BADR

Knowledge Academy

Reconsider your Vue

Wednesday, Sept 20

Sessions Roadmap

1- Vue.js in Depth

Vue.js core concepts, Library Internals, Vue Instance.

2- REST APIs & Routing

Components, REST APIs, Services, What's Routing, Nested Routing

3- State Management & Wrap up

Manage components state, Interaction between different components

Capstone Project

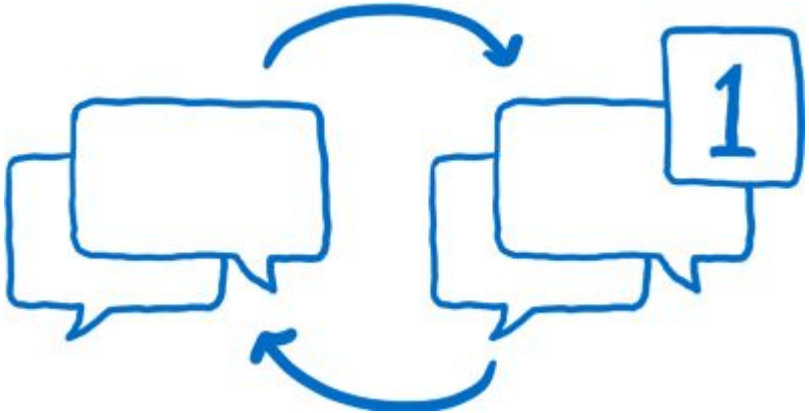
New Tab Dashboard

A New Tab Dashboard with a set of widgets: Weather, Todo List and New Feed Reader.

Session Agenda

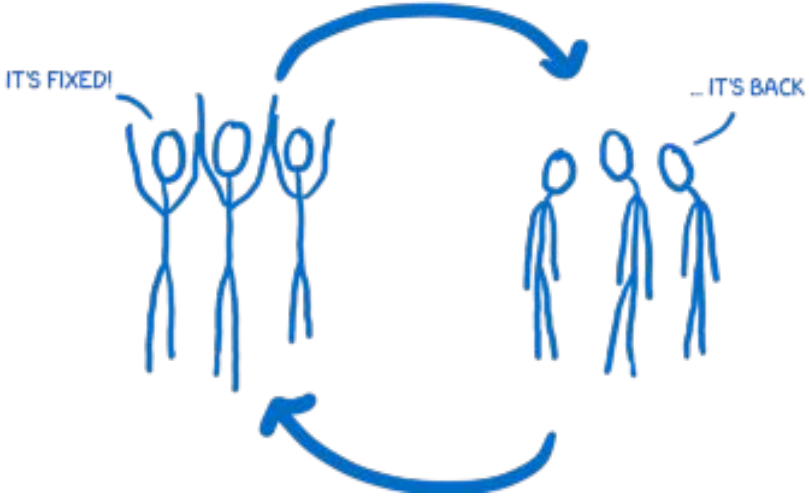
- Background Story
- What’s the problem?
- Solution: Unidirectional Data Flow
- What’s Flux?
- Flux Flow
- What’s Vuex?
- Core Vuex Principles
- Setup & Basic Vuex Store Skeleton
- Short Demo

Background Story



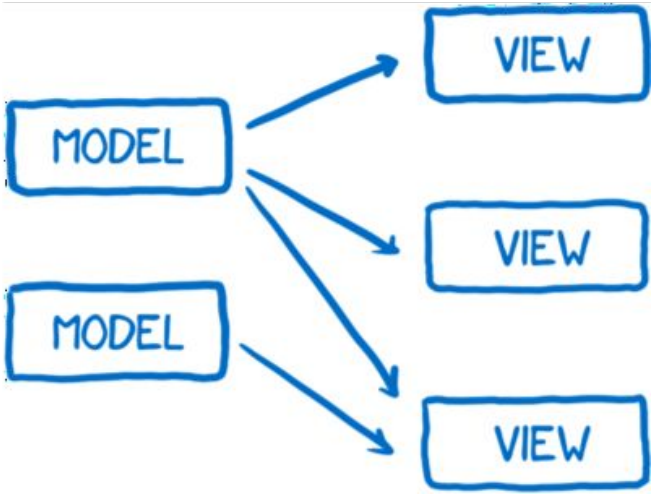
Facebook Messenger Notification Issue

Background Story

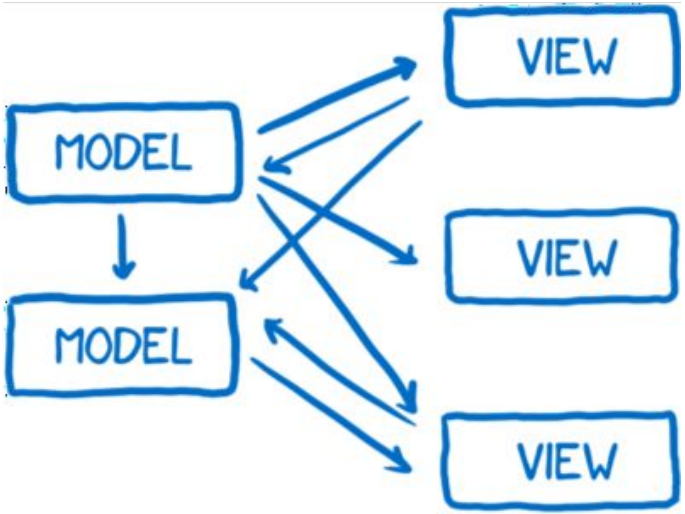


Facebook Messenger Notification Issue

What's the problem?



