Skills Network

Introduction to Redux ToolKit

IBM

What you will learn

- Define the Redux toolkit in the context of React
- Describe the Redux toolkit utilities used to streamline Redux tasks
- Describe the Redux toolkit architecture
- Describe the relationship between a store and a slice

Introduction to Redux toolkit

- Simplifies Redux development
- Includes utilities
- Reduces boilerplate code

Redux toolkit utilities

Simplified store setup

- configureStore() function
- Redux Thunk
- Redux DevTools Extension

Immutability and reducer logic

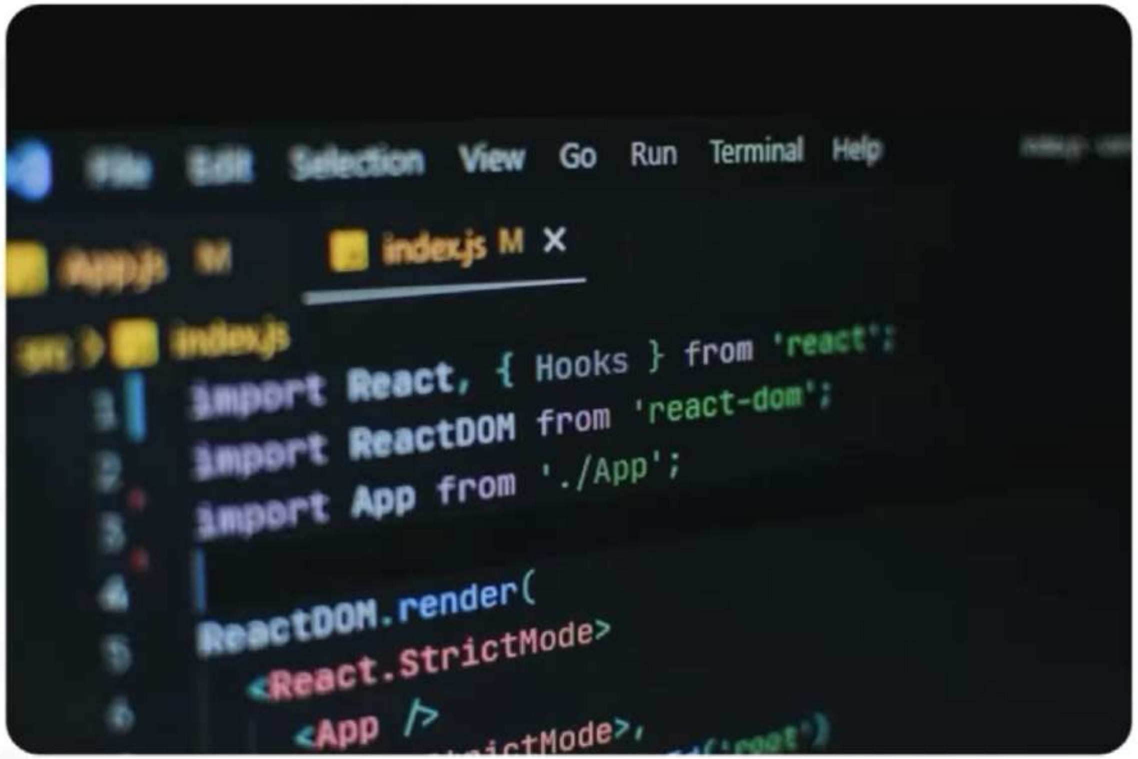
- createSlice() function
- Slice reducers

