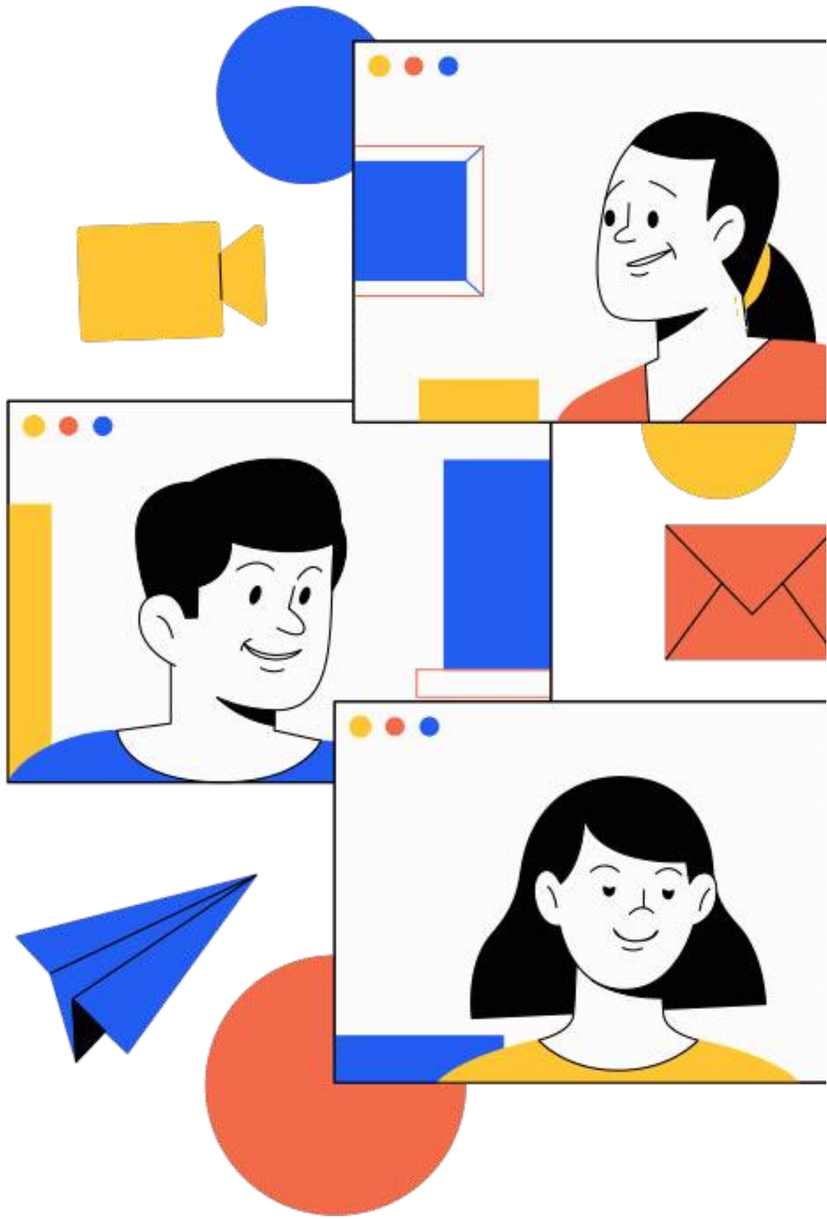
UDACITY

Week 13 - React Native

What about Redux Native?



Ahmed Fouad Lotfy

React Session Lead

Agenda

What we'll cover in this session

- What is Redux and Redux Toolkit?
- Why use Redux with React Native?
- React Native Forms with Redux.
- Handling Async with Redux Sagas.
- Live Demo

What is Redux and Redux Toolkit?

- **The concept of Redux in mobile apps:**

  A predictable state container for JavaScript apps. Redux works with React Native to provide a single source of truth for state management.

- **What is Redux Toolkit?**

  Redux Toolkit makes it easier to write good Redux applications and speeds up development. Same as Redux we used in React Web but with a slight different in syntax and approach.

Redux

React Native

Redux Toolkit Setup

- **Commands for installing Redux Toolkit and dependencies:**

```
npm install @reduxjs/toolkit
```

WEB

- **State Management:** Typically used for managing global state across components in large-scale web applications.
- **Tools:** Redux Toolkit, React-Redux for connecting the store to the React components.
- **Middleware:** Often uses Redux Thunk, Redux-Saga for handling side effects like API calls.
- **Design :** Focus on responsiveness for various screen sizes.

