Welcome to Lecture 23!

Agenda

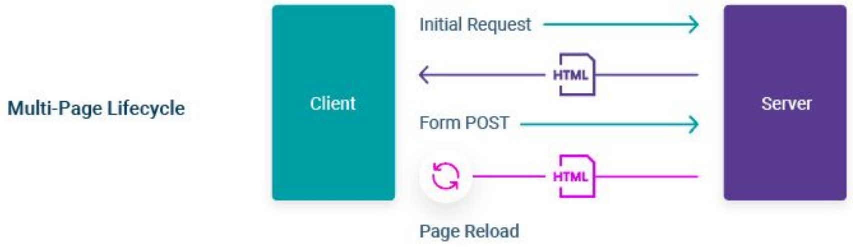
Session Objectives

- What’s a Single-Page Application (SPA)?
  - Introduction to SPA
  - Challenges to building SPA manually
- Introducing the React Toolkit
  - Understanding the npm use case
- Library vs Framework
- Exploring npm & Vite
- Creating your first React app
- Quiz

Introduction to Single-Page Application

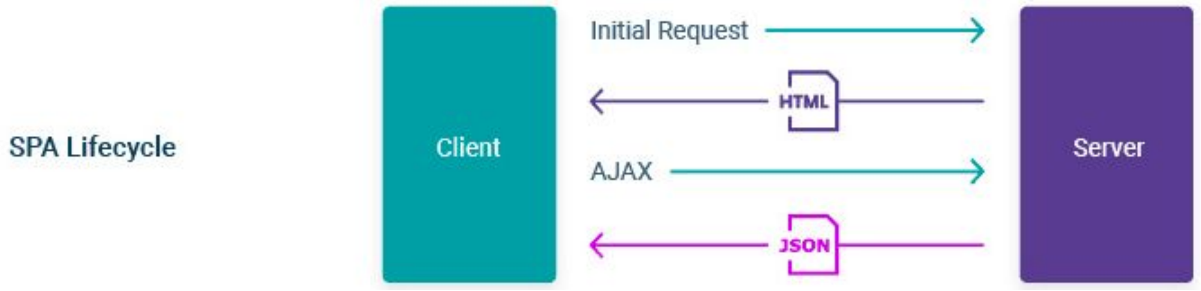
Before SPA, let’s explore Multi-Page Application

- Page-by-Page Load
    - Each distinct route (e.g., /about, /products) is a separate HTML document on the server
- Traditional Navigation
    - An event triggers the browser’s default navigation flow: it discards the current DOM, requests the new page, and recreates the DOM from the returned HTML
- Server-Side Rendering (SSR) by Default
    - Every request goes through the server’s rendering pipeline (templates, controllers, database calls)
- State & Data Handling
    - Application state is typically stored in back-end sessions, cookies, or query parameters
- **Result**: Reliable, time-tested architecture with excellent SEO and straightforward security boundaries, at the cost of extra round-trips and noticeable full-page reloads during navigation



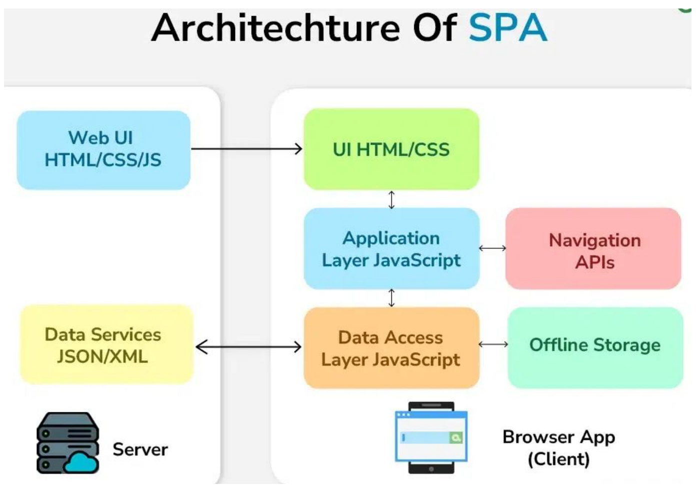
What is a Single-Page Application (SPA)?

- Initial Load
  - An SPA loads a single, primary HTML file (an "application shell") and the associated JavaScript application bundle just once
- Dynamic Updates
  - As the user interacts with the app (e.g., clicks a link), JavaScript intercepts the navigation event. Instead of fetching a new page, it dynamically rewrites only portions of the current page's DOM
- Server Interaction
  - Subsequent communication with the server is primarily done via API calls to fetch or send data (usually in JSON format), not entire HTML pages
- **Result**: A faster, more fluid user experience that eliminates the jarring full-page reloads