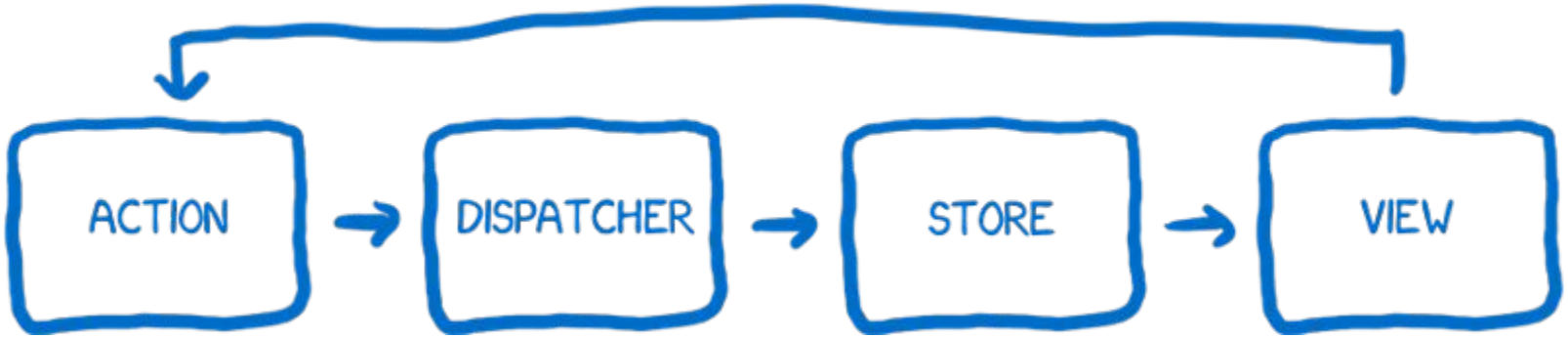
Models pass data to the view layer.



Views update models.
Models update other models.

UI Updates - MVC Architecture Drawbacks

Solution: Unidirectional Data Flow



UI Updates - MVC Architecture Drawbacks

What's Flux?

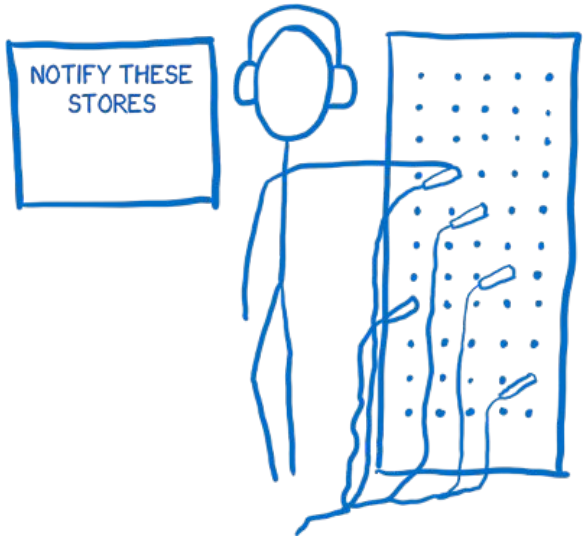
Action Creator

The action creator is like a telegraph operator. It formats your message for you.

- It's in charge of creating actions, which is the path that all changes and interactions should go through.
- You go to the action creator knowing basically what message you want to send, and then the action creator formats that in a way that the rest of the system can understand.
- The action creator creates an action with a type and a payload. The type will be one of the types that you have defined as actions in your system (usually a list of constants).
- Once it has created the action message, the action creator passes that action off to the dispatcher.

What's Flux?

Dispacther



The dispatcher is like a switchboard operator. It knows all the callbacks for the different stores.

- It keeps a list of all of the stores that it needs to send actions to. When an action comes in from the action creator, it will pass the action around to different stores.
- It does this in a synchronous way, which helps with that multi-ball Pong game (Views Mutate Models) effect mentioned earlier.
- The action is sent to all of the registered stores regardless of what the action type is. This means the store doesn't just subscribe to some actions. It hears about all actions and filters out what it cares about and doesn't.

What's Flux?

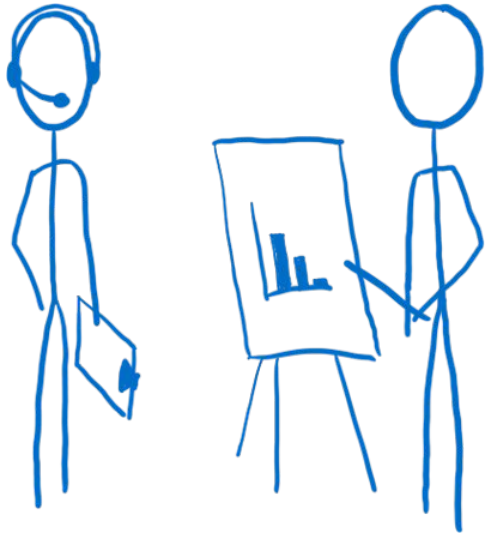
Store

The store is an over-controlling bureaucrat. All changes must go through it.

- The store holds on to all state in the application, and all of the state changing logic lives inside of the stores.
- All state changes must be made by it personally. And you can't directly request that it change the state.
- To request a state change, you must follow proper procedure… you must submit an action via the actior/dispatcher pipeline.
- Once the store has made its changes to the state, it will emit a change event. This will notify the controller view that the state has changed.

What's Flux?

