

### React vs React Native

Usage Scope	Syntax	Animation & Gestures	Routing
React.js is a JS library for building Responsive User Interfaces for Building Web Application. React Native is a framework for creating mobile applications with a native feel.	Both use JSX, but React uses HTML tags while React Native uses <view> <text> etc.	React uses CSS animations on a major scale to achieve animations for a web page while the recommended way for react native is to use the Animated API.	React uses react-router for routing and doesn't have any inbuilt routing capabilities but React Native has a built-in Navigator library.

### useLocalStorage

```
import { useState, useEffect } from 'react';
import AsyncStorage from '@react-native-async-storage/async-storage';
function useLocalStorage(key, initialValue) {
  const [storeValue, setStoreValue] = useState(() => {
    try {
      const item = AsyncStorage.getItem(key);
      return item ? JSON.parse(item) : initialValue;
    } catch (error) {
      console.error(error);
      return initialValue;
    }
  });
  const setValue = async (value) => {
    try {
      const valueToStore = value instanceof Function ? value() : value;
      await AsyncStorage.setItem(key, JSON.stringify(valueToStore));
    } catch (error) {
      console.error(error);
    }
  };
  return [storeValue, setValue];
}
// Usage
// const [username, setUsername] = useLocalStorage('username', 'defaultName');
```

### useDeviceOrientation (cont)

```
>   return () => Dimensions.removeEventListener('change', onChange);
}, []);
return orientation;
}
// Usage:
// const orientation = useDeviceOrientation();
```

### User Input

```

import { useState, useEffect } from 'react';
import { Dimensions } from 'react-native';
function useDeviceOrientation() {
  const [orientation, setOrientation] = useState(
    Dimensions.get('window').width >
    Dimensions.get('window').height
      ? 'LANDSCAPE'
      : 'PORTRAIT'
  );
  useEffect(() => {
    const onChange = ({ window }) => {
      setOrientation(window.width >
        window.height ? 'LANDSCAPE' : 'PORTRAIT');
    };
    Dimensions.addEventListener('change', onChange);
  });
}

```

```

import React, { useState } from 'react';
import { View, TextInput, Button, Text } from 'react-native';
import axios from 'axios';
import AsyncStorage from '@react-native-async-storage/async-storage';
const Login = ({ navigation }) => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');
  const handleLogin = async () => {
    try {
      const response = await axios.post('https://your-api-endpoint/login', {
        email,
        password,
      });
      const token = response.data;
      if (token) {
        await AsyncStorage.setItem('authToken', token);
        navigation.navigate('Home');
      }
    } catch (err) {
      console.error("Login error: ", err);
    }
    setError('Invalid login credentials');
  };
  return (
    <View>
      <TextInput
        placeholder="Email"
        value={email}
        onChangeText={setEmail}
      />
      <TextInput
        placeholder="Password"
        value={password}
        onChangeText={setPassword}
        secureTextEntry
      />
      <Button
        title="Login"
        onPress={handleLogin}
      />
    </View>
  );
}

```



By **aezzat** (rathetsu)  
cheatography.com/rathetsu/

Not published yet.  
Last updated 8th October, 2023.  
Page 2 of 7.

Sponsored by **CrosswordCheats.com**  
Learn to solve cryptic crosswords!  
<http://crosswordcheats.com>

### User Input (cont)

```
>   <TextInput  
placeholder="Password"  
value={password}  
onChangeText={setPassword}  
secureTextEntry  
>  
<Button title="Login" onPress={handleLogin} />  
{error && <Text>Error: {error}</Text>}  
</View>  
);  
};  
export default Login;
```

### Securely Store Auth Token

```
import * as Keychain from 'react-native-keychain';  
// Storing the token securely  
async function storeToken(token) {  
  try {  
    await Keychain.setGenericPassword('User Token', token);  
  } catch (error) {  
    console.error('Error storing token:', error);  
  }  
}  
// Retrieving the token  
async function getToken() {  
  try {  
    const credentials = await Keychain.getGenericPassword();  
    if (credentials) {  
      return credentials.password;  
    }  
  } catch (error) {  
    console.error('Error retrieving token:', error);  
  }  
  return null;  
}  
// Clearing the token  
async function clearToken() {  
  try {  
    await Keychain.resetGenericPassword();  
  } catch (error) {  
  }
```