Core SPA Concepts

- Client-Side Routing:
  - JavaScript handles the application's "routes" (e.g., /home, /about, /profile)
  - It uses the browser's History API to update the URL in the address bar, giving the illusion of separate pages
  - It prevents the browser's default navigation behavior and instead renders new "views" or "components" into the DOM
- State Management
  - The "state" (all the application data, like user info, lists of items, form inputs) is held and managed within the JavaScript application on the client-side.
  - When the state changes (e.g., a new item is added to an array), the UI is automatically updated to reflect that new state



The Challenge of Building SPAs Manually

- Complex DOM Manipulation
  - Manually writing the JavaScript to find the correct DOM elements, create new ones, and update them for every possible user interaction becomes incredibly complex and error-prone at scale
- State-UI Synchronization
  - The hardest problem to solve manually is keeping the UI perfectly in sync with the application's data (state). If a user is added to an array, you must remember to manually write the code to update the user list, the user count, and any other dependent UI element
- Code Organization
  - Without a structured approach, the codebase for a large vanilla JS SPA can become a tangled mess of event listeners and DOM manipulation functions, often called "spaghetti code."

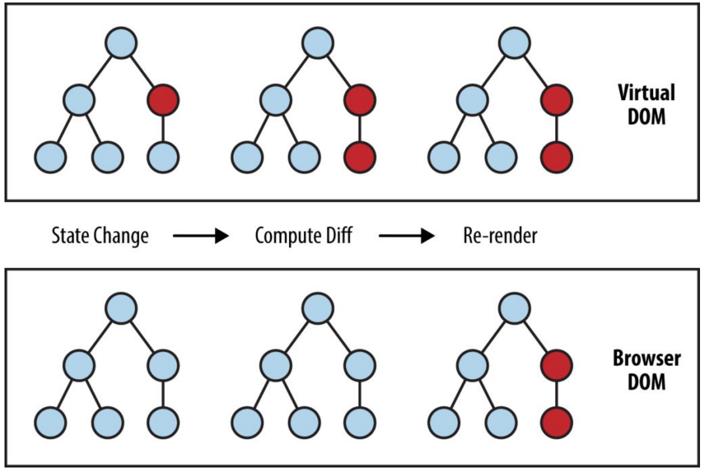
Introducing the React Toolkit

Demo

- Let's demonstrate why manually updating the DOM thousands of times to understand the performance bottleneck large applications must mitigate
  - This is called the "DOM Thrashing" Problem

The Solution: React's Virtual DOM

- React avoids direct DOM manipulation by using a Virtual DOM (V-DOM)
- The V-DOM is a lightweight, in-memory representation of the real DOM, stored as a JavaScript object
- The Process
    - When your application's state changes, React creates a new V-DOM tree reflecting the new state
    - It then compares this new V-DOM tree with the previous one using a highly efficient "diffing" algorithm
    - This algorithm identifies the exact, minimal set of changes required to update the UI
    - Finally, React takes these calculated changes and applies them to the real DOM in one single, optimized batch

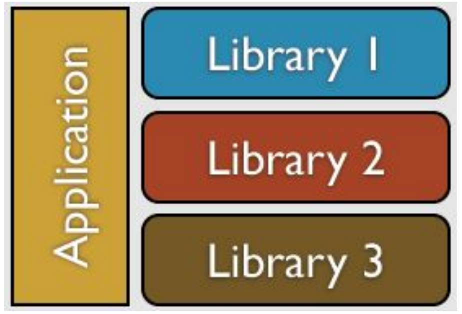


From Manual Updates to Declarative UIs

- The Old Way (Imperative): With vanilla JS, we manually write the exact steps to change the DOM.
  - Eg: "Find this div, create a p, set its text, append it."
- The React Way (Declarative): React introduces a new paradigm. We simply declare what the UI should look like for a given state
- The Shift in Thinking
  - We stop managing the DOM manipulation ourselves. We manage the state
  - When the state changes, React figures out the most efficient way to update the DOM to match our declaration
- As we make a transition in thinking, let’s understand the tool that provides this powerful, declarative approach

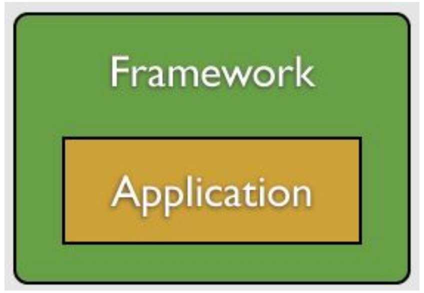
Understanding Libraries

- **Purpose**: A collection of pre-written code (functions, objects) that provides specific functionality to solve a particular problem
- Control Flow
  - You are in control. You import the library and call its functions in your code whenever you need them
- Flexibility
  - High. You can use multiple libraries together and structure your application however you see fit
- Use Cases
  - React: A library for building user interfaces
  - Chart.js: A library for creating charts and graphs
  - Tailwind CSS: A CSS utility library for styling
  - Axios: A library for making HTTP requests



