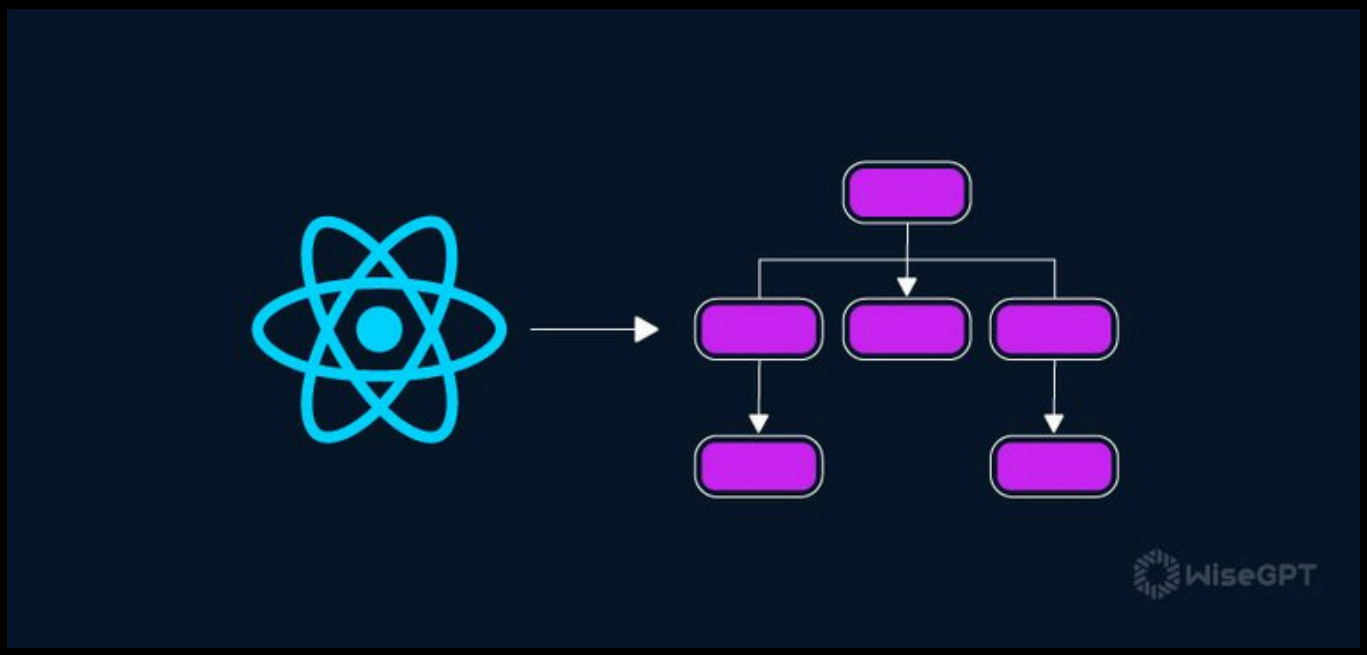
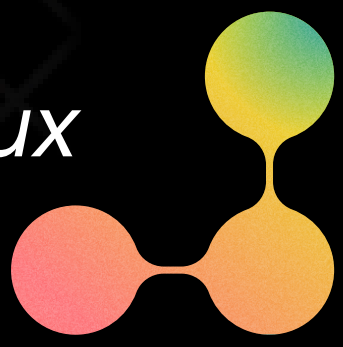
REDUX VS CONTEXT API



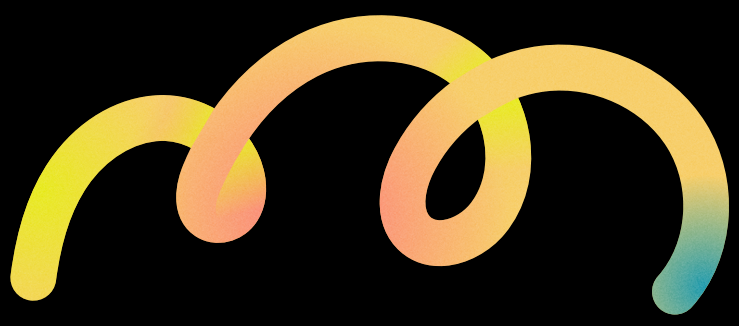
INTRODUCTION

- ***Single Source:*** *State management allows you to manage a single source of state across your application.*
- ***Centralization:*** *Centralized state management helps you to track and manage your states efficiently.*
- ***Managing consistency:*** *By managing states effectively, you ensure that UI consistently reflects current data. Also improved debugging.*
- *Two major ways are using Redux and Context API*