The controller view and the view

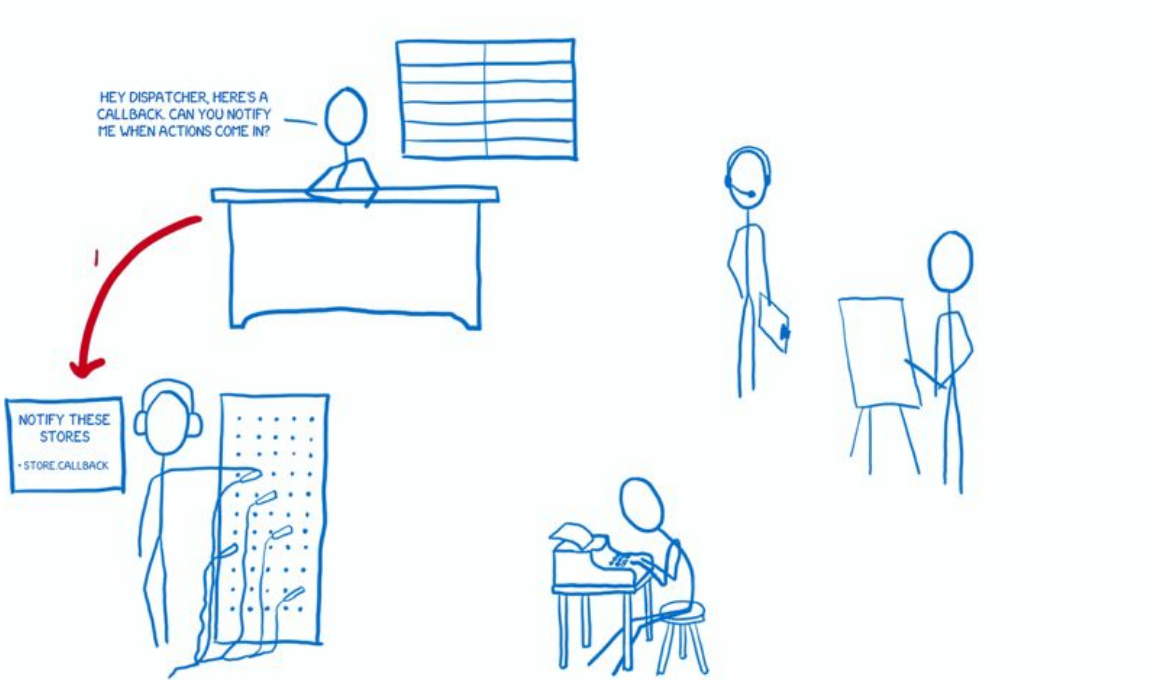


The store is an over-controlling bureaucrat. All changes must go through it.

- The views are in charge of taking the state and rendering it out for the user as well as accepting user input.
- The view is a presenter. It isn't aware of anything in the application, it just knows the data that's handed to it and how to format the data into output that people understand (via HTML).
- The controller view is like a middle manager between the store and the view. The store tells it when the state has changed. It collects the new state and then passes the updated state along to all of the views under it.

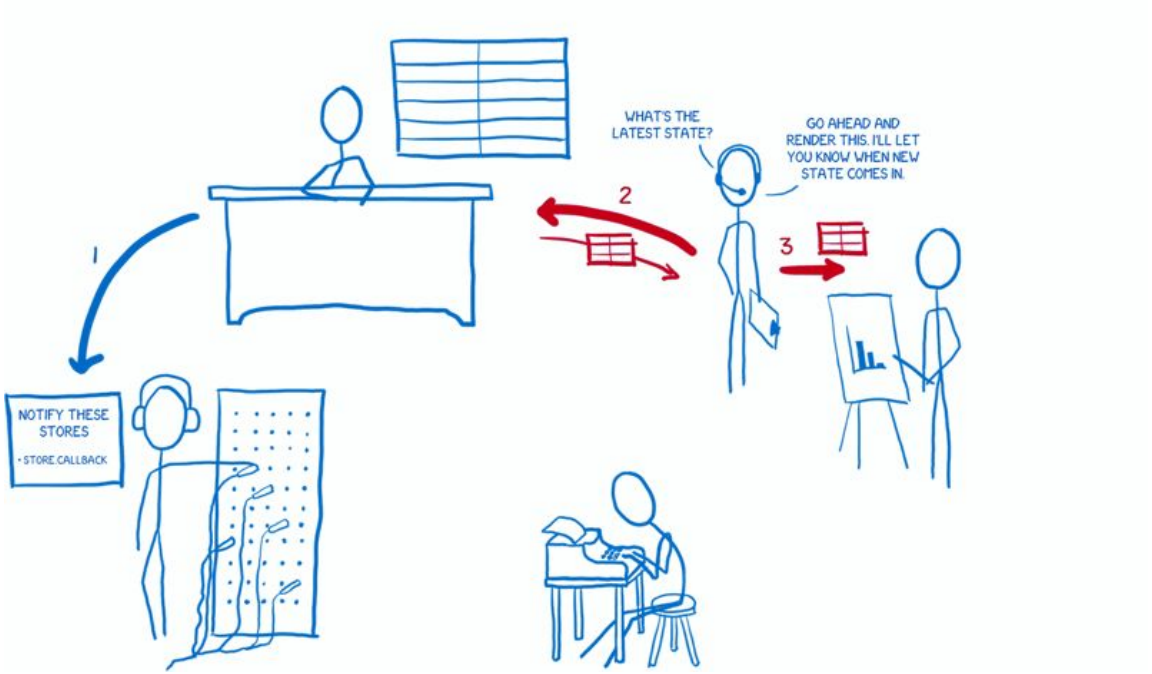
Flux Flow

First there's the setup:
application initialization which only happens once.



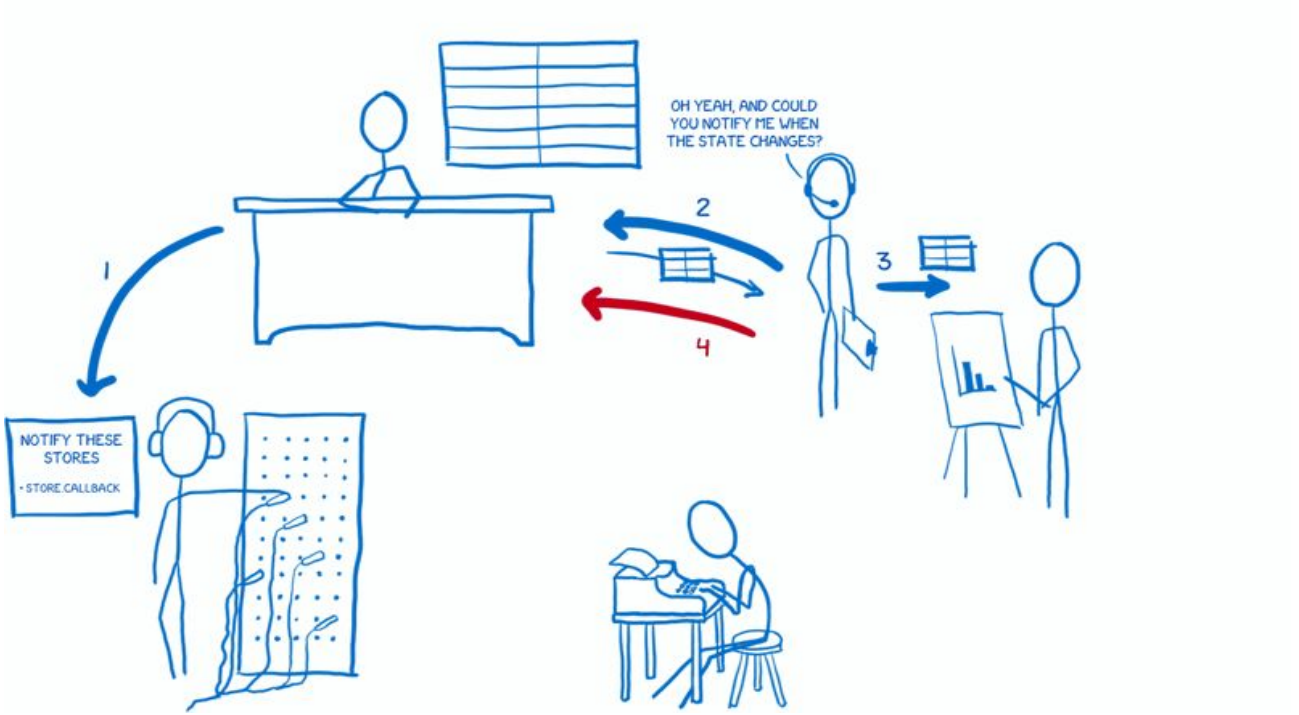
1. Stores let the dispatcher know that they want to be notified whenever an action comes in.

