# React – Question & Answers

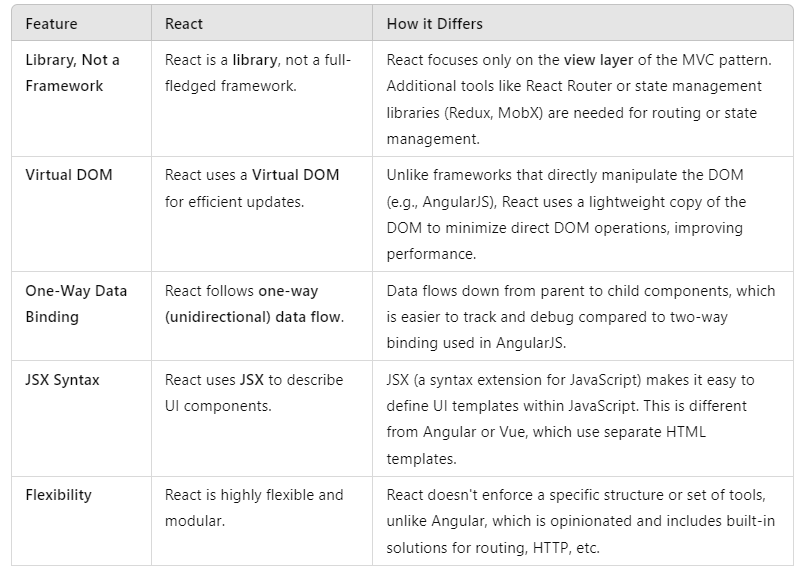
**Question 1:**What is React, and how does it differ from other front-end frameworks?

React is a popular open-source JavaScript library developed by Meta(formerly Facebook) for building user interfaces (UIs), especially for single-page applications. React allows developers to create reusable UI components that manage their own state and update efficiently when the data changes.

* Declarative:   
  React enables developers to describe how the UI should look for a given application state, and it takes care of efficiently updating and rendering the components when that state changes.
* Component-Based:   
  React breaks the UI into small, reusable components, making the code modular, easier to understand, and more maintainable.

* **How React Differs from Other Front-End Frameworks?**

React has unique features and design principles that distinguish it from traditional front-end frameworks like Angular, Vue, and others:



**Question 2:**What are the key benefits of using React in a web application?

* Component-Based Architecture:
* React breaks the UI into small, reusable components. Each component is a self-contained building block, managing its own logic and UI.
* 2.2. Virtual DOM for Performance:
* React uses a Virtual DOM, a lightweight copy of the actual DOM. When changes occur, React:
* Calculates the difference between the current and previous state of the Virtual DOM (called reconciliation).
* Updates only the parts of the real DOM that changed.
* 2.3. Cross-Platform Development:
* With tools like React Native, React code can be reused for building native mobile apps.
* This reduces development time and ensures consistent experiences across platforms.

**Question 3:**   
What is JSX, and how is it used in React?  
  
JSX (JavaScript XML) is a syntax extension for JavaScript used in React to describe what the UI should look like. It combines the power of JavaScript with an XML-like syntax, making it easier to create and visualize React components.

JSX is **not required** to use React, but it is widely adopted because it simplifies writing components and improves code readability.

Behind the scenes, JSX is transpiled into standard JavaScript using tools like Babel.

Example:

  
  
is transpiled to:



* **3.2. Embedding HTML in JavaScript:**  
   JSX allows developers to write HTML-like syntax directly inside JavaScript.

Example:



* **3.3. Supports JavaScript Expressions:**

You can embed JavaScript expressions inside JSX using curly braces {}.

Example:



* **3.4. Why Use JSX?**
* **Improves Readability**: Combines HTML-like syntax with JavaScript, making it easier to visualize the structure of components.
* **Tighter Integration**: Directly integrates JavaScript logic with the UI.
* **React-Specific Optimizations**: JSX is optimized for React, making the development process smoother and more efficient.

**Question 4:**What is the virtual DOM, and how does it improve performance in React?

The Virtual DOM (Document Object Model) is a lightweight, in-

memory representation of the real DOM. It is an abstraction that React uses to optimize updates to the actual DOM, which is often slow to manipulate directly.

**Real DOM**: The traditional browser DOM that represents the UI elements and structure on a web page.  
  
**Virtual DOM**: A JavaScript object that mirrors the structure of the real DOM, enabling React to track changes efficiently.

* **How Does the Virtual DOM Work?**
* 4.1.Initial Rendering:
* When a React application is rendered for the first time, React creates a virtual representation of the DOM (the Virtual DOM).
* 4.2. Updating the Virtual DOM:
* When the state or props of a React component change, React updates the Virtual DOM to reflect the changes.
* React creates a **new Virtual DOM** for the updated state while keeping the old Virtual DOM for comparison.
* 4.3. Updating the Real DOM:
* React applies the calculated changes (patch) to the real DOM in an optimized way, ensuring only the necessary parts of the UI are updated.

**Question 5:**  
What are React components, and how do you create and manage them?

React components are the building blocks of a React application. They encapsulate a piece of the UI and its behavior, making the application easier to build, maintain, and scale.

A component in React can be thought of as a reusable, self-contained piece of UI.

Components accept props (inputs) and manage their own state to render dynamic content.

* 5.1. **Types of React Components:**

React supports two main types of components:

* 5.1.1. Functional Components:
* These are simple JavaScript functions.
* They take props as input and return JSX (the UI).
* Since React 16.8, functional components can also manage state and side effects using **React Hooks**.

Example:

תמונה שמכילה טקסט, גופן, צילום מסך, קו

התיאור נוצר באופן אוטומטי

* 5.1.2. Class Components:  
  Class components are still valid but less frequently used in newer codebases.

