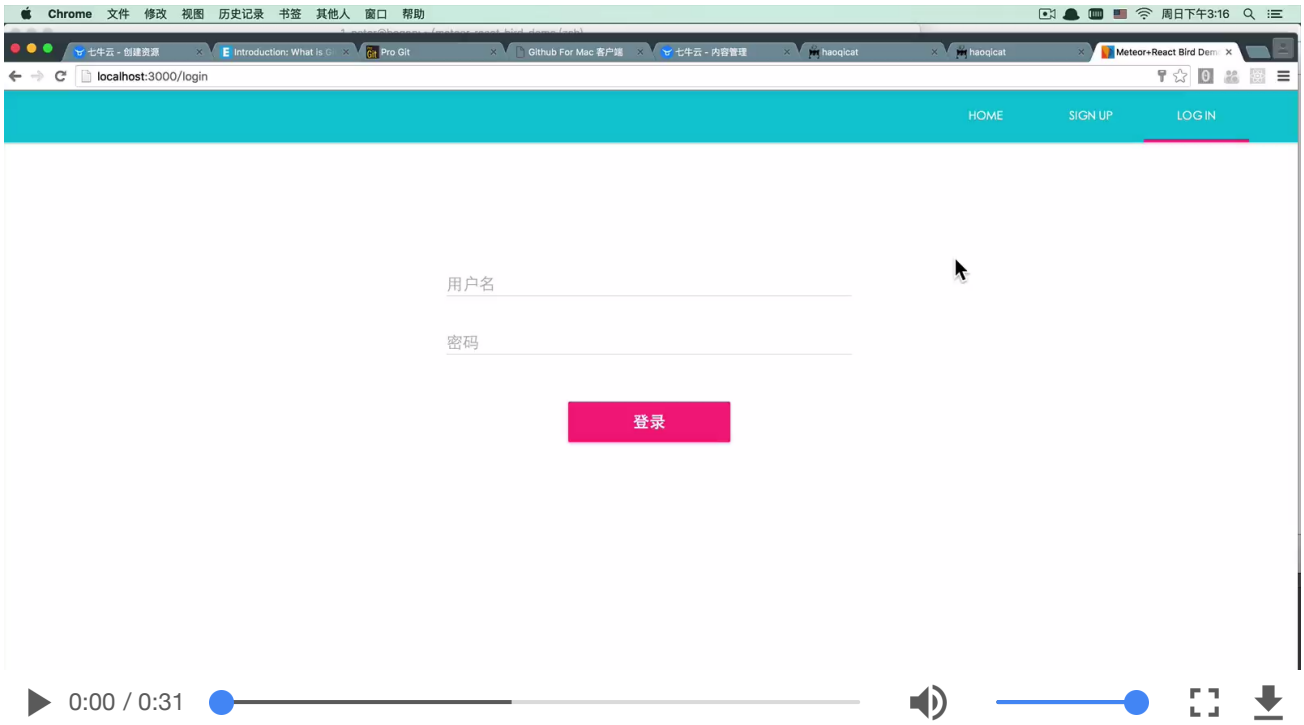

登录和登出

继续完成用户的账号管理功能，前面已经把用户注册介绍完了，本节完成用户的登录和退出登录功能，当然过程中肯定会用到 Meteor 的账号管里相关的接口，但那不是重点，重点是如何让 React 组件状态随着 Meteor 动态数据的变化而改变，这就话很难理解，后面我们会认真的来演示这一点。

我们首先瞄准组件随动的问题，现在还没有登录功能，所以再新注册一个用户，登录进来。打开浏览器 console，执行用户退出登录操作：

```
Meteor.logout()
```

执行这个操作之后，其实 Meteor 的数据已经变了，比如查看当前用户的 id，使用这个接口：

```
Meteor.userId()
```

返回值为空 `null`，但是 React 组件 NavBar 的状态并没有随着 Meteor 动态数据的变化而改变，也就是说 React 组件 NavBar 是不随动的，不 Reactive 的。在 `NavBar.jsx` 文件中，我们看到：

```
let currentUser = Meteor.userId();
```

