# React Native State Management: Context API & Redux for Beginners

**Subtitle:** Learn how to manage and share data across your app efficiently using Context API and Redux.

**Website Name:** haas.dev
**Website Link:** https://dev-roast-app.vercel.app

---

## Introduction

As apps grow, managing state across multiple screens becomes challenging. React Native offers **Context API** for lightweight state sharing and **Redux** for more complex global state management. This guide helps beginners understand both approaches and implement them in real apps.

---

## Step 1: Local Component State

- Start simple with `useState` for component-level state.
- Ideal for **temporary data** that does not need to be shared across screens.

**Example:**

```
import React, { useState } from 'react';
import { View, Text, Button } from 'react-native';

export default function Counter() {
  const [count, setCount] = useState(0);

  return (
    <View style={{ padding: 20 }}>
      <Text>Count: {count}</Text>
      <Button title="Increase" onPress={() => setCount(count + 1)} />
    </View>
  );
}
```

**Exercise:** Add a **decrement button** to the counter.

---

## Step 2: Context API

Context API lets you **share state across components** without prop drilling.

**Example:**

1. Create a context:

```
import React, { createContext, useState } from 'react';

export const AppContext = createContext();

export const AppProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  return (
    <AppContext.Provider value={{ user, setUser }}>
      {children}
    </AppContext.Provider>
  );
};
```

2. Use context in components:

```
import React, { useContext } from 'react';
import { View, Text, Button } from 'react-native';
import { AppContext } from './AppContext';

export default function Profile() {
  const { user, setUser } = useContext(AppContext);

  return (
    <View style={{ padding: 20 }}>
      <Text>User: {user ? user : 'Guest'}</Text>
      <Button title="Login" onPress={() => setUser('Hafsa')} />
    </View>
  );
}
```

**Exercise:** Create a **theme switcher** using Context API that toggles light/dark mode across all screens.

---

## Step 3: Redux Basics

Redux is ideal for **complex apps** with multiple states to manage.

1. Install Redux and React-Redux:

```
npm install redux react-redux
```

2. Create a store:

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

const initialState = { count: 0 };

function counterReducer(state = initialState, action) {
  switch(action.type) {
    case 'INCREMENT':
      return { count: state.count + 1 };
    case 'DECREMENT':
      return { count: state.count - 1 };
    default:
      return state;
  }
}

export const store = createStore(counterReducer);
```

3. Wrap app with `Provider`:

```
import { Provider } from 'react-redux';
import { store } from './store';
import App from './App';

export default function Main() {
  return (
    <Provider store={store}>
      <App />
    </Provider>
  );
}
```

4. Access state and dispatch actions:

```
import { useSelector, useDispatch } from 'react-redux';
import { View, Text, Button } from 'react-native';
```

```
export default function Counter() {
  const count = useSelector(state => state.count);
  const dispatch = useDispatch();

  return (
    <View style={{ padding: 20 }}>
      <Text>Count: {count}</Text>
      <Button title="Increase" onPress={() => dispatch({ type: 'INCREMENT' })} /
      <Button title="Decrease" onPress={() => dispatch({ type: 'DECREMENT' })} /
    </View>
  );
}
```

**Exercise:** Build a **todo app** using Redux to manage the list of tasks.

---

## Step 4: Best Practices

- Use **Context API** for small-to-medium apps.
- Use **Redux** for large-scale apps with complex state.
- Keep state **minimal and organized**.
- Combine **local state and global state** wisely for performance.
- Structure Redux with **actions, reducers, and selectors** for maintainability.

---

Proper state management allows React Native apps to scale effectively while keeping code clean and maintainable. Mastering Context API and Redux is key to building **professional-grade mobile apps**.

Visit **haas.dev** for more React Native guides, tutorials, and project examples.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

---