调试React程序在Chrome上主要由2个插件，分别为React-Devtools以及Redux-Devtools. React-Devtools用于检查插件分布以及state, 而Redux-Devtools用于查看Redux的dataStore以及action发生的情况以及可以控制整个App在某个action的状态，以及action发生前发生后dataStore的状态变化。

**React-Devtools:**

在Chrome extension插件商店中下载并安装React-Devtools.当浏览由React驱动的页面时候，React-Devtools插件将读取组件信息。

**Redux-Devtools:**

第一步：首先在Chrome extension插件商店中下载并安装Redux-Devtools.

第二步：在Redux项目中安装redux-devtools-extension包，并且在创建dataStore时候加入

import { createStore } from 'redux';

import { devToolsEnhancer } from 'redux-devtools-extension'; //devToolsEnhancer是一个enhancer function,而非中间件

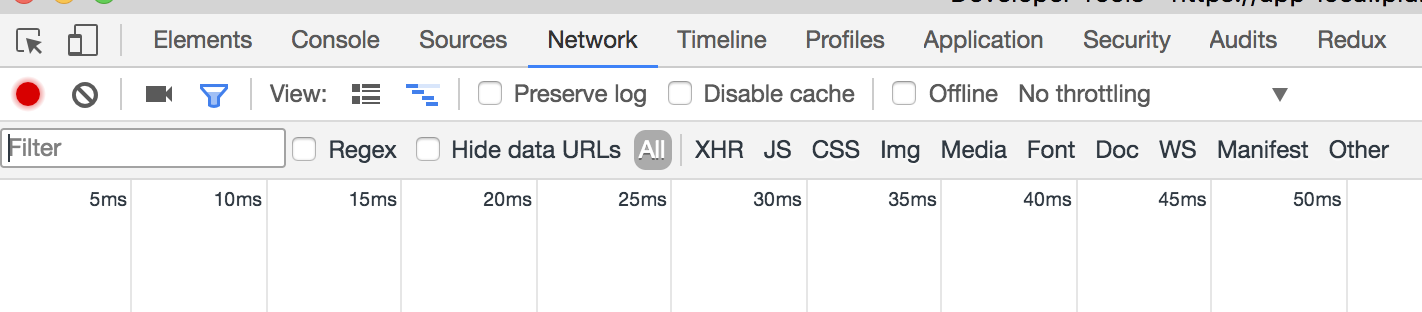
const store = createStore(reducer, /\* preloadedState, \*/ devToolsEnhancer(

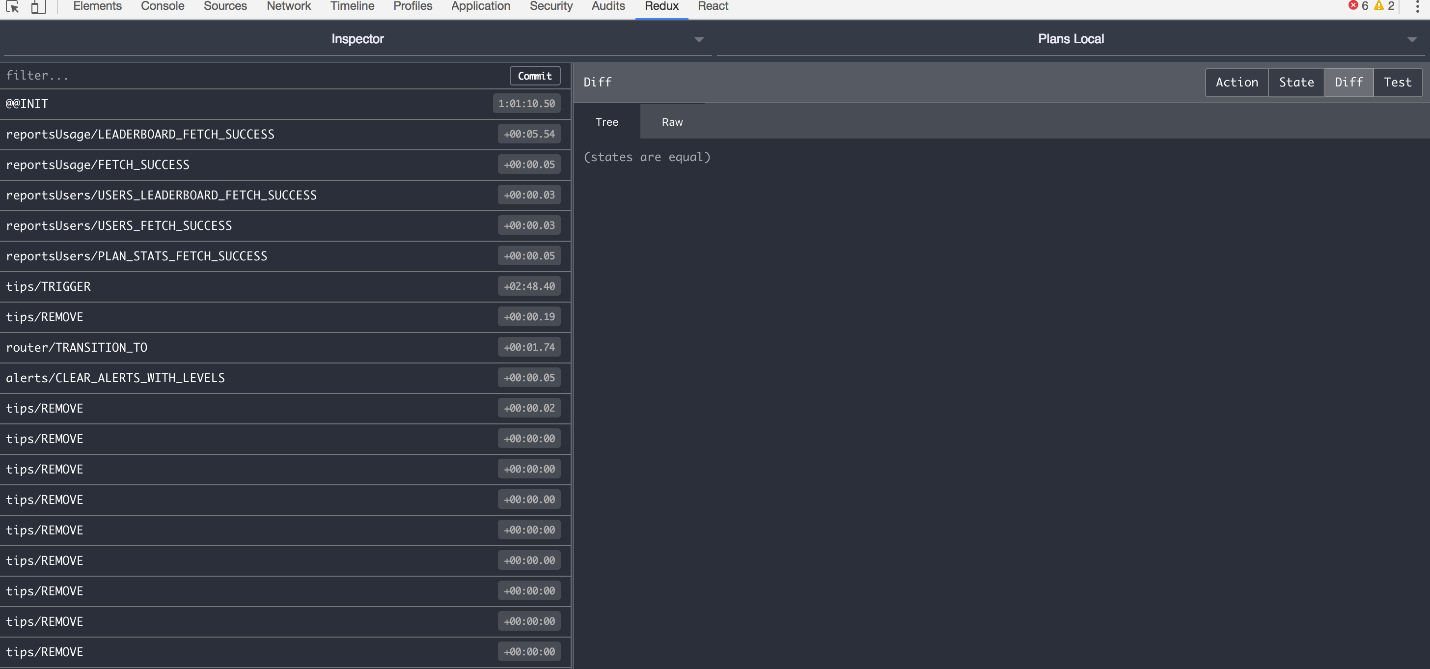
// Specify name here, actionsBlacklist, actionsCreators and other options if needed

