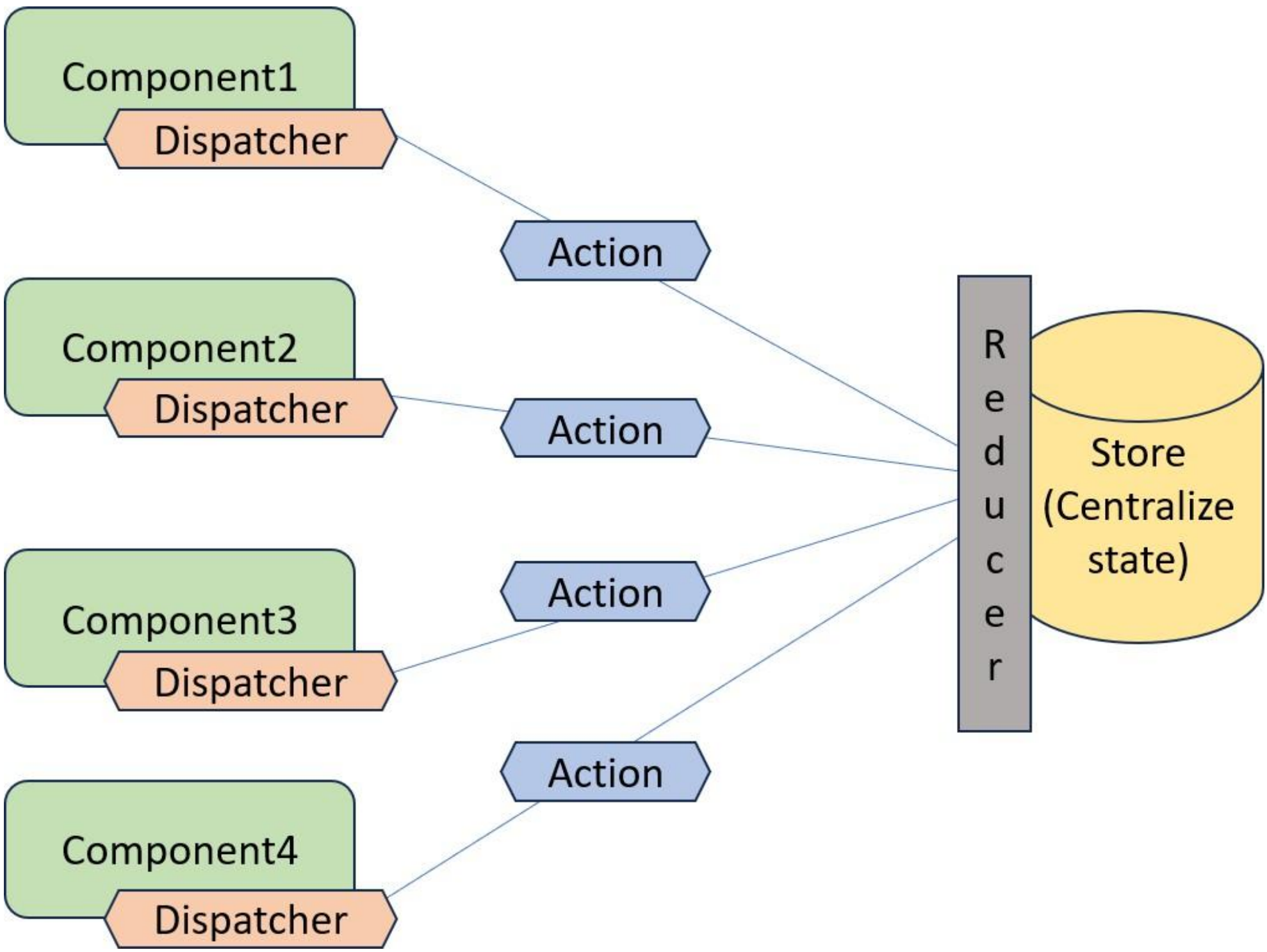
TOP 10
INTERVIEW QUESTIONS

REDUX

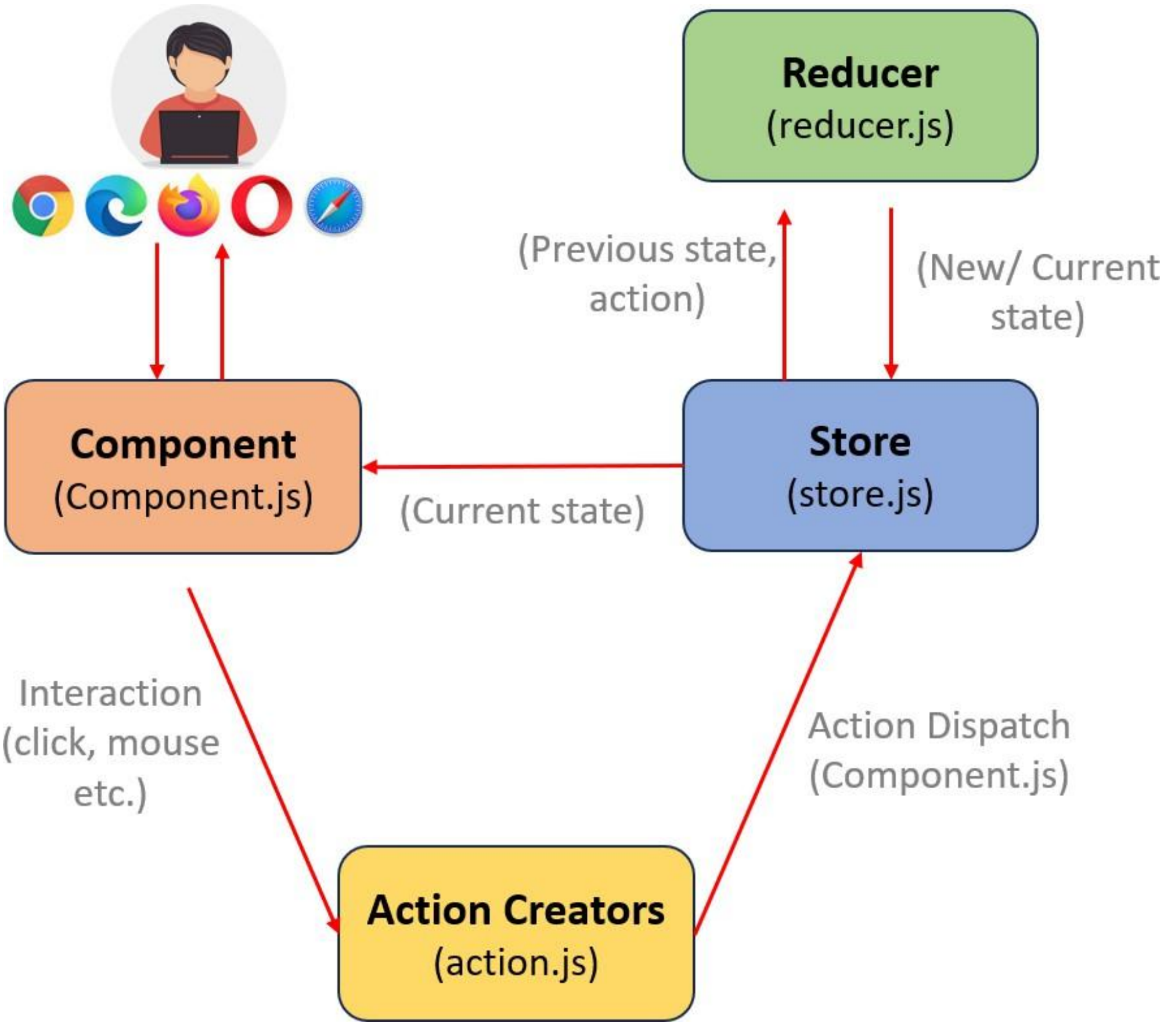
Q. What is the role of Redux in React?

- Redux is an open-source JavaScript library used for **state management**.
- Redux provides a **centralized store** that holds the entire state of an application and allows components to access and update the state in a predictable manner.



Q. What is the Flow of data in React while using Redux?

- **Flow of Data in React-Redux application**



Q. What is the role of Store in React Redux?

- Redux store enables the application to **update state using the defined reducer**.
- Redux Store is a **centralized place** for holding the state of all the components in the application.

