Efficient React State Management

WHICH STATE MANAGEMENT IN REACT HE WILL RENDER JUST THE RELEVANT TREE AND NOT ALL THE TREE GIVE MORE THEN 1 STORE

To achieve state management in React where only the relevant parts of the component tree re-render and not the entire tree, while using more than one store, you have a few options:

1. Zustand

- **Description:** Zustand is a small, fast, and scalable state management library. It allows you to create multiple stores, and only components that use specific slices of the state will re-render when those slices change.
- **Features:**
  - Multiple stores: You can create as many stores as needed.
  - Selectors: You can select specific parts of the state to avoid unnecessary re-renders.
  - No boilerplate: It’s minimalistic and easy to integrate.
- **Usage Example:**

```javascript
import create from 'zustand';

const useStore1 = create(set => ({
  state1: 0,
  increment: () => set(state => ({ state1: state.state1 + 1 })),
}));

const useStore2 = create(set => ({
  state2: 'hello',
  setState2: (newState) => set({ state2: newState }),
}));

function Component() {
  const state1 = useStore1(state => state.state1);
  const state2 = useStore2(state => state.state2);
  return (
    <div>
```

```
      {state1} - {state2}
    </div>
  );
}
```

2. Recoil

- **Description:** Recoil is a state management library for React that provides fine-grained control over state updates. It allows for multiple "atoms" or pieces of state that can be independently managed.
- **Features:**
  - Fine-grained reactivity: Components only re-render when the specific atoms they depend on change.
  - Asynchronous state: It supports asynchronous data fetching and other async operations.
  - Multiple atoms: You can create as many atoms as needed for different parts of your state.
- **Usage Example:**

javascript

```javascript
import { atom, useRecoilState } from 'recoil';

const state1Atom = atom({
  key: 'state1',
  default: 0,
});

const state2Atom = atom({
  key: 'state2',
  default: 'hello',
});

function Component() {
  const [state1, setState1] = useRecoilState(state1Atom);
  const [state2, setState2] = useRecoilState(state2Atom);
  return (
    <div>
      {state1} - {state2}
    </div>
  );
}
```

3. Jotai

- **Description:** Jotai is another minimalistic state management library that allows for multiple atoms or stores. Components will only re-render when the specific atoms they are using change.
- **Features:**
  - Minimal API: It's designed to be simple and intuitive.
  - Multiple atoms: You can create separate atoms for different parts of your state.
  - Fine-grained updates: Components re-render only when their associated atoms change.
- **Usage Example:**

```javascript
import { atom, useAtom } from 'jotai';

const state1Atom = atom(0);
const state2Atom = atom('hello');

function Component() {
  const [state1, setState1] = useAtom(state1Atom);
  const [state2, setState2] = useAtom(state2Atom);
  return (
    <div>
      {state1} - {state2}
    </div>
  );
}
```

Summary:

- **Zustand** is great for simple, scalable state management with multiple stores.
- **Recoil** is excellent if you need advanced features like asynchronous state or want fine-grained control over your state.

- **Jotai** offers a minimalistic approach to managing state with multiple atoms.

Each of these libraries ensures that only the relevant parts of the component tree re-render, helping to optimize performance.