));

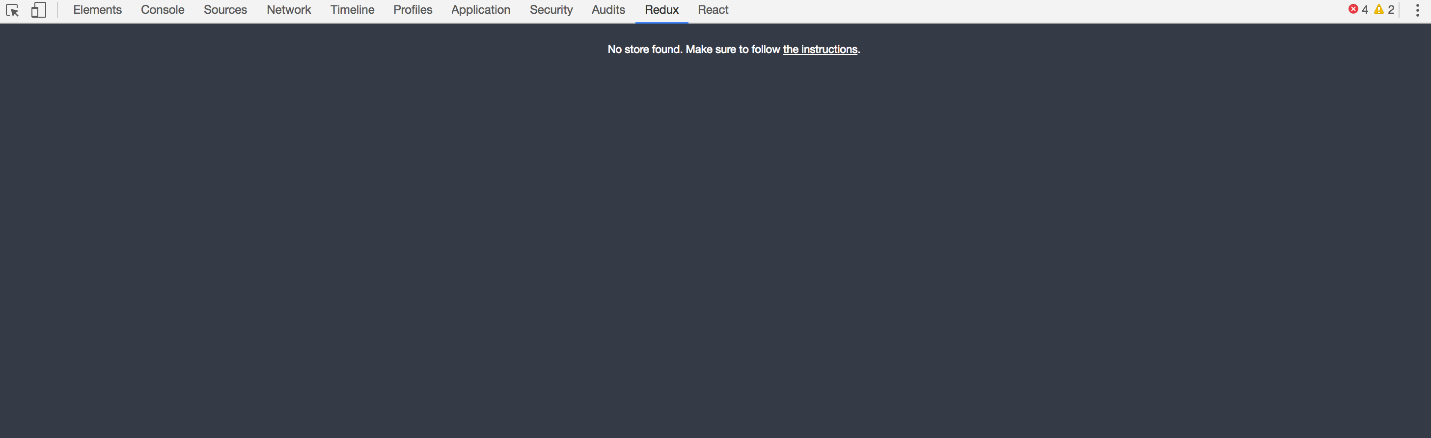
**Redux-DevTools Console Break-down:**

Now that we have everything set up, let’s get into the fun stuff. If you’ve installed the browser extension correctly, you will see a Redux option in the console.

When you click on this new option, you will see one of two things. If you installed everything correctly in your store initialization, it will show you something like this:



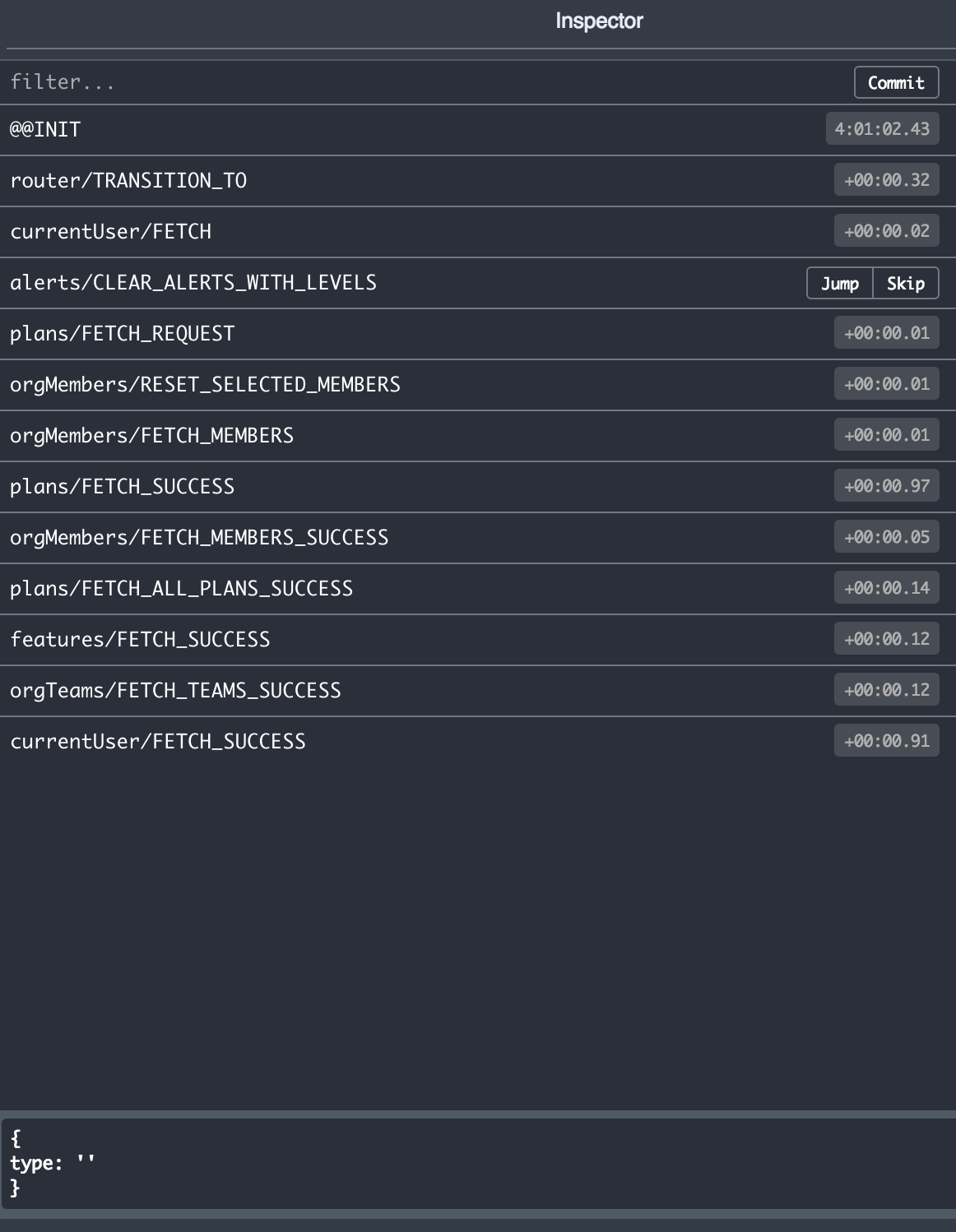
If there is an error in your set-up, including exceptions thrown from your code, you will see this:



Let’s break down whats happening in each section of the dev tools.

Left-Side Console Tools

The left hand column of the dash is showing in real time the actions that are firing on the current page (as defined by the action creator types).



This comes complete with a filter option at the top and the time each action fired after the previous one.

