Redux Principles

A single object as application state, no matter how complicated the application is

Action object, also called change object, describes how to change the current state.

Reducer’s reduce method is a **pure function**, which takes the current state object and an action object and returns (produces) a new state object as the next application state.

- No side effect
- Does not change the input argument, i.e., the current state

Redux Store

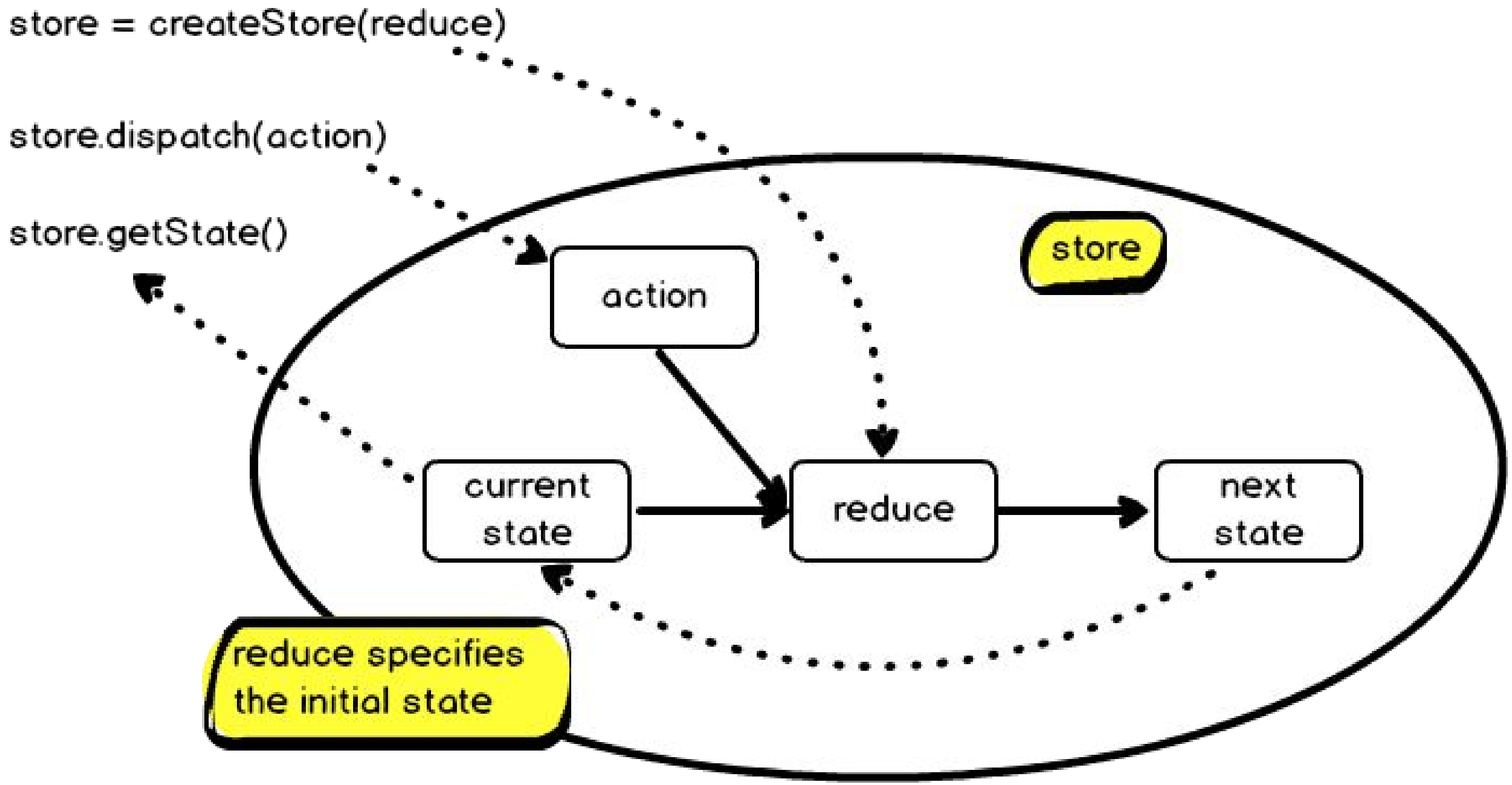
Redux store is an object that

- Holds the application state
- Allows access to state via `getState()`
- Allows state to be updated via `dispatch(actionObj)`

Redux store is created using `createStore(reduce)`, where `reduce` is the pure reduce function.

