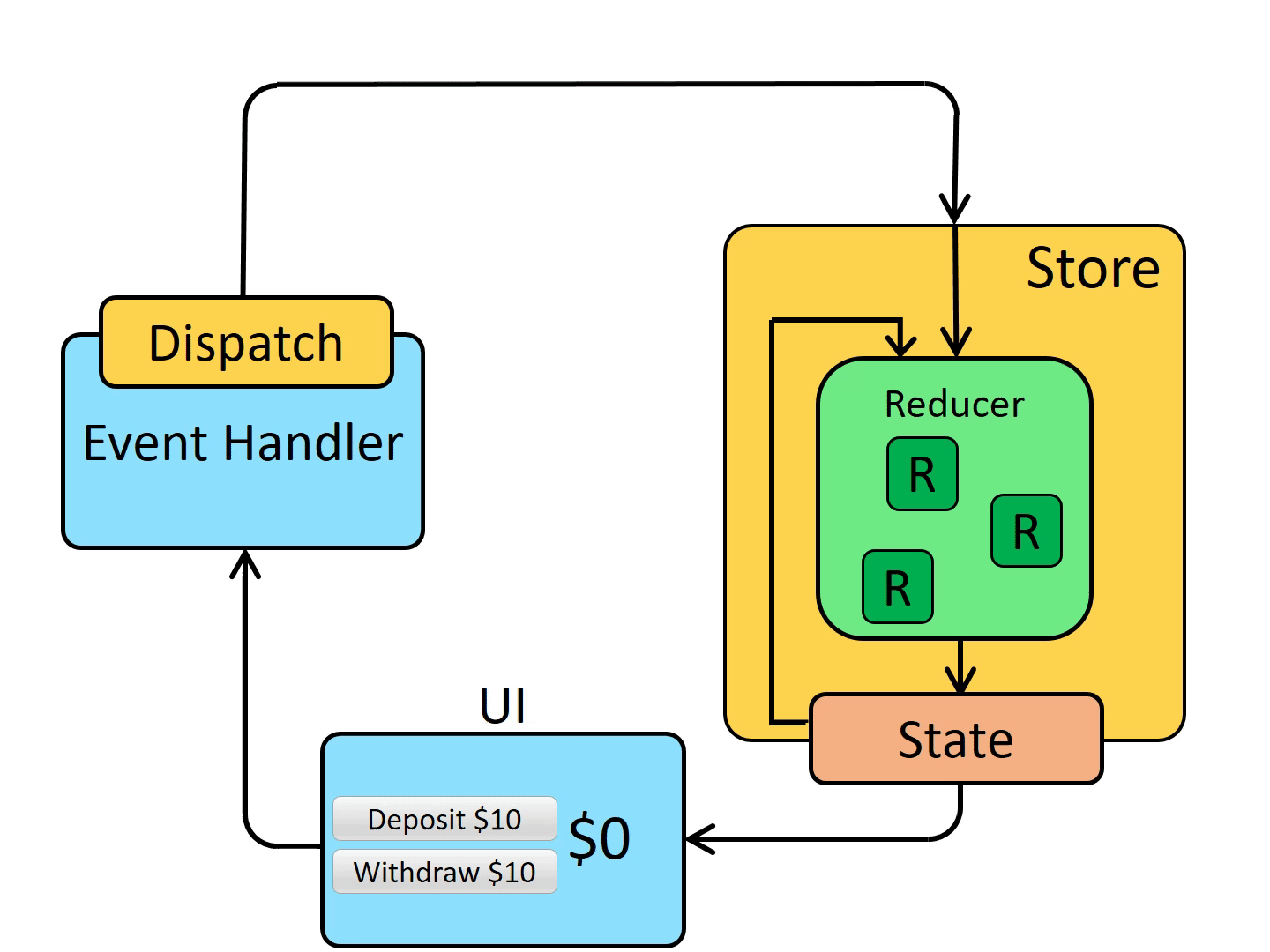
**React Redux: Managing Application State with Predictability**

React Redux is a powerful combination that streamlines state management in React applications. Redux offers a centralized store to hold the application's global state, while React Redux serves as the bridge between React components and the Redux store, enabling:

* **Predictable State Updates:** Redux enforces unidirectional data flow. Actions describe state changes, reducers handle those actions to produce a new state, and the store holds the current state. This predictability makes debugging and reasoning about state changes much easier.
* **Improved Component Reusability:** React components can access and react to updates in the global state, promoting code reusability and maintainability.
* **Testability:** Redux's pure functions (reducers) and separation of concerns make it well-suited for unit testing.

**Core Concepts:**

* **Actions:** Plain JavaScript objects that describe what happened in the application. They have a mandatory type property that identifies the action and optional payload data.
* **Reducers:** Pure functions that accept the current state and an action object. Based on the action's type, they return a new state object, effectively updating the application state.
* **Store:** The central repository for the application state. It holds the current state, dispatches actions to reducers, and notifies subscribed components whenever the state changes.
* **Connect:** A higher-order component (HOC) from React Redux that connects a React component to the Redux store. It injects props containing state data and dispatch functions into the component, allowing it to interact with the global state.



Reference: <https://redux.js.org/>