Redux

Teaching Faculty: Umur INAN

Prepared by Muhyieddin **AL-TARAWNEH**, Umur **INAN**

REDUX

- Redux is a predictable state container for JavaScript apps.
- A platform for developers to build customized state management for their use-cases, while being able to reuse things like the graphical debugger or middleware.

PREDICTABLE

- Redux attempts to make state mutations predictable by imposing certain restrictions on how and when updates can happen.
    - Single Source of Truth
        - The state of whole application is stored as a tree of plain objects and arrays within a single store.
    - State is Read-Only
        - State updates are caused by dispatching an action, which is a plain object describing what happened.
    - Changes are made with pure functions
        - All state updates are performed by pure functions called reducers, which are `(state,action) => newState`

CENTRALIZED

- Having a single store and single state tree enables:
    - Logging of all updates.
    - API handling.
    - Undo/Redo
    - State persistance.

ACTIONS

- To change something in the state, Dispatch an action.
- An action is a plain JS object with a type field.

REDUCERS

- All state updates logic lives in functions called reducers.
- Smaller functions can be composed into larger functions.
- Reducers should be pure functions, with no side effects.
- Reducers need to update data immutably.

STORE

- A Redux store contains the current state value.
- Stores are created using the createStore method, which takes the root reducer function.
- Stores have 3 main methods.
    - dispatch
        - Starts a state update with the provided action object.
    - getState
        - Returns the current stored state value.
    - subscribe
        - Accepts a callback funtion that will be run every time an action is disptched.

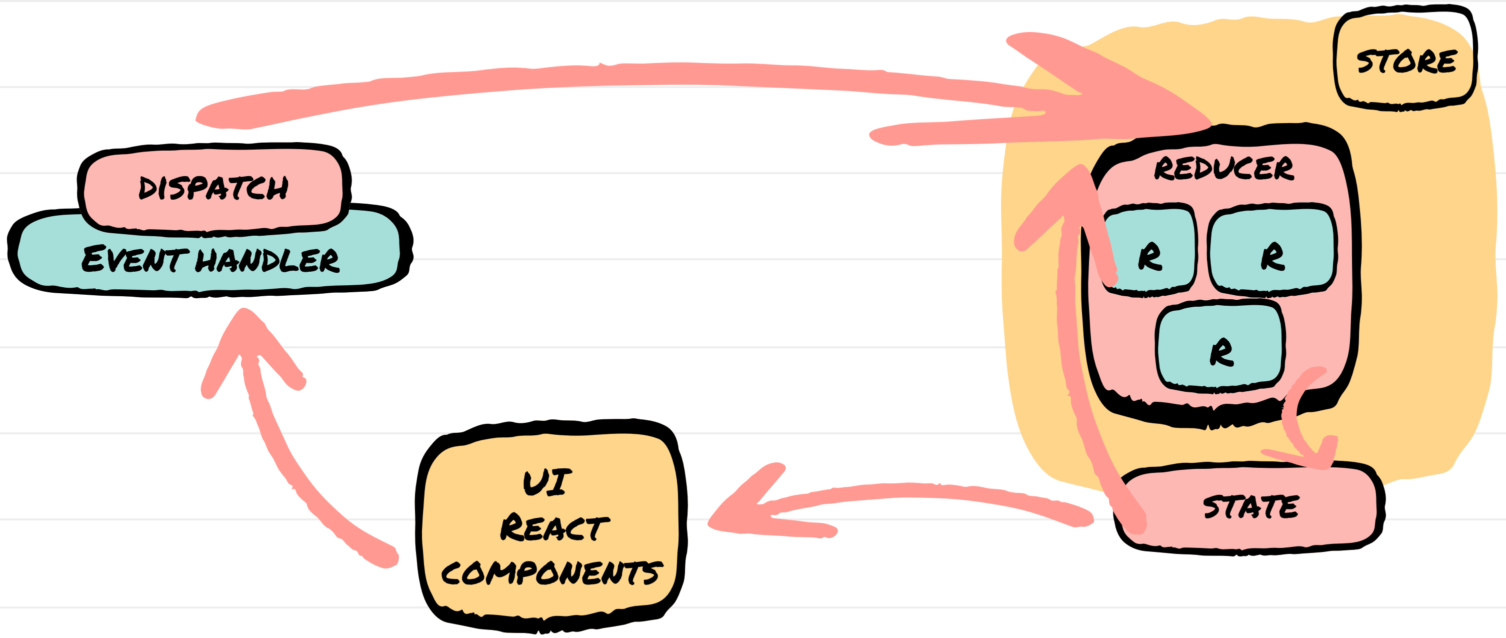
STORE

- To trigger a state update, call dispatch.
    - The store will call reducer and save the result.
    - Add subscription callbacks with subscribe.

