React Developer Tools

Use React Developer Tools to debug your React applications

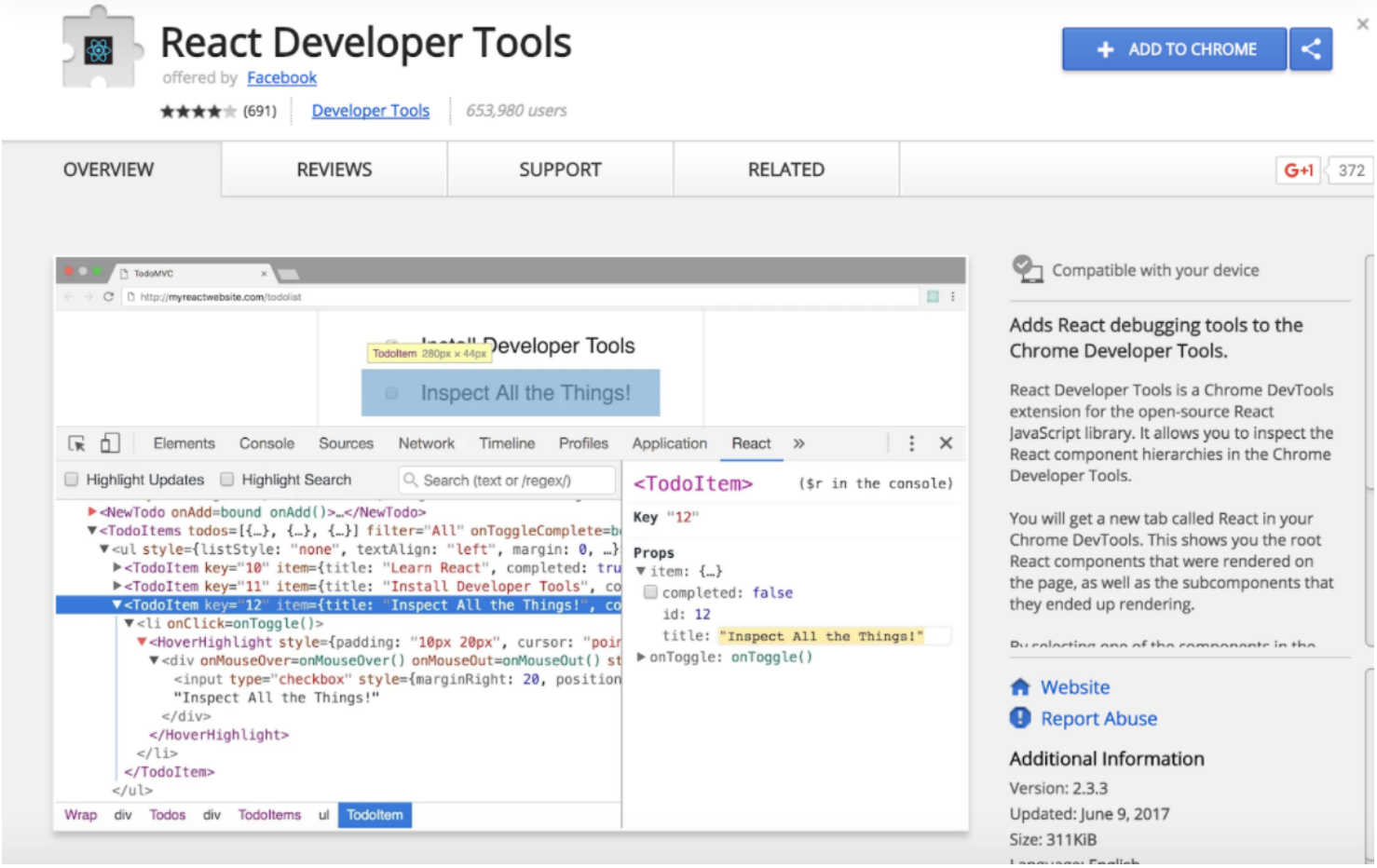
React Developer Tools

Introduction

Effectively debugging applications is a cornerstone of programming. After creating a React App, an important next step is setting up your environment to debug it. We'll cover the basics in this article using the initial skeleton created by using create-react-app. This article assumes you are familiar with create-react-app and Chrome DevTools.