Flux Flow - Initialization



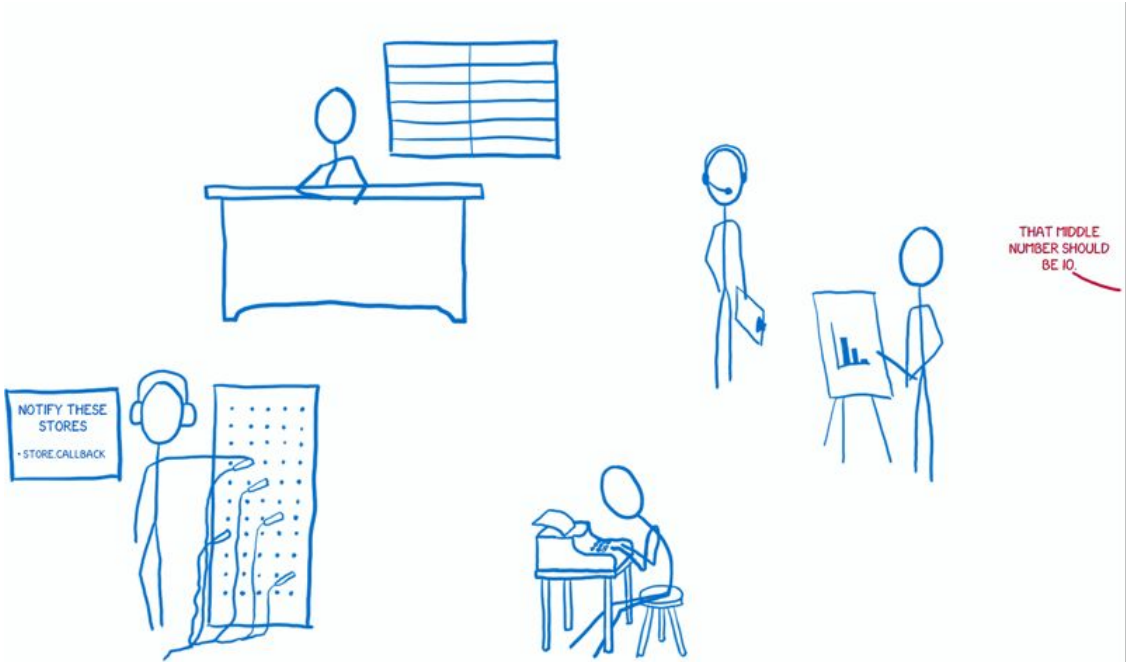
2. Then the controller views ask the stores for the latest state.
3. When the stores give the state to the controller views

Flux Flow - Initialization



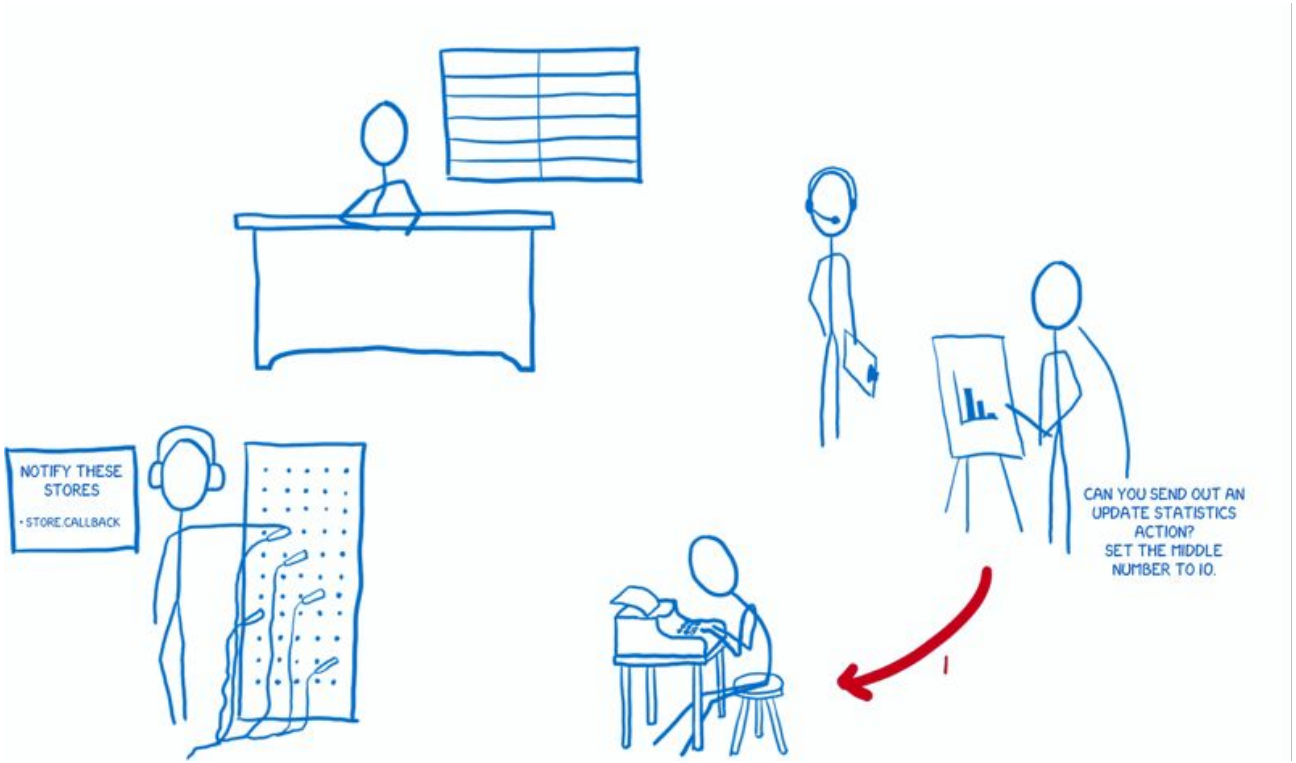
4. The controller views also ask the stores to keep them notified when state changes.

