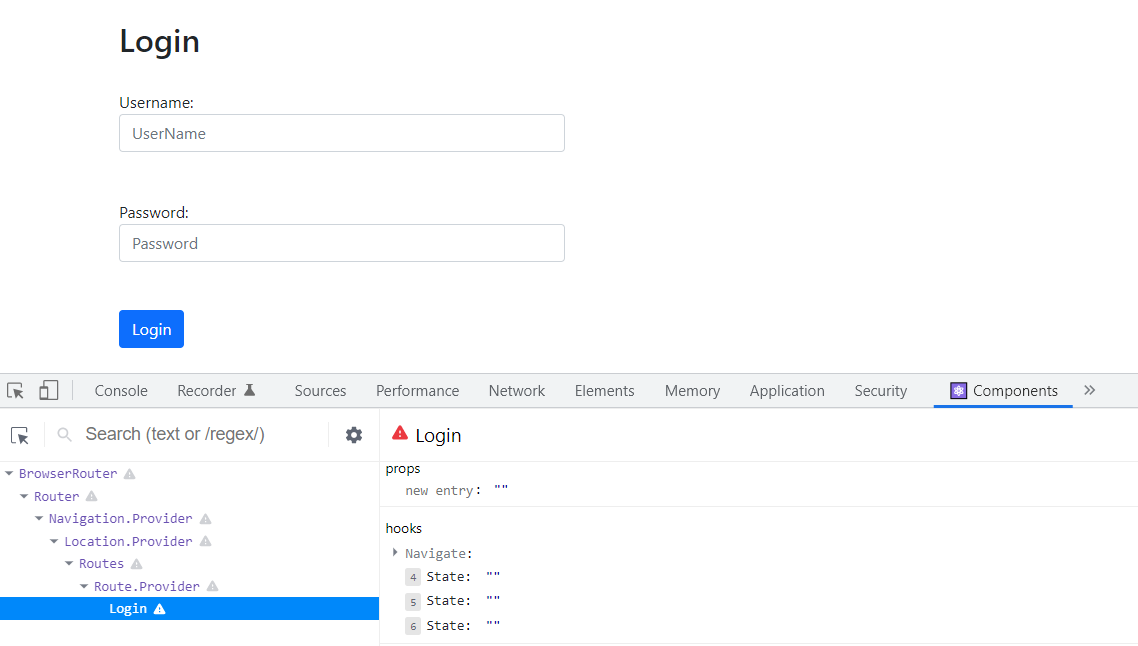
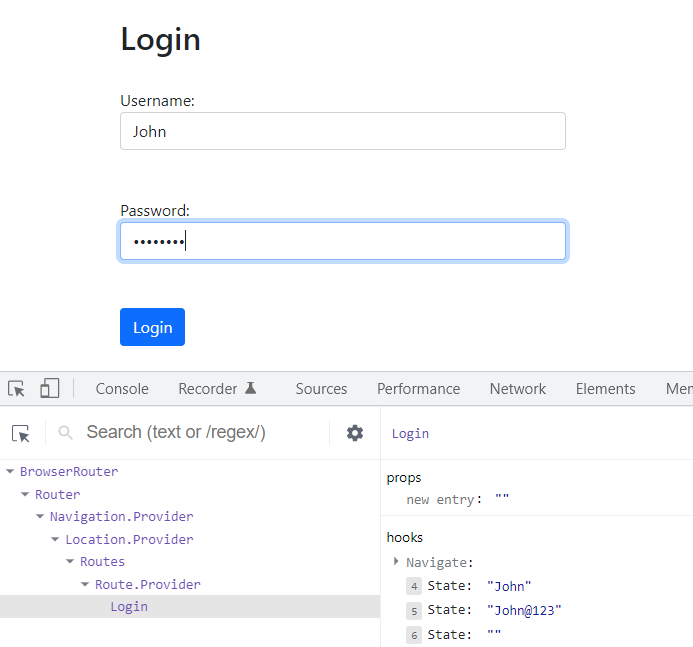
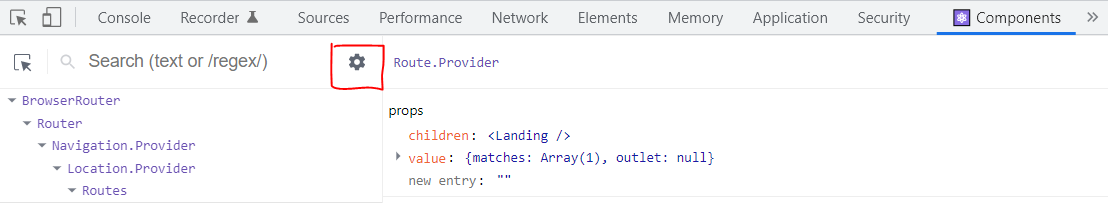
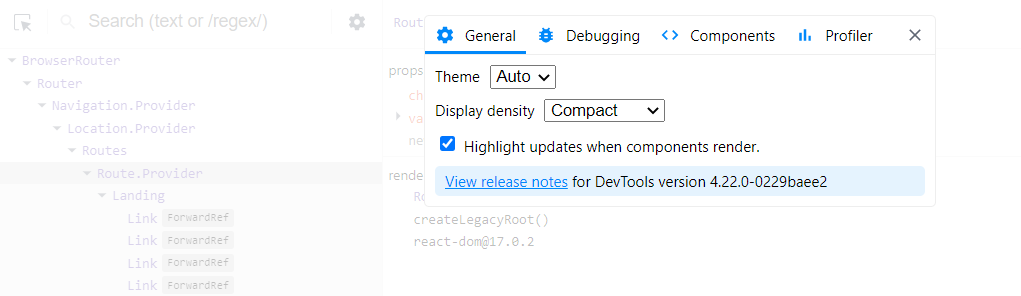
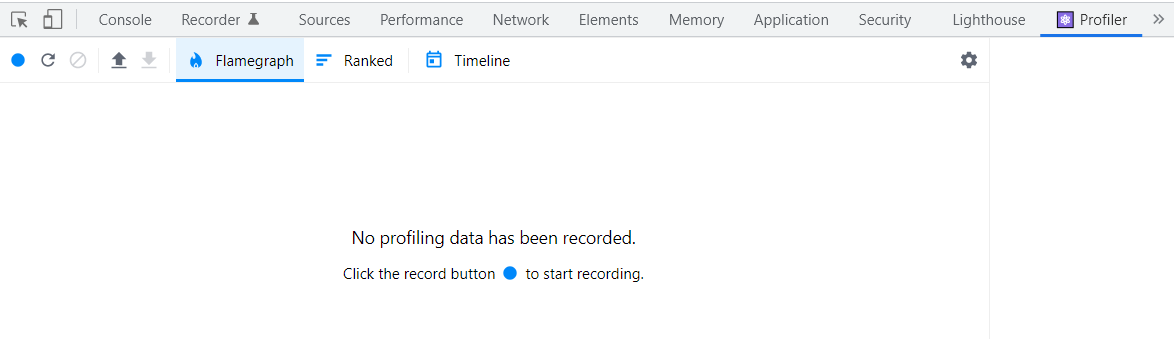
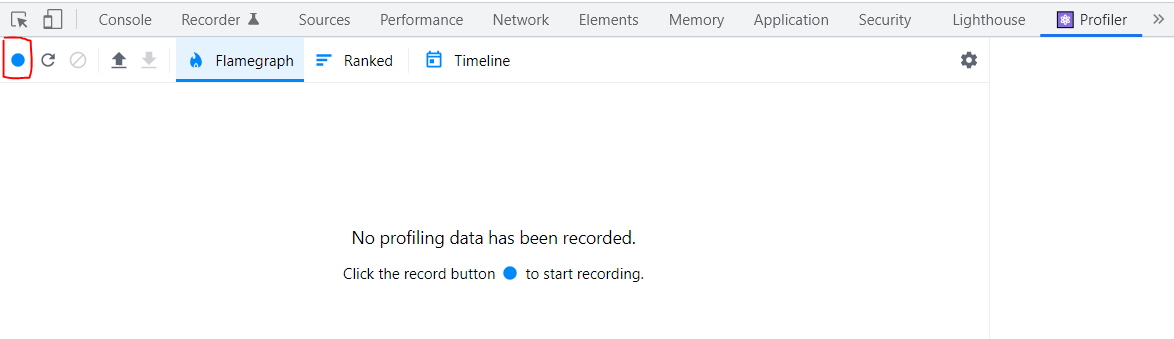
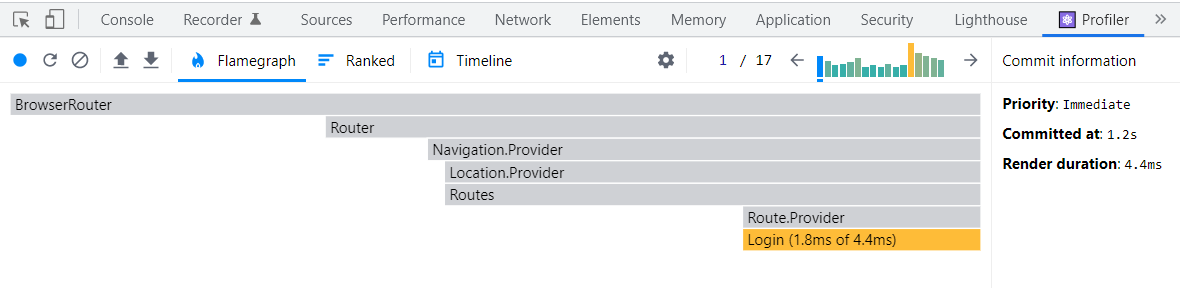
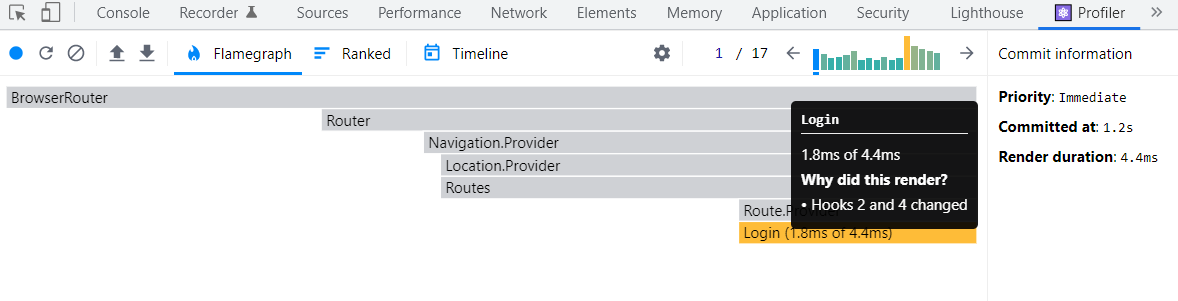
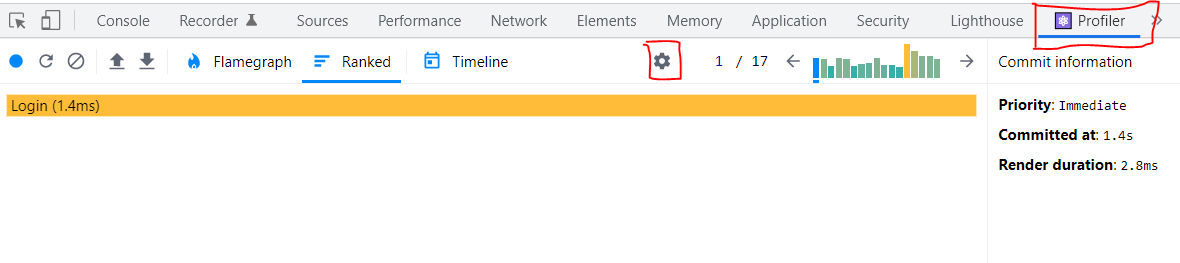
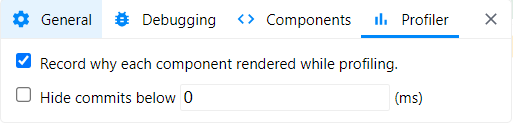
