Node 调试工具入门教程

作者： 阮一峰

日期： 2018年3月20日

JavaScript 程序越来越复杂，调试工具的重要性日益凸显。客户端脚本有浏览器，Node 脚本怎么调试呢？

2016年，Node 决定将 Chrome 浏览器的"开发者工具"作为官方的调试工具，使得 Node 脚本也可以使用图形界面调试，这大大方便了开发者。

本文介绍如何使用 Node 脚本的调试工具。

一、示例程序

为了方便讲解，下面是一个示例脚本。首先，新建一个工作目录，并进入该目录。

```
$ mkdir debug-demo
$ cd debug-demo
```

然后，生成 package.json 文件，并安装 Koa 框架和 koa-route 模块。

```
$ npm init -y
$ npm install --save koa koa-route
```

接着，新建一个脚本 app.js ，并写入下面的内容。

```
// app.js
const Koa = require('koa');
```

```
const router = require('koa-route');

const app = new Koa();

const main = ctx => {
  ctx.response.body = 'Hello World';
};

const welcome = (ctx, name) => {
  ctx.response.body = 'Hello ' + name;
};

app.use(router.get('/', main));
app.use(router.get('/:name', welcome));

app.listen(3000);
console.log('listening on port 3000');
```

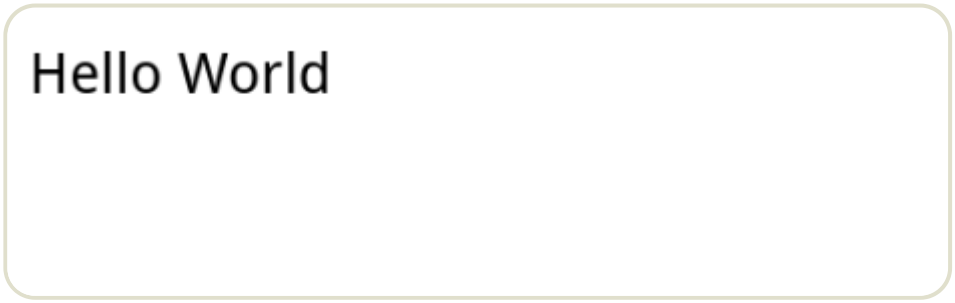
上面代码是一个简单的 Web 应用，指定了两个路由，访问后会显示一行欢迎信息。如果想了解代码的详细含义，可以参考 Koa 教程。

