Thinking in React: State Management

Summary: We will learn

* **State management**
* **When** and **where** to **create state**
* When and how to **derive states**
* **lifting up state**: how to communicate between the parent and child components by lifting up state

**Thinking In React as a Process:**

1. Break the desired UI into components and establish the component tree.
2. Build a static version of React (Without react)
3. Think about state:
   1. When to use state
   2. Type of state: local vs global
   3. Where to place each piece of state
4. Establish data flow:
   1. One-way data flow
   2. Child-to-parent communication
   3. Accessing global state

3 and 4 is known as State Management

**When you know how to “Think in React”, you will be able to answer:**

* How to break up a UI design into components?
* How to make some components reusable?
* How to assemble UI from reusable components?
* What pieces of state do I need for interactivity?
* Where to place state? (What component should “own” each piece of state?)
* What types of state can or should I use?
* How to make data flow through app?

State Management: Can be explained as Deciding when to create pieces of state, what types of state are necessary, where to place each piece of state, and how data flows through the app.

**Types of State: Local VS Global state**

|  |  |
| --- | --- |
| **Local State** | **Global State** |
| State needed **only by one or few components** | State that **many components** might need |
| State that is defined in a component and **only that component and child component** have access to it (by passing via props) | Shared state that is accessible to **every component** in the entire application |
|  | We can define global state using React’s Context API or aan external state management library “Redux” |
| Eg: Input text in the search box, only that component needs to know about that data. Therefore it Is local state | Eg: The shopping cart need to be accessed by all the other components |

We should always start with local state

State: **When** and **Where?**

**Image in phone (19-sept-2024)**

**Thinking about state and lifting state up**

Lifting state Up: Whenever multiple sibling need the access to the same state, we move that piece of state up to the first common parent

**Derived State :** State that is computed from an existing piece of state or from props

See in=mage in phone (Capture)

Then “children” props:

Eg

