**Traditional Front-end Web Development**

The process of building web applications using technologies like HTML, CSS, and JavaScript without using any frameworks or libraries.

**Pain points**

* **State Management Complexity** — As the application grows, managing the state becomes increasingly difficult without a structured approach. This can lead to spaghetti code and make maintenance challenging.
* **Manual DOM Manipulation** — It requires us to manually manipulate the DOM to update the UI, which can be error-prone and time-consuming.
* **Lack of Reusability** — Writing repetitive code for similar UI elements, leading to code duplication. Scaling and maintaining the codebase can become problematic as the application grows.
* **Performance issues** — When dealing with frequent UI updates, the browser (real DOM) must recalculate the layout and repaint the affected elements every time. This process is known as reflow and repaint. It can cause performance problems as slowing down the application.

**Modern Front-end Web Development (React)**

React is a JavaScript library used for building user interfaces in web development.

**Component-Based Architecture**

User interface is divided into reusable and independent components. It enables us to build complex applications by composing smaller components. e.g. Header, Sidebar, Main Content and Footer

* **Modularity** — Components are self-contained. It means dividing the functionality into modules that handle specific tasks. Changes to one component without affecting the entire system. It also enables parallel development, as different teams or developers can work on different components simultaneously.
* **Reusability** — Components can be used in multiple parts of an application. This reduces duplication of code and improves productivity by allowing us to leverage existing components instead of starting from scratch.
* **Testing and Debugging** — Components can be tested independently, allowing for focused unit testing and targeted debugging. It reduces the scope of debugging and faster troubleshooting problems in specific component without affecting the entire system.

**Scalable State Management**

React's ecosystem provides various libraries and patterns for managing application state such as Redux and Context API. These solutions help us handle complex state requirements in scalable ways.

**Separation of Concerns** — The separation of **user interface (UI) logic** and **data (state) management**. This separation helps in organizing and maintaining the state as the application grows, making it easier to manage.

* **User Interface (UI) Logic** — Focus on rendering the task list, capturing user input, and displaying the information to the user. These components are concerned with how the tasks are displayed and their styling.
* **Data (state) Management** — Focus on handling the storage and retrieval of task data. The data management layer provides methods for adding, updating, deleting, and fetching task data. It abstracts the data operations and keep the UI components from dealing directly with data storage or retrieval.

**Virtual DOM**

When a React component is rendered, it creates a copy of the DOM called Virtual DOM. It is exactly the same, but it does not have the power to directly change the layout of the document. This tree is then compared with the previous virtual DOM tree. Once this is done, the virtual DOM calculates the best possible method to make these changes to the real DOM. This ensures that there are minimal operations on the real DOM. Hence, reducing the performance cost of updating the real DOM.

* **Efficient Updates** — When there are changes to the state or props of a component, React compares the previous Virtual DOM with the updated Virtual DOM. It identifies the differences between the two representations.
* **Minimal DOM Updates** — React determines the minimal set of changes required to update the actual DOM based on the identified differences. Instead of updating the entire DOM, it only modifies the specific elements that have changed.
* **Reconciliation** — React performs this process to efficiently update the real DOM. It reconciles the changes from the Virtual DOM and applies them to the actual DOM, ensure the UI reflects the updated state.