Redux toolkit

- npm install @reduxjs/toolkit
- npm i react-redux

REDUX DATA FLOW -- SYNCHRONOUS

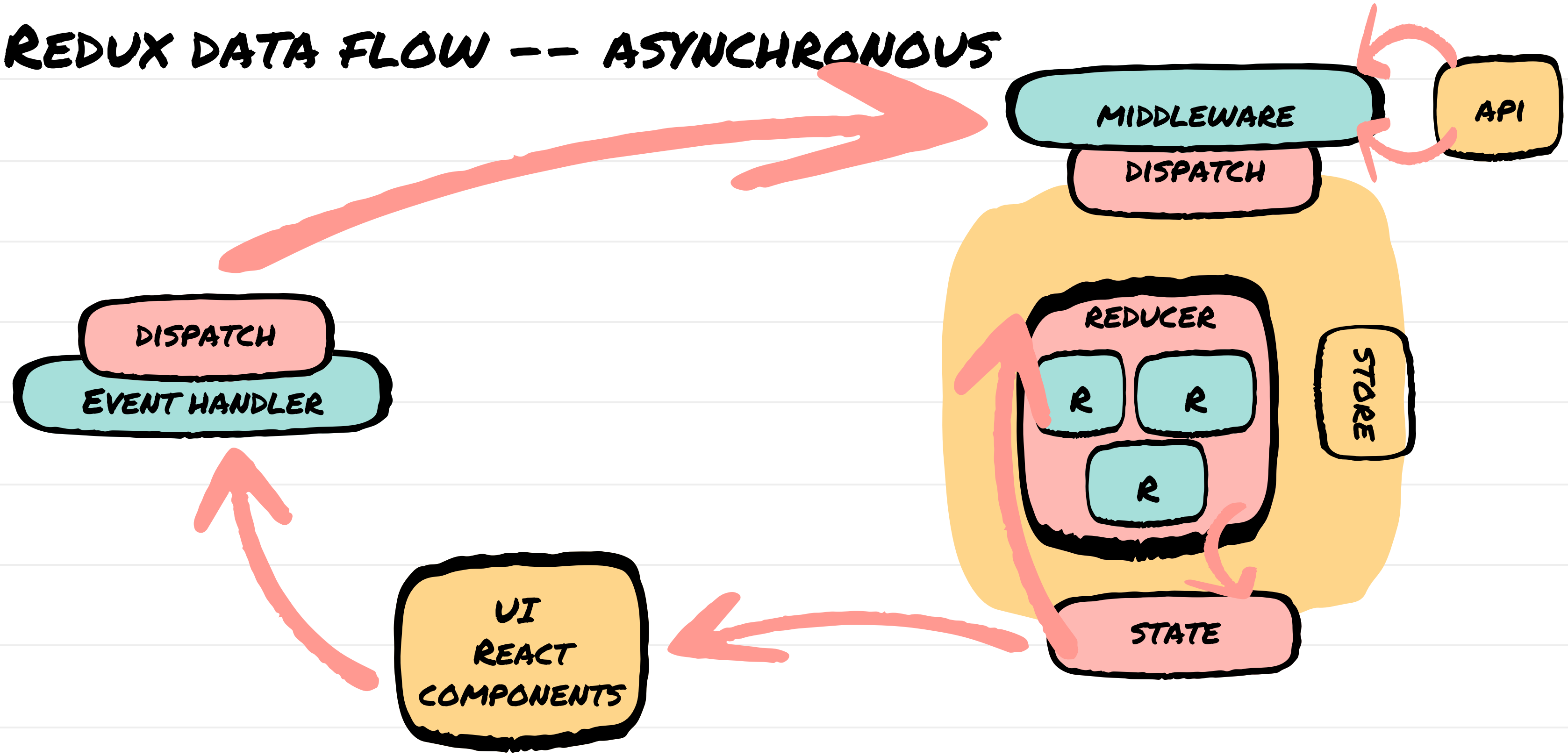


Prepared by Muhyieddin AL-TARAWNEH, Umur INAN

Async logic

- Middleware
    - Middleware are store plugins that wrap dispatch.
    - Redux thunk middleware is standard async middleware.
        - Allows passing functions to dispatch instead of actions.
        - Functions receive (dispatch, getState) as arguments.
        - Can do any sync or async logic inside.

REDUX DATA FLOW -- ASYNCHRONOUS



Prepared by Muhyieddin AL-TARAWNEH, Umur INAN

React redux

- It provides bindings to let React components interact with Redux store.

USESELECTOR

- Extracts a value from the Redux state for use in this component.
  - Accepts a selector function as its argument.
  - Subscribes to the store and re-runs the selector whenever the store state changes.
- Uses reference equality by default.
- Can be called multiple times in one component.

```
const posts = useSelector(state=> state.posts)
```

USEDISPATCH

- Returns the store's dispatch method.

```
const dispatch = useDispatch()
```

PROVIDER

- Makes the Redux store accessible to all components in the app.
- Should be set up in app entry point file and wrap entire app component.
- Set the store property.

Main points

- Frameworks make Web development easier and more effective by providing a secure and reliable foundation on which to build upon.

- The simplest form of awareness, Transcendental Consciousness, provides a strong foundation for a rewarding and successful life.