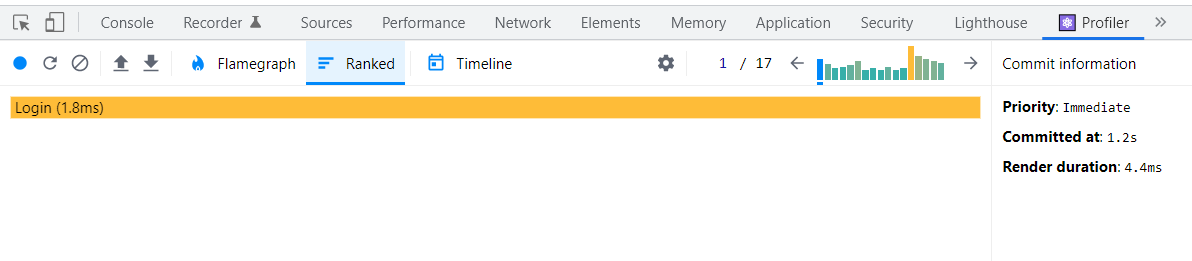
React Developer Tool is a chrome DevTools extension for the ReactJS, which helps us to inspect React component tree in chrome.  
The React Developer Tools is a browser extension with powerful set of utilities to inspect React component tree. You will be able to check the component’s state and identify errors using real data without any console statements or debuggers. You can also use the profiler to explore how components interact with each other, allowing you to identify and optimize components that have slow rendering in your full application.  
Installation and usage:  
1.  Add React Developer tools extension to chrome from chrome web store.  
2. Once it is added, a new tab called Components will appear on chrome developer tool as shown below:  
  