### Securely Store Auth Token (cont)

```
>   console.error('Error clearing token:', error);  
}  
}
```

### Context API

```
import { createContext } from 'react';  
const MyContext = createContext();  
export const MyProvider = ({ children }) => {  
  const [value, setValue] = useState('initial value');  
  return (  
    <MyContext.Provider value={{ value,  
      setValue }}>  
      {children}  
    </MyContext.Provider>  
  );  
}  
// Usage  
import React, { useContext } from 'react';  
const MyComponent = () => {  
  const { value, setValue } = useContext(MyContext);  
  return (  
    <div>  
      {value}  
      <button onClick={() => setValue('newValue')}>Set Value</button>  
    </div>  
  );  
}  
// Wrapping components in the provider  
const App = () => (  
  <MyProvider>  
    <MyComponent />  
  </MyProvider>  
);
```

### FlatList

```
const DATA = [  
  { id: '1', title: 'Item 1' },  
  { id: '2', title: 'Item 2' },  
  // ... more items  
];  
const Item = ({ title }) => (
```



### FlatList (cont)

```
> <View style={styles.item}>
  <Text style={styles.title}>{title}</Text>
</View>
);
// Pull to refresh
const [refreshing, setRefreshing] = React.useState(false);
const onRefresh = React.useCallback(() => {
  setRefreshing(true);
  // Fetch new data and set refreshing to false
  // For example:
  // fetchData().then(() => setRefreshing(false));
}, []);
const MyFlatListComponent = () => (
  <FlatList
    horizontal={false}
    refreshing={refreshing}
    onRefresh={onRefresh}
    data={DATA}
    renderItem={({ item }) => <Item title={item.title} />}
    keyExtractor={item => item.id}
  />
);

```

### Stack Navigation

```
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
const Stack = createStackNavigator();
const App = () => (
  <NavigationContainer>
    <Stack.Navigator initialRouteName="Home">
      <Stack.Screen name="Home" component={HomeScreen} />
      <Stack.Screen name="Details" component={DetailsScreen} />
    </Stack.Navigator>
  </NavigationContainer>
);

```

### Bottom Tab Navigation

```
import { NavigationContainer } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
const Tab = createBottomTabNavigator();
const App = () => (
  <NavigationContainer>
    <Tab.Screen name="Home" component={HomeScreen} />
    <Tab.Screen name="Settings" component={SettingsScreen} />
  </NavigationContainer>
);

```

### Drawer Navigation

```
import { NavigationContainer } from '@react-navigation/native';
import { createDrawerNavigator } from '@react-navigation/drawers';
const Drawer = createDrawerNavigator();
const App = () => (
  <NavigationContainer>
    <Drawer.Navigator initialRouteName="Home">
      <Drawer.Screen name="Home" component={HomeScreen} />
      <Drawer.Screen name="Settings" component={SettingsScreen} />
    </Drawer.Navigator>
  </NavigationContainer>
);

```

### GradientButton



By **aezzat** (rathetsu)  
[cheatography.com/rathetsu/](https://cheatography.com/rathetsu/)

Not published yet.  
Last updated 8th October, 2023.  
Page 4 of 7.

```
import React from 'react';
import { View, TouchableOpacity, Text } from
'react-native';
import LinearGradient from 'react-native-linear-gradient';
const GradientButton = ({  
  hollow = false,  
  borderStart,  
  borderEnd,  
  start,  
  end,  
  onPress,  
  containerStyle,  
  buttonStyle,  
  textStyle,  
  text,
```

Sponsored by **CrosswordCheats.com**  
Learn to solve cryptic crosswords!  
<http://crosswordcheats.com>