二、启动开发者工具

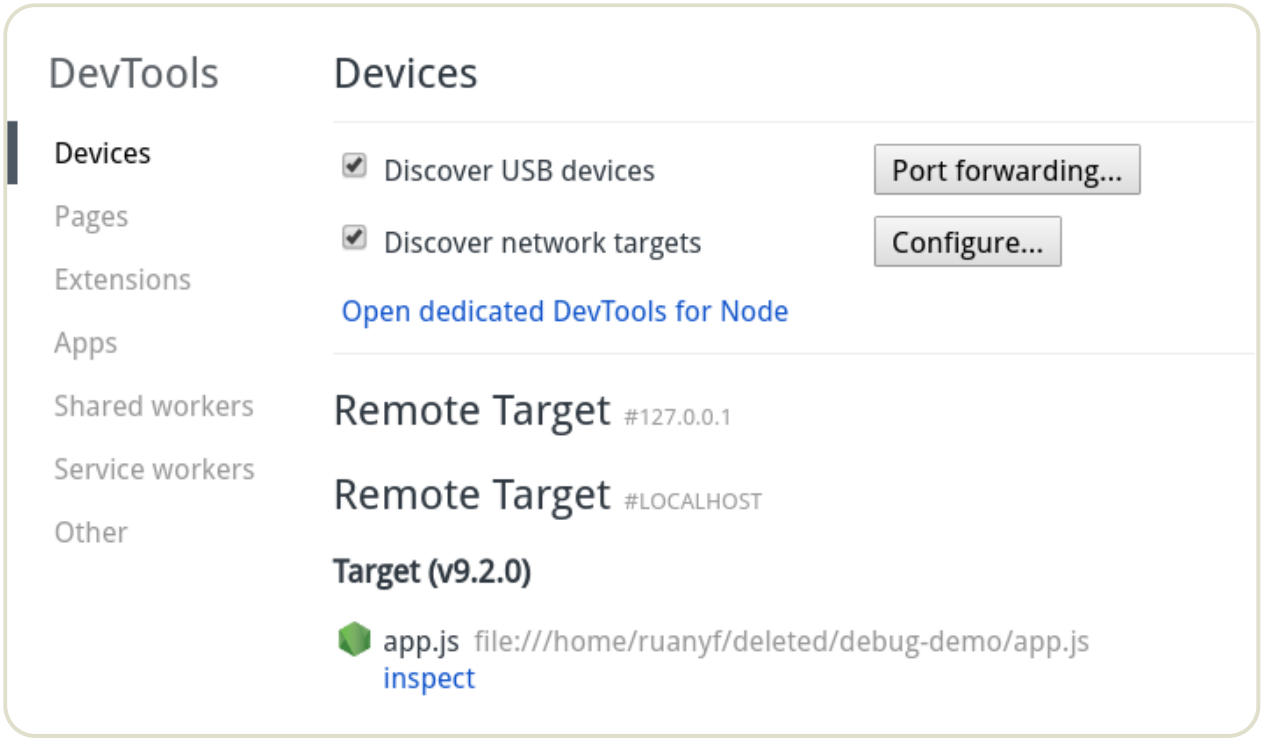
现在，运行上面的脚本。

```
$ node --inspect app.js
```

上面代码中， `--inspect` 参数是启动调试模式必需的。这时，打开浏览器访问 `http://127.0.0.1:3000` ，就可以看到 Hello World 了。

