

# The Dispatcher

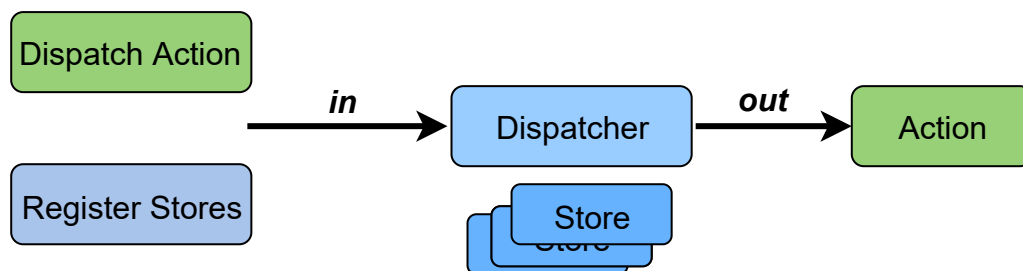
In this lesson, we will look at the Dispatcher as the main component in the Flux architecture, and some code related to it in detail.

## WE'LL COVER THE FOLLOWING ^

- The Dispatcher

## The Dispatcher #

In most cases, we need a single dispatcher because it acts as a sort of glue for the rest of the parts and it, therefore, makes sense to have only one. The dispatcher needs to know two things - actions and stores. The actions are simply forwarded to the stores so we don't necessarily have to keep them. The stores, however, should be tracked inside the dispatcher so we can loop through them:



Following is what we can start with:


```
var Dispatcher = function () {
  return {
    _stores: [],
    register: function (store) {
      this._stores.push({ store: store });
    },
    // checks for updates in each store
    dispatch: function (action) {
      if (this._stores.length > 0) {
        this._stores.forEach(function (entry) {
          entry.store.update(action);
        });
      }
    }
  };
};
```



```
    }  
  }  
};
```

The first thing that we notice is that we *expect* to see an `update` method in the passed stores. It will be nice to throw an error if such method is not there:

```
// throwing an error if no expected update method exists  
register: function (store) {  
  if (!store || !store.update) {  
    throw new Error('You should provide a store that has an `update` method.');
```



---

Let's now move on to Views, Stores and their bounding