USECASE OF STATE MANAGEMENT

- ***User Authentication:*** *Maintains user login status and user role.*
- ***Form Handling:*** *Manage form input values and track form submission and errors.*
- ***Theme and Layout:*** *Maintain consistent theme and layout across application*
- ***Shopping Cart:*** *Add, Remove and Update items and smooth checkout process.*
- ***Data:*** *Real time data across application.*

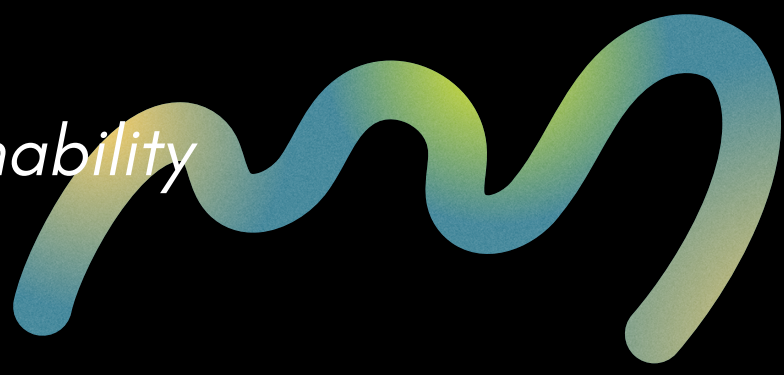


sahu-himanshu

WHAT IS CONTEXT API

Context API is a feature in React which allows you to pass data across component tree without passing states through each component

1. ***Purpose**: It allows you to manage global state and avoid "prop drilling".*
2. ***Main components:***
   - *React.createContext(): Creates a Context object*
   - *Context.Provider: Wraps components that need access to the context*
   - *Context.Consumer or useContext hook: Used to access the context value*
3. ***Benefits**:*
   - *Simplifies state management for certain types of data*
   - *Reduces component complexity*
   - *Improves code readability and maintainability*

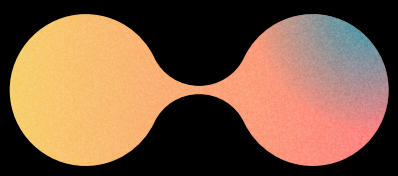


sahu-himanshu

IMPLEMENTATION OF CONTEXT


```jsx
// App.js
import React, { createContext, useState } from 'react';

// Create a context
export const MyContext = createContext();

export const App = () => {
  const [value, setValue] = useState("Hello, World!");
  return (
    <MyContext.Provider value={{ value, setValue }}>
      <Child />
    </MyContext.Provider>
  );
};
```

```jsx
// Child.js
import React, { useContext } from 'react';
import { MyContext } from './MyContext';

const Child = () => {
  const { value, setValue } = useContext(MyContext);
  return (
    <div>
      <p>{value}</p>
      <button onClick={() => setValue("Hello, React!")}>
        Change Value</button>
    </div>
  );
};
export default Child;
```



sahu-himanshu

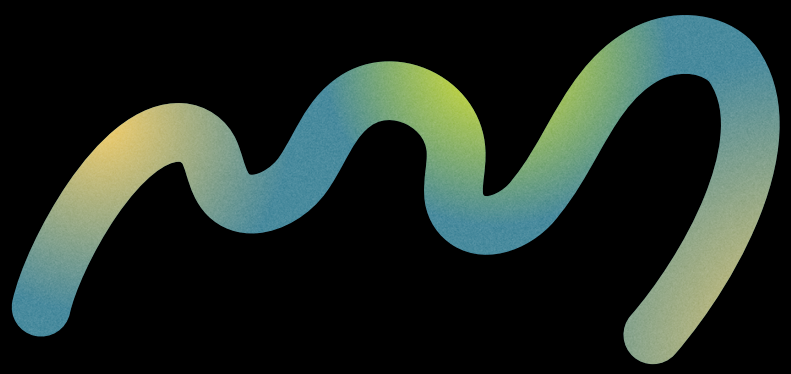
5 WHAT IS REDUX?

