Top 10 React Redux Interview Question

React is a User interface library. The core Redux is built upon action, state, and reducers. All the data resides at the store. Reacts components use actions in order to update the stores. Reducers help the store to update itself. Props make sure the updated stores are available for React.

Now Redux is a knowable state container built for JavaScript apps. It allows managing the application state, and it stands to be open source. Applications that run consistently help run in different environments (client, server, and native) and are easy to test.

Below are the 10 important Redux Interview Questions And Answers that are frequently asked in an interview.

Q1) Benefits of Redux?

- **Maintainability:** The maintenance of Redux becomes easier due to strict code structure and organization.
- **Organization:** code organization is very strict; hence the stability of the code is high, which intern increases the work to be much easier.
- **Server rendering:** This is useful, particularly to the preliminary render, which keeps up a better user experience or search engine optimization. The server-side created stores are forwarded to the client-side.
- **Developer tools:** It is Highly traceable, so changes in position and changes in the application; all such instances make the developers have a real-time experience.
- **Ease of testing:** The first rule of writing testable code is to write small functions that do only one thing and that are independent. Redux's code is made of functions that used to be: small, pure, and isolated.

Q2) How Distinct from MVC and Flux?

As far as MVC structure is concerned, the data, presentation, and logical layers are well separated and handled. Change to an application even at a smaller position may involve many changes through the application. this happens because data flow exists bidirectional as far as MVC is concerned. Maintenance of MVC structures are hardly complex, and Debugging also expects a lot of experience for it.

Flux stands closely related to redux. A story-based strategy allows capturing the changes applied to the application state, the event subscription, and the current state are connected by means of components. Call back payloads are broadcasted by means of Redux.

Q3) Functional programming concepts?

The various functional programming concepts used to structure Redux are listed below,

- Functions are treated as First-class objects.
- Capable of passing functions in the format of arguments.
- Capable of controlling flow using recursions, functions, and arrays.
- Helper functions such as reduce and map filters are used.
- Allows linking functions together.
- The state doesn’t change.
- Prioritizing the order of executing the code is not really necessary.

Q4) Redux change of state?

For a release of action, a change in state to an application is applied; this ensures an intent to change the state will be achieved.

Example:

- The user clicks a button in the application.
- A function is called in the form of a component.
- So now an action gets dispatched by the relative container.
- This happens because the prop (which was just called in the container) is tied to an action dispatcher using mapDispatchToProps (in the container).
- Reducer on capturing the action, it intern executes a function, and this function returns a new state with specific changes.
- The state change is known by the container and modifies a specific prop in the component due to the mapStateToProps function.

Q5) Where can Redux be used?

Redux is majorly used in combination with reacting. It also has the ability to get used to other view libraries too. Some of the famous entities like AngularJS, Vue.js, and Meteor. It can get combined with Redux easily. This is a key reason for the popularity of Redux in its ecosystem. So many articles, tutorials, middleware, tools, and boilerplates are available.

Q6) What is the typical flow of data in a React + Redux app?

Call-back from the UI component dispatches an action with a payload; these dispatched actions are intercepted and received by the reducers. This interception will generate a new application state. From here, the actions will be propagated down through a hierarchy of components from the Redux store. The below diagram depicts the entity structure of a redux+react setup.

Q7) What is store in redux?

The store holds the application state and supplies the helper methods for accessing the state.
Register listeners and dispatch actions. There is only one Store while using Redux. The store is configured via the createStorefunction. The single store represents the entire state. Reducers return a state via action.

```
export function configureStore(initialState) {
    return createStore(rootReducer, initialState);
}
```

The root reducer is a collection of all reducers in the application.

```
const rootReducer = combineReducers({
    donors: donorReducer,
});
```

Q8) Explain Reducers in Redux?

The state of a store is updated by means of reducer functions. A stable collection of reducers form a store, and each of the stores maintains a separate state associated with itself. To update the array of donors, we should define a donor application.
The reducer as follows.

```
export default function donorReducer(state = [], action) {
    switch (action.type) {
        case actionTypes.addDonor:
            return [...state, action.donor];
        default:
            return state;
    }
}
```

The reducers receive the initial state and action. Based on the action type, it returns a new state for the store. The state maintained by reducers is immutable. The below-given reducer it holds the current state and action as an argument for it and then returns the next.

```
state: function handelingAuthentication(st, actn) {
    return _.assign({}, st,
        {
            auth: actn.pyload
        });
}
```

Q9) Redux workflow features?

- **Reset:** Allow to reset the state of the store
- **Revert:** Rollback to the last committed state
- **Sweep:** All disabled actions that you might have fired by mistake will be removed
- **Commit:** Makes the current state the initial state

Q10) Explain action's in Redux?

Actions in Redux are functions that return an action object. The action-type and the action data are packed in the action object. Which also allows a donor to be added to the system. Actions send data between the store and the application. The actions produce all information retrieved by the store.

```
export function addDonorAction(donor) {
    return {
        type: actionTypes.addDonor,
        donor,
    };
}
```