**Redux Observable to the rescue**

Ever since [Redux](https://redux.js.org/) was released by [Dan](https://twitter.com/dan_abramov) and [Andrew](https://twitter.com/acdlite?lang=en) (huge fan ^\_^), it certainly changed the way I used to write code. It opened a whole new door to the way we write our code using pure functions, immutable patterns, time travelling (wicked!!) and so much more. I am assuming you have prior knowledge of redux.

I first used redux for building our internal CRM web application for my company. A CRM tool is no small application of course, as time passed by, I soon realised my code was not so pure as I thought it would be. Coming from a MVC background, I tried creating something very similar using Redux. **My reducers were pure functions which accepted initial state and returned updated state**, **which acted like my data model(M). Actions and routes were like my controllers(C) and of course react components were my views(V).**

But there was a **serious** **flaw** in this architecture. My actions were not clean anymore. All the complicated logic was built in my actions. Although it was easier to track changes through action types, but I was unable to prevent my actions from getting polluted. I was obsessed with keeping everything simple and clean.

Let us dive into the code to understand the problem I faced

|  |
| --- |
| export default function fetchDeliveryAddress(state = initialState, action){ |
|  | switch(action.type){ |
|  | case FETCH\_DELIVERY\_ADDRESS\_LIST\_SUCCESS: |
|  | return { |
|  | ...state, |
|  | deliveryAddressList: action.deliveryAddressList, |
|  | deliveryDetailsState: action.deliveryDetailsState |
|  | } |
|  |  |
|  | default: |
|  | return state; |
|  | } |
|  | } |

Below is an example to demonstrate how complicated my action became

|  |
| --- |
| export function getDeliveryAddress(some\_state) { |
|  | return function(dispatch) { |
|  | axios(some\_config\_object) |
|  | .then(function(response) { |
|  | let deliveryDetailsState; |
|  |  |
|  | const deliveryAddressList = response.data.payload; |
|  |  |
|  | let updatedDeliveryAddressList = deliveryAddressList.map((item, index) => { |
|  | if(index !== 0) { |
|  | // This isn't the item we care about - keep it as-is |
|  | return item; |
|  | } |
|  |  |
|  | return { |
|  | ...item, |
|  | listIndex: 0, |
|  | } |
|  | }); |
|  |  |
|  | dispatch({ |
|  | type: types.FETCH\_DELIVERY\_ADDRESS\_LIST\_SUCCESS, |
|  | some\_state: some\_state, |
|  | deliveryAddressList: updatedDeliveryAddressList, |
|  | }); |
|  | }) |
|  | .catch(function (error) { |
|  | logException(error); |
|  |  |
|  | return dispatch({ |
|  | type: types.FETCH\_DELIVERY\_ADDRESS\_LIST\_FAILURE, |
|  | some\_state: some\_state, |
|  | error |
|  | }); |
|  | }); |
|  | } |
|  | } |

It pained me to write this code, my function was not so clean and pure anymore, but I hate giving up, I had to fix it and find a way to keep my action clean. So after some searching I realised that my **action was handling side effects. I needed a way to watch over the side effects without affecting my actions directly.** Finally I bumped into **Observables**, a perfect match for my problem, that would not only fix complicated api handling but also handle other side effects. Let us understand what exactly is observable.