Redux is a popular state management library used primarily in JavaScript applications. It helps manage and centralize application state, making it easier to handle complex state transitions and data flow.

1. ***Purpose**: The main purpose of Redux is to provide a centralized store for the application's state, ensuring that the state is consistent and predictable throughout the app.*
2. ***Main Components:***

- ***Store:** All state changes are managed through the store.*
- ***Actions:** Plain JavaScript objects that describe what happened in the application.*
- ***Middlewares:** Optional components that sit between dispatching an action and the moment it reaches the reducer.*
- ***Reducers:** Pure functions that take the current state and an action as input and return a new state.*
- ***Selectors:** Functions that extract specific pieces of data from the state.*

3. ***Benefits***:
   - *Predictable State Management: The state is predictable due to the unidirectional data flow, making debugging and testing easier.*
   - *Centralized State: All application state is stored in a single store, providing a global state management solution and eliminating the need for passing props down through the component tree.*
   - *Easy State Sharing: Sharing state between different parts of the application becomes straightforward with Redux.*
   - *Scalability: Redux is highly scalable and works well for applications of all sizes.*



IMPLEMENTATION OF REDUX

Step1: *Install Redux in your app*
npm install redux react-redux

Step2: *Create the Redux Store*

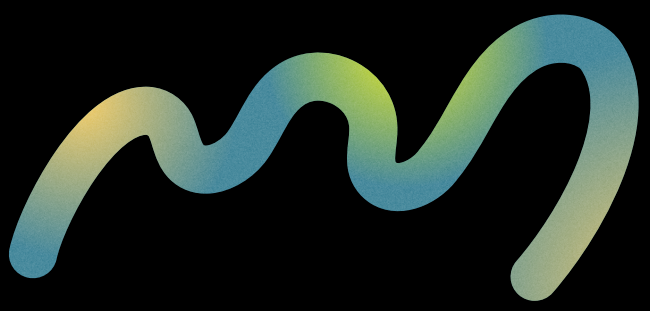
```
// Reducer function
const counterReducer = (state = initialState, action) => {
  switch (action.type) {
    case 'INCREMENT':
      return {
        ...state,
        counter: state.counter + 1,
      };
    case 'DECREMENT':
      return {
        ...state,
        counter: state.counter - 1,
      };
    default:
      return state;
  }
};
```

```
// src/store.js
import { createStore } from 'redux';

// Initial state
const initialState = {
 counter: 0,
};


// Create Redux store
const store =
createStore(counterReducer);

export default store;
```

sahu-himanshu

Step3: Create Action Creators

```
// src/actions.js
export const increment = () => {
  return {
    type: 'INCREMENT',
  };
};

export const decrement = () => {
  return {
    type: 'DECREMENT',
  };
};
```

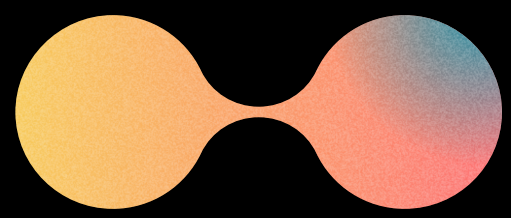


Step4: Set Up the React Application

```
// src/index.js
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')
);
```