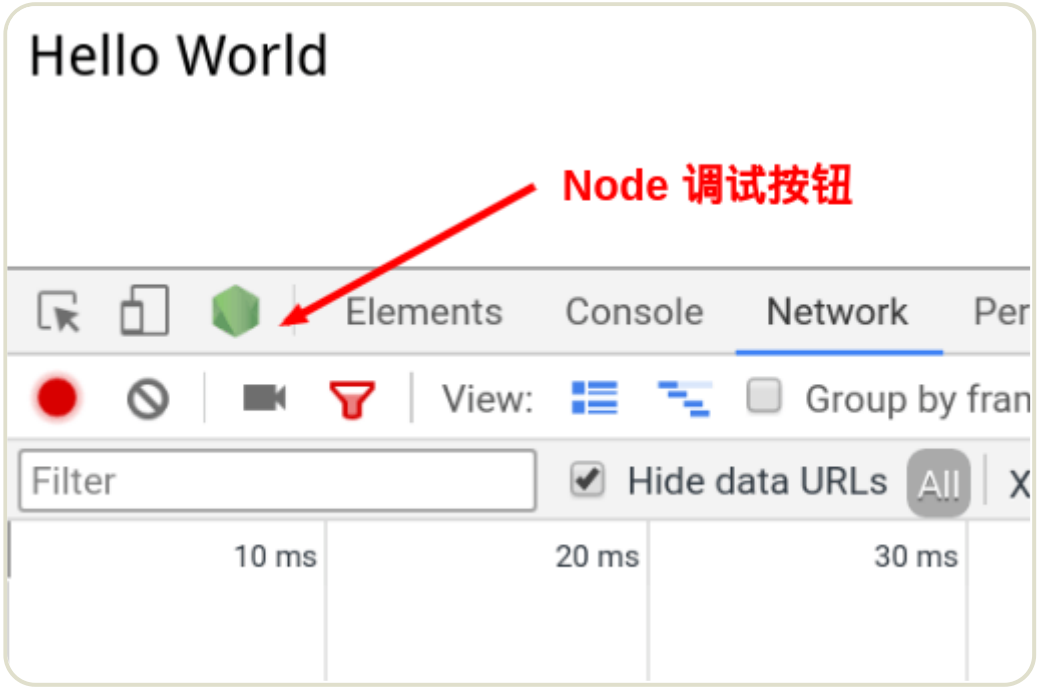


接下来，就要开始调试了。一共有两种打开调试工具的方法，第一种是在 Chrome 浏览器的地址栏，键入 `chrome://inspect` 或者 `about:inspect` ，回车后就可以看到下面的界面。



