Flux, Redux & Vuex

Matt Woodruff

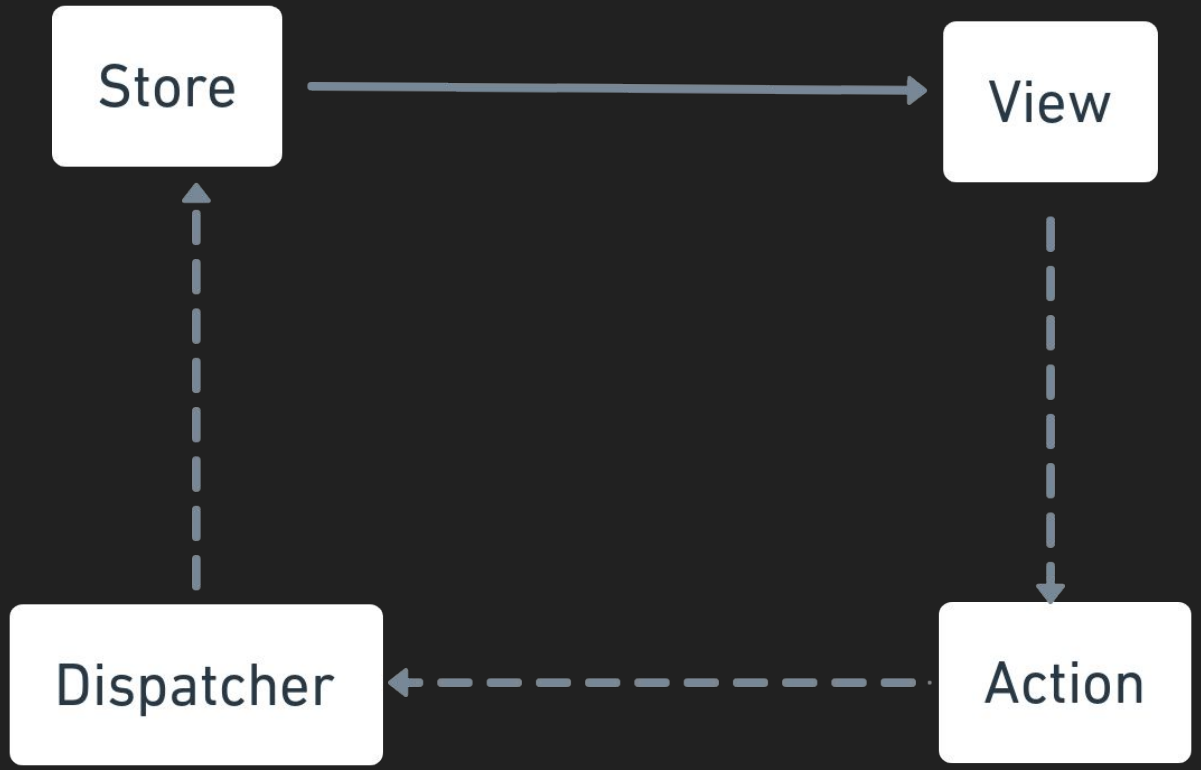
What is the Flux?

Flux is an an application architectural pattern created by Facebook that uses a unidirectional data flow.

Why Flux?

- Simplify scaling issues of MVC
- Keeps your view declarative
- Unidirectional data flow and predictable data consistency make for easier debugging and testing
- Separate data state from component view state

