Logging the State Object

Now we'll write a function which logs all our store updates. This will be helpful in understanding if our contacts component is working properly.

Each time the store updates and invokes render, let’s log the state from the store.

Here’s how:

```
const render = () => {
  fancyLog();
  return ReactDOM.render(<App />, document.getElementById("root"));
};
```

Just call a new function, **fancyLog()** you’ll soon write. Here’s the fancyLog function:

```
function fancyLog() {
  console.log("%c Rendered with 👉 👉 👇", "background: purple; color: #FFF");
  console.log(store.getState());
}
```

Hmmm. What have I done?

Console.log(store.getState()) is the bit you’re familiar with. This will log the state retrieved from the store.

The first line, **console.log("%c Rendered with 👉 👉 👇 ", “background: purple; color: #fff”);** will log the text, ”Rendered with …” plus some emoji, and some CSS style to make it distinguishable. The %c written before the *”Rendered with …”* text makes it possible to use the CSS styling.

Enough talking. Here’s the complete code:

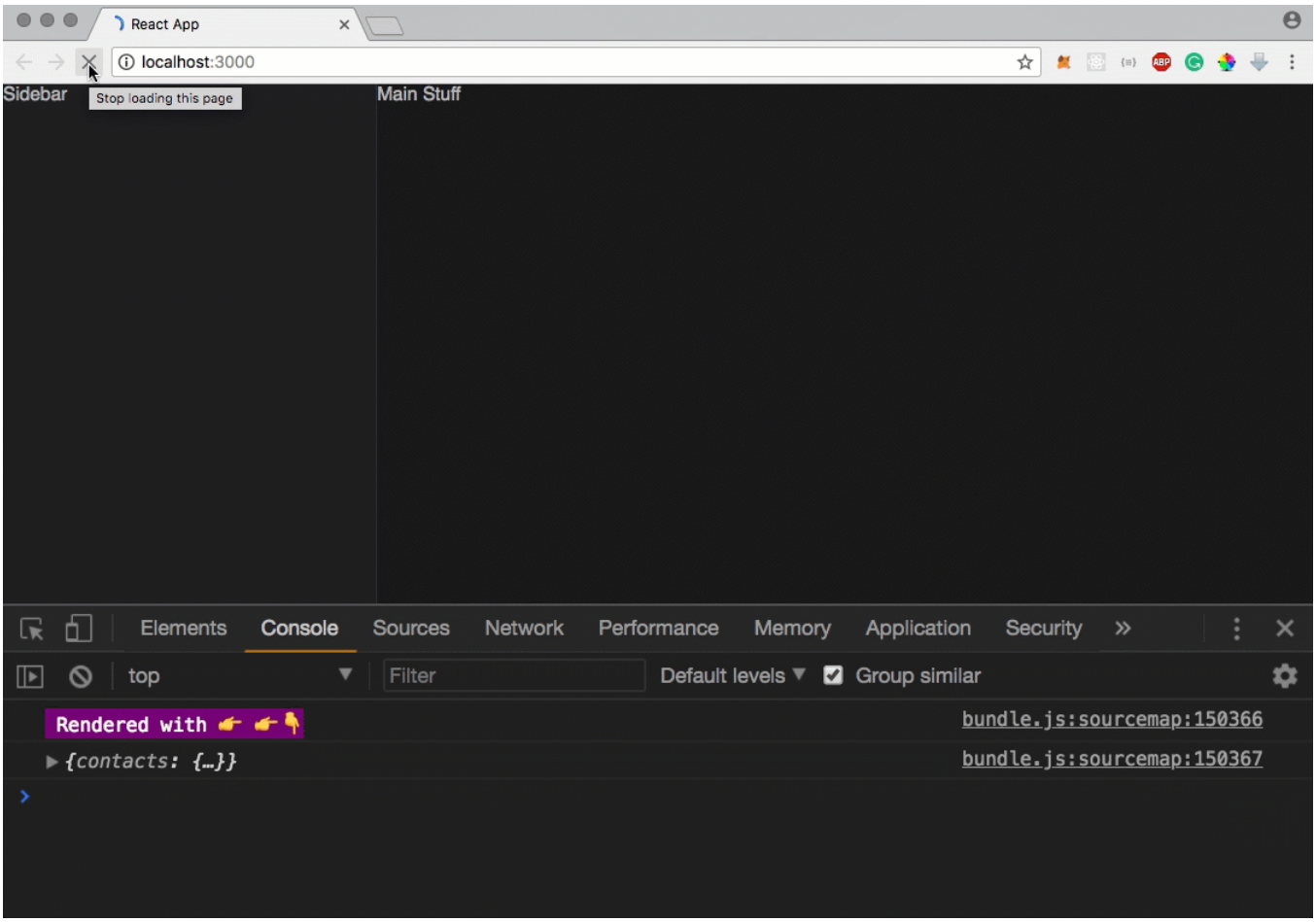
index.js:

```
import ReactDOM from "react-dom";
```

```
import ReactDOM from "react-dom";
import "./index.css";
import App from "./App";
import registerServiceWorker from "./registerServiceWorker";
import store from "./store";
const render = () => {
  fancyLog();
  return ReactDOM.render(<App />, document.getElementById("root"));
};
render();
store.subscribe(render);
registerServiceWorker();
function fancyLog() {
  console.log("%c Rendered with 👉👉👇", "background: purple; color: #FFF"");console.log(stc
}
```

index.js

Here's the state object being logged.



As you can see, within the state object is a **contacts** field that holds the contacts available for the particular user. The structure of the data is as we discussed before now. Each contact is mapped with their **user_id**

We've made decent progress.

Next, we'll figure out a way to pass contact data to the sidebar.