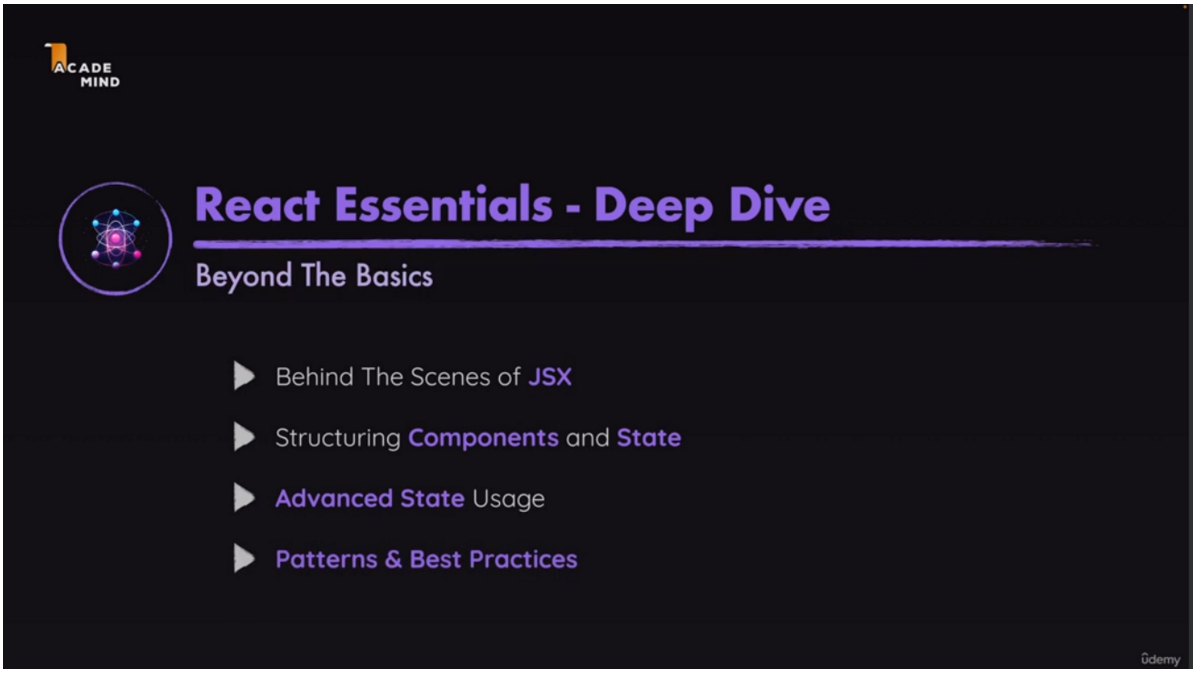
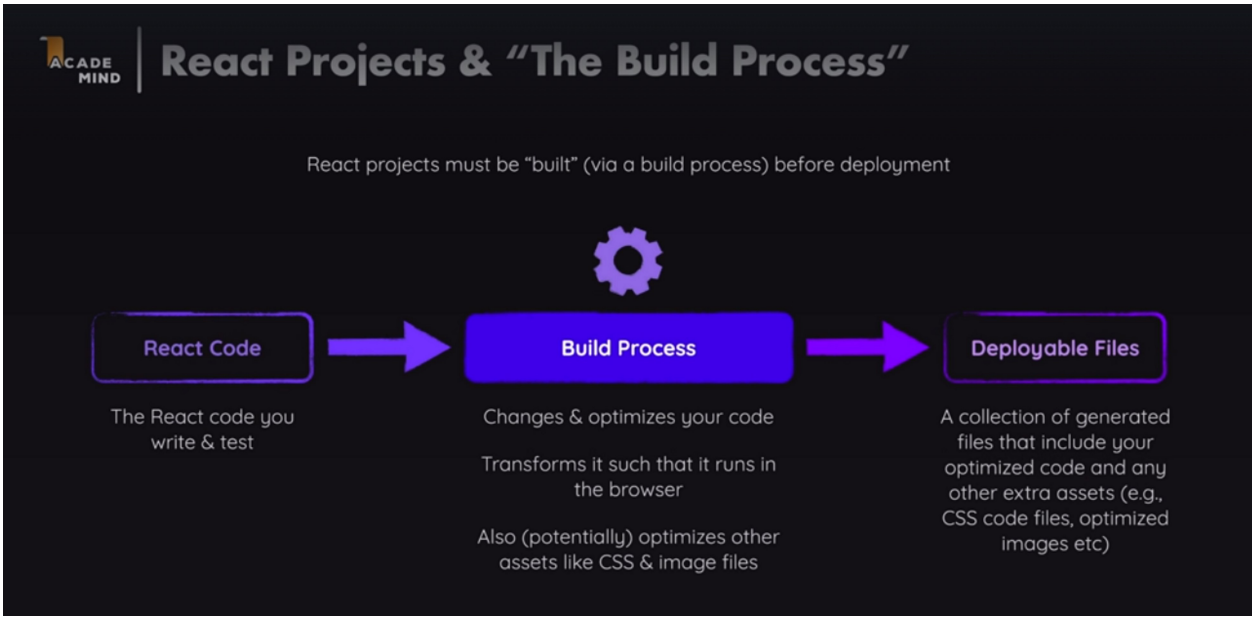
Section 4: React Essentials - Deep Dive