虽然 `Meteor.userId()` 这个可变的 Meteor 数据已经赋值给了 `currentUser` 这个变量，我们希望通过 `Meteor.userId()` 的变化改变 Tab 组件的状态。但是，这会儿 Tab 组件肯定是不会自动渲染的，因为让一个 React 组件基本有两种方式：一种是在组件内部直接改变它的 state 状态值；另外一种方式就是我们使用这个组件的时候，若传给它的 prop 属性值发生了变化，这个组件也会重新渲染一下。我们这里就通过传递属性的方式改变组件状态，在 NavBar 的父组件 App 组件中给 NavBar 组件添加一个属性 `currentUser`，设置如下：

```
<NavBar currentUser={ Meteor.userId() } />
```

然后，到 `NavBar.jsx` 文件中修改代码，重新定义 `currentUser` 这个变量：

```
let currentUser = this.props.currentUser;
```

那这样，我们的 NavBar 组件的状态就可以随着 `Meteor.userId()` 的值变化了吗？答案是不可以。这就要涉及到 Meteor 的一个重要知识点了，如何在 React 组件之中使用

随动的 Meteor 数据。

React 组件中使用 Meteor 的随动数据

在 React 组件中正确的使用 Meteor 随动数据，首先要安装一个 npm 包：

```
npm install --save react-addons-pure-render-mixin@15.0.1
```

注意：目前 react-addons-pure-render-mixin 包的最新版本是 `15.0.2`，会与其它已经安装的 node 包所需要的 React 版本产生冲突，所以指定了安装的版本号 `15.0.1`。

然后，再安装一个 Atmosphere 包 react-meteor-data，它依赖刚才先安装的 node 包。

```
meteor add react-meteor-data
```

这个 Atmosphere 包提供了一个 `createContainer` 方法，通过这个方法可以获取 Meteor 的随动数据，并通过 React props 把获取的数据传递给 React 组件。以下面的代码为例：

```
createContainer(() => {
  return {
    currentUser: Meteor.userId(),
  };
}, App);
```

把 `Meteor.userId()` 接口返回的数据通过 React 属性 `currentUser` 传递给 App 组件。在 App 组件中则可以通过 `this.props.currentUser` 语句得到 `Meteor.userId()` 的数据。这样当 Meteor 的随动数据发生改变的时候，与之对应的 React 属性就会改变，从而包含这个属性的 React 组件就会重新渲染了。请参考文档使用 createContainer。

另外，别忘了在 `App.jsx` 文件开头导入需要的功能模块：

```
import { Meteor } from 'meteor/meteor';
import { createContainer } from 'meteor/react-meteor-data';
```

查看代码更改：React 组件中使用 Meteor 的随动数据

所有修改都完成之后，可以到浏览器中测验一下，重新注册一个用户，注册成功之后，在浏览器的 console 中执行：

```
Meteor.userId()
```

返回当前登录用户的 id，说明用户登录进来了。然后在浏览器 console 中执行语句：

```
Meteor.logout()
```

退出登录。这时会发现导航栏中的 `ACCOUNT` 标签，变成了 `SINGU UP` 标签，说明 React 组件状态果然随着 Meteor 动态数据的变化而改变了。

创建登录表单

修改 `imports/ui/LogIn.jsx` 文件，添加登录表单的代码。

查看更改：创建登录表单

没有新的知识点，和注册表单长的一模一样。

实现用户登录功能

继续修改 `imports/ui/LogIn.jsx` 文件，添加一些代码进来。

查看更改：实现用户登录功能

这里，我们用到了由 accounts-password 包提供的另一个函数接口 Meteor.loginwithpassword，这个接口的第一个参数是只有一个键值对的对象类型，对象的 key 可以是 username、email 或者是 id，我们采取用户名和密码登录，所以用 `{username: userName}` 键值对。

另外，当用户登录成功之后会跳转到 `/chat` 路径，所以使用了 `react-router` 提供的 `context.router` 对象。因为 `/chat` 路径对应的 Chat 组件还没有创建，所以又新建

了一个 imports/ui/Chat.jsx 文件，并添加了 Chat 组件对应的路由。

```
<Route path="/chat" component={Chat} />
```

同时，还修改了 NavBar.jsx 文件，当用户登录成功后，用 chat 标签替换 log in 标签，实现方式与前面讲到的 account 标签替换 sign up 标签的方式相同。

至此，用户登录功能就完成了。

重新组织用户认证相关的文件结构

在继续操作之前，我们稍微调整一下文件结构，把涉及到用户认证功能的代码 SignUp.jsx 文件和 LogIn.jsx 文件都放到 imports/ui/auth 目录中，相应的还要调整一些文件的导入路径问题。

