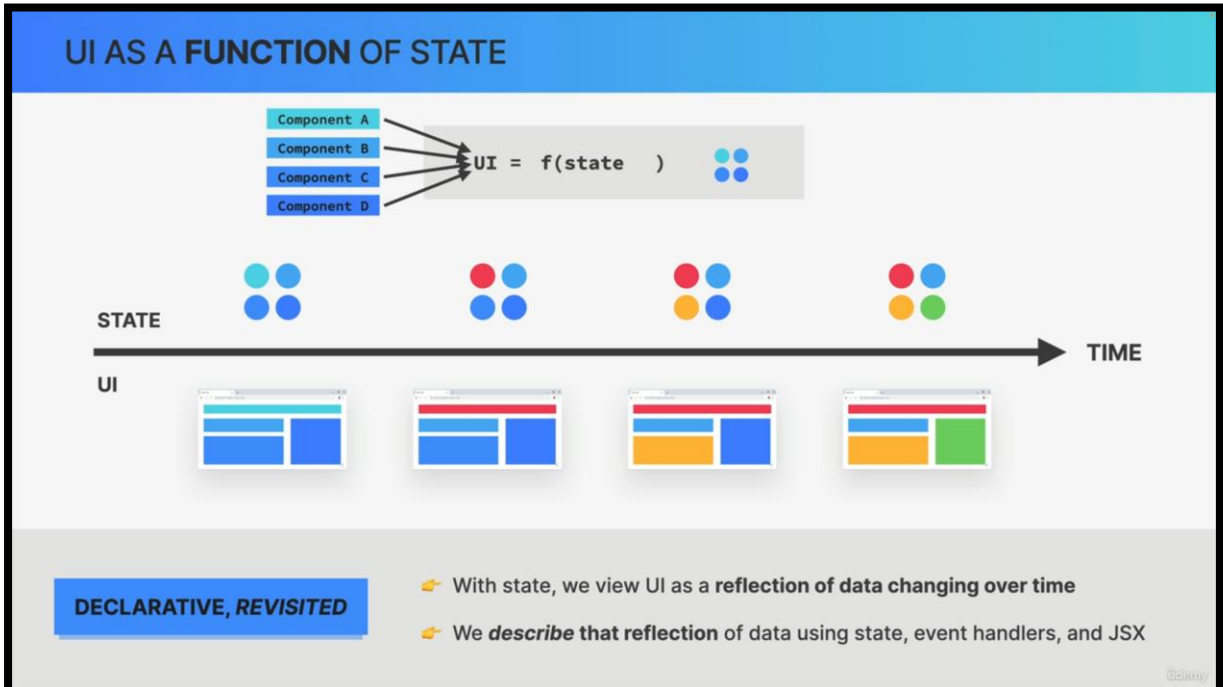


We can basically think of the entire application view, the entire user interface, as a function of state. In other words, the entire UI is always a representation of all the current states in all components.



A React application is fundamentally all about changing state over time, and correctly displaying that state at all times.

This is really what the declarative approach to building user interfaces is all about. Instead of viewing a UI as explicit DOM manipulations, with state, we now view a UI as a reflection of data changing over time.

We describe that reflection of data using state, event handlers, and JSX.

IN PRACTICAL TERMS...

PRACTICAL GUIDELINES ABOUT STATE

- 👉 Use a state variable for any data that the component should keep track of ("remember") over time. **This is data that will change at some point.** In Vanilla JS, that's a `let` variable, or an `[]` or `{}`
- 👉 Whenever you want something in the component to be **dynamic**, create a piece of state related to that "thing", and update the state when the "thing" should change (aka "be dynamic")
 - 👉 **Example:** A modal window can be open or closed. So we create a state variable `isOpen` that tracks whether the modal is open or not. On `isOpen = true` we display the window, on `isOpen = false` we hide it.
- 👉 If you want to change the way a component looks, or the data it displays, **update its state**. This usually happens in an **event handler** function.
- 👉 When building a component, imagine its view as a **reflection of state changing over time**
- 👉 For data that should not trigger component re-renders, **don't use state**. Use a regular variable instead. This is a common **beginner mistake**.