在 Target 部分，点击 inspect 链接，就能进入调试工具了。

第二种进入调试工具的方法，是在 http://127.0.0.1:3000 的窗口打开"开发者工具"，顶部左上角有一个 Node 的绿色标志，点击就可以进入。

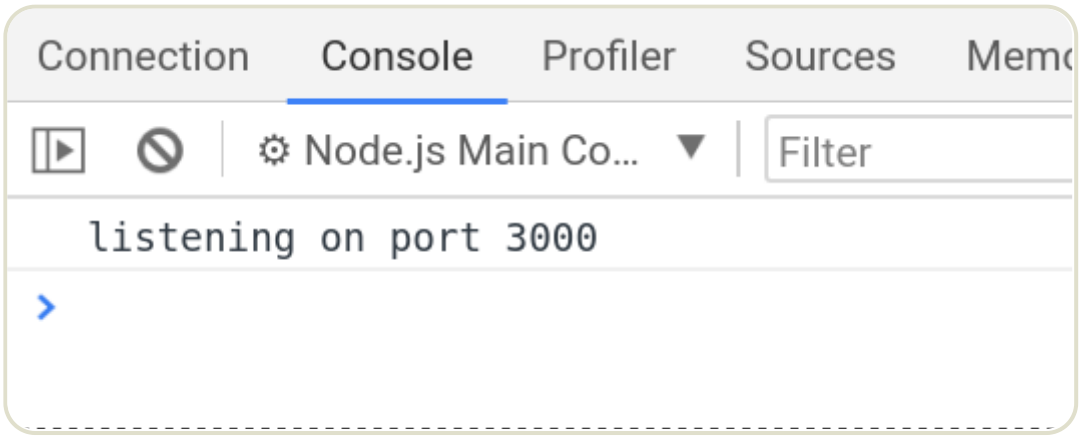


三、调试工具窗口

调试工具其实就是"开发者工具"的定制版，省去了那些对服务器脚本没用的部分。

它主要有四个面板。

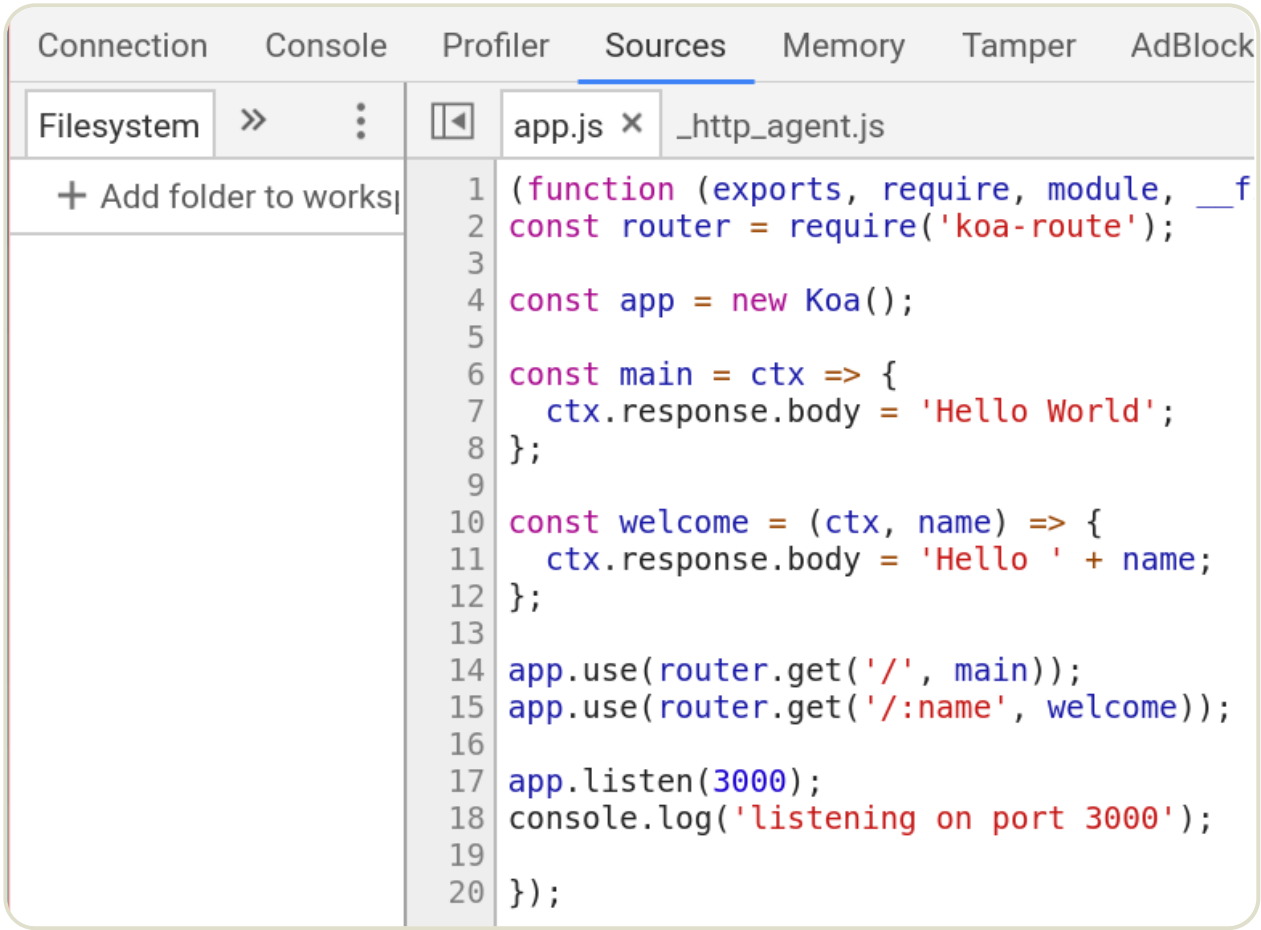
- Console：控制台
- Memory：内存
- Profiler：性能
- Sources：源码