Flux Flow - Normal Flow



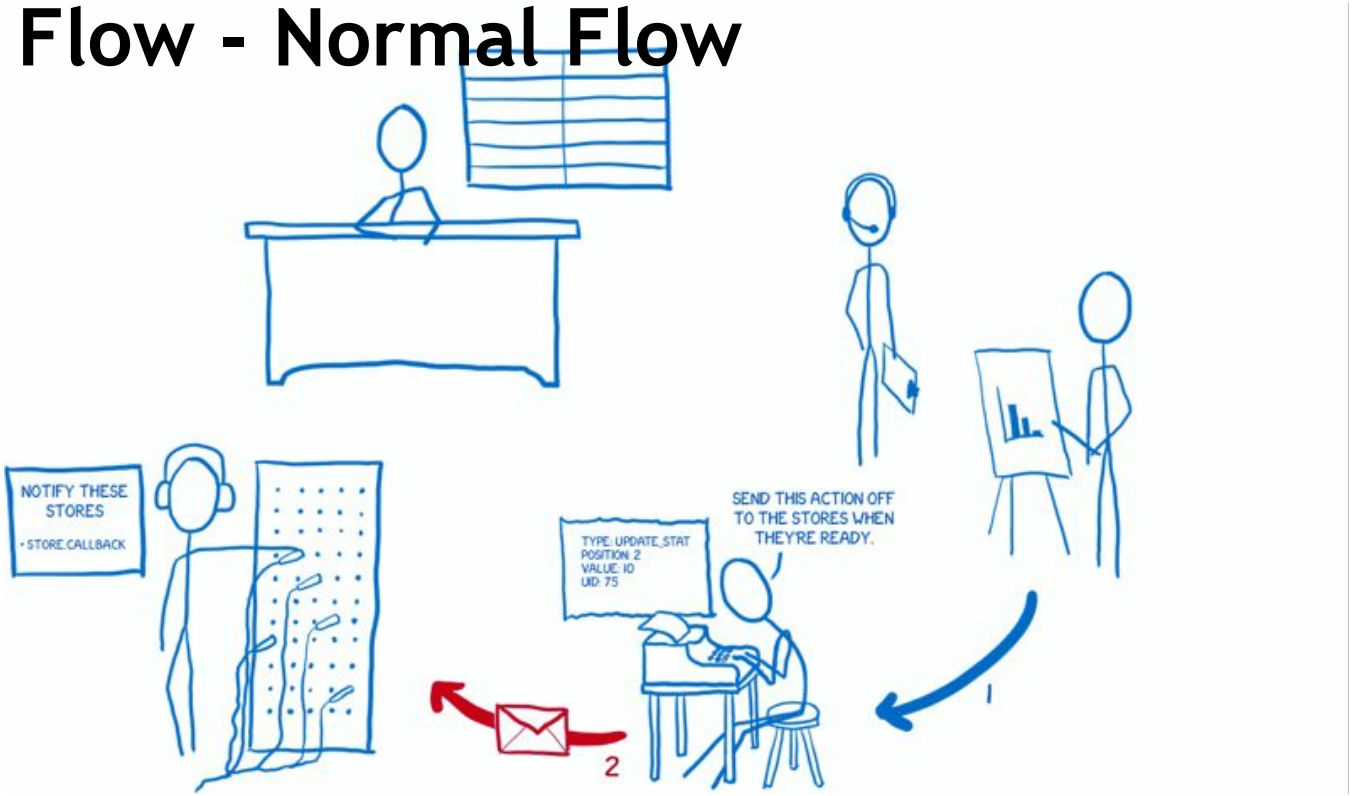
We'll kick off the data flow with a user interaction.