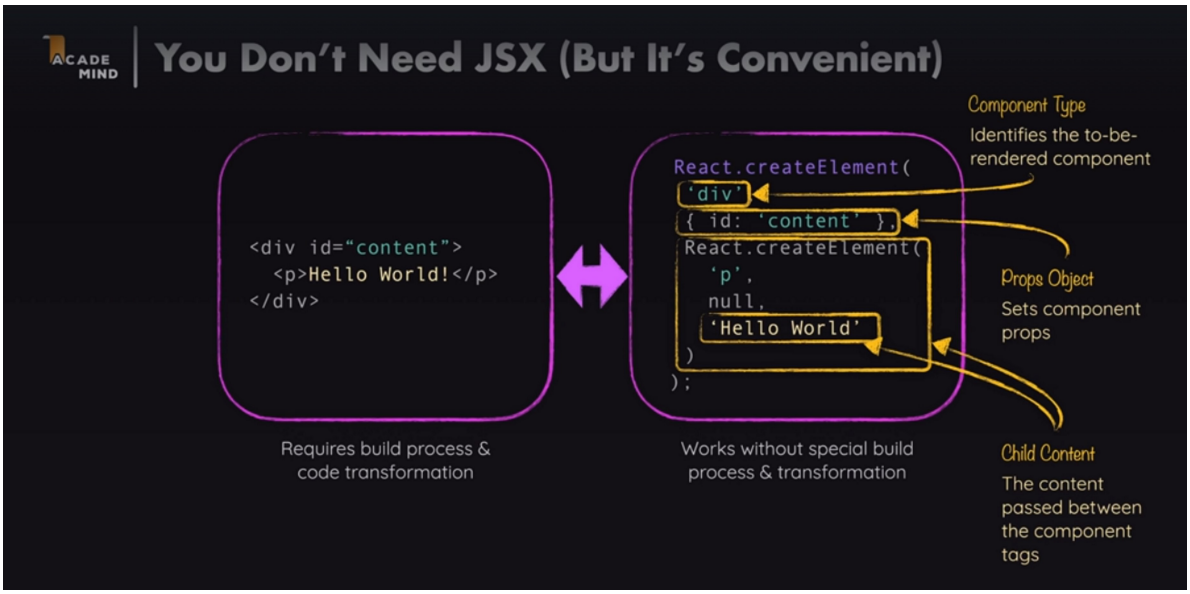
19 September 2024 09:00



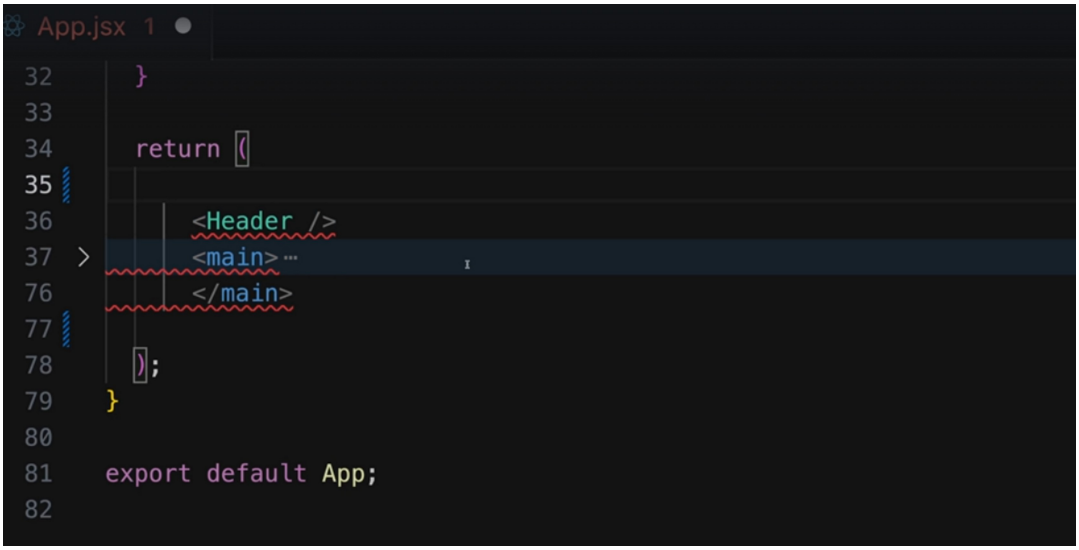
Now we'll get started in this section by taking another look at this JSX code, this HTML in JavaScript code about which you learned in the last course section. Because there I mentioned that this JSX code, of