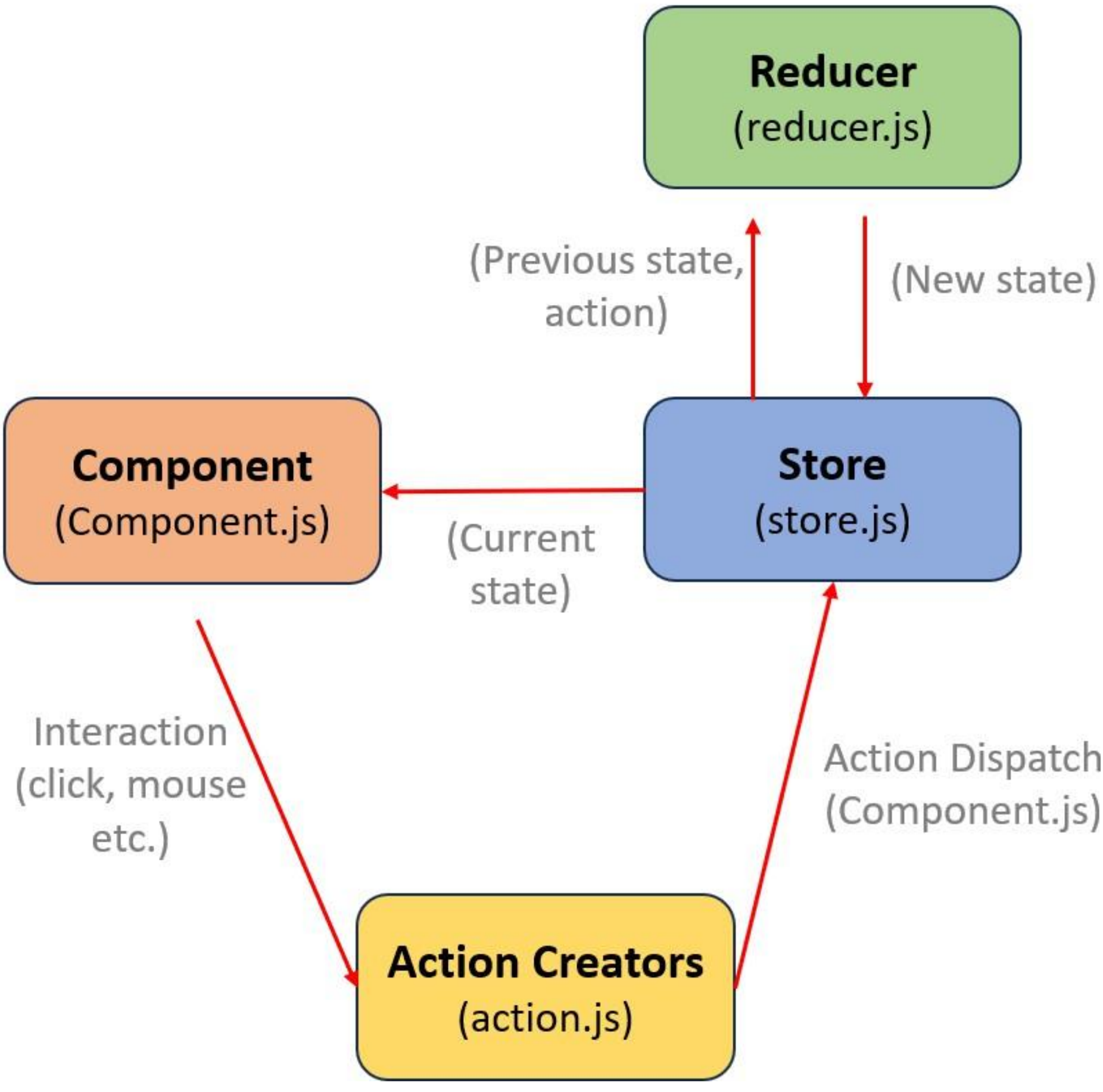
```
// store.js
import { configureStore } from '@reduxjs/toolkit';
import counterReducer from './reducer';

const store = configureStore({
  reducer: {
    counter: counterReducer,
  },
});

export default store;
```

