|  |  |
| --- | --- |
| **ch36.ReactJS Flux Vs MVC** | **Date: 22-02-2022** |

**Topics**

React Props Validation,

# React Props Validation

# React Flux Vs. MVC

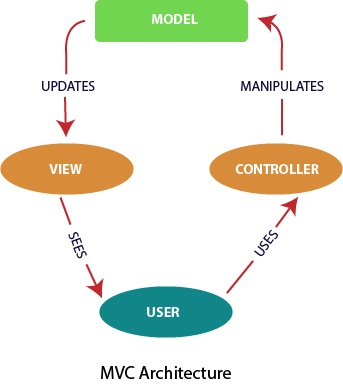
## **MVC**

MVC stands for Model View Controller. It is an architectural pattern used for developing the user interface. It divides the application into three different logical components: the Model, the View, and the Controller. It is first introduced in 1976 in the Smalltalk programming language. In MVC, each component is built to handle specific development aspect of an application. It is one of the most used web development frameworks to create scalable projects.

### **MVC Architecture**

The MVC architecture contains the three components. These are:

* **Model:** It is responsible for maintaining the behavior and data of an application.
* **View:** It is used to display the model in the user interface.
* **Controller:** It acts as an interface between the Model and the View components. It takes user input, manipulates the data(model) and causes the view to update.

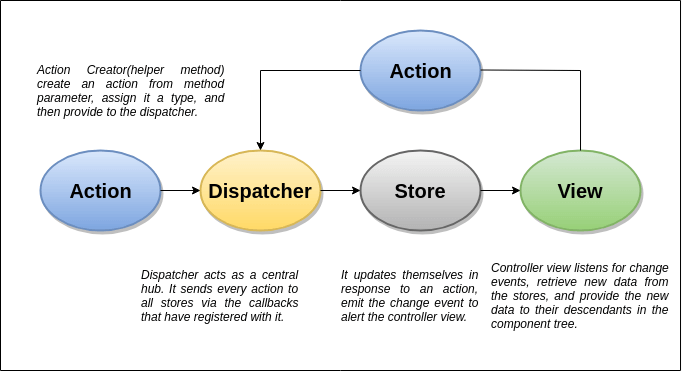


## **Flux**

According to the official site, Flux is the application architecture that Facebook uses for building client-side web applications. It is an alternative to MVC architecture and other software design patterns for managing how data flows in the react application. It is the backbone of all React application. It is not a library nor a framework. It complements React as view and follows the concept of Unidirectional Data Flow model.

Flux Architecture has three major roles in dealing with data:

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



## **MVC Vs. Flux**

|  |  |  |
| --- | --- | --- |
| **SN** | **MVC** | **FLUX** |
| **1.** | It was introduced in 1976. | It was introduced just a few years ago. |
| **2.** | It supports Bi-directional data Flow model. | It supports Uni-directional data flow model. |
| **3.** | In this, data binding is the key. | In this, events or actions are the keys. |
| **4.** | It is synchronous. | It is asynchronous. |
| **5.** | Here, controllers handle everything(logic). | Here, stores handle all logic. |
| **6.** | It is hard to debug. | It is easy to debug because it has common initiating point: Dispatcher. |
| **7.** | It is difficult to understand as the project size increases. | It is easy to understand. |
| **8.** | Its maintainability is difficult as the project scope goes huge. | Its maintainability is easy and reduces runtime errors. |
| **9.** | Testing of application is difficult. | Testing of application is easy. |
| **10.** | Scalability is complex. | It can be easily scalable. |

Next Topic[React Redux](https://www.javatpoint.com/react-redux)