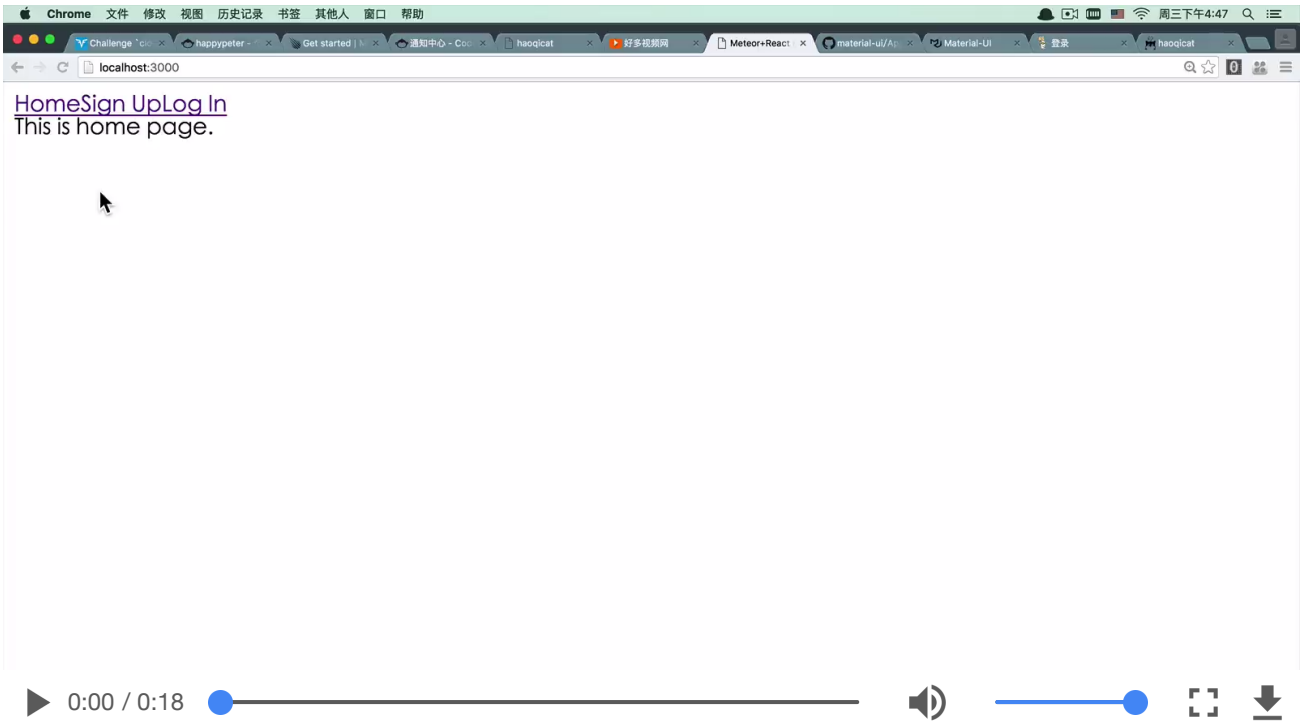

嵌套路由

本节将把各个组件中重复使用的代码抽取成一个导航栏组件，使用 React Router 的嵌套路由功能重构代码，从而可以复用这个导航栏组件。

构建 NavBar 组件

现在 SignUp、LogIn 和 Home 组件中有一段相同的代码。接下来我们就把这段共有代码抽成一个新的组件 NavBar，新建文件 `imports/ui/shared/NavBar.jsx`：

```
import React, { Component } from 'react';
import { Link } from 'react-router';

class NavBar extends Component {
  render() {
    return (
      <div>
        <Link to='/'>Home</Link>
        <Link to='/signup'>Sign Up</Link>
        <Link to='/login'>Log In</Link>
      </div>
    );
  }
}

export default NavBar;
```

这时就可以把 SignUp、LogIn 和 Home 组件中的共用代码删掉了。

查看更改：构建 NavBar 组件

定义布局组件

新建文件 `imports/ui/App.jsx`，添加如下代码：

```
import React, { Component } from 'react';

import NavBar from './shared/NavBar.jsx';

class App extends Component {
  render() {
    return (
      <div>
        <NavBar />
        { this.props.children }
      </div>
```

```
    );
  }
}

export default App;
```

查看更改： 定义 App 布局组件

只要把页面之间共享的导航栏写到布局组件中，就不必在各个页面组件中单独添加了，减少代码冗余。另外，布局组件 App 后续会作为嵌套路由中的父组件，它将包裹其它子组件，`{ this.props.children }` 未来会替换为各个具体的子组件。

编写嵌套路由

修改 `imports/startup/client/routes.jsx` 文件如下：

```
import React from 'react';
import { Router, Route, IndexRoute, browserHistory } from 'react-router

...
import App from '../../ui/App.jsx';

export const renderRoutes = () => (
  <Router history={browserHistory}>
    <Route path='/' component={App}>
      <IndexRoute component={Home} />
      <Route path="/signup" component={SignUp} />
      <Route path="/login" component={LogIn} />
    </Route>
  </Router>
);
```

查看更改：编写嵌套路由

这样就实现了一个嵌套路由，App 布局组件设置为父路由组件，其它子路由对应要嵌入到 App 中的各个组件。这样，路由嵌套和组件嵌套的映射关系是非常明确的。同时通过 IndexRoute 把 Home 组件设置为一个默认组件，也就是访问顶级路由 `/` 的时候要显示的组件。

总结

学会了嵌套路由，那么 react-router 的基本使用我们也就掌握了。