Boilerplate reduction

- Concise code

Installing RTK

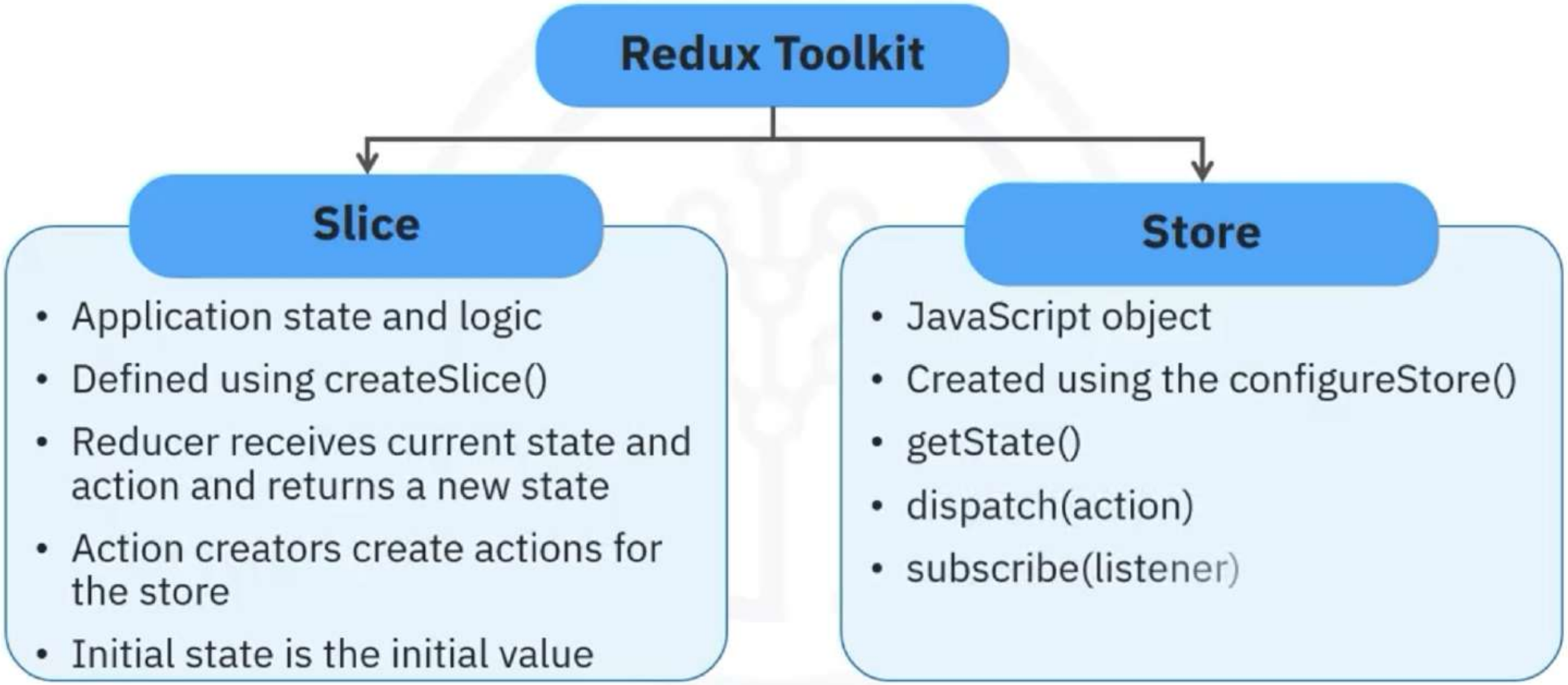
```
npm install
@reduxjs/toolkit
```



```
import React, { Hooks } from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
```

Redux toolkit architecture



Slice

- Application state and logic
- Defined using createSlice()
- Reducer receives current state and action and returns a new state
- Action creators create actions for the store
- Initial state is the initial value

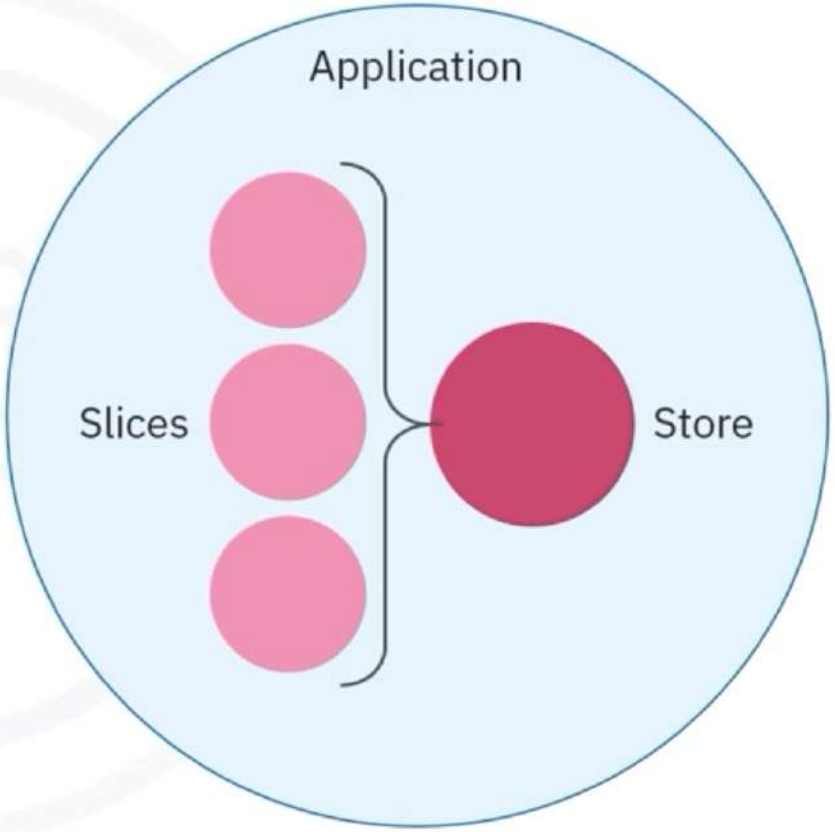
Store

- JavaScript object
- Created using the configureStore()
- getState()
- dispatch(action)
- subscribe(listener)