Example:

תמונה שמכילה טקסט, גופן, צילום מסך, קו

התיאור נוצר באופן אוטומטי

* 5.2. **Props and State in Components:**
* 5.2.1. Props (Properties):
* Props are used to pass data from a parent component to a child component.
* They are immutable (cannot be modified by the child component).

Example:  
תמונה שמכילה טקסט, גופן, צילום מסך, לבן

התיאור נוצר באופן אוטומטי

Usage:

<Greeting name="John" />

* 5.2.2 State:
* State is data that a component manages internally.
* It is mutable (can be updated).

Example (using Hooks):

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

* 5.3. **Managing React Components:**
* 5.3.1. Props Validation:
* Use PropTypes to ensure props have the correct type and structure.

Example:



* 5.3.2. State Management:
* Use React Hooks like useState and useReducer for local state.
* For global state, consider libraries like Redux or React Context.

**Question 6:**  
How does React handle state changes & updates to the user interface?

* **State in React:**

State is a built-in object that React components use to store data that changes over time. Whenever the state of a component changes, React triggers a re-render to update the user interface (UI) to reflect the new state.

* **React's State Management Workflow:**
* State Initialization:
* State is initialized when the component is first rendered.
* In class components, state is defined using this.state
* In functional components, the useState Hook is used.

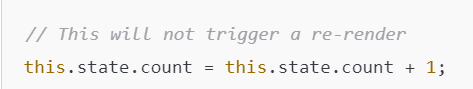
Example:



* State Modification:
* State changes are triggered using methods like this.setState (class) or the state updater function from useState (functional).
* React ensures that these updates are asynchronous, meaning multiple updates can be batched together to improve performance.
* React's Reconciliation:
* React uses a process called **reconciliation** to determine the minimal number of changes needed to update the DOM.
* It compares the **current virtual DOM** with a new version created after the state change.

* **Key Features of State Updates in React:**
* Immutability:
* React's state is immutable, meaning it should not be modified directly. Instead, a new version of the state is created.

Example of incorrect modification:

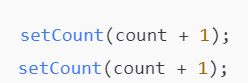


Correct approach:

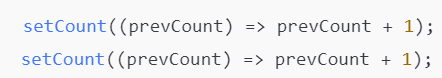


* Asynchronous Updates:
* React batches state updates for performance optimization.

Example:



This may not behave as expected because React may batch updates, so count might not reflect the latest value. Use a function-based update to resolve this:



* Re-rendering:
* React triggers a re-render of the component whenever state or props change.
* React only re-renders the parts of the UI that depend on the changed state, thanks to its virtual DOM.

* **State Change Lifecycle:**
* Trigger:
* State change is triggered via setState or useState.
* Virtual DOM Update:
* React creates a new virtual DOM tree based on the new state.
* Reconciliation:
* The new virtual DOM is compared to the previous one.
* DOM Updates:
* React efficiently applies the necessary changes to the actual DOM.
* Re-render:
* The component re-renders with the updated UI.
* **Best Practices for Handling State Changes:**
* Use Functional Updates:
* When the new state depends on the previous state, use the functional form of setState:



* Avoid Mutating State Directly:
* Always return a new object or array when updating state.
* Split State:
* Break down state into smaller, manageable pieces for better performance and readability.
* Minimize Re-renders:
* Use React.memo or useCallback to optimize performance when needed.
* Use Controlled Components:
* Manage form inputs and other UI elements with React's state.

**Question 7:**  
What is the concept of a “single source of truth” in React, and how does it relate to state management?

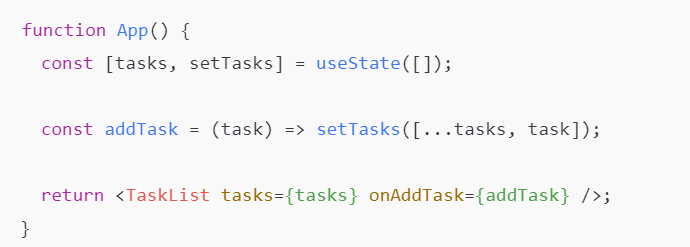
The "single source of truth" is a principle in React (and software design in general) that emphasizes maintaining a single, authoritative data source for the state of your application. This ensures consistency and clarity in how data is managed and used.

In React, the state of a component or application often serves as the "single source of truth." The state is stored in a specific location (e.g., in the useState hook or a global state management tool like Redux or Context API), and all components or parts of the application reference this state to derive their behavior or display.

* **How It Relates to State Management:**
* Local State as a Single Source of Truth:
* In small applications, the local state (managed using useState or this.setState) in a specific component can act as the single source of truth.

Example:  


* Here, the count state is the single source of truth. The UI always reflects its current value.
* Shared State in Parent Components:
* For larger applications, shared state is often lifted to a parent component to act as the single source of truth for multiple child components.

Example:  


**Question 8:**Can you explain the difference between a controlled and uncontrolled component in React?

In React, controlled components and uncontrolled components refer to how form data is managed within the application. The distinction lies in whether React directly controls the form data or if the DOM handles it.

* **Controlled Components:**A controlled component is a form element (e.g., input, textarea, select) where React manages the value through its state. The form element's value is bound to a React state variable, and any updates to the input's value trigger a change in state.
* **Uncontrolled Components:**An uncontrolled component is a form element where the DOM itself manages its state. React does not directly control the value of the input; instead, you access the value using references (ref).

A screenshot of a computer

Description automatically generated  
  
Controlled components provide more control and are ideal for complex forms where React state needs to reflect the form’s values. Uncontrolled components are simpler to implement and useful for scenarios where React doesn’t need to manage the form data explicitly. Choosing between the two depends on the requirements of your application.