Understanding Frameworks

- **Purpose**: A framework provides a complete, opinionated structure for building an entire application. It's a blueprint, not just a set of tools
- Control Flow
  - The framework is in control. It provides a skeleton and calls your code at specific, predefined points
- Flexibility
  - Lower. A framework often dictates how you handle things like routing, data management, and state, providing its own built-in solutions
- Examples
  - Angular, Vue.js, Ruby on Rails are popular web frameworks



Building your React Projects

Introducing Vite: The Modern Build Tool

- What is it?
  - Vite is a modern, extremely fast tool for building web projects. It is not a framework, but a tool that sets up our development environment
- What problem does it solve?
  - Manually setting up a modern JavaScript project with support for features like JSX (a file format used by React) and optimized code bundling used to be very complex
- Key Benefits
  - Blazing Fast Development Server: Provides instant feedback in the browser when you save changes on your machine, permitting faster testing
  - Pre-configured Templates: Allows you to start a new React (or Vue, Svelte, etc.) project with one command
  - Optimized Builds: When you're ready to deploy your site, Vite bundles your code into small, efficient files for production
- How do we install libraries like Vite, React & Tailwind?

Introduction to NPM

NPM (Node Package Manager)

- What is NPM?
  - NPM is the standard package manager for the JavaScript ecosystem. It is a critical tool for all modern web development
- Its Two Main Parts
  - An online registry (a massive public database of open-source JavaScript code packages)
  - A command-line tool (npm) used to install, manage, and share those packages
- How can you use an npm package in your project?
  - The package.json File: This file is the manifest for your project. It lists all the packages your project depends on (like react, tailwind, vite, etc.) and their versions

Understanding Packages & Modules

- A package (or module) is simply a folder containing JavaScript code and a package.json file.
- This file acts as a manifest, describing the package and its dependencies.
- Packages are reusable pieces of code that solve specific problems (e.g., react for building UIs, axios for network requests).
- Local vs. Global Packages
  - <u>Local</u> (Default): Installed directly within your project within a *node_modules* folder. This is the standard for most packages you'll install (react, vite, etc.)
  - <u>Global</u>: Installed on your computer system-wide. Used only for command-line tools you want to run from anywhere (e.g., http-server, vite)

Essential NPM Commands

- npm init -y
  - Initializes a new Node.js project by creating a package.json file in your current directory
- npm install <package-name>
  - Downloads a package from the registry into your node_modules folder and adds it to your dependencies in package.json. Eg: npm i react
- npm uninstall <package-name>
  - Removes a package from node_modules and package.json, removing it from your project
- npm run <script-name>
  - Runs a command defined in the "scripts" section of package.json (e.g., npm run dev)

Break Task: Install Node.js & NPM

- Visit to nodejs.org
  - Download the LTS (Long-Term Support) version for your operating system
- Run the installer and follow the on-screen instructions (accepting defaults is fine)
- To verify, open your terminal (or Git Bash) and run node -v and then npm -v. You should see version numbers appear
- If you've done the above, you can install nvm, which allows us to switch between different versions of node

Setting Up Our First React Project

That’s it for today.
Questions?

Quiz!

Question 1

- What is the primary difference between a Single-Page Application (SPA) and a Multi-Page Application (MPA)?
  - A. SPAs are always faster than MPAs
  - B. MPAs use JavaScript for routing, while SPAs rely on server-side routing
  - C. SPAs load a single HTML shell and dynamically update content, while MPAs request a new HTML page from the server for each navigation
  - D. SPAs cannot have multiple "pages" or views

Correct Answer: C

Question 2

- How is React best described?
  - A. Complete, opinionated framework that dictates your application's structure.
  - B. JavaScript library for building user interfaces.
  - C. Back-end language for managing servers.
  - D. CSS utility library for styling components.

Correct Answer: B

Question 3

- What is the main performance benefit of React's Virtual DOM?
  - A. It eliminates the need to write any JavaScript.
  - B. It directly manipulates the real DOM faster than vanilla JS.
  - C. It allows you to write HTML inside your JavaScript files.
  - D. It calculates the minimal changes in memory and updates the real DOM in one efficient batch, avoiding "DOM thrashing".

Correct Answer: D

Question 4

- How would you install vite in your React project?
  - A. npm install vite
  - B. npm i --save-dev vite
  - C. npm init vite
  - D. A & B

Correct Answer: D

Question 5

- A package in your package.json is listed with the version "^4.1.3". Which of the following new versions would be installed if you run npm install?
    - A. 4.2.0
    - B. 5.0.0
    - C. 4.1.2
    - D. 3.9.9

Correct Answer: A

Question 6

- What is the main purpose of the package-lock.json file?
  - A. To list a high-level overview of your project's main dependencies
  - B. To lock the exact versions of all installed packages and sub-dependencies, ensuring consistent installations across all environments
  - C. It's a file you manually edit to prevent certain packages from being updated
  - D. To speed up the npm install command by caching packages locally

Correct Answer: B