查看更改：重新组织用户认证相关的文件结构

实现用户退出登录功能

用户登录已经实现了，接下来添加退出登录功能。

新建一个 LogOutMenu 组件，位于 imports/ui/auth/LogOutMenu.jsx 文件中，它将负责完成用户退出登录功能。在 LogOutMenu 组件中，仍然都是 Material-UI 组件库的基本使用，唯一要解释的是 ActionAccountCircle 这个小组件，涉及到的参考文档就是 Material-UI 的 SVG 图标 这一部分内容。Material-UI 通过 SvgIcon 组件的形式包含了谷歌所有的材料图标， ActionAccountCircle 就是一个预先定义好的 SvgIcon 组件， SvgIcon 组件的命名方式根据谷歌材料图标的类别和图标名称组合而成，所以从 ActionAccountCircle 这个组件名可以知道，它对应谷歌材料图标 Action 类中的 accout circle 图标，一个圆形的小人头像图标。

当点击这个小人头像的时候，会弹出一个快捷菜单，菜单中有两个条目，一个显示当前用户的用户名，用户名通过给 LogOutMenu 组件传递的 username 属性获得。另一个显示 退出 字样，点击 退出 ，会触发 MenuItem 的 handleTouchTap 事件处理器，从而执行语句：

```
Meteor.logout();
this.context.router.push('/');
```

之前在浏览器 console 中，我们已经使用了多次 `Meteor.logout()` 语句让用户退出登录，这次真正在代码中使用起来了。用户退出登录后，当前页面切换到首页，渲染 Home 组件。

然后，到 `NavBar.jsx` 文件中使用组件 LogOutMenu，不过这个组件的显示是有条件的，代码如下：

```
{ currentUser ? <LogOutMenu username={} /> : '' }
```

当用户处于登录状态的时候，显示 LogOutMenu 组件，要不然就什么也不显示。不过，现在的 LogOutMenu 组件的 `username` 属性值为空。这里，我们要用到 Meteor 的另一个随动数据 `Meteor.user()`，到浏览器 console 中，执行：

```
Meteor.user()
```

若有用户登录的时候，上述语句会返回一个对象，包含当前用户的用户名。若没有用户登录，返回值为 `null`。因此，我们可以通过 `Meteor.user()` 得到当前用户的用户名，仍然按照在 React 组件中使用 Meteor 随动数据的方式获取 `Meteor.user()` 的数值。修改 `App.jsx` 文件，在 `createContainer` 方法中添加一行代码：

```
userInfo: Meteor.user()
```

然后，把 `userInfo` 属性值传递给 NavBar 组件使用：

```
<NavBar userInfo={this.props.userInfo} />
```

在 NavBar 组件中，当传入的 `userInfo` 属性值不为空的时候，获得当前用户的用户名，并传递给 LogOutMenu 组件使用：

```
<LogOutMenu username={this.props.userInfo ? this.props.userInfo.usernam
```

这样，在 LogOutMenu 组件中就可以显示当前用户的用户名了。到浏览器中测试一下，用已经注册的用户登录进来，导航栏右边出现一个小人头像图标，点击图标，弹出一个快捷菜单，点击菜单中的 `退出` 选项，当前页面切换到首页，用户退出登录。

查看更改：实现用户退出登录功能

从小屏幕设备上退出登录

因为我们的导航栏是响应式的，在小屏幕设备上显示的是 AppBar 组件，所以我们也应该在 AppBar 组件上添加 LogOutMenu 组件。

```
this.props.currentUser ? this.getLoginAppBar() : this.getAppBar())
```

若用户登录，则渲染带有 LogOutMenu 组件的 AppBar 组件；若用户没有登录，则渲染普通的 AppBar 组件。

另外，还得调整一下 AppDrawer 组件的显示状态，用户登录之后，显示的条目应该变成 `Home`、`Account` 和 `Chat` 三项。调整方式与 NavBar 组件标签的调整方式一样。

```
<ListItem
  value={ currentUser ? '/account' : '/signup' }
  primaryText={ currentUser ? 'Account' : 'Sign up' } />
```

查看更改：从小屏幕设备上退出登录

到此为止，用户的账号管理功能就完工。