course, is a non-standard feature. It's not supported by the browser
and therefore the code you write in your code files here is not the code that ends up in the browser. Instead, as you learned, we got a build process.



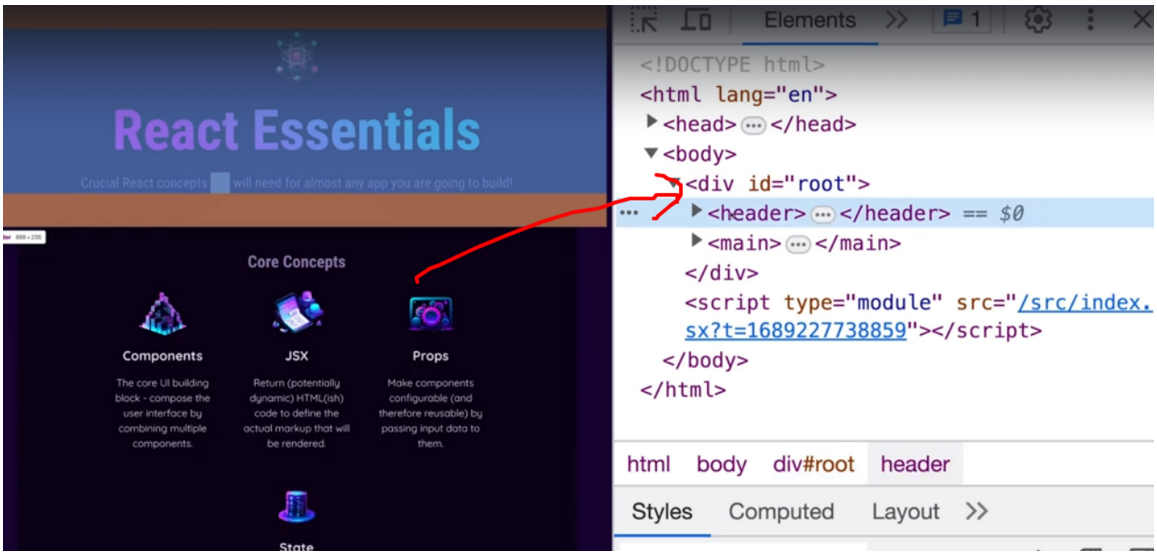
You could also build React Apps without using JSX.