Slices and store relationship

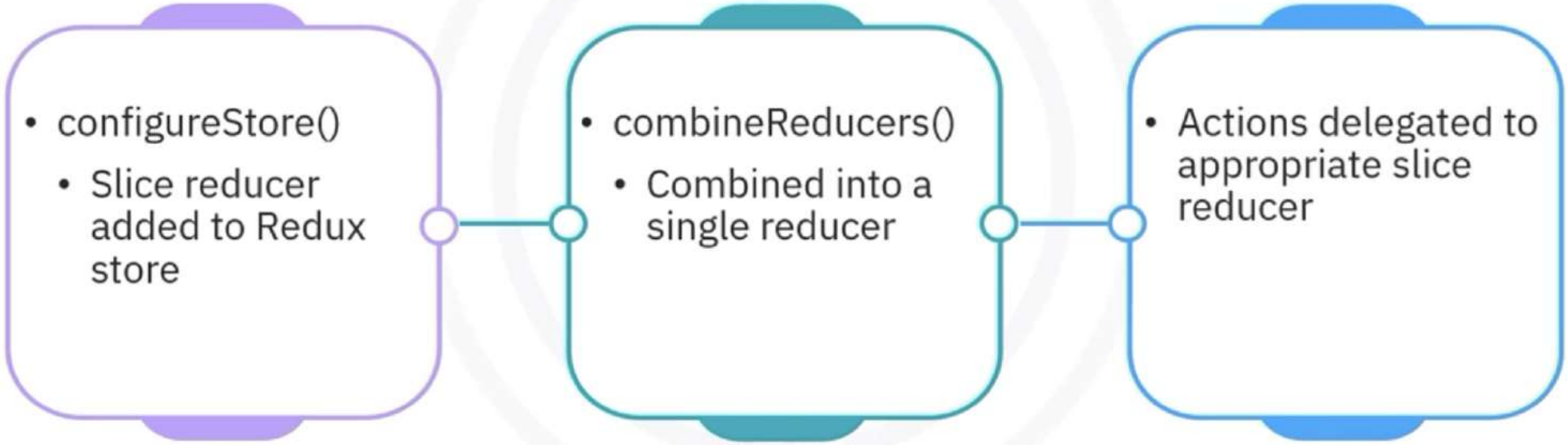
Relationship

- Slices define parts of the application state and logic to update them
- Store combines slices to form application state tree



Slices and store relationship

Integration



- configureStore()
  - Slice reducer added to Redux store

- combineReducers()
  - Combined into a single reducer

- Actions delegated to appropriate slice reducer

Slices and store relationship example

E-commerce application

- Incremental product quantity
- Total bill amount
- Number of super coins

Components

- App. jsx
- ProductQuantity.jsx
- CartValue.jsx
- CounterSlice.jsx
- Store. jsx
- Main. jsx

Components of the application

ProductQuantity.jsx

```
import React from 'react'
import { useDispatch, useSelector } from 'react-redux'
import { decrement, increment } from '../CounterSlice';
function ProductQuantity() {
  const dispatch=useDispatch();
  const counter=useSelector((state)=>state.counter.counter);
return (
  <>
  <h1>In Cart Product</h1>
  <div className="container">
  <h1> Products Number</h1>
   <div className="quantity">
    <div>Product Quanity</div>
```

Skills Network

IBM

Components of the application

ProductQuantity.jsx

```
<a href="#" onClick={()=>dispatch(increment())}><span>-</span></a>
      <input type="text" value={counter}/>
      <a href="#" onClick={()=>dispatch(decrement())}><span>+</span></a>
     </div>
    </div>
    </>
  )
}
export default ProductQuantity
```

Components of the application

CartValue.jsx

```
import React from 'react'
import { useSelector } from 'react-redux';

export const CartValue = () => {
    const counter = useSelector((state) => state.counter.counter);

let totalAmount=counter*100+(Math.random()*15);
  return (
   <>
<h2>The Total amount is {totalAmount}</h2>
   </>
  )
}
```

Components of the application

CounterSlice.jsx

```
import { createSlice } from "@reduxjs/toolkit";
export const CounterSlice=createSlice({
    name:'counter',
    initialState:{
        counter:0
    },
    reducers:{increment:(state)=>{state.counter+=1
        },     decrement:(state)=>{state.counter-=1;
        },     }
});
export const{increment,decrement}=CounterSlice.actions;
export default CounterSlice.reducer;
```

Components of the application

Store. jsx

```
import { configureStore } from "@reduxjs/toolkit";
import counterReducer from './CounterSlice'
export default configureStore({reducer:{counter:counterReducer
    }
})
```

Components of the application

main. jsx

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'
import store from './store.js'
import { Provider } from 'react-redux'
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <Provider store={store}>
    <App />
    </Provider>    </React.StrictMode>,
)
```

Recap

In this video, you learned that:

- In the context of React, the Redux toolkit (RTK) is an official package, the Redux team provides, to simplify Redux development and make it more efficient
- Redux toolkit provides a configureStore() function that combines several pieces of Redux setup logic into a single function call
- Redux toolkit introduces the createSlice() function, which allows developers to define “slice reducers” that automatically handle immutable updates to the state
- A slice in the Redux toolkit represents a piece of your application state and the logic to update it
- The Redux store is a single JavaScript object that holds the complete state tree of your application