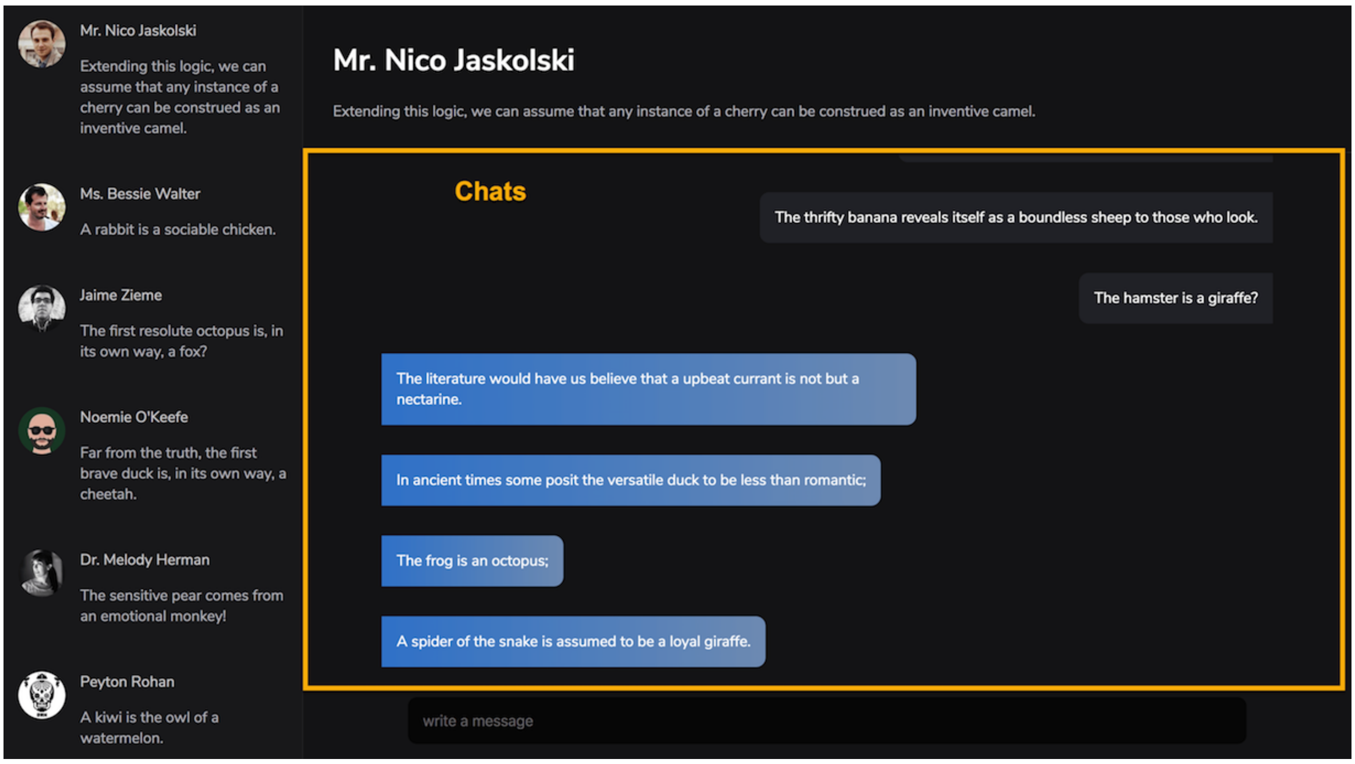
The Chats Component

Just as we've done so far, our first step in making this component will be to create a reducer for the messages and make sure that it is a visible field in the consoleLog.

Let’s move on to building the <Chats /> component.



Mr. Nico Jaskolski

Extending this logic, we can assume that any instance of a cherry can be construed as an inventive camel.

Ms. Bessie Walter

A rabbit is a sociable chicken.

Jaime Zieme

The first resolute octopus is, in its own way, a fox?

Noemie O'Keefe

Far from the truth, the first brave duck is, in its own way, a cheetah.

Dr. Melody Herman

The sensitive pear comes from an emotional monkey!

Peyton Rohan

A kiwi is the owl of a watermelon.

Mr. Nico Jaskolski

Extending this logic, we can assume that any instance of a cherry can be construed as an inventive camel.

Chats

The thrifty banana reveals itself as a boundless sheep to those who look.

The hamster is a giraffe?

The literature would have us believe that a upbeat currant is not but a nectarine.

In ancient times some posit the versatile duck to be less than romantic;

The frog is an octopus;

A spider of the snake is assumed to be a loyal giraffe.

write a message

The component is essentially a rendered list of a user’s conversations.

So, where do we get these conversations from?

Yeah, from the state of the application.

Like I explained earlier, a real world app will fetch the user conversations from a server. However, my approach to learning Redux is that you eliminate as many complexities as possible when learning the fundamentals.

To that effect, there’ll be no server fetching resource here. We’ll hook up the data using some helper functions I have created for random user data generation.

Let’s start by hooking up the needed data to the state of the application. The process is the same as we've done multiple times already.

> 1. Create a Reducer
> 2. Using ES6, add a default parameter value to the reducer
> 3. Include the reducer in the combineReducers function call.

Will you try that out before moving on to my solution?

Here comes my solution, anyway.

Create a new file, messages.js in the reducers directory. This will be the messages reducer.

Here is the content of the messages reducer.

reducers/messages.js:

```
import { getMessages } from "../static-data";
export default function messages(state = getMessages(10), action) {
  return state;
}
```

reducers/messages.js

To generate random messages, I have imported the `getMessages` function from static-data.

This function takes an amount, represented by a number. The getMessages function will then generate that amount of messages for each user contact.

For example, `getMessages(10)` will generate 10 messages per user contact.