Working With Fragments



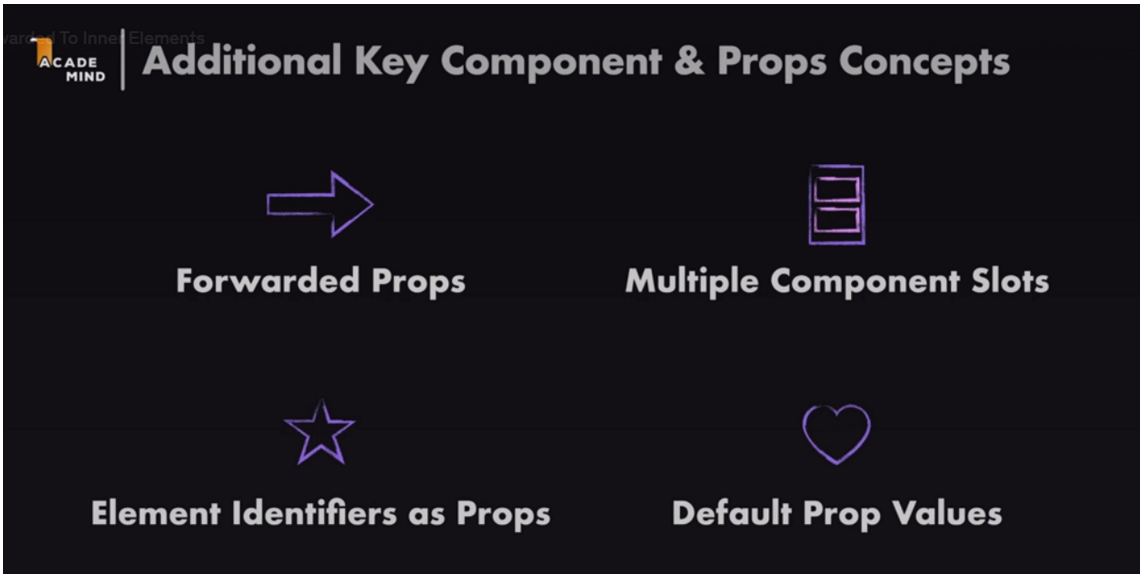
you can't return multiple items in a return statement.