1. Install React Developer Tools

Facebook created a Chrome extension to help with debugging React Apps. It is called *React Developer Tools* and allows developers to inspect React components, view their properties, and interact with them while looking at the application in Google Chrome. You can add this functionality to Chrome by navigating to the extension page here, selecting "ADD TO CHROME", and following the installation prompts.



2. Inspect React Components

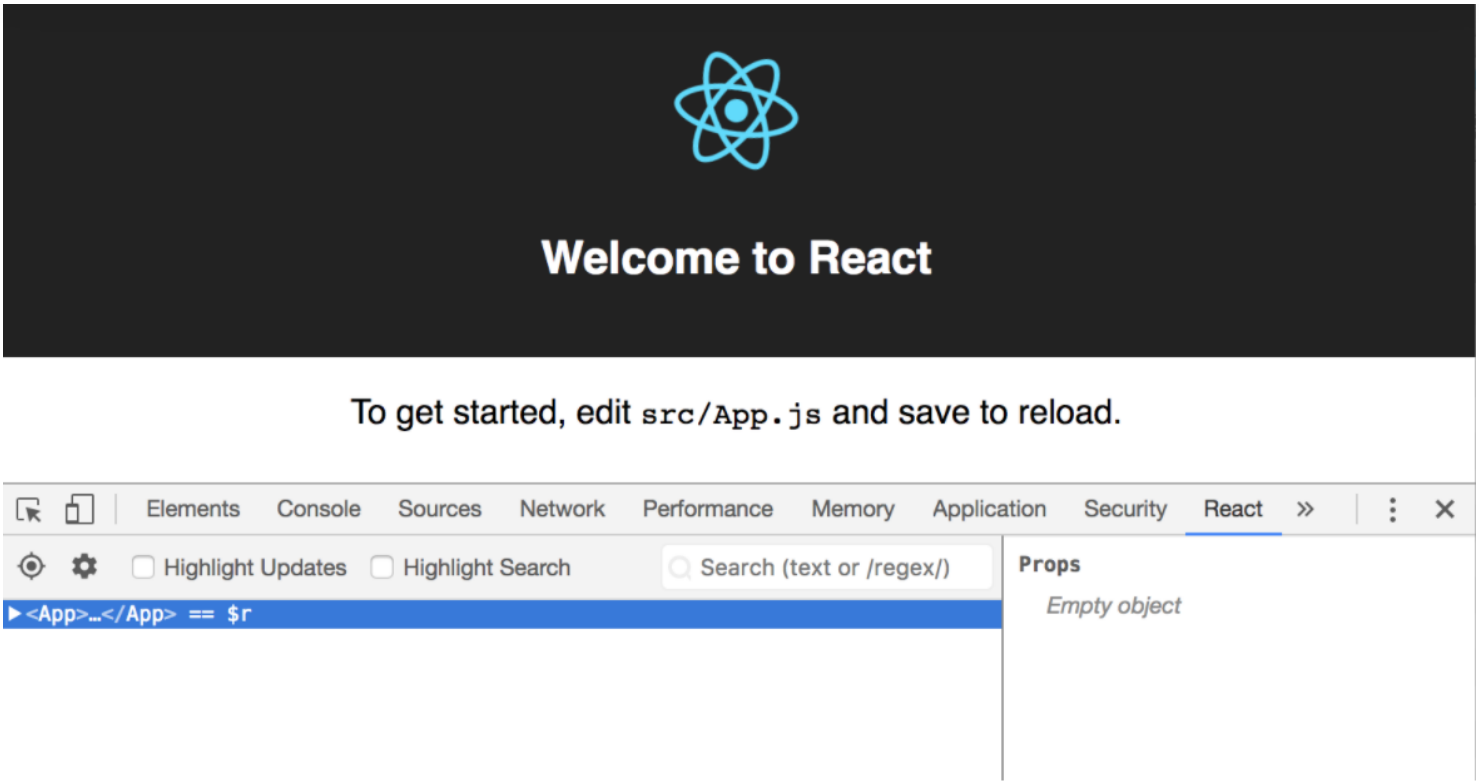
With the extension installed, if you start your React App (npm start) and visit the site in Chrome, the React Developer Tools icon in the Chrome menu bar should change from inactive:

to active:

. This indicates that the site you are browsing is a React App.

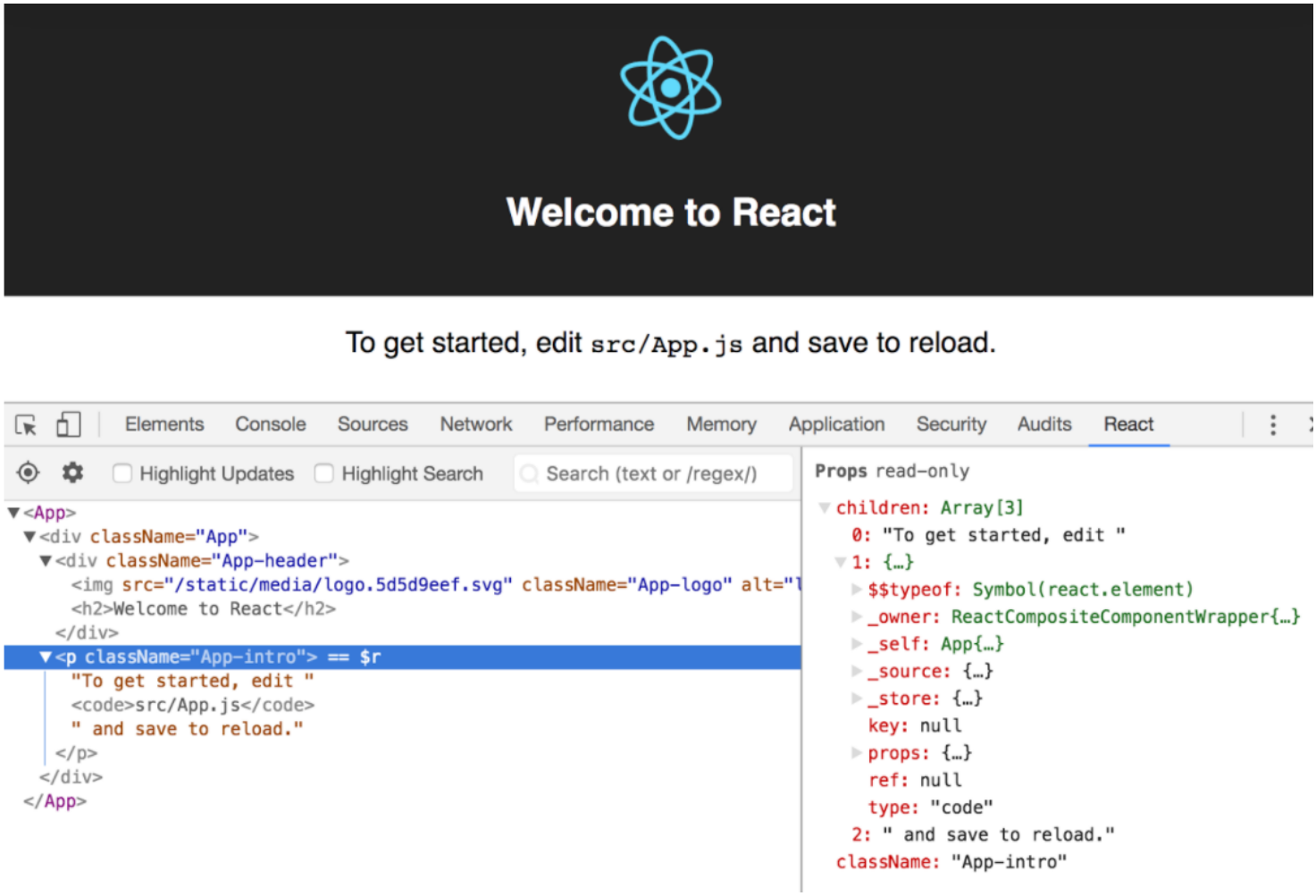
To open the React Developer Tools, first open Chrome DevTools (View > Developer > Developer Tools) and then select the “React” tab on the right:

Next Get Help

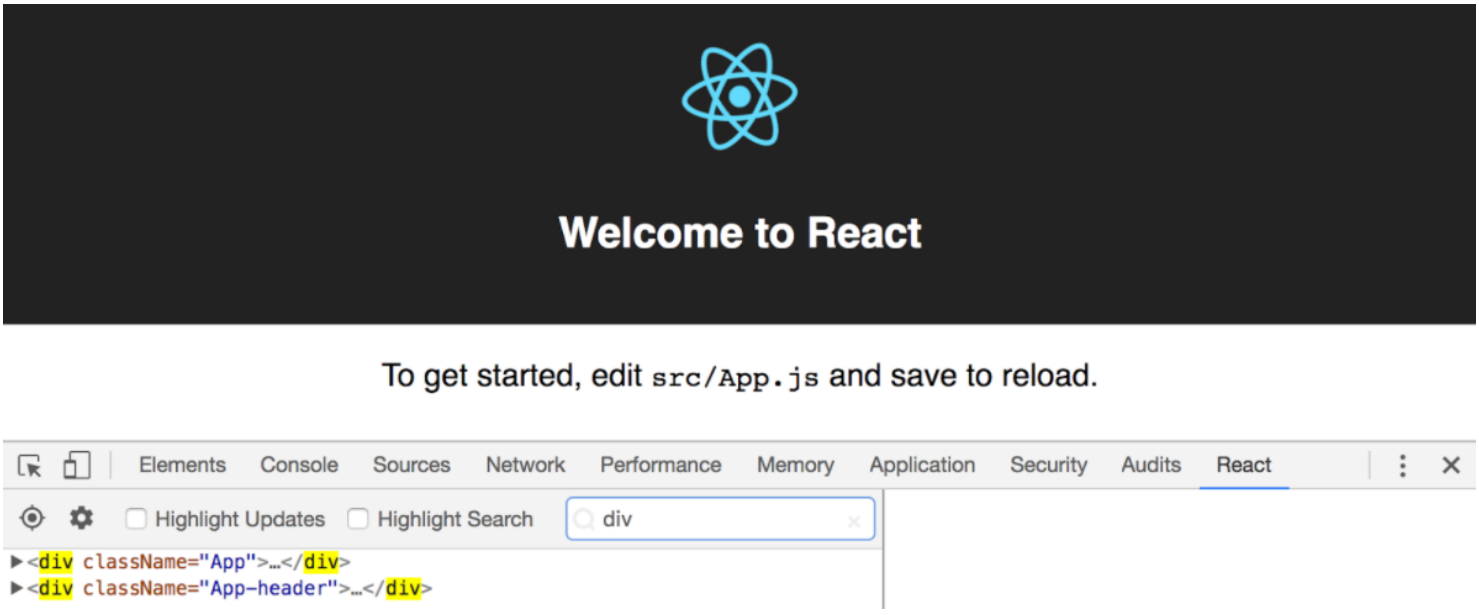


Note that this tab will only appear on sites using React. If you expand `<App>...</App>`, you will see a tree of all the rendered React components. As you hover over components on the left, they are highlighted in the rendered view, similar to Chrome DevTools. If you click on components in the left side of the window, their properties are exposed on the right side:

Next Get Help



You can also use the search box to locate components by name:



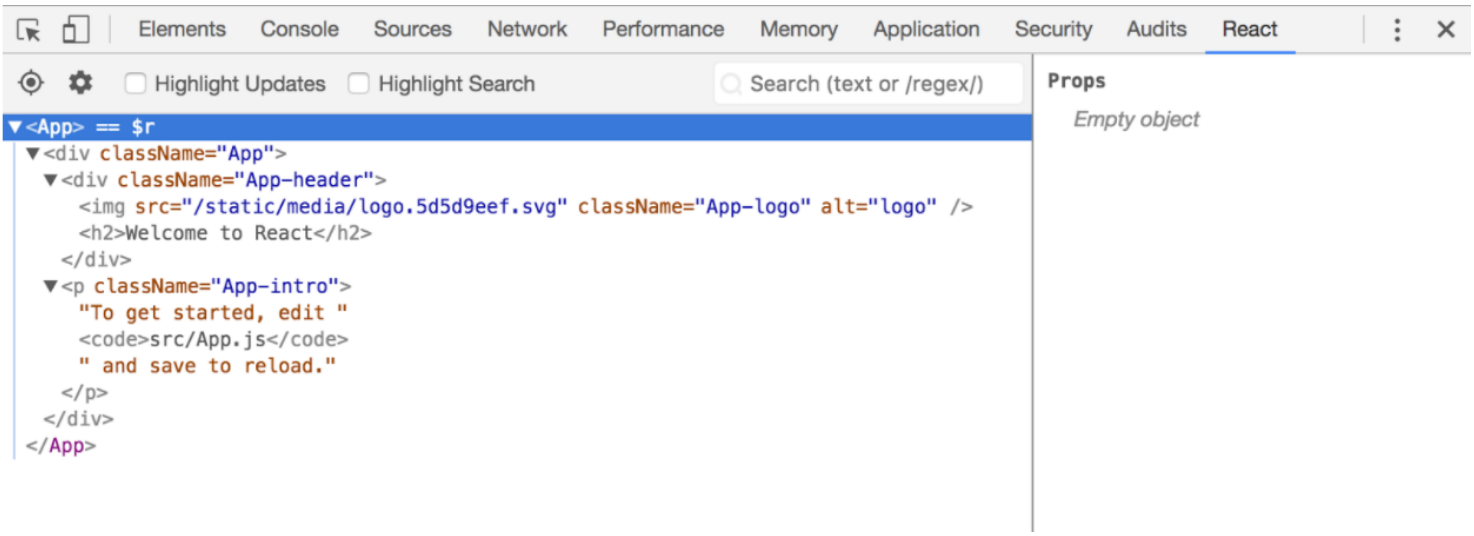
Further details on how to use these tools are available through the official

Next Get Help

3. Modify Components with Javascript

With React Developer Tools and the console, it is possible to modify rendered React components. This allows you to experiment with changing component values, calling methods, and testing interaction between components.

As an example, select the main `<App>` component in React Developer Tools. You'll see that it appends `== $r` to the right side of the component name in the tree:



Now, if you switch over to the console view, you can access this component using `$r`. By logging $r, you can see that this is indeed the component selected in the React tab. You can do things like access its `state`, update its `state`, and access its `props`:

```
> $r
◂ ▶App {props: Object, context: Object, refs: Object, updater: Object, _reactInternalInstance: ReactCompositeComponentWrapper…}
> $r.setState({foo:"bar"})
◂ undefined
> $r.state
◂ ▶Object {foo: "bar"}
> $r.props
◂ ▶Object {}
>
```

With these tools you're now ready to begin debugging React Apps!

Next Get Help

Next

Get Help