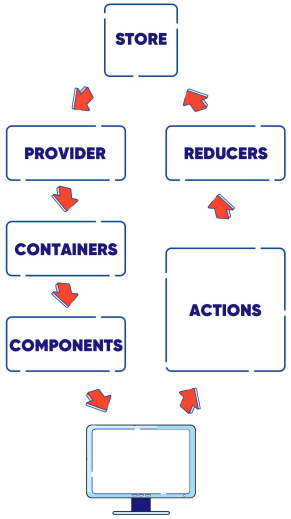
Assembler
School of Software Engineering

React Hooks and Redux

Redux

- A predictable state container.
- Single source of truth.
- Consistent and portable.
- Easy to Maintain and Debug.
- Separate presentation layer from logic and data.
- Scalable - Complexity through composition.

Redux



Fundamentals

- **Store**: A Store is an object that holds the whole state tree of your application.
- **State**: holds data which can be rendered to the user.
- **Action:** Plain JavaScript objects, and they must have a property to indicate the type of action to be carried out. They must also have a payload that contains the information that should be worked on by the action. Actions are sent using store.dispatch() method.
- **Reducers:** Are pure functions that take the previous state of the app and return a new state based on the action passed to it. It takes two parameters: the current state and action
- **Provider**: It's a component that has a reference to the Store and provides the data from the Store to the component it wraps.
- **Middleware:** It's a function that is able to intercept, and act accordingly, our actions, before they reach the reducer.

State Management

- Local state -> Can only be used within the components where they were defined
- Global state -> Can be shared across multiple components

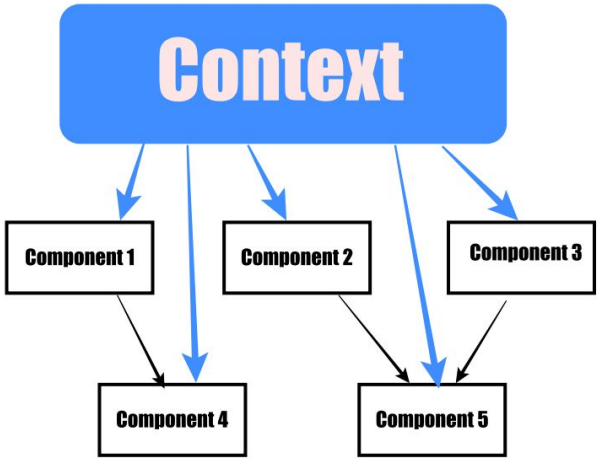
Presentational vs Container

	Presentational Components	Container Components
Purpose	How things look (markup, styles)	How things work (data fetching, state updates)
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
Are written	By hand	Usually generated by React Redux

React Hook and Redux

- `useSelector` Allow us to extract data from a Redux store, and is called whenever its containing component renders.
- `useCallback` It's a utility hook that will memoize any function, and will only create a new reference to the wrapped function if a dependency changes.
- `useDispatch` Simply returns a reference to the `dispatch()` function internal to Redux. Just define the hook within a component to start using it.

State Management with React Hooks and Context



The React Context API allows you to easily access data at different levels of the component tree. It creates a parent most component which is Context and it store all the data which is accessible from all the components of the project.

State Management with React Hooks and Context

Available since React v16.3.0 through the Context API

`useState` is recommended for handling simple values like numbers or strings

`useReducer`, is recommended for handling a state object that contains multiple values with different data types in a tree-like structure. You will need to declare functions that can change one or more of these state values. *** IMPORTANT: Immutability needs to always be present in reducers

React Hooks Context or Redux

The winner is…

- Redux DevTools
- Offers a performance advantage over the Hooks API through more efficient re-rendering
- Hooks provide an excellent, functional, compact way for managing state within a component