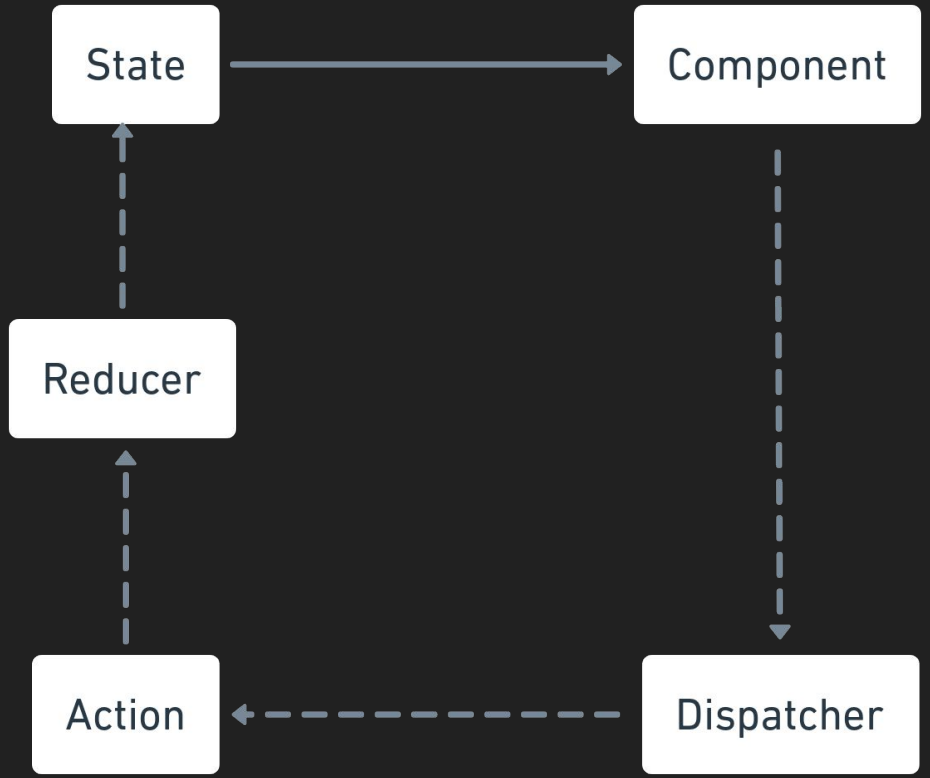
Flux Pattern



Redux

React.js Flux Implementation

Redux Flow



React Component

```tsx
const App: React.FC = () => {
  const { count } = useSelector((state: RootState) => state.count);

  const incrementCount = () => store.dispatch(increment())
  const decrementCount = () => store.dispatch(decrement())
  const resetCount = () => store.dispatch(reset())

  return (
    <div className="App">
      <h1>Counter</h1>
      <p>{count}</p>
      <button onClick={incrementCount}>Increment</button>
      <button onClick={decrementCount}>Decrement</button>
      <button onClick={resetCount}>Reset</button>
    </div>
  );
}

export default App;
```

Redux Actions

```
export const increment: ActionCreator<IncrementCounterAction> = () => ({
    type: COUNTER_ACTIONS.INCREMENT,
})

export const decrement: ActionCreator<DecrementCounterAction> = () => ({
    type: COUNTER_ACTIONS.DECREMENT,
})
export const reset: ActionCreator<ResetCounterAction> = () => ({
    type: COUNTER_ACTIONS.RESET,
})
```