- `reduce` specifies the initial state.

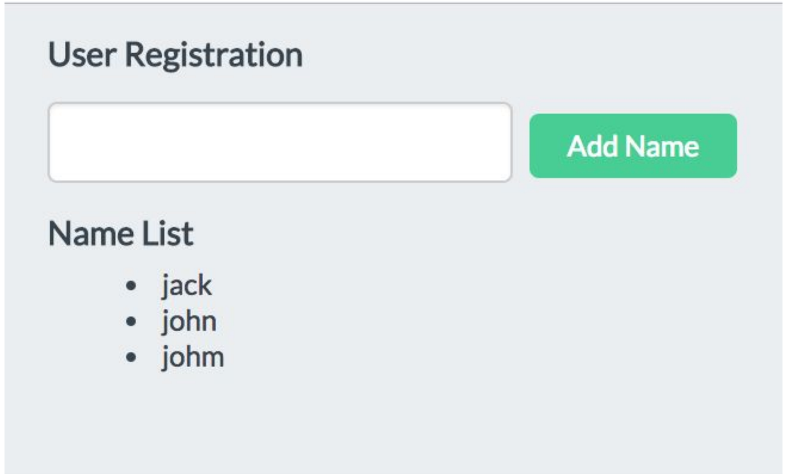
Redux Store

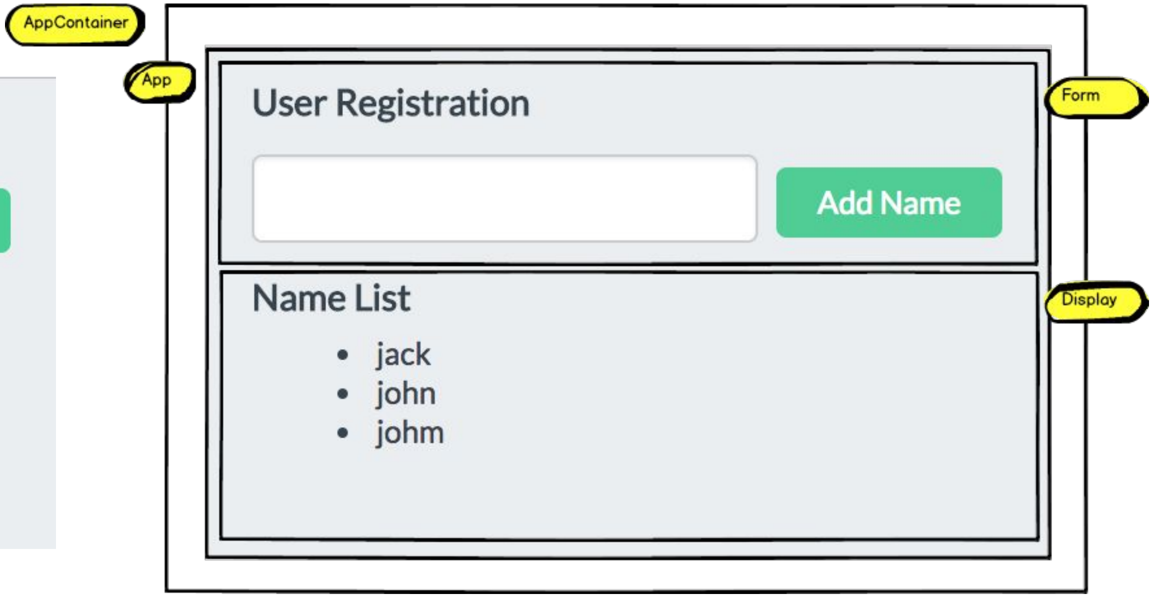

store = createStore(reduce)
store.dispatch(action)
store.getState()
store
action
current state
reduce
next state
reduce specifies the initial state

Two Types of Component

	Presentational Components (easy to code and predictable)	**Container Components (root)**
Purpose	How things look	How things work
Aware of Redux	No	Yes
To read data	Read data from props	Subscribe to Redux state
To change data	Invoke callbacks from props	Dispatch Redux actions

Redux Basics





`AppContainer` is the root component that only holds the application state.