You can check the current state and props of the component   
3. When you enter values for username and password the state gets updated   
   
4. You can track the component re-rendering across interactions  
To observe how components respond to changing data, click on the settings icon.  
   
And then check the option “Highlight updates when components render” under General tab.  
   
When you make any changes, React Developer Tools will highlight components that re-render

**Profiler**

When you install React Developer tools with Chrome, you get a Profiler Tab along with Components tab in the developer tools.  
   
Profiler helps in checking how long each component takes to render. This helps in identifying components that are slow.  
To use Profiler, click on the blue circle present on the left side to begin recording and click it again to finish recording   
   
When you stop recording you will find a graph under Flamegraph tab, which shows how long each component took to render.   
   
You can hover over the component to know the reason why the component was re-rendered  
 ​​​​​​​

To know the reason why the component was re-rendered check the option “Record why each component rendered while profiling”. This option can be found by clicking on the settings tab under the Profiler tab

When you click on the Ranked tab you will find how much more time the component has taken, when compared to all other components.  
​​​​​​​

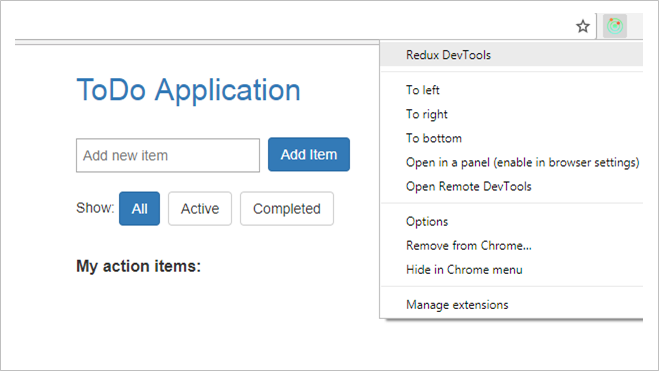
REdux DEvtools extension:

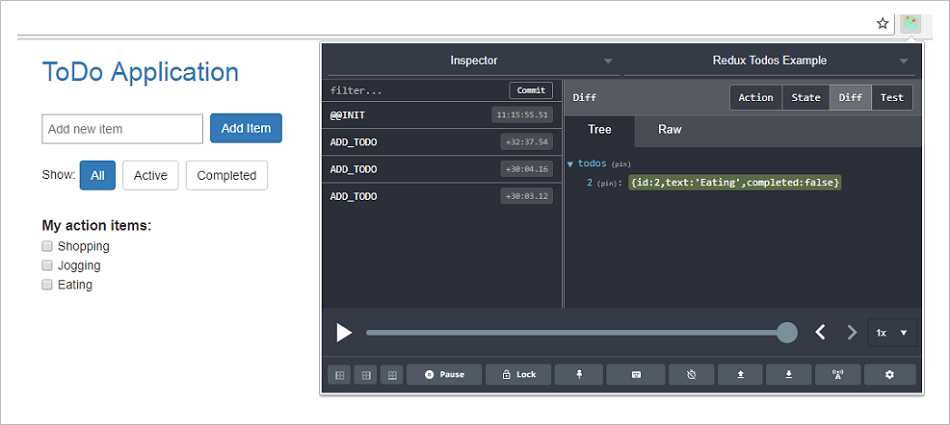
**Time Travel Debugging:** Time travel debugging is a powerful way to see exactly how your application state is changing over time. You can travel through time as you debug, so you can go back in history and see each specific state change. You can also undo specific state changes and observe how the final state changes. You can even turn off individual actions that occurred, so that you observe what would be the state object if a specific action had not happened. You can play all the interactions back with the click of a button and even select the speed at which it plays back.​​​​​​​

Time Travel debugging can be done using Redux developer tools which is available as an add-on to the browser. Once the Redux DevTools is installed in the browser to use it you need to include the below code in your application.

