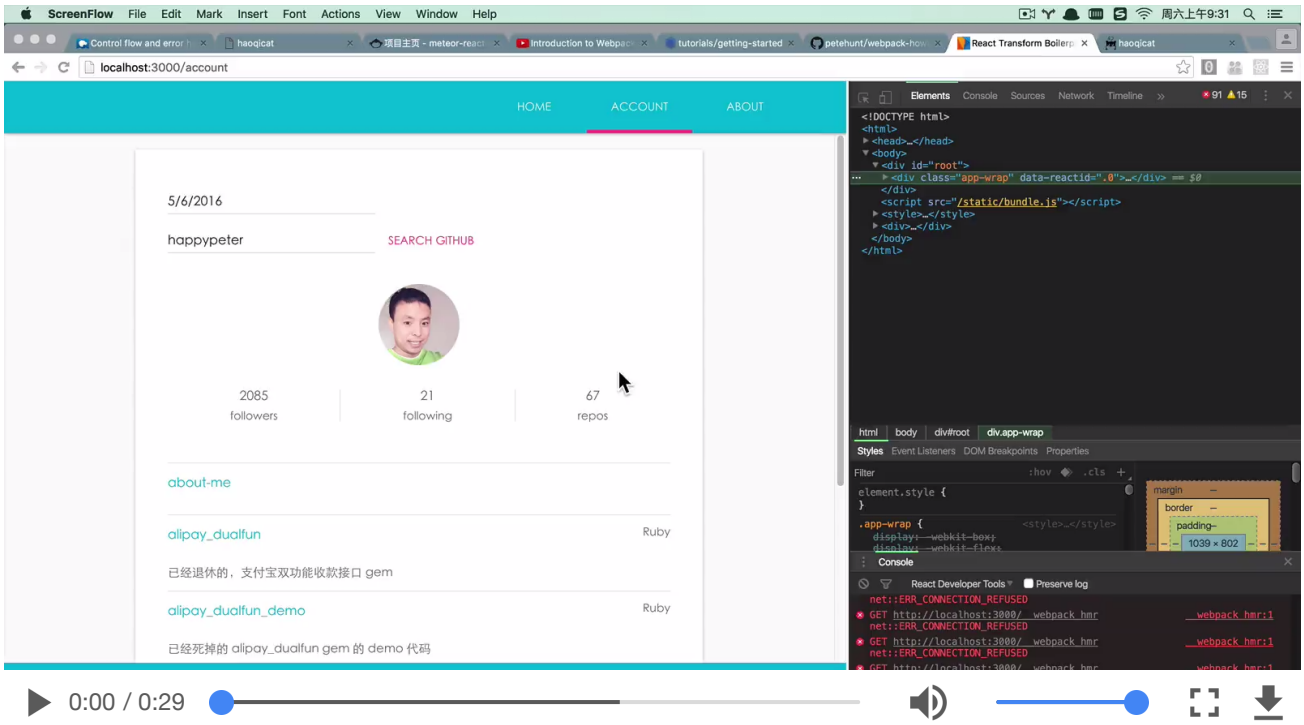

0:00 / 0:29

使用 material-ui 的 Tabs 组件

使用 react-router 实现“多页面”的单页面应用

react 的 State 以及 Lifecycle Methods

react 下实现 media query

添加更多组件进来

第三章 Ajax 请求第三方 API

部署项目到服务器

本地开发环境下的操作

在第二阶段的 demo 目录下，执行命令 `npm run build` 的时候，读取的是 `webpack.config.prod.js` 配置文件，所以我们要对这个文件做些修改：

```
var path = require('path');
var webpack = require('webpack');

module.exports = {
  devtool: 'source-map',
  entry: [
    './src/index'
  ],
  output: {
    path: path.join(__dirname, 'dist'),
    filename: 'bundle.js',
    publicPath: '/dist/'
  },
  plugins: [
    new webpack.optimize.OccurenceOrderPlugin(),
    new webpack.DefinePlugin({
      'process.env': {
        'NODE_ENV': JSON.stringify('production')
      }
    }),
    new webpack.optimize.UglifyJsPlugin({
      compressor: {
        warnings: false
      }
    })
  ],
  module: {
    loaders: [{
      test: /\.js$/,
      loaders: ['babel'],
      include: path.join(__dirname, 'src')
    },
    {
      test: /\.scss$/,
      loader: 'style!css!autoprefixer!sass'
    },
    {
      test: /\.(jpe?g|png)$/,
      loader: 'file-loader'
    }]
```

```
  }
};
```

目的是为了添加项目所需要的各种 loader。同时，把编译生成的文件存放到 dist 目录中。

修改 index.html 文件

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title>React Transform Boilerplate</title>
    <meta name="viewport" content="width=device-width, initial-scale=1"
  </head>
  <body>
    <div id="root"></div>
    <script src="/dist/bundle.js"></script>
  </body>
</html>
```

注意：生产环境下，bundle.js 文件存放在 dist 目录下。

配置 express 服务

在项目根目录下，新建一个名为 server.js 的文件，内容如下：

```
var express = require('express');
var path = require('path');
var webpack = require('webpack');
var app = express();

var static_path = path.join(__dirname, 'public');

app.use(express.static(static_path))
  .get('*', function (req, res) {
    res.sendFile('index.html', {
      root: static_path
```

```
  });
}).listen(process.env.PORT || 8080, function (err) {
  if (err) { console.log(err) };
  console.log('Listening at localhost:8080');
});
```

把本地做的修改都 push 到 GitHub 项目仓库中，以备后用。

服务器上的操作

把项目部署到阿里云上，环境是 ubuntu + nginx + express。

把代码 clone 到服务器上，运行命令：

```
git clone git@github.com:your_github_account/react-transform-boilerpla
```

接下来，执行以下命令，安装所需要的 npm 模块，以及编译生成在生产环境用到的文件：

```
npm install
npm run build
```

命令执行完毕，会在当前目录下生成一个 dist 目录，目录中有 bundle.js 文件，还有图片。

另外，在项目根目录下创建一个 public 目录，把 index.html 文件 和 dist 目录移动到 public 目录下面。然后执行命令：

```
node server.js
```

项目跑起来之后，访问 your_domain_name:8080 地址，就能看到应用跑起来了。

配置 nginx

虽然应用跑起来了，但是我们一关闭窗口应用就终止了，那如何让其运行在后台，并且有个美观的域名，不用在网址中输入端口号呢？

要想让应用在服务器后台执行，可以使用 tmux 来实现。先终止应用运行，在项目根目录下执行命令：

```
tmux
```

这样会打开一个 tmux 命令行窗口，在此窗口中执行命令

```
node server.js
```

这样启动的应用，即使本地与服务器断开连接之后，其仍然在运行。

接下来，就是配置 nginx web 服务器了，到 `/etc/nginx/sites-enabled` 目录下，添加一个新文件，比方说 `app.conf`，内容如下：

```
server {
    listen          80;
    server_name your_server_domain_name;

    location / {
        proxy_pass http://localhost:8080;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $http_x_forwarded_host;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_read_timeout 3m;
        proxy_send_timeout 3m;
    }
}
```

这里，`server_name` 就是域名，比方说 `s2.haoduoshipin.com`。为了让配置文件生效，执行命令：

```
sudo service nginx reload
```

到此，项目就算部署成功了！