|  |  |
| --- | --- |
| **ch35.ReactJS Flux Concept** | **Date: 22-02-2022** |

**Topics**

React Props Validation,

# React Props Validation

# React Flux Concept

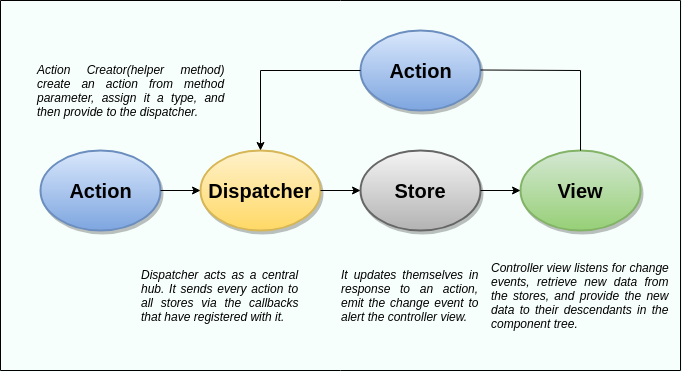
Flux is an application architecture that Facebook uses internally for building the client-side web application with React. It is not a library nor a framework. It is neither a library nor a framework. It is a kind of architecture that complements React as view and follows the concept of Unidirectional Data Flow model. It is useful when the project has dynamic data, and we need to keep the data updated in an effective manner. It reduces the runtime errors.

Flux applications have three major roles in dealing with data:

1. Dispatcher
2. Stores
3. Views (React components)

Here, you should not be confused with the Model-View-Controller (MVC) model. Although, Controllers exists in both, but Flux controller-views (views) found at the top of the hierarchy. It retrieves data from the stores and then passes this data down to their children. Additionally, action creators - dispatcher helper methods used to describe all changes that are possible in the application. It can be useful as a fourth part of the Flux update cycle.

## **Structure and Data Flow**



In Flux application, data flows in a single direction(unidirectional). This data flow is central to the flux pattern. The dispatcher, stores, and views are independent nodes with inputs and outputs. The actions are simple objects that contain new data and type property. Now, let us look at the various components of flux architecture one by one.

### **Dispatcher**

It is a central hub for the React Flux application and manages all data flow of your Flux application. It is a registry of callbacks into the stores. It has no real intelligence of its own, and simply acts as a mechanism for distributing the actions to the stores. All stores register itself and provide a callback. It is a place which handled all events that modify the store. When an action creator provides a new action to the dispatcher, all stores receive that action via the callbacks in the registry.

The dispatcher's API has five methods. These are:

|  |  |  |
| --- | --- | --- |
| **SN** | **Methods** | **Descriptions** |
| 1. | register() | It is used to register a store's action handler callback. |
| 2. | unregister() | It is used to unregisters a store's callback. |
| 3. | waitFor() | It is used to wait for the specified callback to run first. |
| 4. | dispatch() | It is used to dispatches an action. |
| 5. | isDispatching() | It is used to checks if the dispatcher is currently dispatching an action. |

### **Stores**

It primarily contains the application state and logic. It is similar to the model in a traditional MVC. It is used for maintaining a particular state within the application, updates themselves in response to an action, and emit the change event to alert the controller view.

### **Views**

It is also called as controller-views. It is located at the top of the chain to store the logic to generate actions and receive new data from the store. It is a React component listen to change events and receives the data from the stores and re-render the application.

### **Actions**

The dispatcher method allows us to trigger a dispatch to the store and include a payload of data, which we call an action. It is an action creator or helper methods that pass the data to the dispatcher.

### **Advantage of Flux**

* It is a unidirectional data flow model which is easy to understand.
* It is open source and more of a design pattern than a formal framework like MVC architecture.
* The flux application is easier to maintain.
* The flux application parts are decoupled.

Next Topic[Flux Vs. MVC](https://www.javatpoint.com/react-flux-vs-mvc)