Flux Flow - Normal Flow



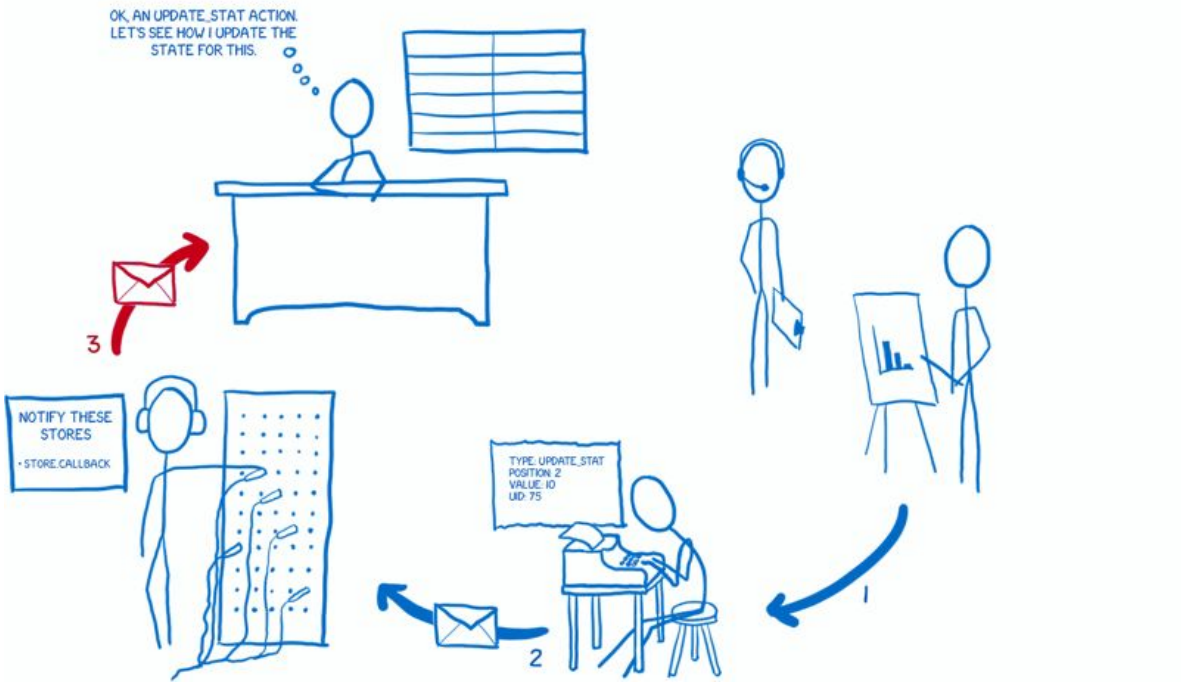
The view tells the action creator to prepare an action

Flux Flow - Normal Flow



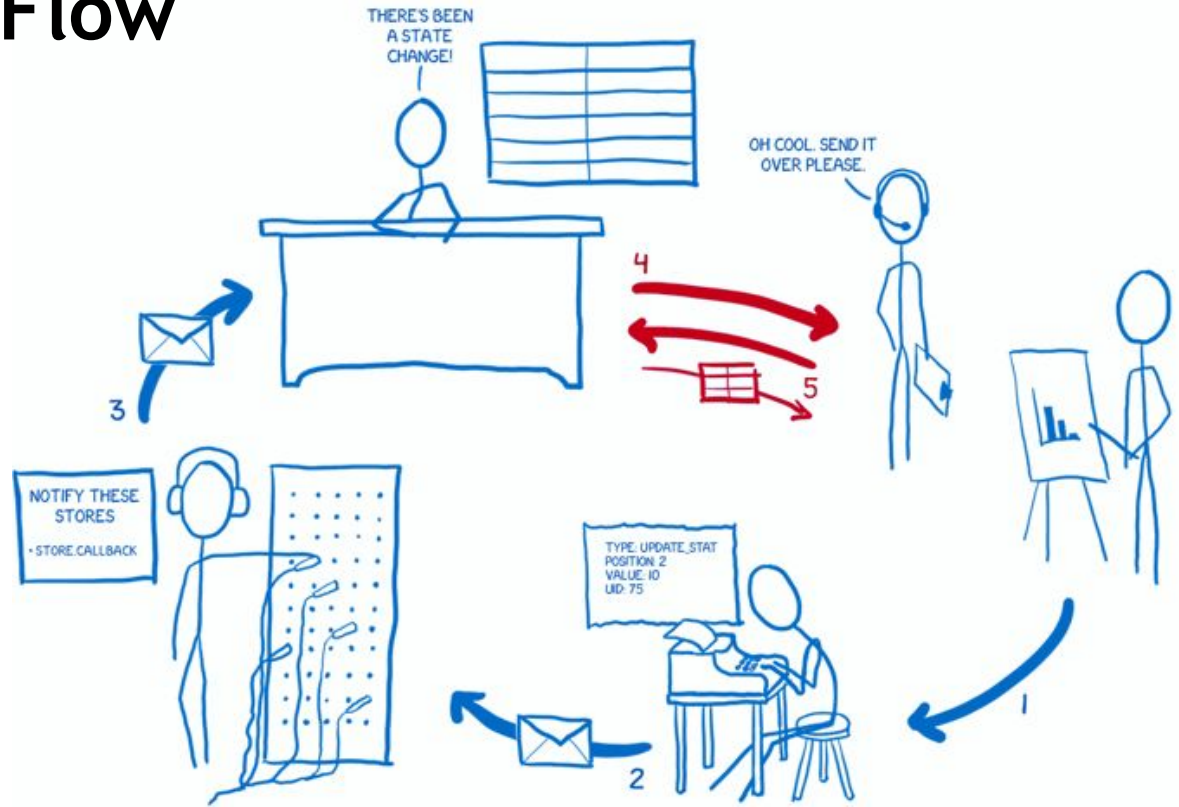
The action creator formats the action and sends it off to the dispatcher.