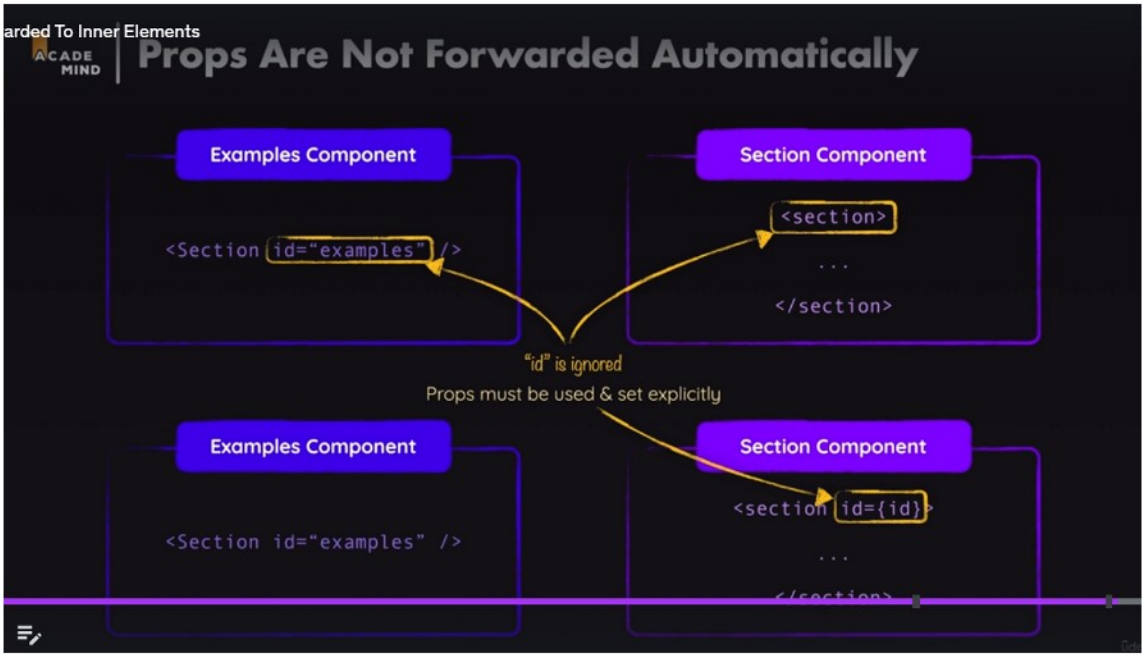


no extra div!

Splitting Components By Feature & State

for preventing unnecesary Header and CoreConcepts Component rendering, we have split the components.