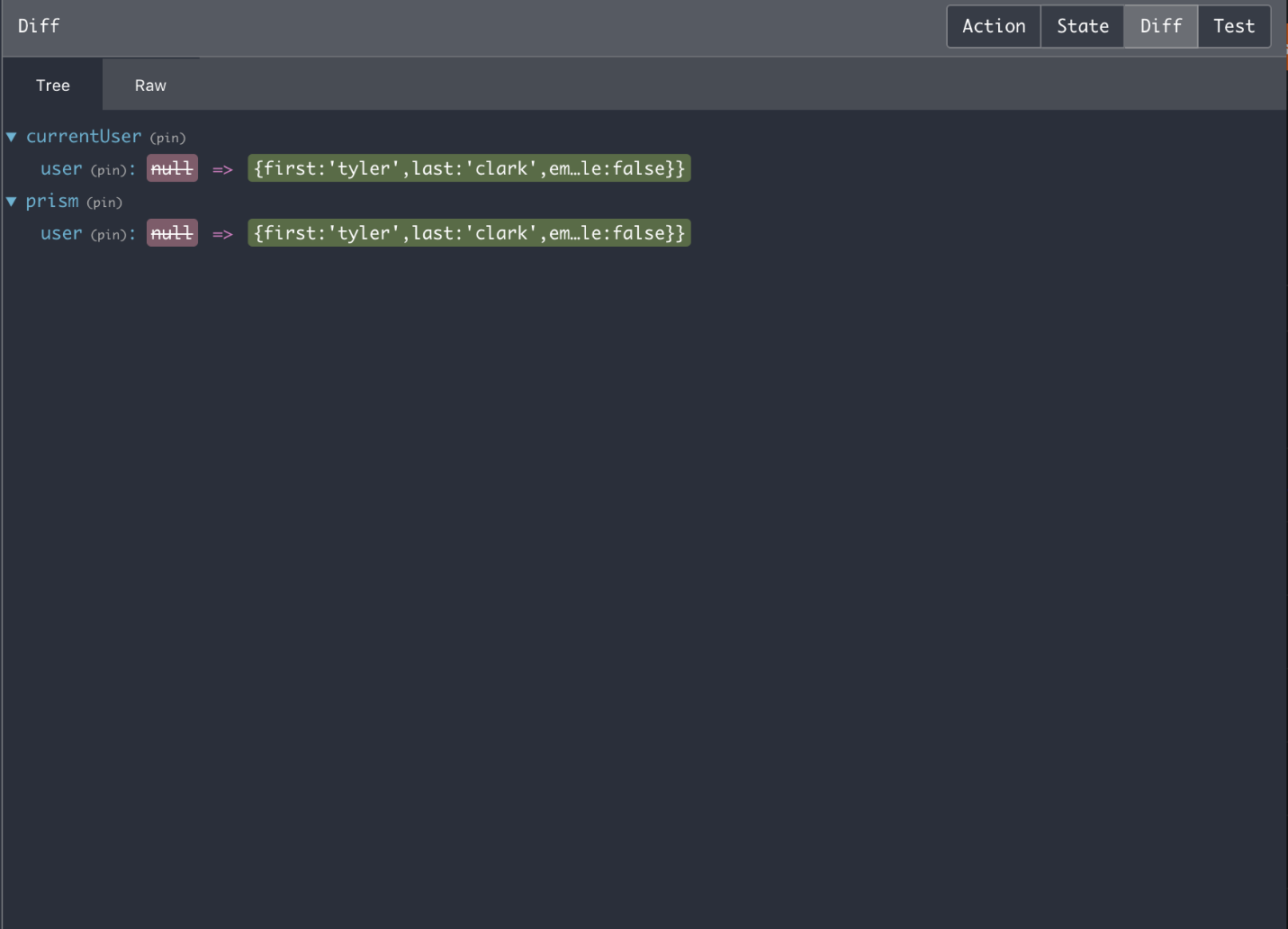
When you click on an individual action it displays two options (Jump and Skip). This is the first introduction to time traveling and changing the application view. Jump will take your application to the state of the app at the time this action fired. The skip will cross out the action and show you your app without that action.

Right-Side Console Tools

The four tabs at the top right are in my opinion the most helpful of the tools.

Diff

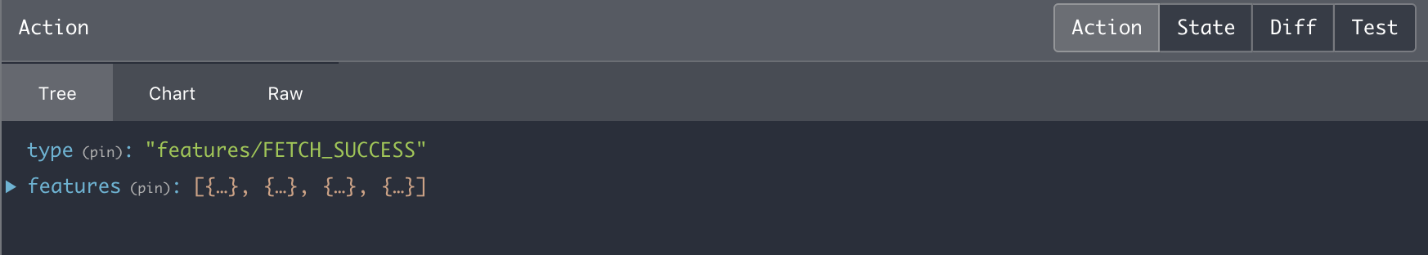
The right side of the console has multiple options to see the state of the application and the relation of each action to the state.