Flux Flow - Normal Flow



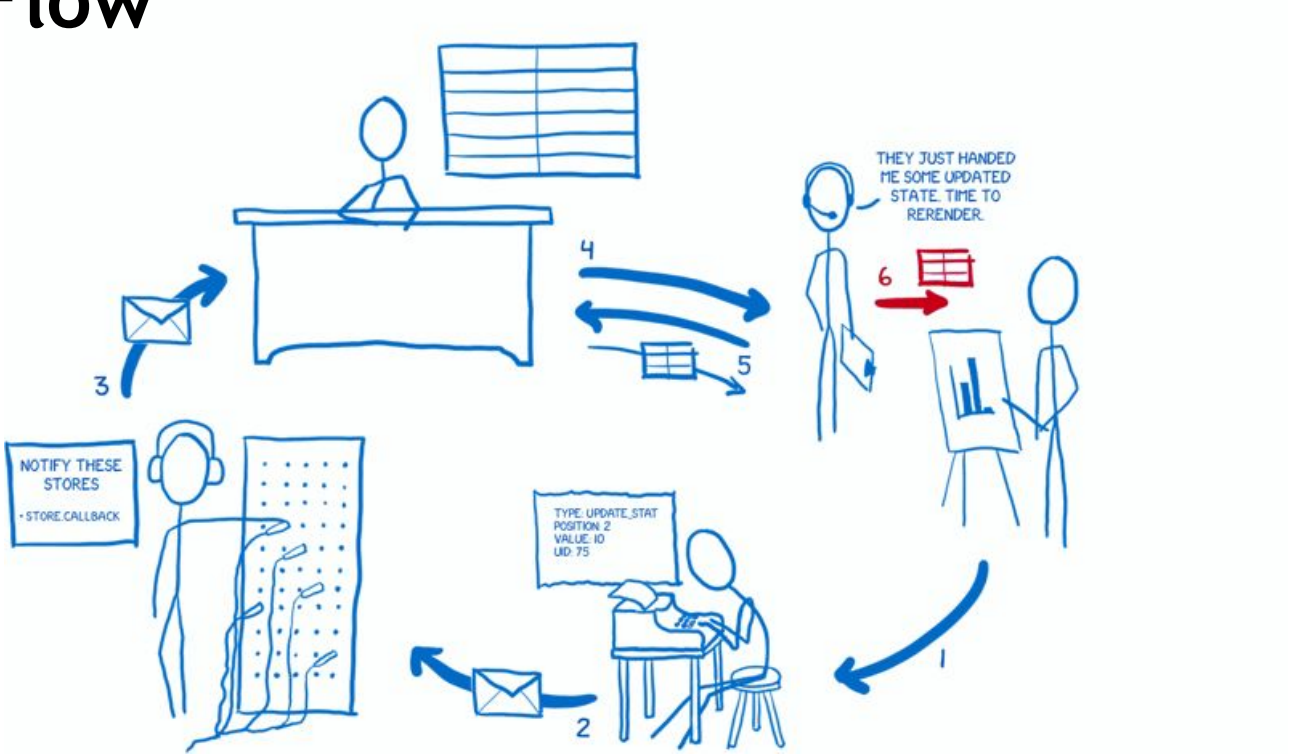
The dispatcher sends the action off to the stores in sequence

Flux Flow



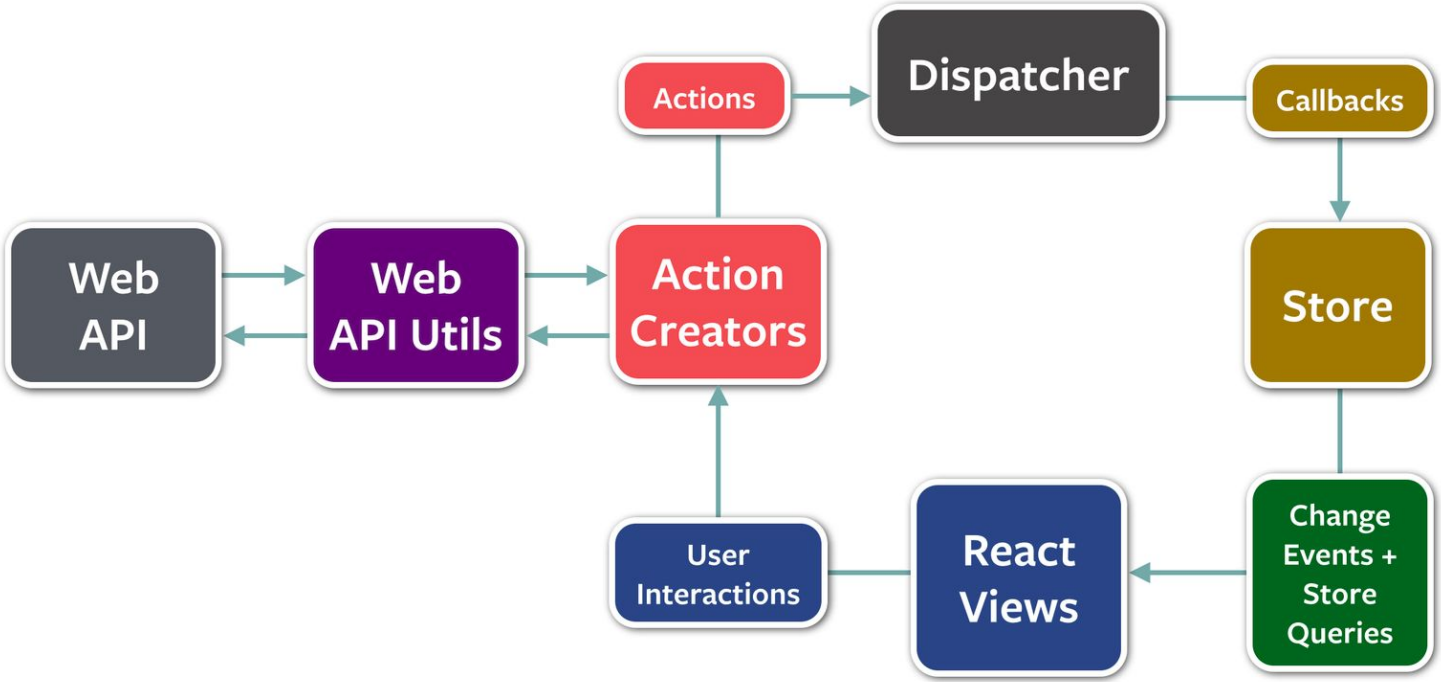
Those view controllers will then ask the store to give them the updated state.

Flux Flow



After the store gives it the state, the view controller will tell its child views to rerender based on the new state.