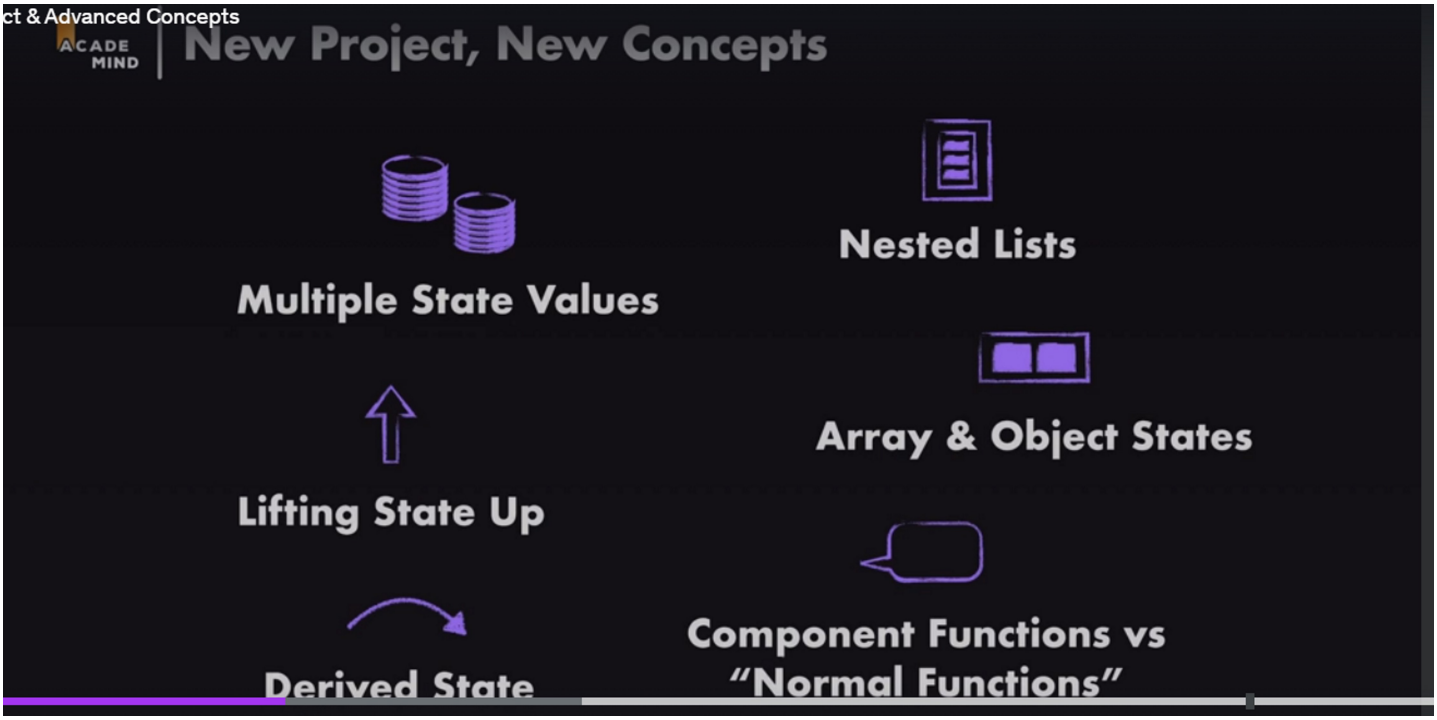


Forwarding Props To Wrapped Elements

```
return (
    <Section title="Examples" id="examples">
        <menu>
            <TabButton
                isSelected={selectedTopic === 'components'}
                onClick={() => handleSelect('components')}>
                Components
            </TabButton>
            <TabButton
                isSelected={selectedTopic === 'jsx'}
                onClick={() => handleSelect('jsx')}>
                JSX
            </TabButton>
            <TabButton
                isSelected={selectedTopic === 'props'}
                onClick={() => handleSelect('props')}>
                Props
            </TabButton>
```

But this three dots thing here is built into JavaScript and basically tells JavaScript to collect all other props
that might be received on this section component and merge them into a props object. So into one JavaScript object

Onwards To The Next Project & Advanced Concepts



Updating State Based On Old State Correctly

Well, the problem with this code here is that React behind the scenes is, in the end, scheduling those state updates
you're triggering with those state updating functions, like setIsEditing. So this state update here is not performed instantly.
Instead, it's scheduled by React to be performed in the future. Now, the future will probably be in one or two milliseconds.

So it's really fast, but it's not instant. That's important. Now, as a React developer, it's really important to understand that React behaves that way and that behavior also explains why when you copy this state updating function
and you paste it in immediately after the first state updating function, you might not get the result you might expect.

React Is Scheduling State Updates

Somewhere in your code
Code snippets in different places in your app's code

React's State Updating Schedule
Managed behind the scenes by React

```
setIsEditing(true);  ──→ 1 → Update state to true
...
setIsEditing(false); ──→ 2 → Update state to false
...
setIsEditing(true);  ──→ 3 → Update state to true
...
setIsEditing(false); ──→ 4 → Update state to false
```

State updates are **not performed instantly** but at some point in the future
(when React has time for it)