When an action is selected in the left hand side of the console, the Diff option will show only what that individual action changed in the state tree.

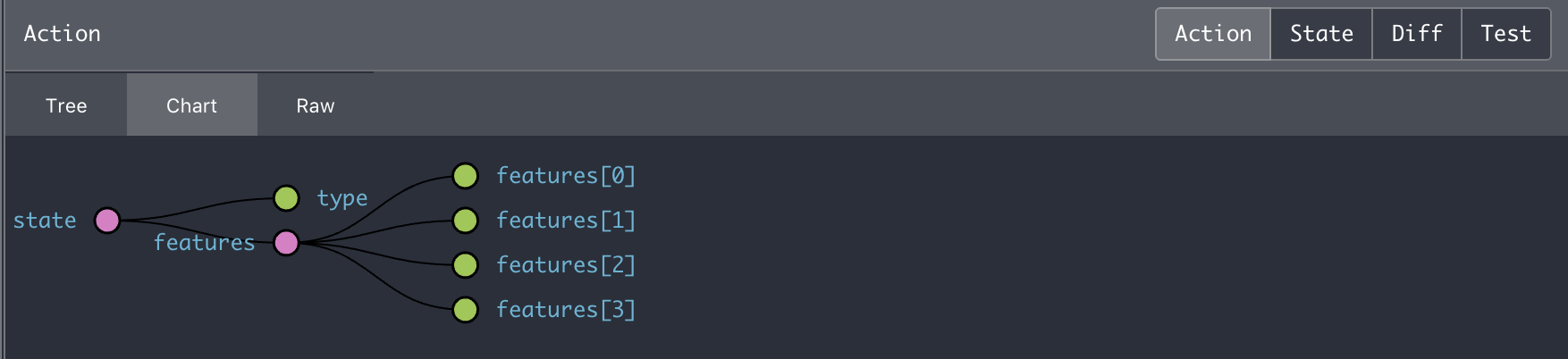
Action

Clicking on the Action tab will show the individual action type and any data it is carrying along to the reducers.



In the example above, this action has a type features/FETCH\_SUCCESS and also carries data, the requested features to the reducer. You might have noticed that there are three sub tabs Tree , Chart , Raw within this master Action tab. These display the same data just in different formats.

* The Tree view is the default view and summarizes the data into single lines.
* The Chart tab shows the up-to-date store tree. Including all of the combined reducers and their corresponding data branches.



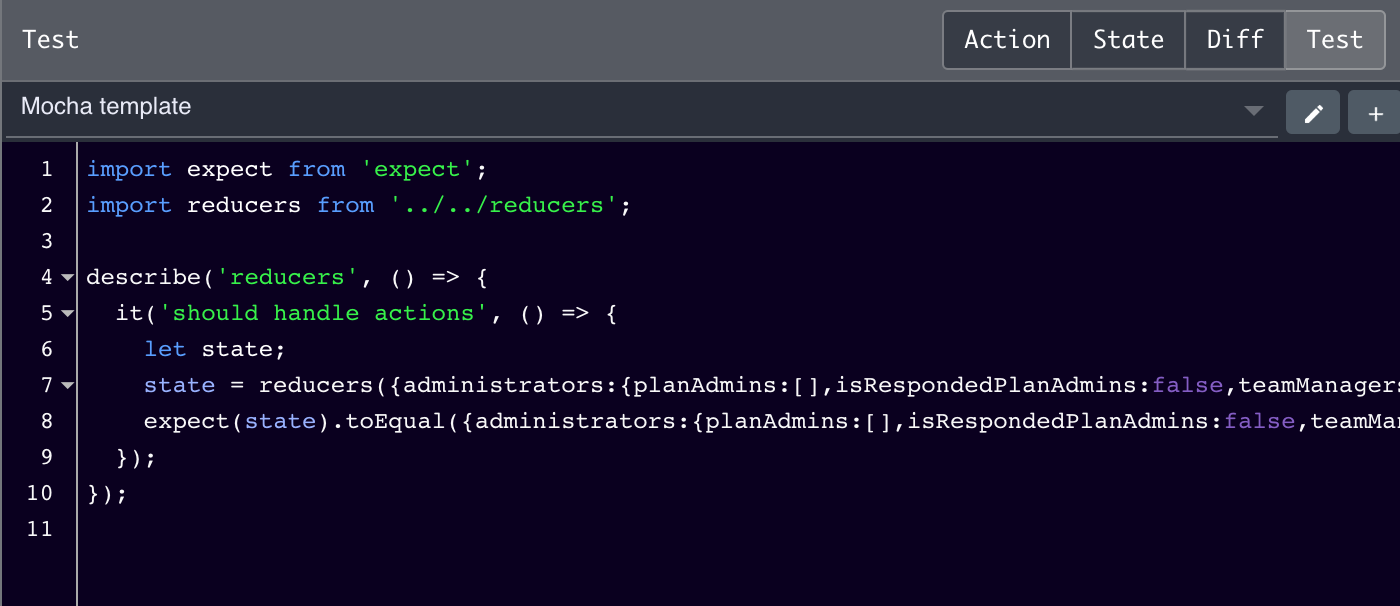
* The Raw tab shows the action creator in a code view.

