**[ GAMECHANGER TIPS AND TRICK 🚀] REACT JS**

**JavaScript Topic use in React.js**

* **Array**
* **Objects**
* **Arrow Function**
* **Map**
* **Filter**
* **Reduce**
* **Mutability and Immutability**
* **Destructuring**

**React.js Topics**

* **Component:** A **component** is a reusable piece of code in React used to build the UI, like a **Navbar**, **Sidebar**, or **Card**. It helps organize the UI into independent, manageable pieces.  
  Components can store their own data using **useState**.
* **Map:** The **map()** function is used to **render lists of components dynamically**. For example, if you want to display 10 cards, you can map over an array of data and render a card for each item.
* **Data Usage - JSX – Dataflow:**
* **JSX (JavaScript XML)** looks like HTML but allows JavaScript expressions using {}

Example: <p>{2 + 2}</p> // Output: 4

* **Dataflow in React is one-way**: data flows **from parent to child** through **props**.
* **Props: Props (Properties)** are used to **pass data or functions from one component to another**, usually from **parent to child**.

**Example use case:**

* Component A sends data to Component B via props.
* B can pass that data to its child C.

If too many levels are involved, this becomes hard to manage. To solve this, we use **Redux** or **Context API**.

* **State - useState: State** is the data **local to a component**. You use **useState** to create and update state. When state changes, the component **automatically re-renders** with the new data.
* **Effect – useEffect:** useEffectruns after the component mounts (i.e., after it appears on the screen).  
  It’s commonly used for:
* Fetching data from an API
* Running code only once when the page loads
* Updating something when a state or prop changes
* **Redux (At the end): Redux** is a state management tool used when **many components need access to the same data**. Instead of passing data down through multiple levels using props, Redux provides a **central store** for global data access. It simplifies the flow of data in large applications.
* **Context API (At the end):** The Context API is a built-in feature in React that allows you to share data between components without passing props manually at every level. It's like a shortcut for props, especially helpful when data needs to be accessed by many components (like themes, language, user info, etc.).