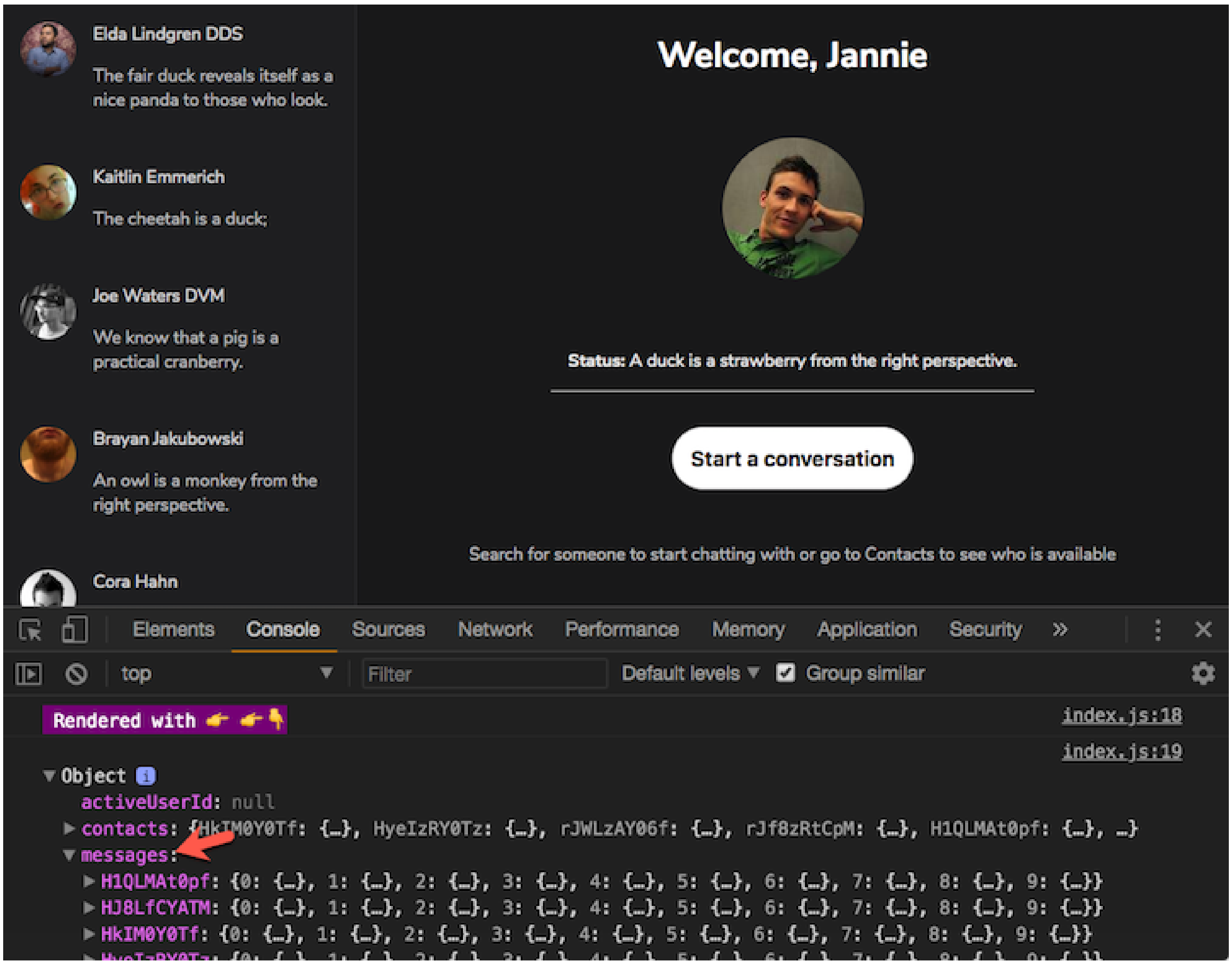
Now, include the reducer in the combineReducers function call in reducers/index.js

reducers/index.js:

```
import messages from "./messages";
export default combineReducers({
  user,
  messages,
  contacts,
  activeUserId
});
```

Doing this will include a messages field in the state object.

Here’s a look at the logs. You’ll now find messages as seen below:



Elda Lindgren DDS
The fair duck reveals itself as a nice panda to those who look.

Kaitlin Emmerich
The cheetah is a duck;

Joe Waters DVM
We know that a pig is a practical cranberry.

Brayan Jakubowski
An owl is a monkey from the right perspective.

Cora Hahn

Welcome, Jannie

Status: A duck is a strawberry from the right perspective.

Start a conversation

Search for someone to start chatting with or go to Contacts to see who is available

```
Rendered with 👉 👉👇                                                              index.js:18
                                                                                  index.js:19
▼Object ℹ
   activeUserId: null
  ▶contacts: {HkIM0Y0Tf: {…}, HyeIzRY0Tz: {…}, rJWLzAY06f: {…}, rJf8zRtCpM: {…}, H1QLMAt0pf: {…}, …}
  ▼messages:
    ▶H1QLMAt0pf: {0: {…}, 1: {…}, 2: {…}, 3: {…}, 4: {…}, 5: {…}, 6: {…}, 7: {…}, 8: {…}, 9: {…}}
    ▶HJ8LfCYATM: {0: {…}, 1: {…}, 2: {…}, 3: {…}, 4: {…}, 5: {…}, 6: {…}, 7: {…}, 8: {…}, 9: {…}}
    ▶HkIM0Y0Tf: {0: {…}, 1: {…}, 2: {…}, 3: {…}, 4: {…}, 5: {…}, 6: {…}, 7: {…}, 8: {…}, 9: {…}}
```

With that in place, we can safely resume building the Chats component.