let store = applyMiddleware(thunk)(createStore) (rootReducer,window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_ && window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_())

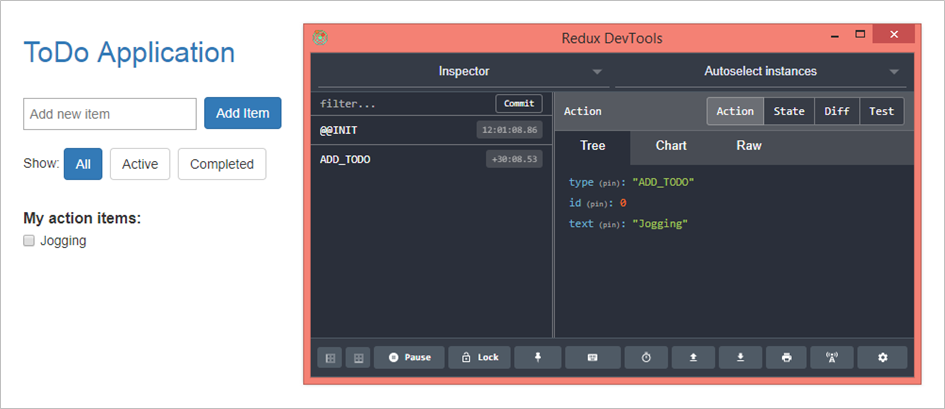
By clicking on the Redux dev tools, you can debug the application as shown below.



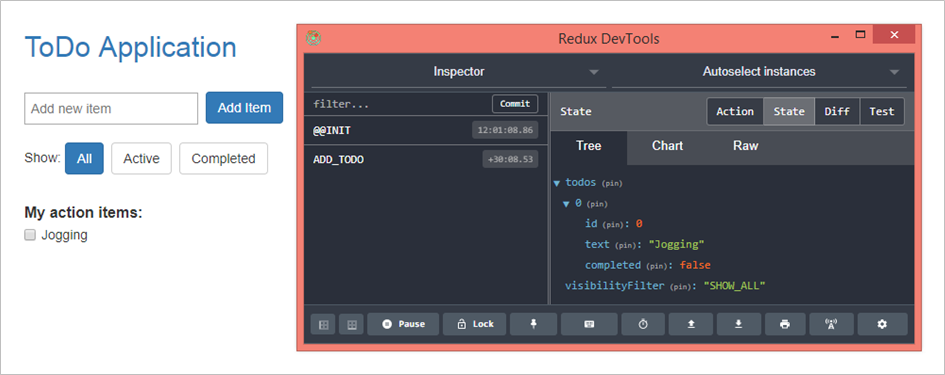


With time travel debugging you can go back to previous points of the state. This helps in testing very specific parts of your interaction.

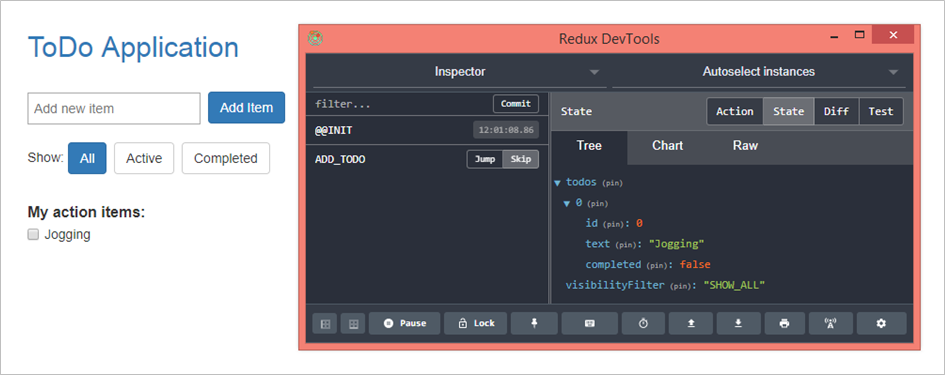
In the todo application after adding a todo item, you can see the action by clicking on the Action tab in the Redux developer tools as shown below:



The current state can be viewed by clicking on the State tab in the Redux developer tools as shown below. The state can be viewed as a tree, chart or Raw by clicking on the corresponding tabs.



You can skip a particular action by clicking on the skip option present next to the action in the Inspector tab as shown below



When you skip ADD\_TODO action above, the Jogging item will be removed from the view.

Redux developer tools help in debugging your application by moving back and forth with the state which is because of state immutability in Redux.