Mobile

- **State Management:** Commonly used to manage state across screens and navigation in mobile apps.
- **Tools:** React Native with React-Redux, React Navigation for state persistence during navigation.
- **Middleware:** Often uses Redux Thunk, Redux-Saga for handling side effects like API calls.
- **Design:** Focuses on optimizing performance, handling offline states, and syncing data.

Why use Redux with React Native?

- **Key benefits of using Redux toolkit:**
  - *Centralized state*
  - *Consistent Across Screens*
  - *Predictable Changes*
  - *Offline Support*
  - *DevTools*

Redux

React Native

Why use Redux with React Native?

- **Key features:**

  - Simplified Setup: Reduces boilerplate for faster setup.

  - Immutable Updates: Built-in Immer makes state managemen

  - Async Logic: createAsyncThunk simplifies async actions.

  - Debug: Enhanced debugging experience

Create Store using Redux Toolkit

- **Commands for Creating Redux Toolkit Store:**

```
import { configureStore } from '@reduxjs/toolkit';
import counterReducer from './counterSlice';

export const store = configureStore({
  reducer: {
    counter: counterReducer,
  },
});
```

Form Management with Redux

- **A simple form in React Native that updates Redux state on form input.**

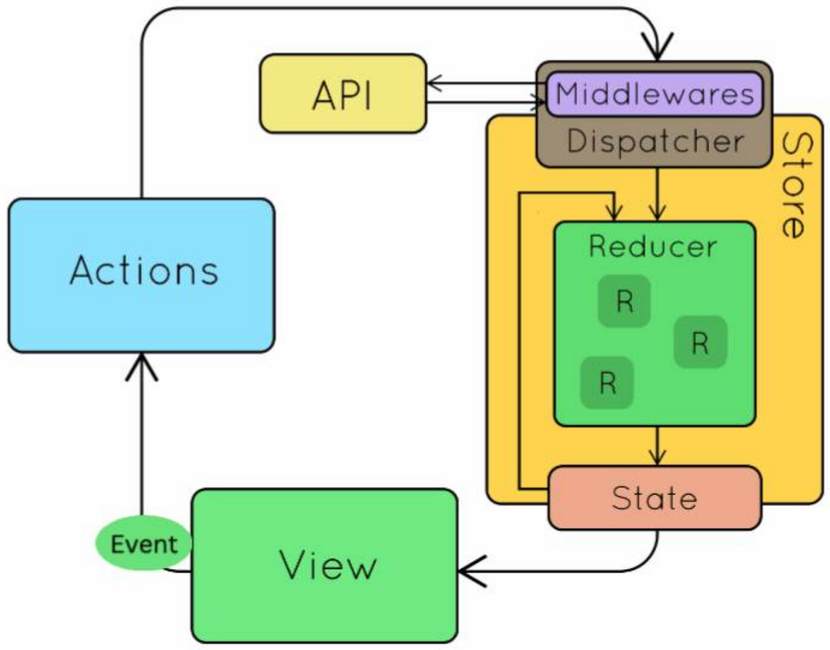
```
const Form = () => {
  const dispatch = useDispatch();
  const name = useSelector((state) => state.form.name);
  const [inputValue, setInputValue] = useState('');

  return (
    <View>
      <TextInput
        value={name ? name : inputValue} onChangeText={(text) => setInputValue(text)} />
      <Button title="Submit" onPress={() => dispatch(updateName(inputValue))} />
    </View>
  );
};
```

Handling Async with Redux Sagas

Why Redux Saga is effective for handling async flows?

Uses generator functions to manage complex async flows and side effects, offering a more powerful alternative for handling async logic.



Handling Async with Redux Sagas

Example of a action creators functions using Redux that fetches data

```
const FETCH_DATA_REQUEST = 'FETCH_DATA_REQUEST';
const FETCH_DATA_SUCCESS = 'FETCH_DATA_SUCCESS';
const FETCH_DATA_FAILURE = 'FETCH_DATA_FAILURE';
```

```
export const fetchDataRequest = () => ({
  type: FETCH_DATA_REQUEST,
});

export const fetchDataSuccess = (data) => ({
  type: FETCH_DATA_SUCCESS,
  payload: data,
});

export const fetchDataFailure = (error) => ({
  type: FETCH_DATA_FAILURE,
  payload: error,
});
```