这些面板的用法，基本上跟浏览器环境差不多，这里只介绍 Sources （源码）面板。

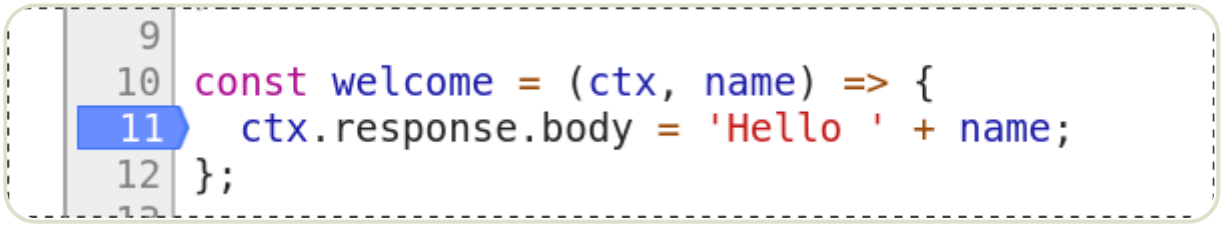
四、设置断点

进入 Sources 面板，找到正在运行的脚本 `app.js` 。

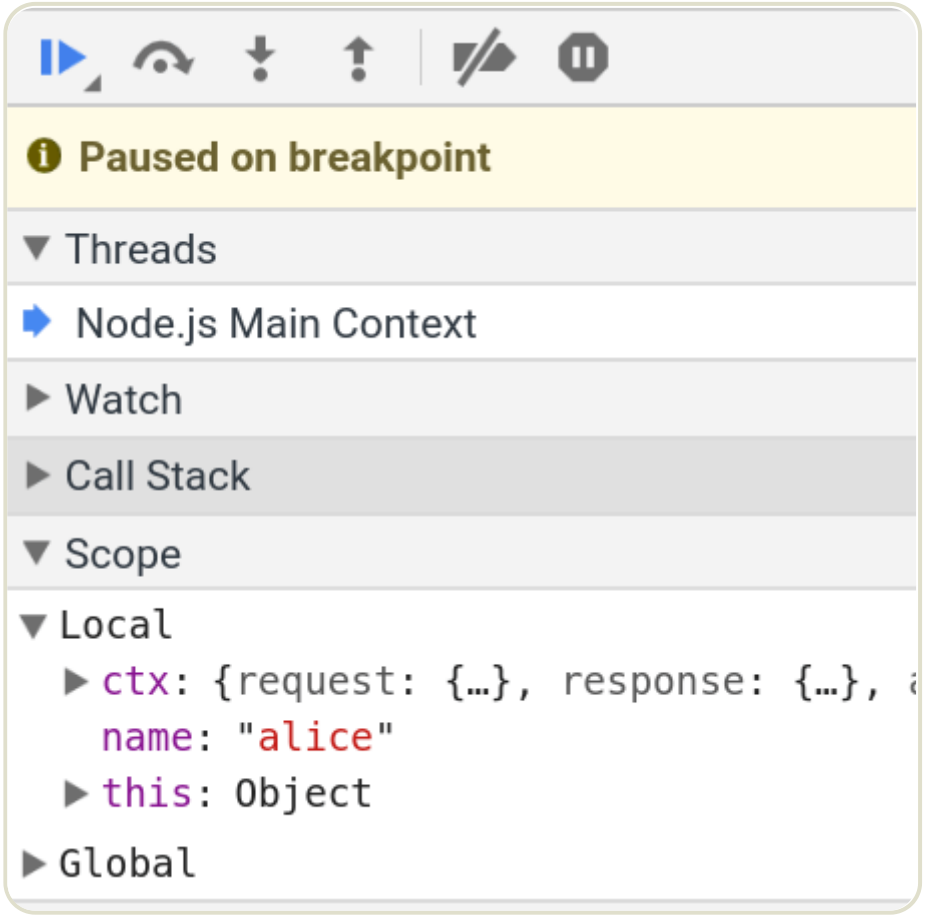


在第11行（也就是下面这一行）的行号上点一下，就设置了一个断点。

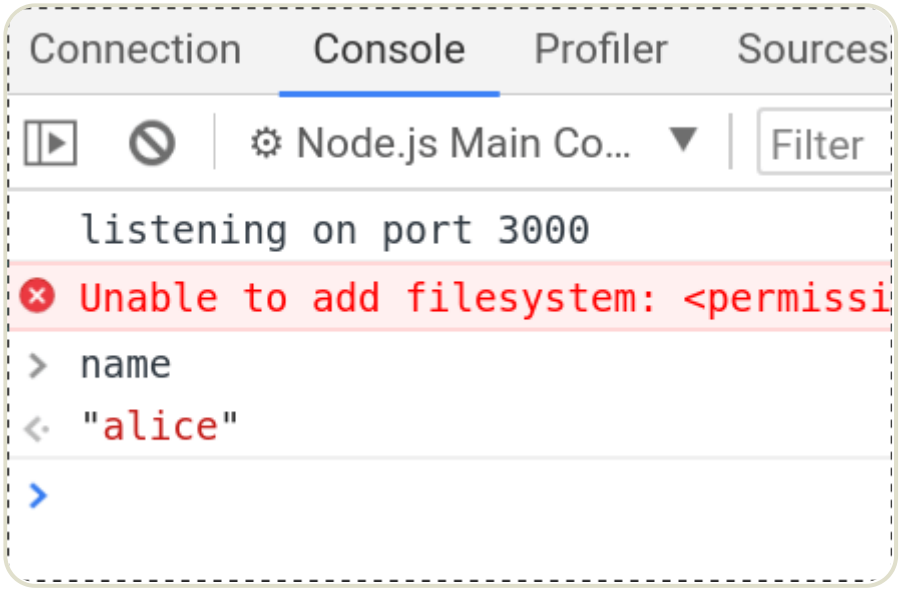
```
ctx.response.body = 'Hello ' + name;
```