State

The state tab shows the entire state tree at the time of the action selected in the left hand side of the console.

Test

Back to the upper tabs, there is a test option. This will create a test format in some pre-provided testing frameworks. It takes your root state and provides a written test on what the end state should hold



Top Console Tools



The top of the console offers two more sets of tabs. The right tab lets you flip between different instances of running apps on the page. If you have different stores running on the same page.

The left set of tabs shows different modes:

* Inspector: the most used and is the default mode. Shows all the tools that have been discussed here.
* Log Monitor shows complete state and actions, as opposed to the Inspector mode where it just shows the diffs. (Try using the slider while in this view and watch your state tree grow/shrink!)
* Chart shows the state in a tree-like structure.

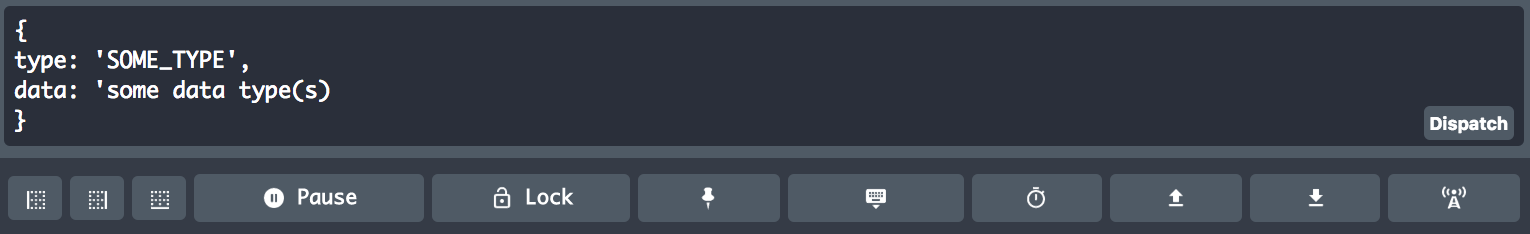
Magic Toolbar

Image for post

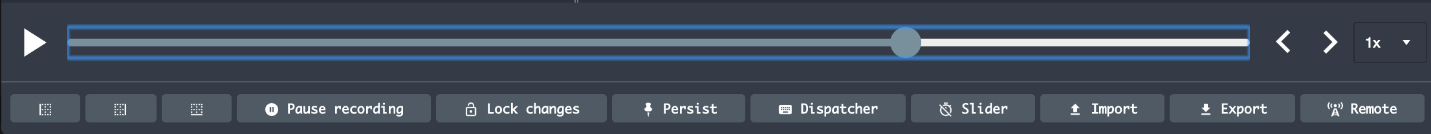
Image for post

The bottom of the console provides more functionality for diving into your actions and state. It allows for time travel, dispatching actions, importing/exporting state, and [remote control.](https://github.com/zalmoxisus/remote-redux-devtools) Starting from the far left, the first three options are for creating a new console view ether to the left, right, or bottom.

* The Pause recording does not stop other actions from firing within your app, it is just stopping the recording of actions within the console. Unlike other abilities within the console it does not control the state of your running browser page.
* The Lock changes will actually freeze the running app’s other/future actions. So it takes Pause recording to the next level by locking any future state changes in your app.
* The Persist option will keep the current state even when the page reloads
* The Dispatcher option will either show or hide the dispatching module. Which can be used to fire a custom action with custom data.



* The Slider option will give you a smooth scroll through the actions. This will time travel through the actions that have fired and update the running app to show the current state and certain points in time. This comes packed with an auto-play, one step action arrows, and a speed interval play-through time.



* The import and export options are for importing/exporting state as JSON into the application. Which enables you to run your app off of custom state.
* The Remote option creates a separate console, not connected to the browser console tools. This will run the same actions but “remotely” away from the dev-tools.