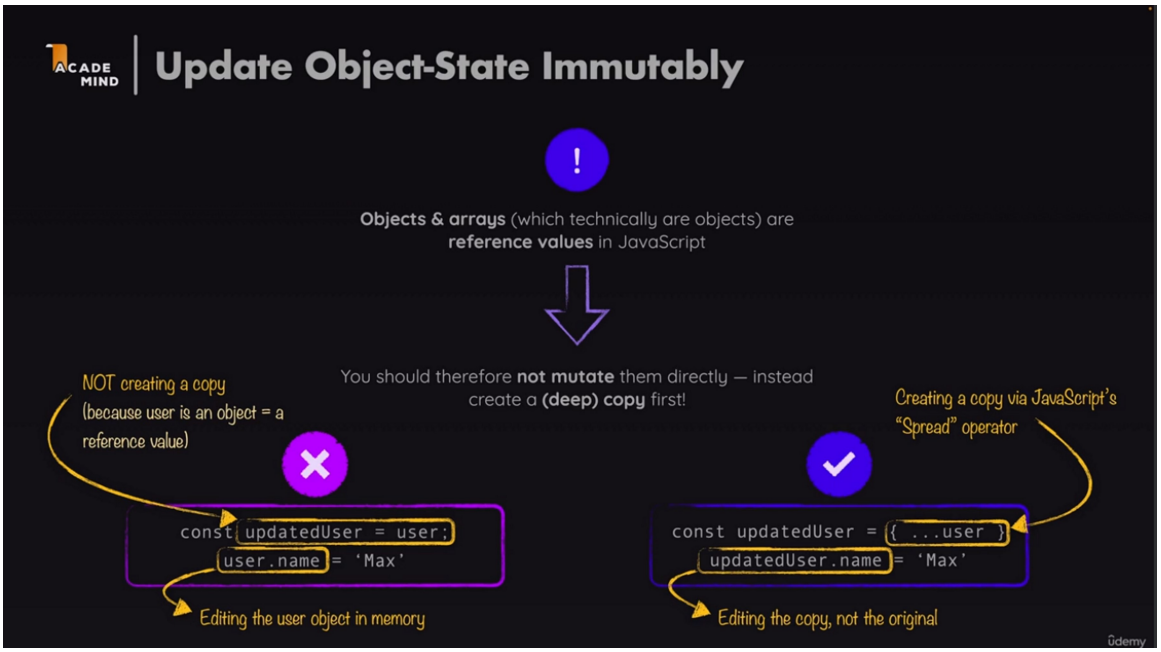
In most cases, those state updates of course still are executed **almost instantly**

udemy

```jsx
export default function Player({ name, symbol }) {
  const [isEditing, setIsEditing] = useState(false);

  function handleEditClick() {
    setIsEditing(!isEditing); // => true
    setIsEditing(!isEditing); // => false
  }

  let playerName = <span className="player-name">{name}</span>;
  // let btnCaption = 'Edit';

  if (isEditing) {
    playerName = <input type="text" required value={name} />;
    // btnCaption = 'Save';
  }
```

false

false




```
const GameBoard = () => {
    const [gameBoard, setGameBoard] = useState(initialGameBoard);

    function handleSelectSquare(rowIndex, colIndex) {
        //not recommended.
        setGameBoard((prevGameBoard) => {
            prevGameBoard[rowIndex][colIndex] = 'X';
            return prevGameBoard;
        });
    }
```

you're dealing with a reference value in JavaScript. and therefore if you would be updating it like this, you would be updating the old value in -memory immediately, even before this scheduled state update was executed by React. And this can again lead to strange bugs or side-effect.

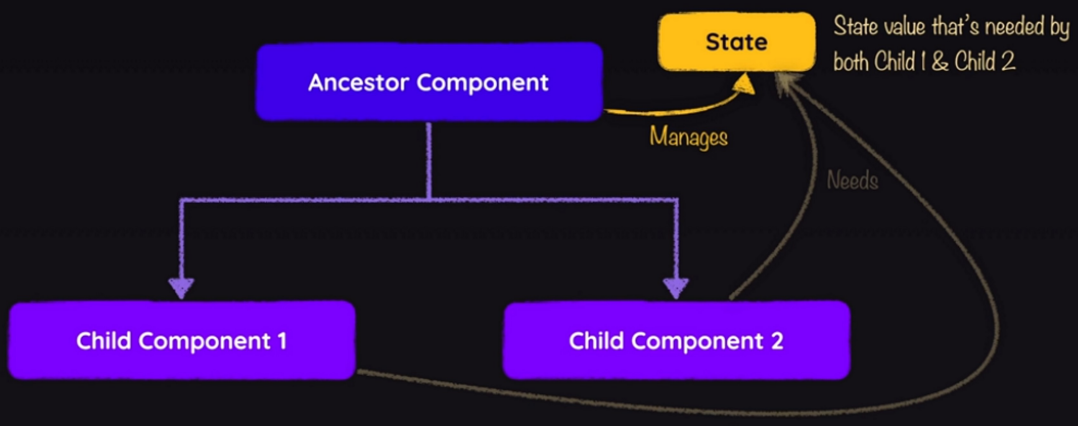
https://academind.com/tutorials/reference-vs-primitive-values