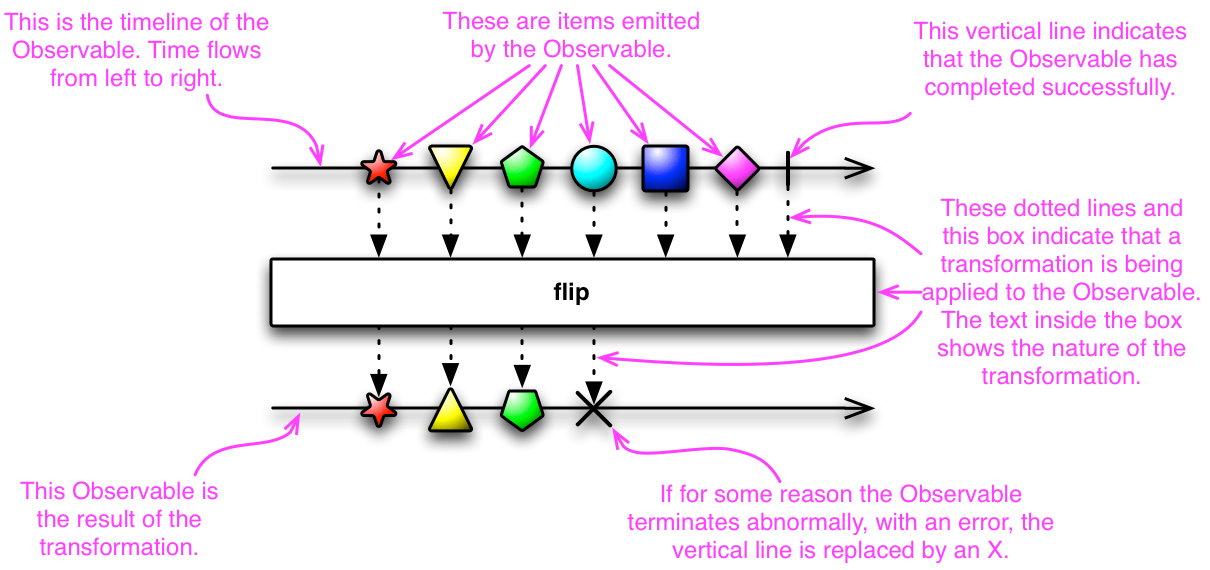
**Observables**

It is a concept of [Reactive programming](https://gist.github.com/staltz/868e7e9bc2a7b8c1f754). To answer what it is I will quote it directly from the author himself

*Reactive programming is programming with asynchronous data streams.*

Reactive programming is beyond the scope of this article, so I will explain it in very simple terms.

The below image from the official doc explains what [Observables](http://reactivex.io/documentation/observable.html) are



I personally like visual representation, makes it much easier to understand.

Now combine Observables with Redux, we have a new Super saiyan 3 Redux Observable :)



Redux Observable :)

[**Redux Observable**](https://redux-observable.js.org/)

It opened the door to a new style of programming called the reactive way! It is the sexiest thing I have come across. I won’t get into the details of it for now(may be in a different tutorial), I just wanted to discuss architecture. So let us dive into code and look at how beautiful a code can be through this.

Action is simple again yay!

Epic looks really cool @\_@

The reducer stays as it is earlier. This helps me in keeping actions and reducers clean.

As reactive programming famously quotes,

*Everything is a stream!*

Imagine all the actions happening through the course of application is a stream, as soon as a event is detected in the stream, observables masqueraded as epics in our framework watch over those actions and emit events. For us the event that has happened is of action type FETCH\_DELIVERY\_ADDRESS\_LIST\_SUCCESS.

Our **getDeliveryAddressEpic** with help of **Observable** looks for the action in the event stream of type(**[ofType](https://redux-observable.js.org/docs/basics/Epics.html" \t "_blank) a operator in redux-observable**) FETCH\_DELIVERY\_ADDRESS\_LIST\_SUCCESS. Argument **action$** is acting as values provided to the epic from action **getDeliveryAddress** for handling side effects and returning updated value accordingly.

This might seem overwhelming, but if you have to observe the code from a top level view, look at the simplicity of the operation being handled.

Actions and reducers are clean, side effects and the dirty work is handled by our epics, which even makes the dirty work really clean! That is beauty of [rxjs](https://www.gitbook.com/book/btroncone/learn-rxjs" \t "_blank). [Rxjs](https://www.gitbook.com/book/btroncone/learn-rxjs" \t "_blank) is different topic all together. My intention is to discuss the approach we can take to write cleaner and manageable code.