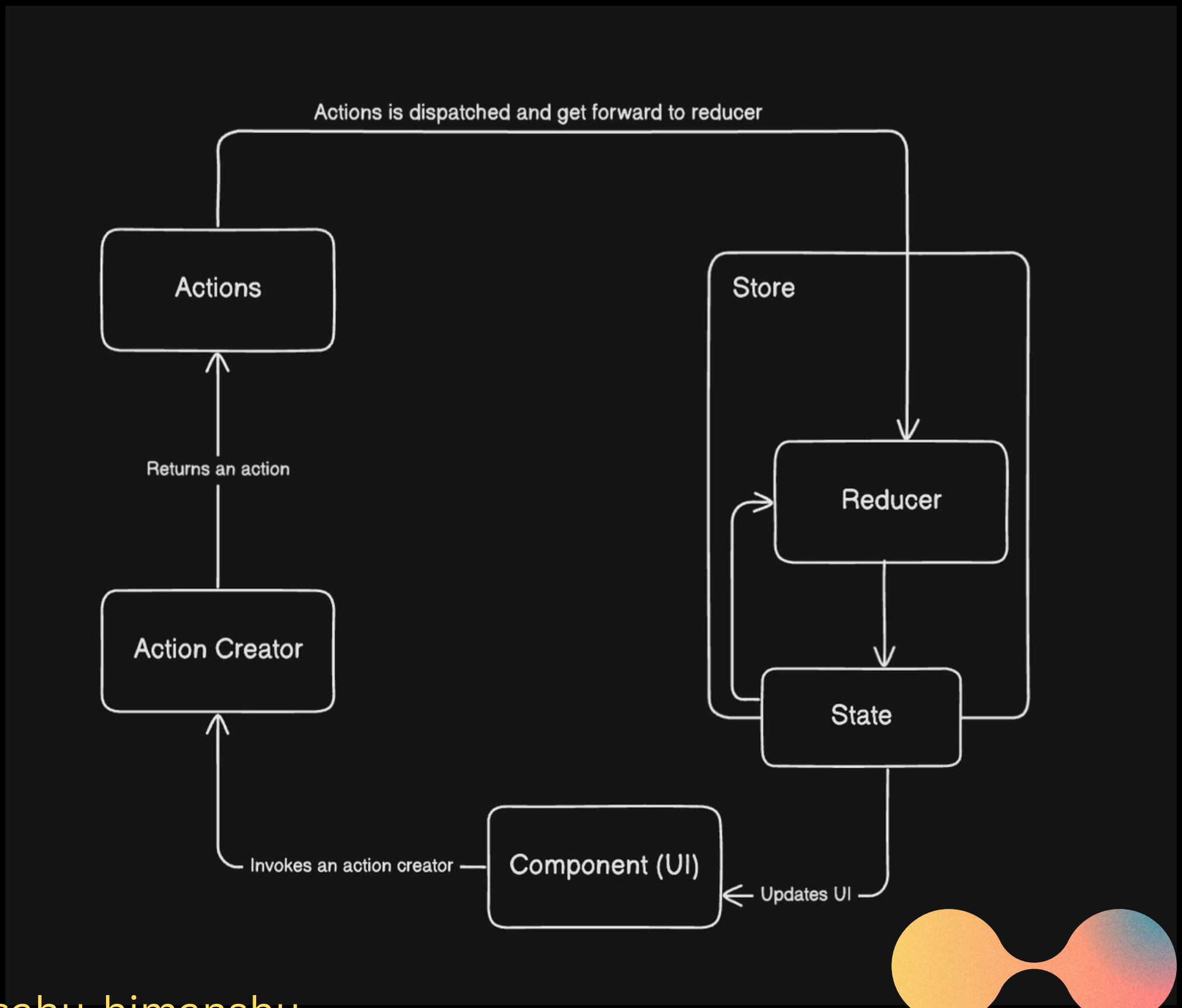
Step5: *Connect React Components to Redux*

```
// src/App.js
import React from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { increment, decrement } from './actions';

function App() {
  const counter = useSelector((state) => state.counter);
  const dispatch = useDispatch();

  return (
    <div>
      <h1>Counter: {counter}</h1>
      <button onClick={() => dispatch(increment())}>Increment</button>
      <button onClick={() =>
dispatch(decrement())}>Decrement</button>
    </div>
  );
}

export default App;
```

10

Redux Flowchart

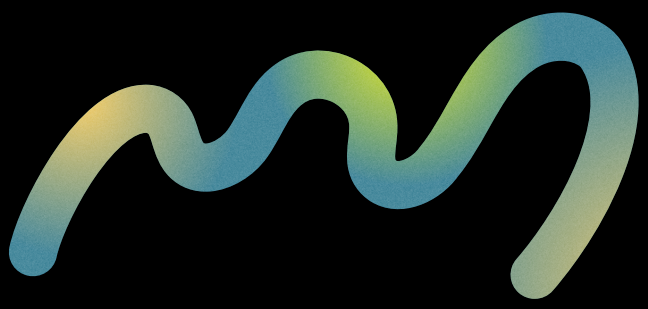


sahu-himanshu

Difference: Redux vs Context

Context API: *Simplifies state management for small applications or specific components. It provides a way to pass data through the component tree without manually passing props.*

Redux*: A more robust solution for managing global state in complex applications. It centralizes state management, making it easier to handle complex state logic and interactions with middleware for side effects.*



sahu-himanshu

Thanks

Guys

sahu-himanshu