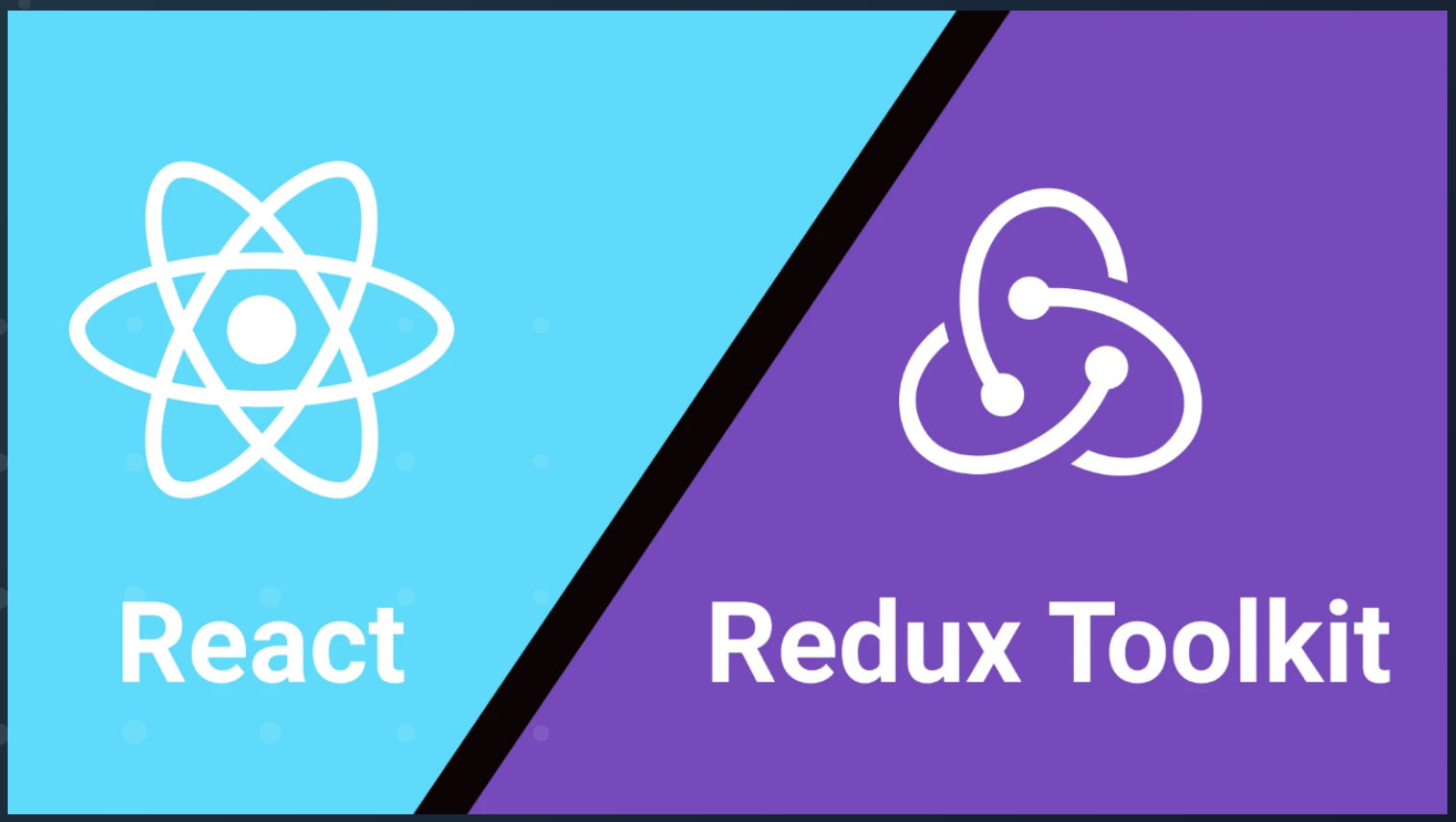
Redux Toolkit

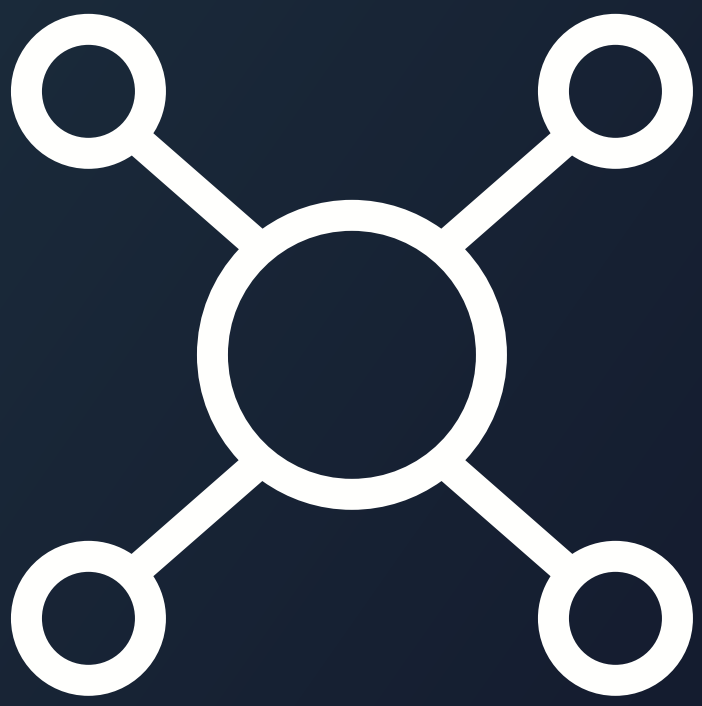
Easiest Explanation Ever



like and share

Have you ever played a game where you have to keep track of a bunch of different things, like how many coins you've collected, how many lives you have left, and what level you're on? It can be hard to remember all of those things and keep track of them, right?

Well, programming can be kind of like that sometimes, especially when we're working with something called "state." State is just a fancy word for "all the things we need to keep track of in our program." And when we're working with state in a program, it can be hard to keep track of everything and make sure everything stays organized.



That's where Redux Toolkit comes in! Redux Toolkit is a tool that helps us manage our state and keep everything organized. It provides a bunch of different functions and tools that make it easier to write code that works with state, so we don't have to worry about keeping track of everything ourselves.

Think of Redux Toolkit like a big toolbox that has all the tools we need to build something. Instead of trying to find all the tools we need ourselves, we can just open up the toolbox and use the tools that are already there. That makes it a lot easier to build something and make sure everything works correctly!

So, that's what Redux Toolkit is: it's a toolbox full of tools that helps us manage our state and keep everything organized in our programs.

Lets understand Redux-Toolkit with an easy peasy code example.

Step #1

First, we need to install Redux Toolkit and some other dependencies. We can do that by running the following command in our terminal:

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

Step #2

