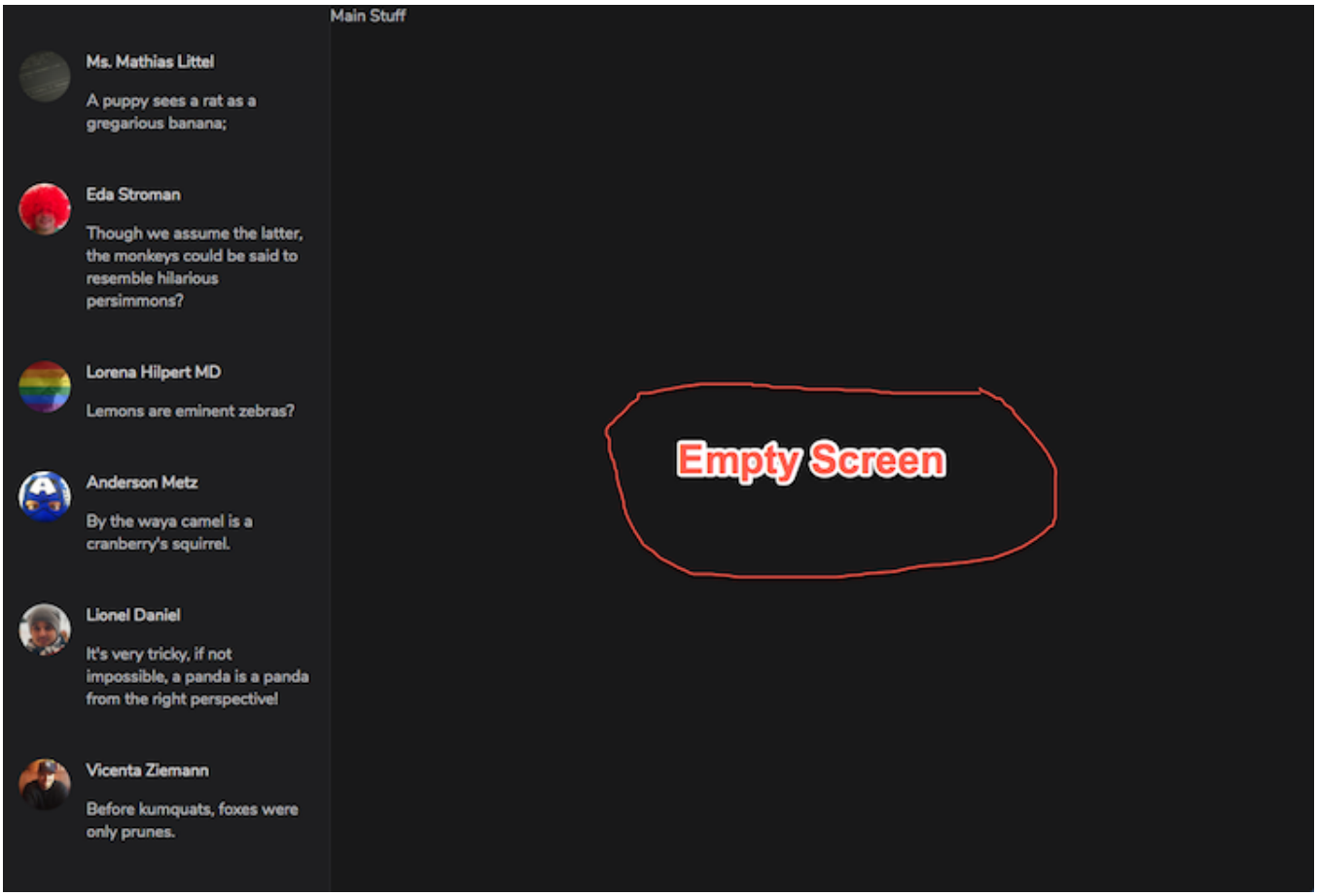
# The Active User

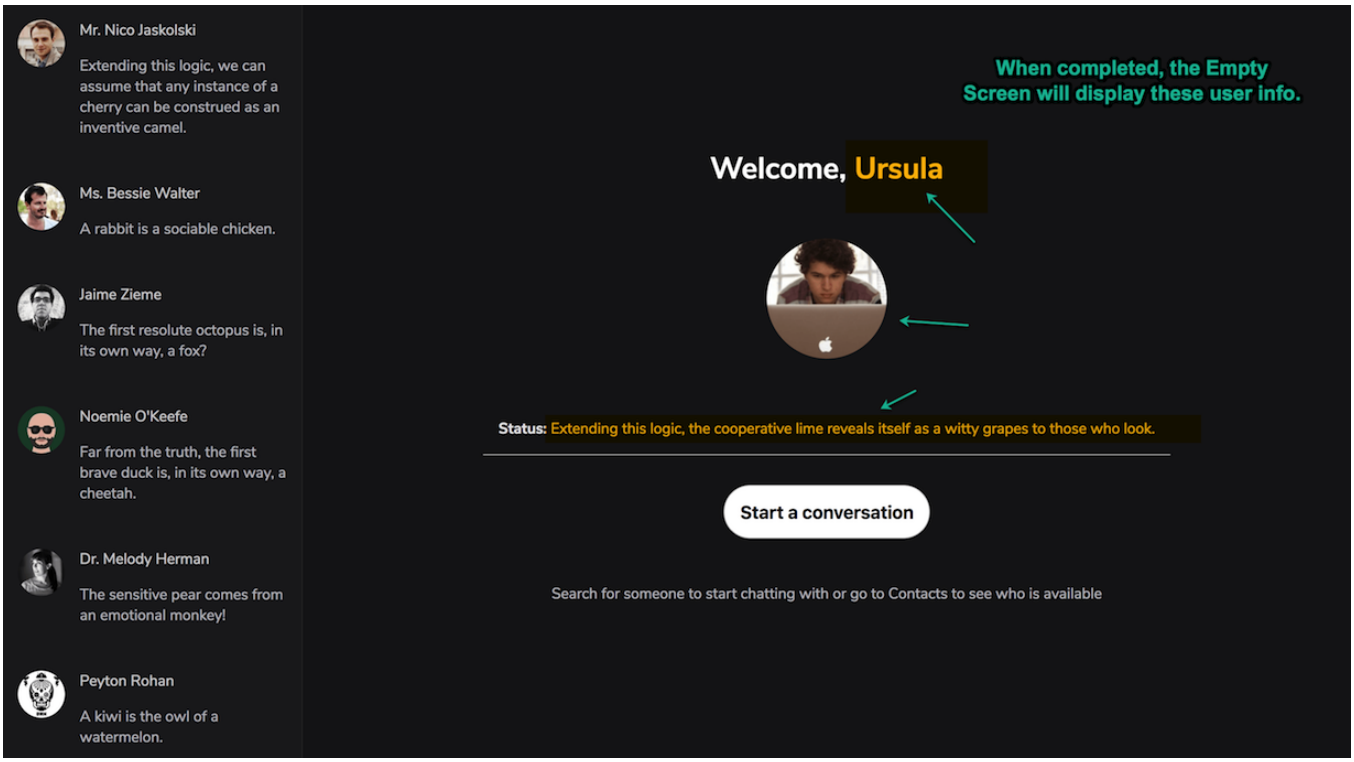
For the first step of building our Empty Screen, we will create a new field in the state object called "activeUserID". This will act as a template for viewing a contact. If set to NULL (initial state), the empty screen will show our profile (Visual Representation below).

> Right now, the Main component just displays the text, main stuff.
>
> This isn't what we want.
>
> The end goal is to show an empty screen, but show user messages when a contact is clicked on.




> Let's build the empty screen.
>
> For this, we'll need a new component called, **Empty.js**. While at it, also create a corresponding CSS file, **Empty.css**.
>
> Please create these in the **components** directory.

<Empty /> will render the markup for the empty screen. To do this, it will require a certain user prop.




Definitely, the user is to be passed in from the state of the application. Don't forget the overall structure of the state object we resolved earlier:

So, here's the current content of the <Main /> component:

```
import React from "react";
import "./Main.css";
const Main = () => {
  return <main className="Main">Main Stuff</main>;
};
export default Main;
```

It just returns the text, Main Stuff.

The <Main /> component is responsible for displaying the <Empty /> component when no user is active. As soon as a user is clicked, <Main /> renders the conversations of the clicked user. This could be represented by a component, <ChatWindow />.

For this render toggle to work i.e for <Main /> to render either <Empty /> or <ChatWindow />, we need to keep track of certain activeUserId.

For example, by default activeUserId will be null, then <Empty /> will be shown.

However, as soon as a user is clicked, the activeUserId becomes the user_id of the clicked contact. Now, <Main /> will render the <ChatWindow /> component.

Cool, huh?

For this to work, we will keep a new field in the state object, activeUserId.

By now, you should know the drill already. To add a new field to the state object, we'll have this set up in the reducers.