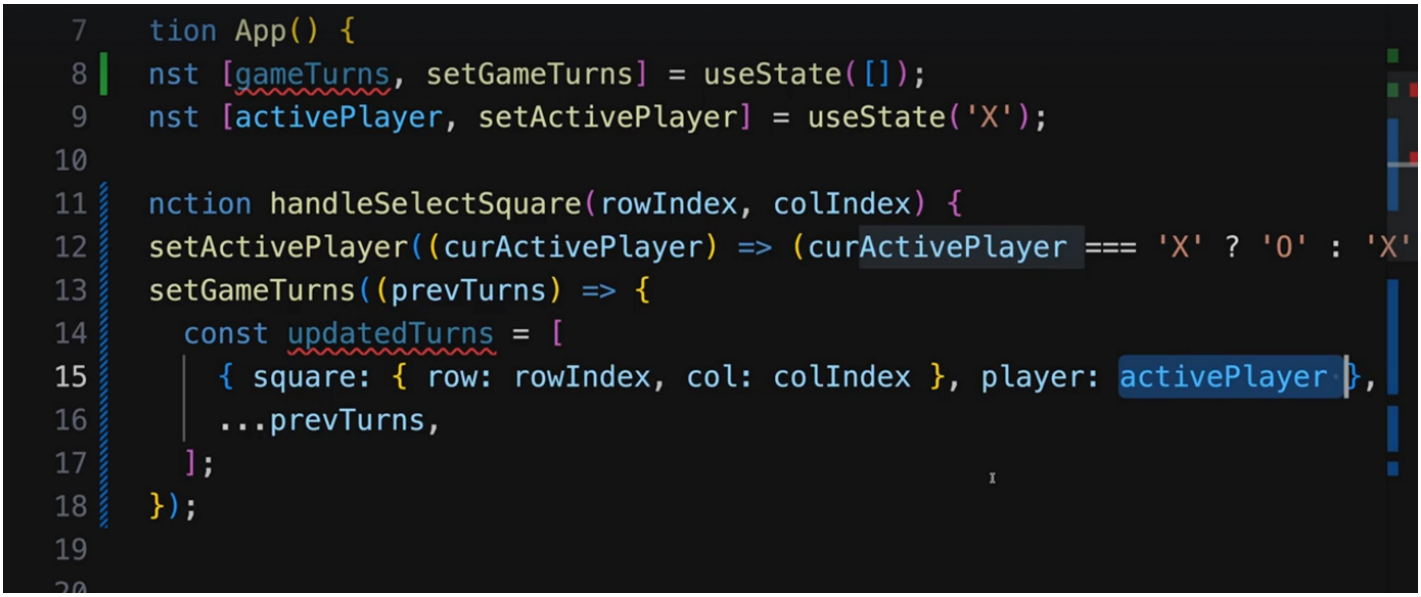
Lifting State Up

Lift the state up to the closest ancestor component that has access to all components that need to work with that state

Lifting State Up

Lift the state up to the closest ancestor component that has access to all components that need to work with that state




tion App() {
nst [gameTurns, setGameTurns] = useState([]);
nst [activePlayer, setActivePlayer] = useState('X');
nction handleSelectSquare(rowIndex, colIndex) {
setActivePlayer((curActivePlayer) => (curActivePlayer === 'X' ? 'O' : 'X'
setGameTurns((prevTurns) => {
const updatedTurns = [
{ square: { row: rowIndex, col: colIndex }, player: activePlayer },
...prevTurns,
];
});