Next, we need to set up our Redux store. The store is where all of our state is kept. We can create a store using the configureStore function from Redux Toolkit. Here's an example:

```
import { configureStore } from '@reduxjs/toolkit';
import rootReducer from './reducers';

const store = configureStore({
  reducer: rootReducer,
});
```

This code creates a new store using configureStore, and passes in our root reducer, which is responsible for managing all of our state.

Step #3

We also need to wrap our app with the Provider component from react-redux, which makes our store available to all the components in our app. Here's an example:

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

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root')
);
```

Step #4

Now that our store is set up, we can start using it in our components. We can use the useSelector hook from react-redux to select data from the store, and the useDispatch hook to dispatch actions to the store. Here's an example:

```
import React from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { increment } from './reducers/counter';

function Counter() {
  const count = useSelector(state =>
  state.counter.value);
  const dispatch = useDispatch();

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => dispatch
      (increment())}>Increment</button>
    </div>
  );
}

export default Counter;
```

In this example, we're using the useSelector hook to select the value property of our counter slice of state. We're also using the useDispatch hook to dispatch an increment action when the button is clicked.

And that's it!

With Redux Toolkit, we can easily set up and manage our state in a React app.

Useful ? Let me know in the comments

like and share

Sunil Vishwakarma

Frontend Developer

and **Follow** for more