState() - R|W - mutable

- React State



useContext...

special function

- React Hooks ***

Global store - share data

- Intro to Redux External library

sharing data

- Redux vs Context

input - text / number / date, select

- Handling User Inputs

used for multi-component applications

- React Router

parameterized routes

- Advanced Router

fetch, axios

- Consuming REST APIs

React + Express

- Uploading Files

components with styles

- Styled Components

Email password → caching → encryption

- Authentication

JWT → token

- Authorization

- Session Handling

React UI + Express + DB

- Pagination

AWS cloud

- Cloud Deployment

About Instructor

- 16+ years of experience
- Associate Technical Director at Sunbeam
- Freelance Developer working in various domains using different technologies
- Developed 180+ mobile applications on iOS and Android platforms → React Native
- Developed various websites using PHP, MEAN and MERN stacks
- Languages I love and use in every programming: C, C++, Python, JavaScript, TypeScript, PHP Go

Introduction

SPA → React (Facebook) / Angular (Google) / VueJS (Alibaba)

- React, also known as ReactJS, is a popular and powerful JavaScript library used for building dynamic and interactive user interfaces, primarily for single-page applications (SPAs)
- It was developed and maintained by Facebook and has gained significant popularity due to its efficient rendering (→ fast) techniques, reusable components, and active community support
- React is a declarative, component based library that allows developers to build reusable UI components and It follows the Virtual DOM (Document Object Model) approach, which optimizes rendering performance by minimizing DOM updates. React is fast and works well with other tools and libraries
  - → Redux / Router / axios
- Prerequisite of React
  - For learning React first you have a clear understanding of HTML, CSS and JavaScript
    - → JS
      - → function → named / arrow
      - → OOP – class, object, reusable
  - As React is a JavaScript library and uses most of its concept so you really have to understands the major concepts of it
  - HTML and CSS
  - JavaScript and ES6
  - JSX (JavaScript XML) →
  - Node + NPM
  - Git

SPA → Single Page Application

- → app loads the whole UI when loading first time
- → never need to send any request for UI again
- → sends request to load data only when needed
- → faster than MPA app
- → slower while loading for the first time
- → can be cached (stores data in memory)

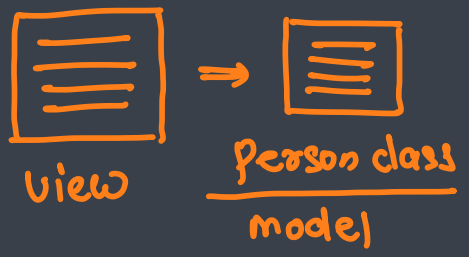
History

- It was created by **Jordan Walke,** who was a software engineer at **Facebook**
- It was initially developed and maintained by Facebook and was later used in its products like **WhatsApp** & **Instagram**
- Facebook developed ReactJS in **2011** in its newsfeed section, but it was released to the public in the month of **May 2013**
- Today, most of the websites are built using MVC (model view controller) architecture
- In MVC architecture, React is the 'V' which stands for view, whereas the architecture is provided by the Redux or Flux

Model → represents data from database [classes]

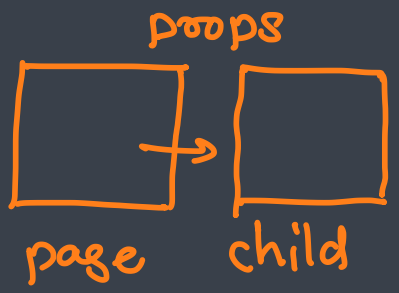
View → User interface → angular / React / vueJS

Controller → controls view by binding it with model



Features

- **Declarative UI** [JSX] → code = UI + logic (JSX)
  - React allows developers to design user interfaces in a declarative way
  - Developers describe what the UI should look like for any given state, and React updates the DOM to match that state automatically
- **Component-Based Architecture** → component - reusable entity < class / function
  - Applications in React are built as a collection of small, reusable components
  - Encourages modularity
  - Makes code reusable and easier to test and maintain
- **JSX (JavaScript XML)** ⇒ JS + HTML
  - React uses **JSX**, a syntax extension that lets you write HTML-like code inside JavaScript
  - Improves readability and allows developers to embed JavaScript expressions directly within the markup
- **Virtual DOM** → in memory representation → document object of JS
  - React uses a **virtual DOM**, a lightweight copy of the actual DOM
  - Efficiently updates the real DOM by minimizing changes
  - Enhances performance, especially in dynamic applications

