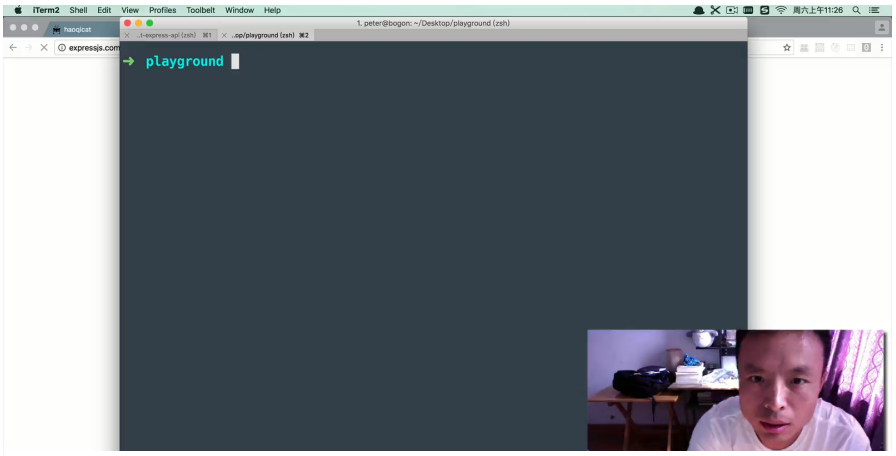

00:00 -07:09

创建一个简单的 Express 应用

假定你已经在系统中安装了 Node.js (>=5.4.1)

创建应用目录

首先创建课程案例目录 `react-express-api-demo`，然后在案例目录中新建 `server` 目录，存放实现后端 API 服务的代码，

```
mkdir react-express-api-demo && cd react-expı
```

接下来，生成应用需要的 `package.json` 文件：

```
npm init -y
```

参考 npm init 命令的用法

安装 express 包

```
npm install --save express
```

- Express 是一个轻便灵活的 Node.js 应用开发框架，本应用将构建一个提供 API 服务的 Express 应用

添加应用入口文件

新建文件 `index.js`

```
touch index.js
```

然后打开 `index.js` 文件，先添加下面两行代码：

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

上面两行代码的功能是：导入 `express` 功能模块，创建一个 `Express` 应用实例名为 `app`

继续添加代码：

```
app.get('/', function(req, res) {
  res.send('Hello world!');
})
```

app.get() 接口会响应 HTTP GET 请求，当访问路径与 `/` 相匹配的时候，则执行上述代码，并通过 res.send() 接口向客户端发送 `Hello world!` 字符串。

```
app.listen(3000, function() {
  console.log('Your server is running on port
});
```

app.listen() 方法会创建一个 HTTP server 实例，用来监听来自本地3000端口的所有请求。

保存 `index.js` 文件并退出，在命令行中运行命令：

```
node index.js
```

让刚创建好的 Express 应用运行起来。打开 Chrome 浏览器，在地址栏中输入网址 `http://localhost:3000`，网页中会显示 `Hello World!` 字样。

至此一个最简单的 Express 应用就搭建好了，在后面的课程中，我们会逐步完善它的功能，让它提供 API 服务。

使用 nodemon 提高开发效率

这时候，你可以试着修改一下代码，比如把 `Hello World` 改成 `Hello Wild World`，然后刷新页面，会

发现没有变化。解决这个问题，需要先关闭刚才已经启动的应用，然后再运行命令：

```
node index.js
```

让应用重新启动之后，我们所做的修改才能生效。若在应用开发过程中每次修改代码都要重启应用，那就太不方便了！莫担忧，可以借助工具 nodemon 排除烦恼。这个 nodemon 工具可以助力 node 应用的开发效率，因为它能监测 node 应用目录中的各个文件，若文件有改动，nodemon 会自动重启你的 node 应用，再也不用手动重启了。

```
npm install -g nodemon
```

上面命令中用到了 `-g` 选项，说明要全局安装 nodemon 包，这样新创建的 node.js 应用都能使用 nodemon 运行起来了。

尽然 nodemon 已经安装好了，那如何使用呢？非常简单，在应用根目录下，先终止运行 `node index.js` 命令，然后在命令行中输入一个新的命令：

```
nodemon index.js
```

通过 nodemon 命令启动应用之后，应用中的各个文

件就被 nodemon 监测了。即使应用中要安装新的 npm 包，nodemon 也会重启应用。不妨试着修改一下 `index.js` 文件，看一下效果吧。