Handling Async with Redux Sagas

Example of a reducers functions using Redux that fetches data

```
const initialState = {
  data: null,
  loading: false,
  error: null,
};
```

```
const dataReducer = (state = initialState,
                     action) => {
  switch (action.type) {
    case FETCH_DATA_REQUEST:
      return {
        ...state,
        loading: true,
        error: null,
      };

    case FETCH_DATA_SUCCESS:
      return {
        ...state,
        loading: false,
        data: action.payload,
        error: null,
      };

    case FETCH_DATA_FAILURE:
      return {
        ...state,
        loading: false,
        error: action.payload,
      };

    default:
      return state;
  }
};
```

Handling Async with Redux Sagas

Example of a combining reducers functions using Redux

```
import { combineReducers } from 'redux';

const rootReducer = combineReducers({
  data: dataReducer,
  // other reducers go here
});

export default rootReducer;
```

Handling Async with Redux Sagas

What is createSlice?

- createSlice is a utility from Redux Toolkit that simplifies the process of creating reducers and action creators.

- It combines reducers, initial state, and action creators into one single function.

Handling Async with Redux Sagas

Key Features:

- Automatic Action Creators.

- Simplifies Reducers

- Action Types

- Built-in Initial State

Handling Async with Redux Sagas

```
import { createSlice } from '@reduxjs/toolkit';

const counterSlice = createSlice({
  name: 'counter',       // Slice name (used as action prefix)
  initialState: { value: 0 }, // Initial state
  reducers: {
    increment: (state) => {   // Reducer function
      state.value += 1;       // State mutation handled by Immer
    },
    decrement: (state) => {
      state.value -= 1;
    },
    incrementByAmount: (state, action) => {
      state.value += action.payload;
    },
  },
});

export const { increment, decrement, incrementByAmount } = counterSlice.actions; // Generated actions
export default counterSlice.reducer; // Slice reducer
```

Handling Async with Redux Sagas

Example of a saga that fetches data

```
function* fetchDataSaga(action) {
  try {
    //const user = yield call(Api.fetchData, action.payload.Id);
    const data = yield call(Api.fetchData);
    yield put({ type: 'FETCH_SUCCEEDED', data });
  } catch (e) {
    yield put({ type: 'FETCH_FAILED', message: e.message });
  }
}
```

Handling Async with Redux Sagas

Example of a saga that fetches data

```
function* fetchDataSaga() {
  try {
    const data = yield call(api.fetchData); // Call the API function
    yield put(fetchDataSuccess(data)); // Dispatch success action
  } catch (error) {
    yield put(fetchDataFailure(error)); // Dispatch failure action
  }
}
```

Handling Async with Redux Sagas

Example of a saga that watch fetches data requests

```
// Watcher saga: Watches for FETCH_DATA_REQUEST actions
function* watchFetchData() {
  yield takeLatest(FETCH_DATA_REQUEST, fetchDataSaga);
}
```

Navigation with Redux Sagas

Example of a Navigation with Redux: Handling state across multiple screens in a mobile app using React Navigation and Redux.

```
const Stack = createStackNavigator();

function App() {
  return (
    <Provider store={store}>
      <NavigationContainer>
        <Stack.Navigator>
          <Stack.Screen name="Home" component={HomeScreen} />
          <Stack.Screen name="Profile" component={ProfileScreen} />
        </Stack.Navigator>
      </NavigationContainer>
    </Provider>
  );
}
```

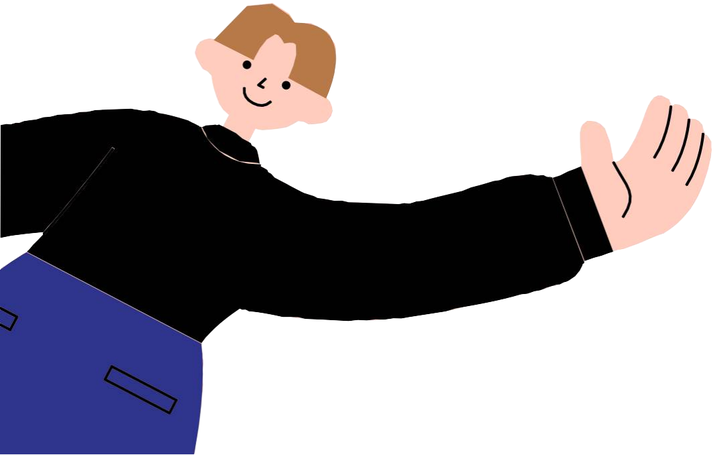
Sagas vs Thunk

1. ***Redux Sagas:*** Manage more complex asynchronous flows using generator functions with greater control.
2. ***Redux Thunk:*** A middleware for handling simple async logic by returning functions

