Flux Concepts

Overview

Flux is an architectural pattern that follow a unidirectional flow of data through your Application mostly used with React, Introduced by ‘Facebook’. It is neither a framework nor a library.

It was designed at Facebook along with the React View Library. It focuses on creating explicit and understandable paths for your application's data. Which makes tracing changes during development simpler and makes bugs easier to track down and fix.

React is library, not a framework, used to represent ‘VIEW’ part of your application. React is not capable to manage the state of data in a complex application. To manage such a complex application’s data, we used FLUX architecture.

Parts

- Dispatchers
- Stores
- Action
- View

Dispatchers

The dispatcher receives actions and dispatches them to stores that have registered with the dispatcher. **Every store will receive every action.** There should be only one singleton dispatcher in each application.

Example:

1. User types in title for a todo and hits enter.
2. The view captures this event and **dispatches** an "add-todo" action containing the title of the todo.
3. **Every store** will then receive this action.

Stores

Collection of many objects

2 types of data flows

- Bidirectional- MVC
- unidirectional -best

Bidirectional places a large amount of stress on the MVC(Model view controller) as it tries to maintain application state and data.

Unidirectional - Any changes in the view layer will have changes to the data layer

How does it work?

Store registers itself with the dispatch and provides it with callback
Callback receives action as a parameter
Used in the store's methods.
Allowing the action to update the state

Actions

Action creator

- Send out requests/commands to the dispatcher
- To trigger the data flow, send the relevant data
- Interact with the application

Action type

- Use to define actions with types

```
{
  type: 'ADD_TODO',
  text: 'TODO content'
}
```

Views

- Independent framework, i.e. does not need to use React or similar framework as the dispatcher.
- Listens and reacts to the change events by dispatcher.
- Actions are dispatched from views as the user interacts with the interface.
- Every component that reflect changes in the view is linked to the store.

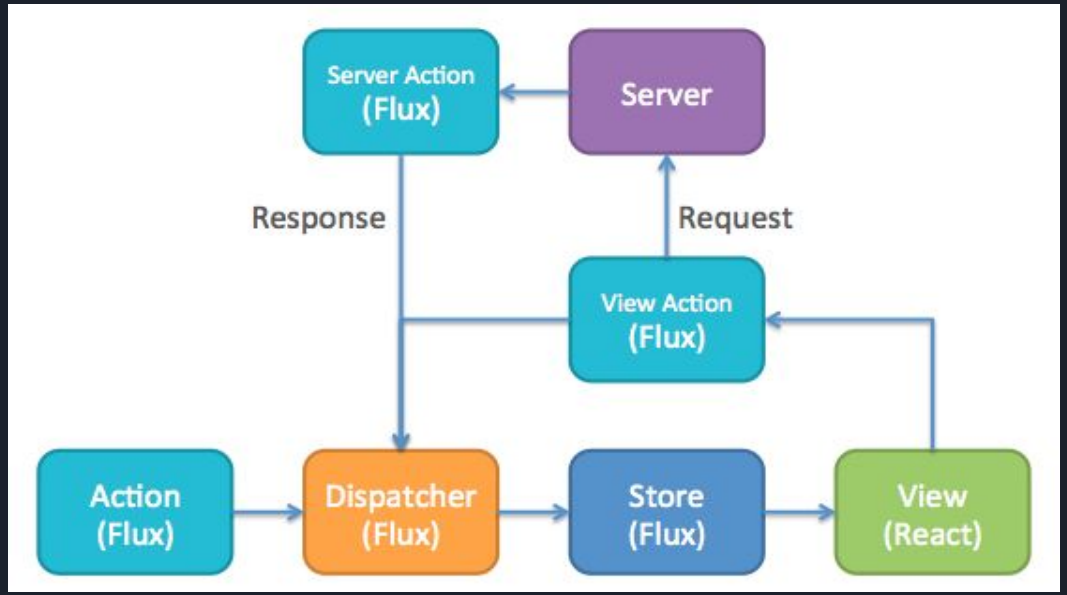
Eg: When writing text on an editor, the view gets the input from user and the dispatcher renders them to the store, and finally the view shows the changes in the editor.

setstate()

forceUpdate()

Flow of Data

- Data flows through the Dispatcher as a central hub.
- Actions are provided to the Dispatcher, and most often originate from user interactions with the Views.
- The Dispatcher sends Actions to all Stores. Stores respond to whichever Actions are relevant to the state they maintain.
- The View Action sends information in two directions - the request to the server, and notification to the Dispatcher.
- Views retrieve data from the Stores in an event handler. The Views re-render themselves and all of their descendants.



How Do I get started?

Create React App - https://reactjs.org/docs/create-a-new-react-app.html

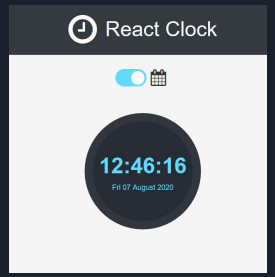
$ npx create-react-app my-app

$ cd my-app

$ npm start

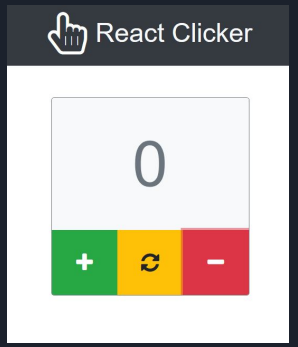
$ npm run build

Examples of React Applications

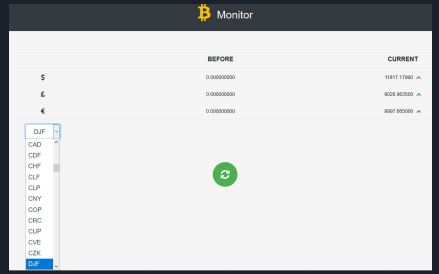


Clock - http://react-clock-basic.drminnaar.me/

Click Counter - http://react-clock-basic.drminnaar.me/



Bitcoin monitor - http://react-bitcoin-monitor.drminnaar.me/



Thank you!

Victoria Chan

Ben Foord

Mark Christison

Abhimanu Saharan

Warren Lourens

Prapti Mane