Restructuring the State Object

In this lesson, we'll turn our three independent components into objects and figure out a way to link them.

In the previous lesson, we decided to make our Messages and Contacts components independent.

We now need to figure out a way to connect the two in order to know which messages belong to which user.

Here's one way to handle this:

```
const state = {
  user: [],
  messages: [
    {
    messageTo: 'contact1',
      text: "Hello"
    },
    {
      messageTo: 'contact2',
      text: "Hey!"
    }
  ],

  contacts: ['Contact1', 'Contact2']
}
```

So, all I've done is make the **messages** field an array of message objects. These objects will have a **messageTo** key.

This key shows which contact a particular message belongs to.

We are getting close. Just a bit of refactoring, and we are done.

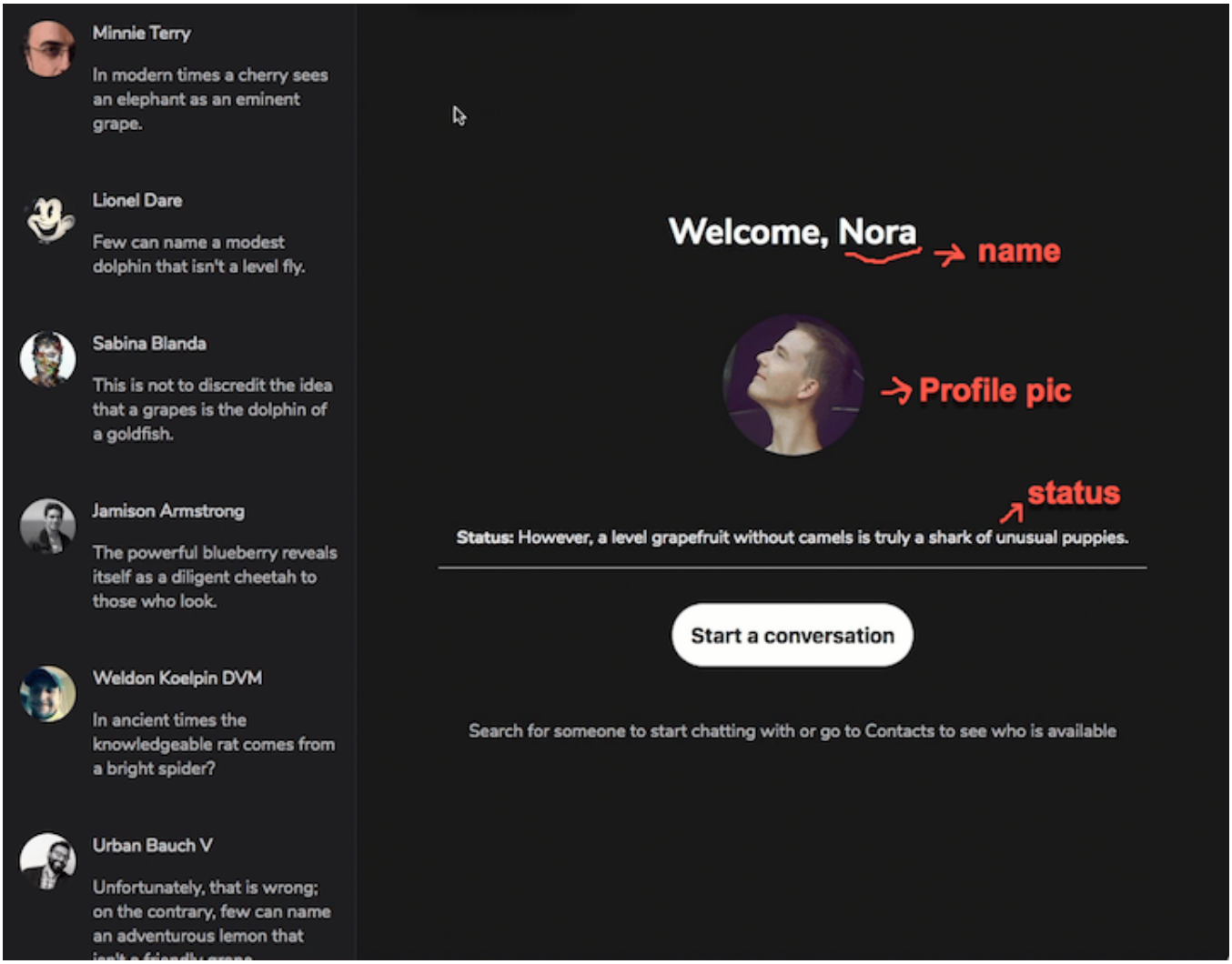
Instead of just an array, a user may be better described by an object - a user object.

```
user: { name,
  email,
  profile_pic,
```

```
  status:,
  user_id
}
```

User Object

A user will have a name, email, profile picture, fancy text status and a unique user ID. The user ID is important - and must be unique for each user.



Think about it. The contacts of a person may also be represented by a similar user object.

So, the **contacts** field within the state object may be represented by a list of user objects.

```
contacts: [
  {
       name,
       email,
       profile_pic,
       status,
       user_id
  },
  {
       name,
```

```
      email,
      profile_pic,
      status,
      user_id_2
  }
]
```

Okay. So far so good.

The contacts field is now represented by a huge array of user objects.

Instead, we can have the contacts represented by an object too. Here's what I mean.

Instead of wrapping all the user contacts in a giant array, they could also be put in an object. See below:

```
contacts: {
     user_id: {
       name,
       email,
       profile_pic,
       status,
       user_id
     },
     user_id_2: {
       name,
       email,
       profile_pic,
       status,
       user_id_2
     }
}
```

Since objects must have a key value pair, the unique IDs of the contacts are used as keys to their respective user objects.

Makes sense?

There's some advantages to using objects over arrays. There's also downsides.

In this application, I'll mostly be using objects to describe the fields within the state object. If you're not used to this approach, this lovely video explains some of the advantages to this approach.

Like I said earlier, there are a few disadvantages to this too approach but I'll show you how to get over them.

We have resolved how the contacts field will be designed within the application state object. In the next lesson, we will move on to the messages field.