Flux Summary



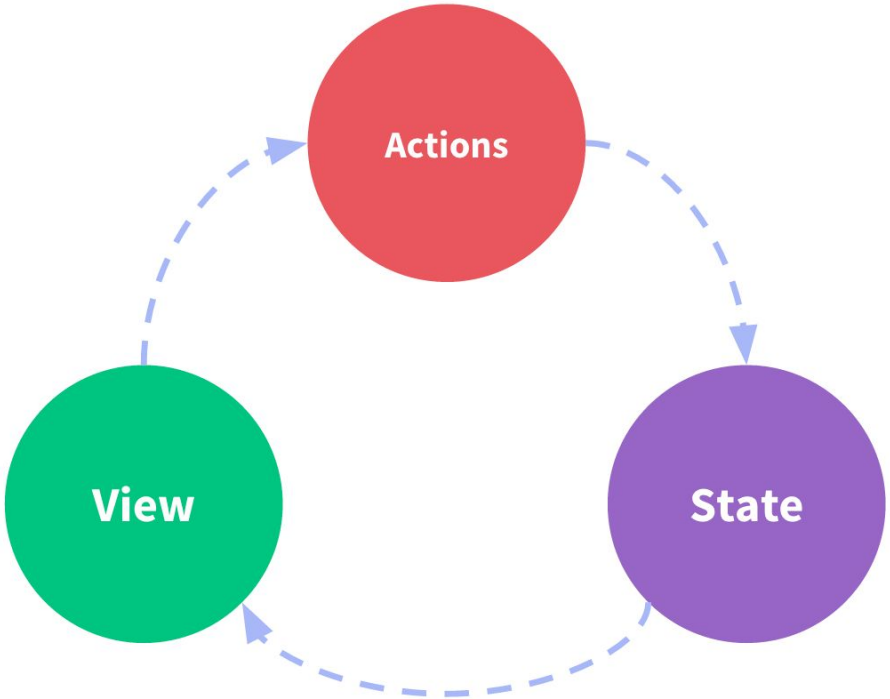
Flux Summary

- Single source of truth
- No data duplication
- All actions are visible
- Updates never cascade
- Many stores can react to same action
- Cache data aggresively
- Record and replay bugs
- Pre-specify any actions from server
- Test components and stores separately

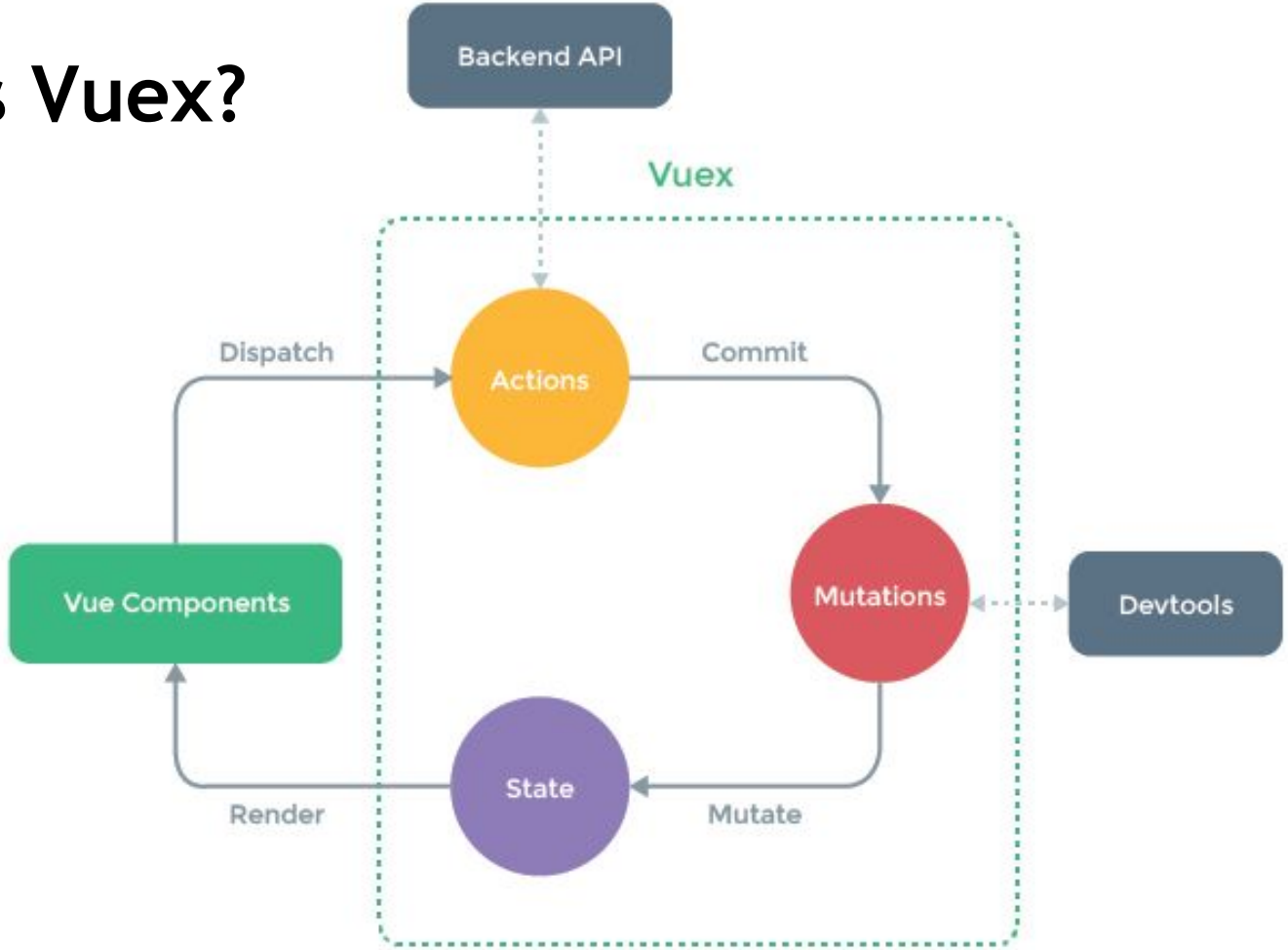
What’s Vuex?

- Vuex is a state management pattern and library for Vue.js applications
- It also integrates with Vue's official devtools extension to provide advanced features such as zero-config time-travel debugging and state snapshot export / import.

What’s Vuex?



What's Vuex?


Backend API
Vuex
Dispatch
Actions
Commit
Vue Components
Mutations
Devtools
State
Render
Mutate

Vuex Principles

Describe application state using a global singleton that enforces rules for changes

You can use an initial state object to describe the entire application

Vuex Principles

Changes in the system happen by dispatching actions and committing mutations

State is mutated in the global singleton, using predictable rules

Vuex Principles

A library implementation that takes advantage of Vue's granular reactivity system

Mutations are done on observable objects that trigger updates in components

Setup & Quick Demo

```
> npm install vuex
> npm run dev
```

That’s All for now

That’s our last session,
Thank you for your time