**Conclusion**

Of course this article might seem opinionated, I am sure there are other ways to handle side effects like [redux saga](https://github.com/redux-saga/redux-saga). I hope this article gave you insight on how we can combine the concept of Observable into Redux and make your framework go super saiyan 3 :D

Please clap to encourage this article and show some ❤!

Follow me on [twitter](https://twitter.com/daslusan) to get more updates regarding new articles and to stay updated in latest frontend developments.

**Some helpful links to get started**

1. [Introduction to reactive programming](https://gist.github.com/staltz/868e7e9bc2a7b8c1f754)
2. [Redux official doc](https://redux.js.org/)
3. [Redux Observable Official doc](https://redux-observable.js.org/)
4. [Learning Rxjs](https://www.gitbook.com/book/btroncone/learn-rxjs/details)

See you again soon and happy coding!! ❤

* [JavaScript](https://codeburst.io/tagged/javascript?source=post)
* [Redux](https://codeburst.io/tagged/redux?source=post)
* [React](https://codeburst.io/tagged/react?source=post)
* [Reactjs](https://codeburst.io/tagged/reactjs?source=post)
* [Redux Observable](https://codeburst.io/tagged/redux-observable?source=post)

One clap, two clap, three clap, forty?

By clapping more or less, you can signal to us which stories really stand out.

392

 Follow

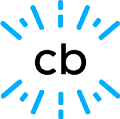
[[](https://codeburst.io/@daslusan?source=footer_card)](https://codeburst.io/@daslusan?source=footer_card" \o "Go to the profile of Lusan Das)

[**Lusan Das**](https://codeburst.io/@daslusan)

Medium member since Dec 2017

Senior Software Engineer || FrontEnd Engineer || React || EDM || Scuba Lover || Travel || Food

 Follow

[](https://codeburst.io/?source=footer_card)

[**codeburst**](https://codeburst.io/?source=footer_card)

Bursts of code to power through your day. Web Development articles, tutorials, and news.

[More from codeburst](https://codeburst.io/how-to-not-react-common-anti-patterns-and-gotchas-in-react-40141fe0dcd?source=placement_card_footer_grid---------0-41)

[How to NOT React: Common Anti-Patterns and Gotchas in React](https://codeburst.io/how-to-not-react-common-anti-patterns-and-gotchas-in-react-40141fe0dcd?source=placement_card_footer_grid---------0-41)

[Go to the profile of Arfat Salman](https://codeburst.io/@arfatsalman)

[Arfat Salman](https://codeburst.io/@arfatsalman?source=placement_card_footer_grid---------0-41)

2.8K

[More from codeburst](https://codeburst.io/25-html-css-tutorials-6a864f387185?source=placement_card_footer_grid---------1-41)

[25 HTML & CSS Tutorials](https://codeburst.io/25-html-css-tutorials-6a864f387185?source=placement_card_footer_grid---------1-41)

[[Go to the profile of Brandon Morelli](https://codeburst.io/@bmorelli25)](https://codeburst.io/@bmorelli25)

[Brandon Morelli](https://codeburst.io/@bmorelli25?source=placement_card_footer_grid---------1-41)

1.2K

[More from codeburst](https://codeburst.io/25-node-js-tutorials-1db3b1da0260?source=placement_card_footer_grid---------2-41)

[25 Node JS Tutorials](https://codeburst.io/25-node-js-tutorials-1db3b1da0260?source=placement_card_footer_grid---------2-41)

[[Go to the profile of Brandon Morelli](https://codeburst.io/@bmorelli25)](https://codeburst.io/@bmorelli25)

[Brandon Morelli](https://codeburst.io/@bmorelli25?source=placement_card_footer_grid---------2-41)

5K

**Responses**

julio cornelio

Be the first to write a response…

julio cornelio

* 392