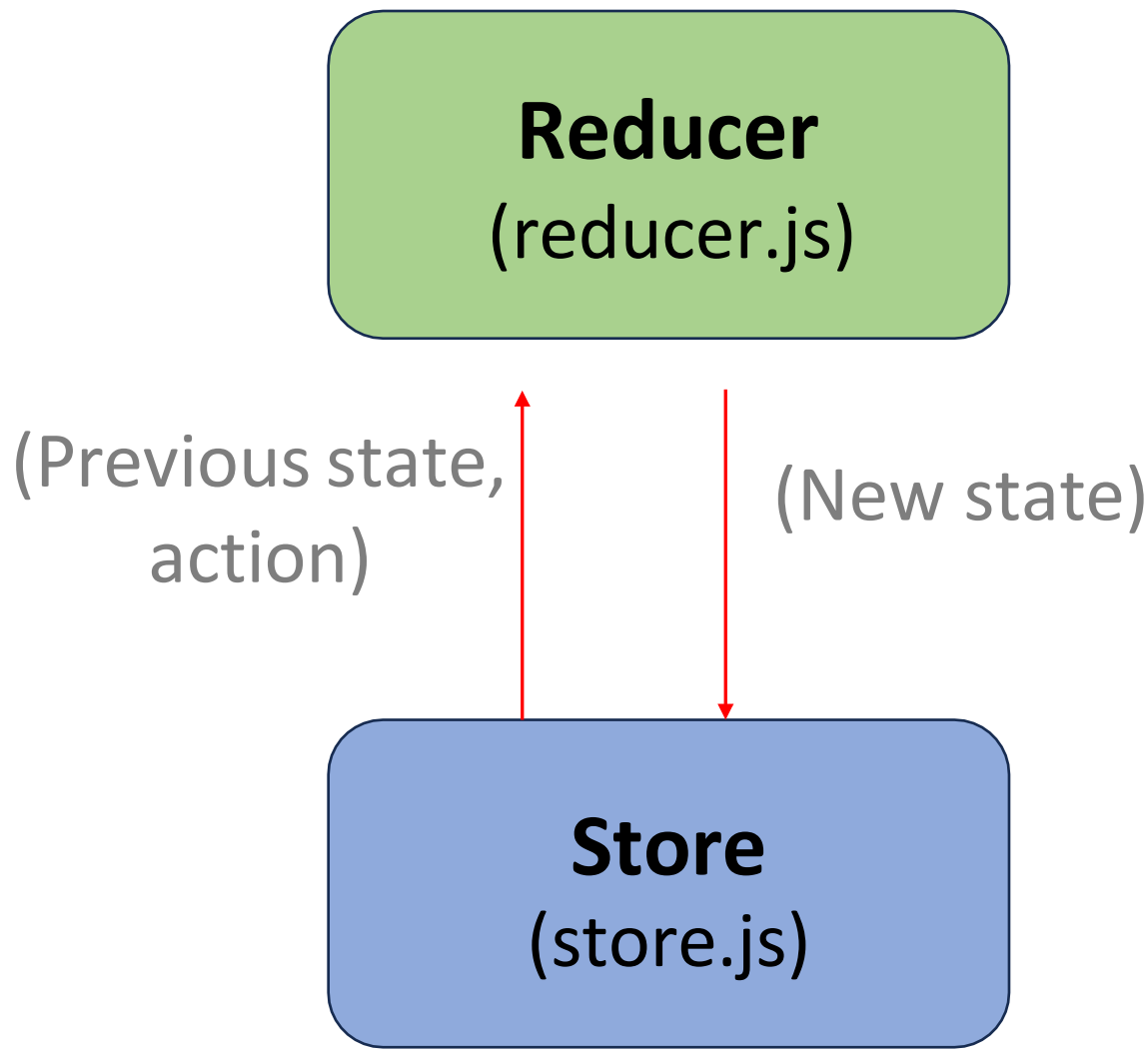
Q. What is the role of Store in React Redux?