Aspect	Redux Sagas	Redux Thunk
Pros	- Handles complex async flows - More powerful for tasks like retries, cancellations, etc. - Decouples side effects from components	- Simpler to implement - Easier for small to medium apps
Cons	- More complex syntax (generator functions) - Steeper learning curve - More code to maintain	- Can get messy with deeply nested async calls - Tightly couples logic with components
When?	- Large apps with complex async tasks - Need control over concurrency, retries, or cancellation of actions	- Small to medium apps - Simple async logic like fetching data or basic side effects

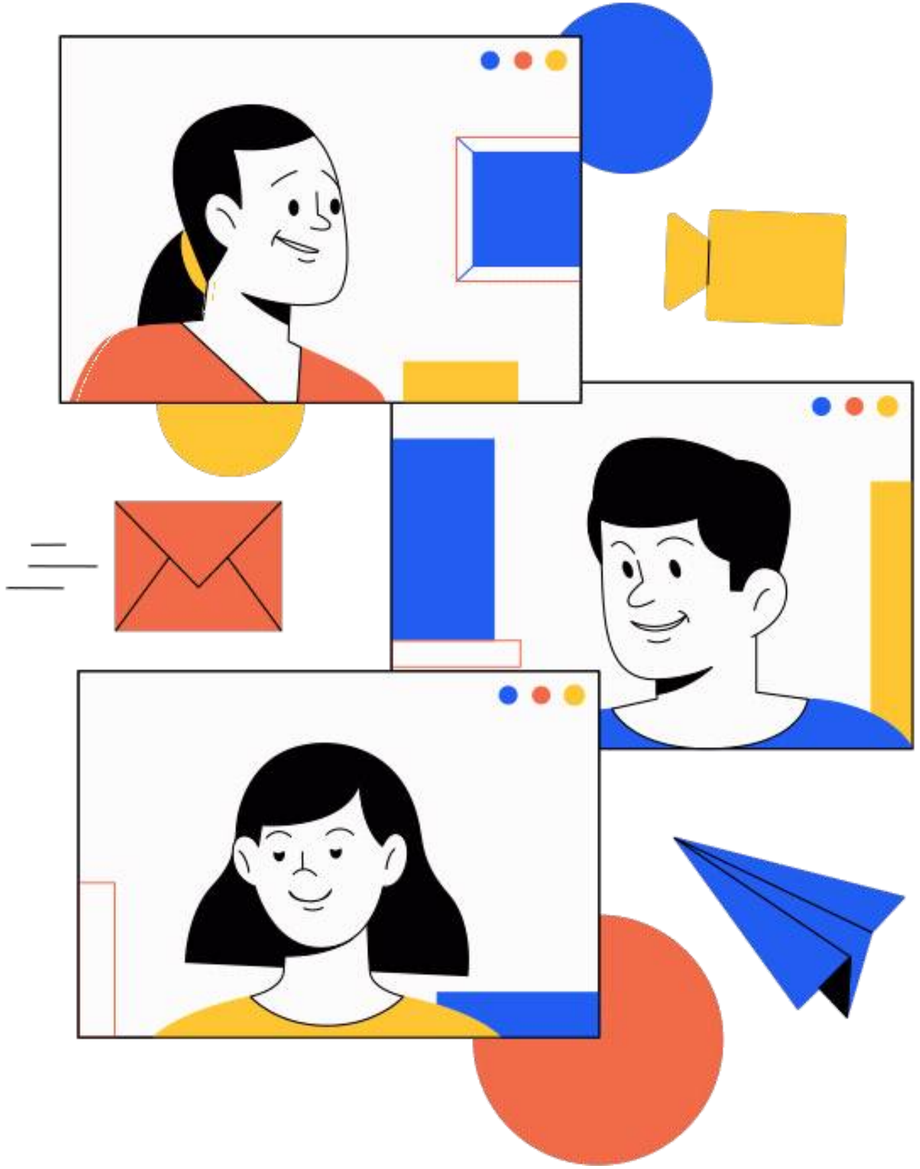
Demo Time

Any Questions?



Next Session:

React Native Features



Thank you for attending!

Feel free to email at **a.lotfy@fci-cu.edu.eg** or reach me at circle anytime for any questions or clarifications!

Follow me on Github **@ahmeddxfouad**

code and slides are found at this **github repo**