Debugging with React Developer Tools

In this lesson, we'll learn how to debug React code with React Developer Tools

Introducing React Developer Tools

React Developer Tools lets you inspect the React components hierarchy, props and state. It comes as a browser extension for Chrome and Firefox and as a standalone app that works with other environments. Once installed, the extension icon will light up on websites using React. On such pages, you will see a tab called “React” in your browser’s developer tools.

Debugging the Hacker News App

Let’s try it on our Hacker News application. On most browsers, a quick way to bring the *dev tools* up is to right-click on the page and than hit “Inspect”. Do it when your application is loaded, then click on the “React” tab. You should see its elements hierarchy, being `<App>` the root element. If you expand it, you will find instances of your `<Search>`, `<Table>` and `<Button>` components, as well.

The extension shows on the side pane the component’s state and props for the selected element. For instance, if you click on `<App>`, you will see that it has no props, but it already has a state. A very straightforward debugging technique is monitoring your application’s state changes from user interaction.

First, check the “Highlight Updates” option, usually above the elements tree. Second, type a different search term in the application’s input field. Only `searchTerm` will be changed in the component’s state. We already knew that would happen, but now we can see it working as planned.

Finally, press the “Search” button. The `searchKey` state will immediately changes to same value as `searchTerm`, and then the response object is added to `results`. The asynchronous nature of your code is now visible.

If you right-click on any element, a dropdown menu will show several useful options. For instance, you could copy the element’s props or name, find the

corresponding DOM node, or jump to the application’s source code in the browser. The last option is very useful for inserting breakpoints and debugging your JavaScript functions.

Exercises

- Install the React Developer Tools extension on your favorite browser
    - Run your Hacker News Clone application and inspect it using the extension
    - Experiment with state and props changes
    - Watch what happens when you trigger an asynchronous request
    - Perform several requests, including repeated ones. Watch the cache mechanism working

Further Reading:

- Read about how to debug your JavaScript functions in the browser
- Read about (and use) the React Profiler