- **Flow of Data in React-Redux application**



Q. What is the role of Reducer in Redux?

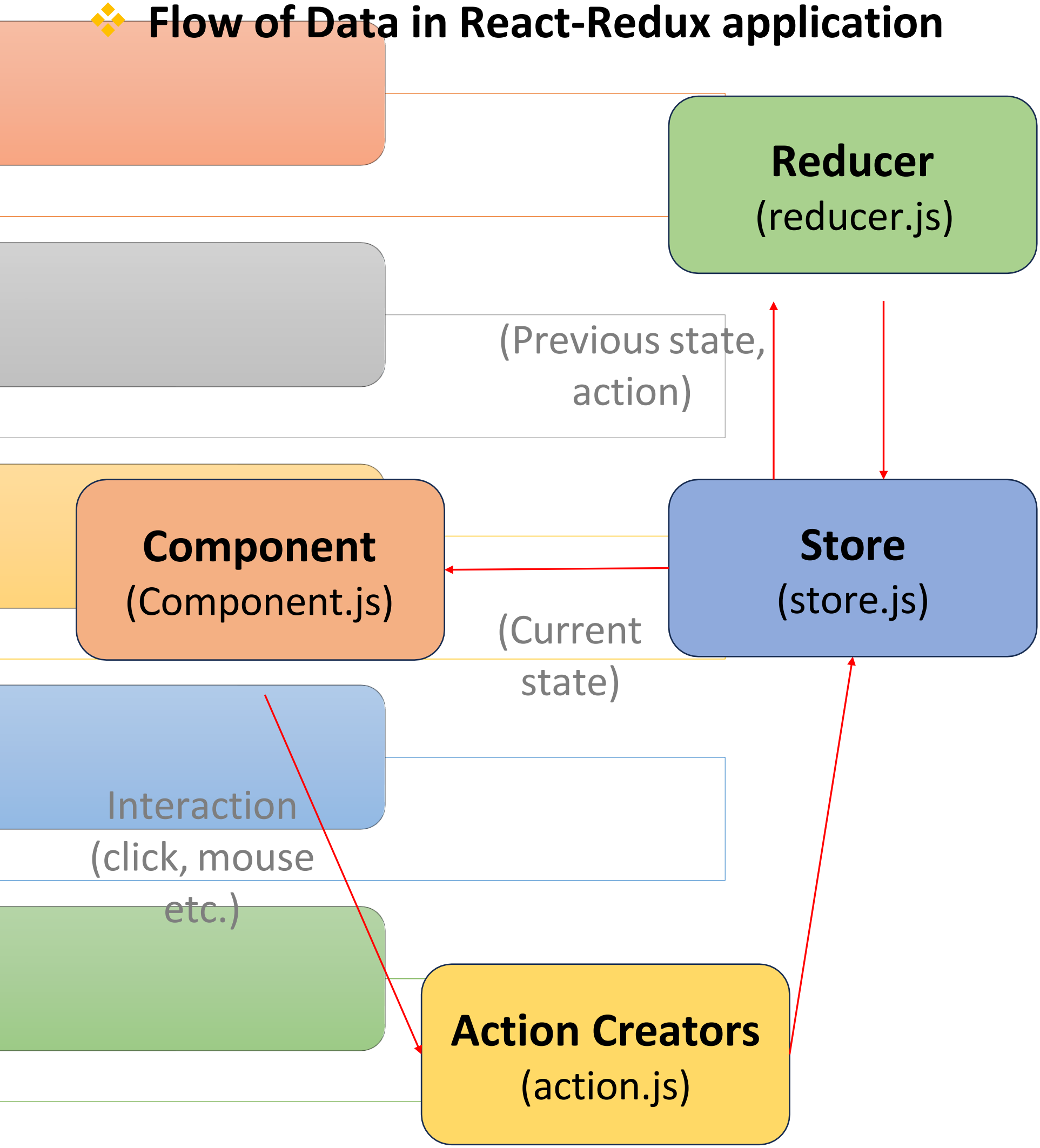
- A reducer is a function that takes the **previous state** and **an action** as arguments and returns the new state of the application.




```
// reducer.js
const counterReducer = (state = { count: 0 }, action) => {
  switch (action.type) {
    case "INCREMENT":
      return { count: state.count + 1 };
    case "DECREMENT":
      return { count: state.count - 1 };
    default:
      return state;
  }
};
export default counterReducer;
```