Create a new file, activeUserId.js in the reducers folder.

And here's the content of the file:

reducers/activeUserID.js:

```
export default function activeUserId(state = null, action) {
  return state;
}
```

reducers/activeUserID.js

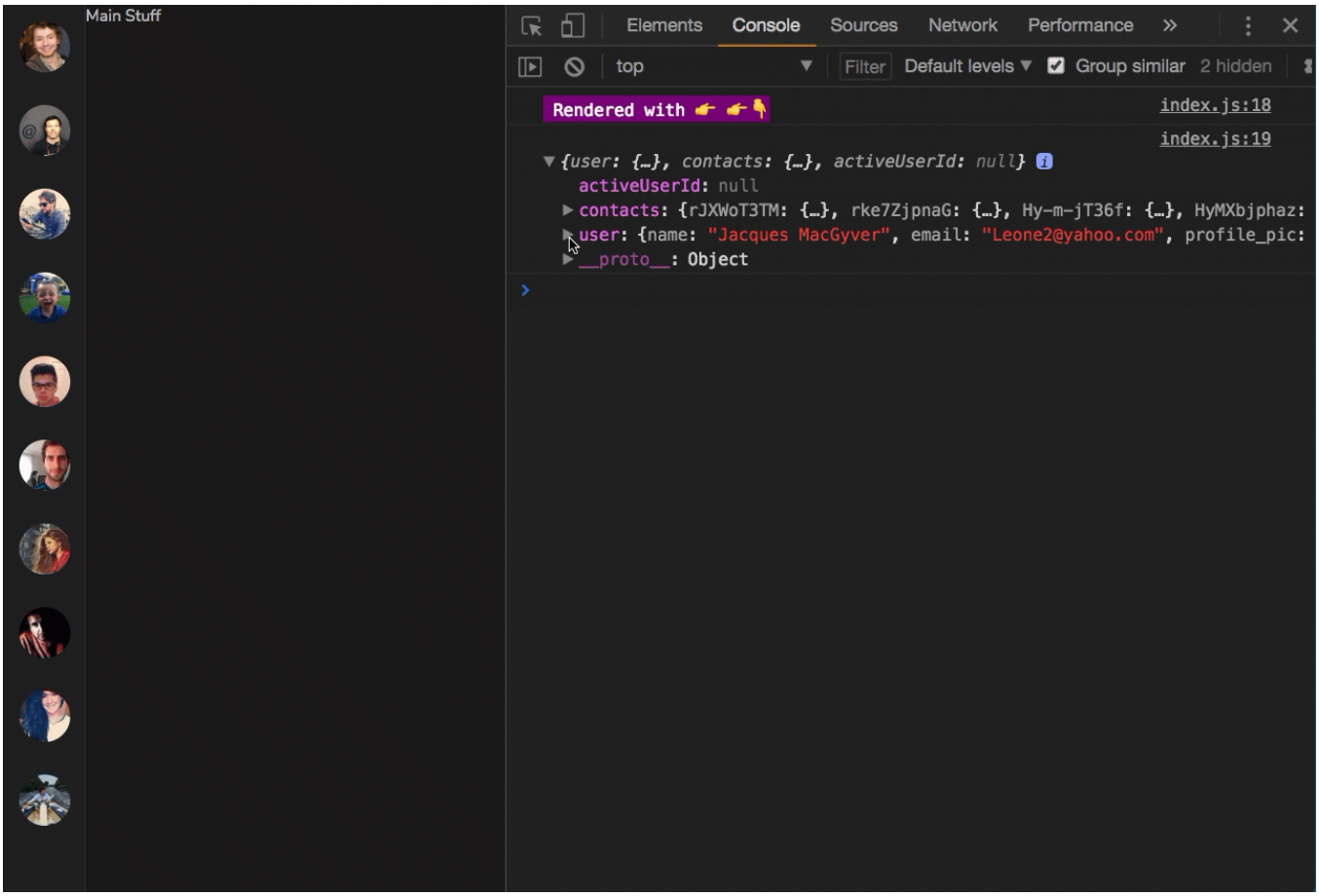
By default, it returns null.

Now, hook this newly created reducer to the combineReducer method call like this:

```
...
import activeUserId from "./activeUserId";

export default combineReducers({
  user,
  contacts,
  activeUserId
});
```

Now if you inspect the logs, you'll find activeUserId right there in the state object.




This is a lot to digest. Take your time understanding our progress up till now.

We'll pick up from here in the next lesson.