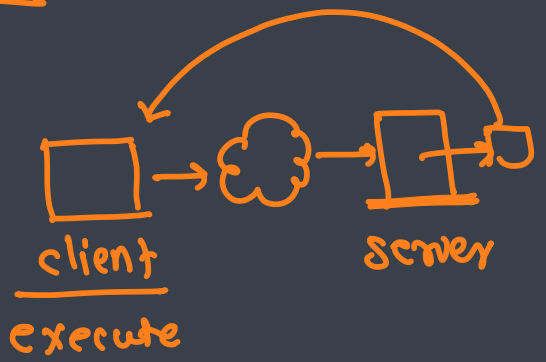


- **Unidirectional Data Flow**
  - React enforces a **unidirectional data flow**, meaning data flows in a single direction (from parent to child)
  - Makes debugging easier and improves control over how data is passed and managed

Features

useState, useEffect, useCallback ...

- **React Hooks** → special function which embed functionality dynamically
  - Hooks are functions like useState and useEffect that allow function components to use React features such as state and lifecycle methods
  - Simplifies code by reducing the need for class components and makes managing state and side effects straightforward
- **React Router**
  - React Router is used for implementing dynamic routing in React applications
  - Allows the creation of single-page applications (SPAs) with seamless navigation (from one component to another)
- **Context API** → sharing data
  - The Context API enables global state management without needing third-party libraries like Redux
  - For managing themes, authentication, or other data shared across multiple components
- **Code Splitting and Lazy Loading**
  - React supports code splitting through React.lazy() and dynamic imports
  - Loads only the required code for a specific page or feature
  - Reduces initial load time and improves performance
- **Server-Side Rendering (SSR)** → Next.JS
  - With frameworks like Next.js, React supports server-side rendering
  - Improves SEO and reduces time-to-interactive for end users
- **Performance Optimization** → Virtual DOM
  - React offers built-in tools and techniques for optimization
  - Concurrent Rendering: React 18 introduced concurrent rendering to handle complex UIs efficiently
- **Cross-Platform Development**
  - With React Native, developers can build mobile applications using React
  - Share code between web and mobile platforms

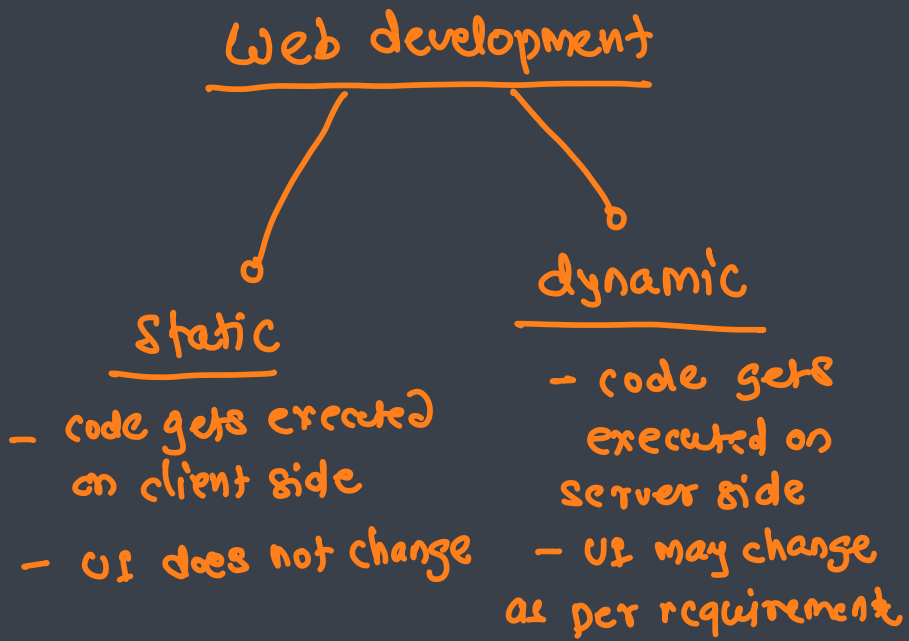


React JS → Web app

React Native → mobile app

Advantages

- Easy to learn and use
- Creating dynamic web application becomes simpler
- Reusable components
- Performance enhancements
- Support of handy tools and libraries
- Benefits of being a JS library
- Easy to unit test the application
  - Jasmine



Disadvantages

- The high pace of development
- Poor documentation
- Its only about the View [REST - axios, global state - redux]
- Learning curve for JSX