Q. Explain the Core Principles of Redux?

- **Flow of Data in React-Redux application**



Q. Explain the Core Principles of Redux?

1. Single Source of Truth (Store)

2. State is Read-Only (Unidirectional)

3. Changes using Pure Functions (Reducers)

4. Actions Trigger State Changes (Actions)

5. Predictable State Changes (Actions)

Q. Explain the Core Principles of Redux?

1. Single Source of Truth:	The entire application state is stored in one place, simplifying data management and ensuring a consistent view of the application.
2. State is Read-Only:	State cannot be directly modified. To make changes to the state, you need to dispatch an action. This ensures that the state transitions are explicit and traceable.
3. Changes using Pure Functions (Reducers):	This ensures predictability and consistency because pure functions returns the same result if the same arguments are passed.
4. Actions Trigger State Changes:	Plain JavaScript objects (actions) describe state changes, guiding the store to invoke reducers and update the application state accordingly.
5. Predictable State Changes with Actions:	State changes are determined by actions, fostering a predictable flow of data and simplifying debugging in response to specific actions.

Q. What are the differences between local component state & Redux state?

Local Component State	Redux State
1. Scope: Limited to the component where defined.	Global and accessible across components.
2. Management: Managed internally by the component.	Managed externally by the Redux store.
3. Performance: Generally, more performant for small-scale applications.	More performant for large applications.
4. Complexity: Simpler to set up and manage.	Comparatively complex to manage.
5. Testing: Simpler to test with component-specific state.	Requires more comprehensive testing due to global nature and interactions between components.

Q. What are the challenges or disadvantages while using Redux?

1. Boilerplate Code
2. Learning Curve
3. Verbosity and Complexity
4. Overhead for Small Projects
5. Global State for Local Components
6. Integration with Non-React Libraries

Q. What are the challenges or disadvantages while using Redux?

1. Boilerplate Code:	Implementing Redux requires writing extensive boilerplate code in action, reducer, store, increasing code volume and complexity.
2. Learning Curve:	Understanding Redux concepts can be challenging, posing a learning curve for developers, especially those new to React state management.
3. Verbosity and Complexity:	As projects grow, Redux code may become verbose and complex, demanding careful management of actions and reducers.
4. Overhead for Small Projects:	In small projects, Redux may introduce unnecessary complexity, potentially outweighing its benefits in development efficiency.
5. Global State for Local Components:	Overusing Redux for local state introduces unnecessary complexity, as not all state requires a global scope.
6. Integration with Non-React Libraries:	Integrating Redux with non-React libraries or frameworks may demand additional effort and customization, potentially adding complexity to the project.

Q. What is Provider Component? How components getting the state from Redux store?

- Provider component of react-redux will **make the Redux store available** to all connected components.

```
// index.js
import { Provider } from "react-redux";
import store from "./Redux/store";

const root = ReactDOM.createRoot(
    document.getElementById("root"));
root.render(
  <Provider store={store}>
    <CounterComponent />
  </Provider>
);
```

```
// CounterComponent.js
// Map Redux state to component props
const mapStateToProps = (state) => {
  return {
    count: state.counter.count,
  };
};
```

Q. What is the role of Connect function in React-Redux?

- The connect function is used to **make the connection** between a React component and the Redux store.

```jsx
import React from "react";
import { connect } from "react-redux";

// 3. Map Redux actions to component props
const mapDispatchToProps = {
  increment,
  decrement,
};

// 4. Map Redux state to component props
const mapStateToProps = (state) => {
  return {
    count: state.counter.count, //From store
  };
};

// 5. Connect the component to Redux store
export default connect(mapStateToProps,
  mapDispatchToProps)(CounterComponent);
```

Q. Explain the concept of Middleware in React-Redux?

- Middleware provides a mechanism to **add extra functionality** to the Redux store.
- Middleware can intercept actions, modify them, or execute additional logic in actions before they reach the reducers.

```
// Configure the Redux store
// with middleware
const store = configureStore({
  reducer: rootReducer,
  middleware: [thunk],
});
```

Q. Explain the concept of Middleware in React-Redux?

- Flow of Data in React-Redux application with Middleware