### GradientButton (cont)

```
> }) => {
const computedBorderStart = borderStart || '#4780B5AA';
const computedBorderEnd = borderEnd || '#4C88C1AA';
const computedStart = start || '#4780B5';
const computedEnd = end || '#4C88C1';
return (
<TouchableOpacity onPress={onPress} style={containerStyle}>
<LinearGradient
colors={[computedBorderStart, computedBorderEnd]}
start={{ x: 0, y: 1 }}
end={{ x: 1, y: 0 }}
style={{ borderRadius: 100 }}>
<View style={{ padding: 4 }}>
<LinearGradient
colors={hollow ? ['white', 'white'] : [computedStart, computedEnd]}
start={{ x: 0, y: 1 }}
end={{ x: 1, y: 0 }}
style={{ borderRadius: 100 }}>
<View style={buttonStyle}>
<Text style={textStyle}>
{text}
</Text>
</View>
</LinearGradient>
</View>
</LinearGradient>
</TouchableOpacity>
);
}
export default GradientButton;
```

### Redux (cont)

```
> });
export const fetchDataSuccess = (data) => ({
type: FETCH_DATA_SUCCESS,
payload: { data },
});
export const fetchDataFailure = (error) => ({
type: FETCH_DATA_FAILURE,
payload: { error },
});
// reducers/dataReducer.js
const initialState = {
data: [],
loading: false,
error: null,
};
export default function dataReducer(state = initialState, action) {
switch (action.type) {
case FETCH_DATA_BEGIN:
return {
...state,
loading: true,
error: null,
};
case FETCH_DATA_SUCCESS:
return {
...state,
loading: false,
data: action.payload.data,
};
case FETCH_DATA_FAILURE:
return {
...state,
loading: false,
error: action.payload.error,
data: [],
};
default:
return state;
}
}
```

### Redux

```
// npm install redux react-redux redux-thunk
// action s/t ypes.js
export const FETCH_DATA_BEGIN = 'FETCH_DA TA_ - BEGIN';
export const FETCH_DATA_SUCCESS = 'FETCH _DA - TA_ S UCCESS';
export const FETCH_DATA_FAILURE = 'FETCH _DA - TA_ FAI LURE';
// action s/d ata Act ions.js
export const fetchD ata Begin = () => ({
type: FETCH_ DAT A_B EGIN,
```



By **aezzat** (rathetsu)  
cheatography.com/rathetsu/

Not published yet.  
Last updated 8th October, 2023.  
Page 5 of 7.

Sponsored by **CrosswordCheats.com**  
Learn to solve cryptic crosswords!  
<http://crosswordcheats.com>

### Redux (cont)

```
> }
// reducers/index.js
import { combineReducers } from 'redux';
import dataReducer from './dataReducer';
export default combineReducers({
  data: dataReducer,
});
// configureStore.js
import { createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';
import rootReducer from './reducers';
const initialState = {};
const middleware = [thunk];
const store = createStore(
  rootReducer,
  initialState,
  applyMiddleware(...middleware)
);
export default store;
// App.js
import React from 'react';
import { Provider } from 'react-redux';
import store from './configureStore';
import Home from './Home'; // your main component
const App = () => (
  <Provider store={store}>
    <Home />
  </Provider>
);
export default App;
// Usage in components
import { connect } from 'react-redux';
import { fetchDataBegin } from './actions/dataActions';
const MyComponent = ({ data, fetchData }) => {
  // use data or call fetchData as needed
};
const mapStateToProps = state => ({
  data: state.data.data,
});
```

### Redux (cont)

```
> const mapDispatchToProps = dispatch => ({
  fetchData: () => dispatch(fetchDataBegin()),
});
export default connect(mapStateToProps, mapDispatchToProps)(MyComponent);
```

### React Native app feel smooth on animations

```
// The primary reason and an important one why well-built native apps feel so smooth
// are by avoiding expensive operations during interactions and animations.
// React Native has a limitation that there is only a single JS execution thread,
// but you can use Interacti onManager to make sure long-running work is scheduled
// to start after any interactions /animations have completed.
import React, { useState, useEffect } from 'react';
import { Text, InteractionManager } from 'react-native';
const ExpensiveComponent = () => {
  const [isDataLoaded, setDataLoaded] = useState(false);
  useEffect(() => {
    InteractionManagerана ger.ru nAfterInteraction(() => {
      loadData();
    });
  }, []);
  const loadData = () => {
    // Simulate a data loading process
    setTimeout(() => {
      setDataLoaded(true);
    }, 2000);
  };
  if (!isDataLoaded) {
    return <Text> Loading...</Text>;
  }
  return <Text>Data Loaded !</Text>;
};
export default ExpensiveComponent;
```

