**What’s Flux:**

Flux is an architectural pattern which enforces the uni-directional data flow. It controls derived data and enables communication between multiple components using a central Store which has authority for all data. Any update in data throughout the application must occur here only. Flux provides stability to the application and reduces run-time errors.

**What is Redux?**

Redux is one of the hottest libraries for front-end development in today’s marketplace. It is a predictable state container for JavaScript applications and is used for the entire applications state management. Applications developed with Redux are easy to test and can run in different environments showing consistent behavior.

**What are the three principles that Redux follows?**

1. ***Single source of truth:***The state of the entire application is stored in an object/ state tree within a single store. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.
2. ***State is read-only:***The only way to change the state is to trigger an action. An action is a plain JS object describing the change. Just like state is the minimal representation of data, the action is the minimal representation of the change to that data.
3. ***Changes are made with pure functions:*** In order to specify how the state tree is transformed by actions, you need pure functions. Pure functions are those whose return value depends solely on the values of their arguments.

### ****List down the components of Redux****

Redux is composed of the following components:

1. **Action** – It’s an object that describes what happened.
2. **Reducer**–  It is a place to determine how the state will change.
3. **Store** – State/ Object tree of the entire application is saved in the Store.
4. **View** – Simply displays the data provided by the Store.

### ****Show how the data flows through Redux?****



### ****How are Actions defined in Redux?****

Actions in React must have a type property that indicates the type of ACTION being performed. They must be defined as a String constant and you can add more properties to it as well. In Redux, actions are created using the functions called Action Creators. Below is an example of Action and Action Creator:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | function addTodo(text) {         return {                  type: ADD\_TODO,                   text      }  } |

### ****Explain the role of Reducer.****

Reducers are pure functions which specify how the application’s state changes in response to an ACTION. Reducers work by taking in the previous state and action, and then it returns a new state. It determines what sort of update needs to be done based on the type of the action, and then returns new values. It returns the previous state as it is, if no work needs to be done.

### ****What is the significance of Store in Redux?****

A store is a JavaScript object which can hold the application’s state and provide a few helper methods to access the state, dispatch actions and register listeners. The entire state/ object tree of an application is saved in a single store. As a result of this, Redux is very simple and predictable. We can pass middleware to the store to handle the processing of data as well as to keep a log of various actions that change the state of stores. All the actions return a new state via reducers.

### ****How is Redux different from Flux?****

|  |  |
| --- | --- |
| Flux vs Redux | |
| **Flux** | **Redux** |
| 1. The Store contains state and change logic | 1. Store and change logic are separate |
| 2. There are multiple stores | 2. There is only one store (in object and tree structure) |
| 3. All the stores are disconnected and flat | 3. Single store with hierarchical reducers |
| 4. Has singleton dispatcher | 4. No concept of dispatcher |
| 5. React components subscribe to the store | 5. Container components utilize connect() |
| 6. State is mutable(changeable) | 6. State is immutable(read only) |

### ****What are the advantages of Redux?****

Advantages of Redux are listed below:

* **Predictability of outcome –**Since there is always one source of truth, i.e. the store, there is no confusion about how to sync the current state with actions and other parts of the application.
* **Maintainability –**The code becomes easier to maintain with a predictable outcome and strict structure.
* **Server-side rendering –** You just need to pass the store created on the server, to the client side. This is very useful for initial render and provides a better user experience as it optimizes the application performance.
* **Developer tools –**From actions to state changes, developers can track everything going on in the application in real time.
* **Community and ecosystem –**Redux has a huge community behind it which makes it even more captivating to use. A large community of talented individuals contribute to the betterment of the library and develop various applications with it.
* **Ease of testing –**Redux’s code is mostly functions which are small, pure and isolated. This makes the code testable and independent.
* **Organization –**Redux is precise about how code should be organized, this makes the code more consistent and easier when a team works with it.

### ****What is React Router?****

React Router is a powerful routing library built on top of React, which helps in adding new screens and flows to the application. This keeps the URL in sync with data that’s being displayed on the web page. It maintains a standardized structure and behavior and is used for developing single page web applications. React Router has a simple API.

### ****Why**** ****is switch keyword used in React Router v4?****

Although a **<div>** is used to encapsulate multiple routes inside the Router. The ‘switch’ keyword is used when you want to display only a single route to be rendered amongst the several defined routes. The **<switch>**tag when in use matches the typed URL with the defined routes in sequential order. When the first match is found, it renders the specified route. Thereby bypassing the remaining routes.

### ****Why do we need a Router in React?****

A Router is used to define multiple routes and when a user types a specific URL, if this URL matches the path of any ‘route’ defined inside the router, then the user is redirected to that particular route. So basically, we need to add a Router library to our app that allows creating multiple routes with each leading to us a unique view.

|  |  |
| --- | --- |
| 1  2  3  4  5 | <switch>      <route exact path=’/’ component={Home}/>      <route path=’/posts/:id’ component={Newpost}/>      <route path=’/posts’   component={Post}/>  </switch> |

### ****List down the advantages of React Router.****

Few advantages are:

1. Just like how React is based on components, in React Router v4, the API is ‘All About Components’. A Router can be visualized as a single root component (**<BrowserRouter>**) in which we enclose the specific child routes (**<route>**).
2. No need to manually set History value: In React Router v4, all we need to do is wrap our routes within the **<BrowserRouter>** component.
3. The packages are split: Three packages one each for Web, Native and Core. This supports the compact size of our application. It is easy to switch over based on a similar coding style.