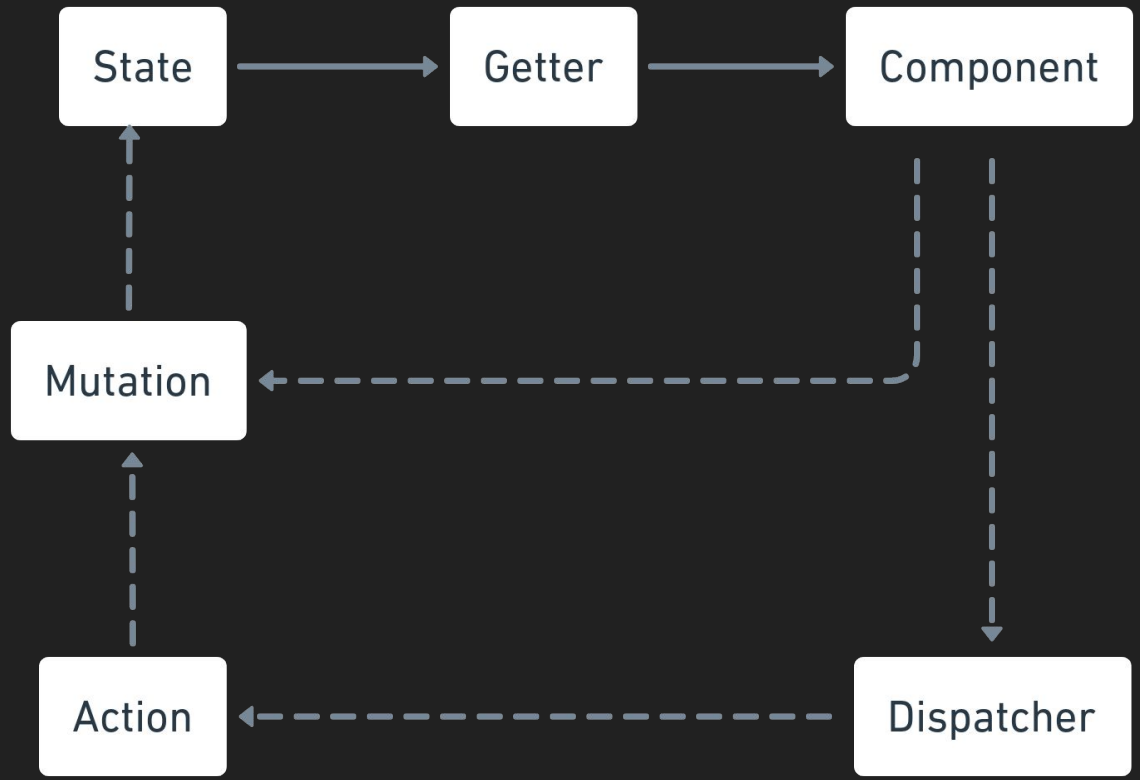
Redux Reducer

```
export const countReducer = (state: CounterState = initialCounterState, { type }: CounterActions): CounterState => {
    switch(type) {
        case COUNTER_ACTIONS.RESET: {
            return { ...initialCounterState };
        }
        case COUNTER_ACTIONS.INCREMENT: {
            const count = state.count + 1;
            return { ...state, count };
        }
        case COUNTER_ACTIONS.DECREMENT: {
            const count = state.count - 1;
            return { ...state, count };
        }
        default: {
            return { ...state };
        }
    }
}
```

Vuex

Vue.js Flux Implementation

Vuex Flow



Vue Component

```vue
<template>
  <div class="home">
    <h1>Counter</h1>
    <p>{{ count }}</p>
    <button v-on="{ click: () => setCount(count + 1) }">Increment</button>
    <button v-on="{ click: () => setCount(count - 1) }">Decrement</button>
    <button v-on:click="loadCount">Load Random</button>
    <button v-on:click="reset">Reset</button>
  </div>
</template>

<script lang="ts">
import Vue from "vue";
import Component from "vue-class-component";
import { Getter, Action, Mutation } from "vuex-class";
import { CountActions } from "@/store/count/actions";
import { CountGetters } from "@/store/count/getters";
import { CountMuations } from "@/store/count/mutations";

@Component
export default class Counter extends Vue {
  @Getter(CountGetters.count) count!: number;
  @Mutation(CountMuations.setCount) setCount!: (value: number) => void;
  @Mutation(CountMuations.reset) reset!: () => void;
  @Action(CountActions.loadCount) loadCount!: () => void;
}
</script>
```

Vuex Action

```
export const CountActions = {
    loadCount: '[count] loading count from api request'
}

export const actions: ActionTree<CountState, RootState> = {
    [CountActions.loadCount]: ({ commit }) => {
        const mockApiResponse = Math.round(Math.random() * 100);
        commit(CountMutations.setCount, mockApiResponse);
    }
}
```

Vuex Mutations

```
export const mutations: MutationTree<CountState> = {
    [CountMutations.setCount]: (state: CountState, payload: number) => {
        state.count = payload;
    },
    [CountMutations.reset]: (state: CountState) => {
        Object.assign(state, countStateDefaults);
    },
};
```

Vuex Getters

```ts
export const CountGetters = {
    count: '[count] returns count state as computed property',
};

export const getters: GetterTree<CountState, RootState> = {
    [CountGetters.count]: ({ count }: CountState) => count,
}
```

Redux <> Vuex

Redux

- Immutable
- Uses reducers to return new slice of state
- Must call action to change state
- Need to import thunk for async

Vuex

- Mutable
- Can call mutation directly without action
- Actions can be async

Resources

Flux: https://facebook.github.io/flux/

Redux: https://redux.js.org/

Vuex: https://vuex.vuejs.org/

Github Examples: https://github.com/woody34/flux