**📘 Chapter 4 – Talk is Cheap, Show me the Code**

**✅ 1. Is JSX mandatory for React?**

* ❌ No.
* React works with React.createElement() directly.
* JSX just makes writing React code **simpler and more readable**.

A screenshot of a computer screen

AI-generated content may be incorrect.

**✅ 2. Is ES6 mandatory for React?**

* ❌ No.
* But most React codebases use **ES6 features** like import/export, arrow functions, let/const.
* Without ES6, writing React code becomes verbose.

**✅ 3. {TitleComponent} vs {<TitleComponent/>} vs {<TitleComponent></TitleComponent>} in JSX**

* {TitleComponent} → Refers to the component **function itself** (not rendered).
* {<TitleComponent/>} → Executes the function → returns JSX → renders component.
* {<TitleComponent></TitleComponent>} → Same as above, longer syntax.

**✅ 4. How can I write comments in JSX?**

{/\* This is a comment in JSX \*/}

**✅ 5. What is <React.Fragment></React.Fragment> and <> </>?**

* A **Fragment** lets you group multiple JSX elements **without adding extra divs** in DOM.
* ✅ Better for styling (CSS doesn’t get messed with unwanted divs)  
  ✅ Slight performance gain

📌 Example:

A screenshot of a computer program

AI-generated content may be incorrect.

**✅ 6. What is Reconciliation in React?**

* **Reconciliation** = React’s process of **updating the DOM efficiently**.
* It compares the **Virtual DOM** with the previous one and only updates changed parts.

**✅ 7. What is React Fiber?**

* React Fiber = The **new reconciliation algorithm** (since React 16).
* Helps React do updates in a **non-blocking, incremental way** → smoother UI.

**✅ 8. Why do we need keys in React?**

* **Keys help React identify elements uniquely** when rendering lists.
* Without keys, React re-renders everything → slower.
* With keys, React updates only changed items.

📌 Example:

A computer code on a black background

AI-generated content may be incorrect.

**✅ 9. Can we use index as keys in React?**

* ✅ Possible, but ❌ not recommended.
* Problem: If items **reorder/add/delete**, React gets confused → wrong UI updates.
* Example: A todo list → when one item is deleted, indexes shift → React mismatches.
* Better: Use a **unique id** from data for consistent rendering.

**✅ 10. What is props in React? Ways to use it.**

* **Props (Properties)** = Way to **pass data from parent to child component**.
* Read-only (cannot be changed by child).

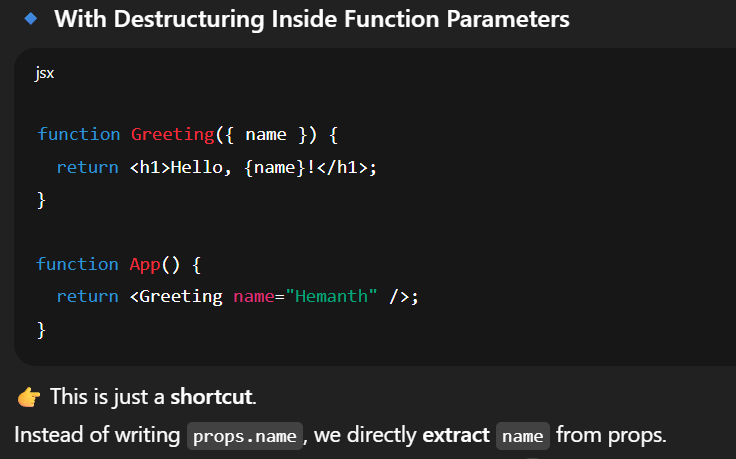
📌 Example:

A computer screen shot of a code

AI-generated content may be incorrect.

✅ Ways:

1. Pass as attributes → <Comp name="John" />
2. Destructure inside → function Comp({ name }) {}
3. Pass objects/arrays/functions as props.



**✅ 11. What is Config Driven UI?**

[**Config** : Config files are files that store instructions often in JSON to control how an application works without changing its code]

* Instead of hardcoding UI, you generate it dynamically from **JSON config** (from API).

(**In simple words:**👉 Instead of writing UI elements directly in code, we define a **config file (data structure)**, and React builds the UI from that config)

* Useful when UI depends on backend response.
* Example: Amazon product cards come from config, not hardcoded

📌 Example:

A screen shot of a computer program

AI-generated content may be incorrect.

**🔹 Reconciliation vs Fiber**

* Reconciliation → React’s **process** of updating DOM efficiently.
* Fiber → React’s **new architecture** for reconciliation (from React 16).
* Fiber makes UI **smoother** because it can **pause, prioritize, and resume rendering** tasks.

✅ Interview Answer: Reconciliation is the *what*, Fiber is the *how*.

**🔹 Props vs State (Trick question)**

* **Props** → data from parent (read-only).
* **State** → data inside component (can change).

**Digital Notes Unclear Concepts :**

**✅ Optional chaining -**

“Optional chaining (?.) allows us to safely access nested object properties. If a property doesn’t exist, it returns undefined instead of throwing an error. It’s especially useful in React when dealing with uncertain API data.”

