**Lists , keys , Api Project**

**Question 1: How do you render a list of items in React? Why is it important to use keyswhen rendering lists?**

**✅ Rendering a List in React**

To render a list of items in React, you typically use the .map() function to iterate over an array and return a JSX element for each item.

**const fruits = ['Apple', 'Banana', 'Orange'];**

**function FruitList() {**

**return (**

**<ul>**

**{fruits.map((fruit, index) => (**

**<li key={index}>{fruit}</li>**

**))}**

**</ul>**

**);**

**}**

### 🔑 Why Use ****Keys**** When Rendering Lists?

**Keys** help React identify which items have changed, been added, or removed. This improves performance and ensures the **correct behavior** during re-renders.

#### Importance of Keys:

* 🧠 **Efficient Reconciliation:** React uses keys to match elements in the virtual DOM with elements in the real DOM.
* 🛠️ **Preserves Component State:** If the list changes (e.g., reordering), keys prevent loss of local state (like form inputs or animations).
* ⚠️ **Avoid Index as Key When Possible:** Using index as a key can cause issues if items are reordered or removed, leading to unexpected UI bugs.

**Question 2: What are keys in React, and what happens if you do not provide a unique key?**

**🔑 What Are Keys in React?**

In React, **keys** are special string attributes you add to list elements when creating components dynamically. They help React **identify** which items have changed, been added, or removed in a list.

### 💡 Why Are Keys Important?

* **Optimize Rendering:** React uses keys to **efficiently update** the DOM by reusing existing elements where possible.
* **Track Component State:** Keys help preserve the **component's local state** (e.g., form inputs, animations) when the list changes.
* **Prevent Bugs:** Without unique keys, React may incorrectly **reuse or reorder components**, causing unexpected behavior.

### ⚠️ What Happens If You Don't Provide a Unique Key?

1. **Performance Issues:** React has to guess how to update elements, which can lead to inefficient re-renders.
2. **UI Bugs:** Components may lose their state or display incorrect data due to incorrect element matching.
3. **Console Warnings:** React will show a warning like:

**Context API**

**Question : What is the Context API in React? How is it used to manage global state acrossmultiple components?**

### What is the Context API in React

The **Context API** is a built-in feature in React that allows you to **share state or data globally** across multiple components — without having to pass props manually at every level (called "prop drilling").

It’s commonly used for:

* Theme settings (light/dark mode)
* User authentication state
* Language/localization settings
* Any global state shared by many components

### 🌐 Why Use Context for Global State

* 🔁 **Avoids Prop Drilling:** You don't have to pass props through every nested component.
* 📦 **Centralizes Shared State:** Easier to manage and update global data.
* 🧼 **Cleaner Code:** Components stay decoupled and focused on their logic.

**Question 2: Explain how createContext()and useContext()are used in React forsharing state.**

### 📌 What is createContext()

createContext() is a function provided by React that creates a **Context object**. This object allows you to **share state or data** across components without passing props manually at every level (i.e., avoiding prop drilling).

### 📌 What is useContext()

useContext() is a React hook that allows a component to **access the current value** of a Context. It must be used **inside a component that is a descendant** of the corresponding Context.Provider.

### 📍 Use Case Example:

* Managing a logged-in user's information across multiple components
* Sharing theme settings (light/dark mode)
* Passing down app-wide configurations like language or layout options