这时，浏览器访问 http://127.0.0.1:3000/alice ，页面会显示正在等待服务器返回。切换到调试工具，可以看到 Node 主线程处于暂停（paused）阶段。



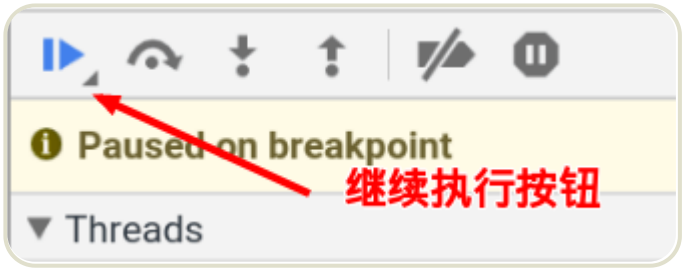
进入 Console 面板，输入 name，会返回 alice。这表明我们正处在断点处的上下文（context）。



再切回 Sources 面板，右侧可以看到 Watch、Call Stack、Scope、Breakpoints 等折叠项。打开 Scope 折叠项，可以看到 Local 作用域和 Global 作用域里面的所有变量。

Local 作用域里面，变量 `name` 的值是 `alice` ，双击进入编辑状态，把它改成 `bob` 。

然后，点击顶部工具栏的继续运行按钮。



页面上就可以看到 Hello bob 了。

命令行下，按下 ctrl + c，终止运行 `app.js` 。

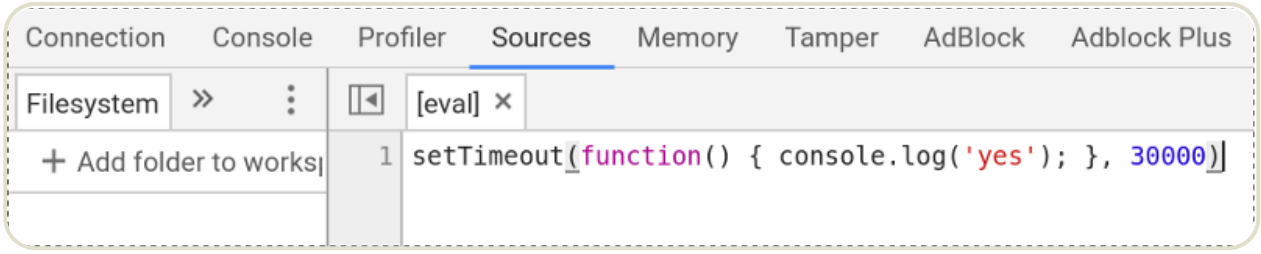
五、调试非服务脚本

Web 服务脚本会一直在后台运行，但是大部分脚本只是处理某个任务，运行完就会终止。这时，你可能根本没有时间打开调试工具。等你打开了，脚本早就结束运行了。这时怎么调试呢？

```
$ node --inspect=9229 -e "setTimeout(function() { console.log('yes'); }, 30000)"
```

上面代码中， `--inspect=9229` 指定调试端口为 9229，这是调试工具默认的通信端口。 `-e` 参数指定一个字符串，作为代码运行。

访问 `chrome://inspect` ，就可以进入调试工具，调试这段代码了。



代码放在 `setTimeout` 里面，总是不太方便。那些运行时间较短的脚本，可能根本来不及打开调试工具。这时就要使用下面的方法。

```
$ node --inspect-brk=9229 app.js
```

上面代码中， `--inspect-brk` 指定在第一行就设置断点。也就是说，一开始运行，就是暂停的状态。

六、忘了写 --inspect 怎么办？

打开调试工具的前提是，启动 Node 脚本时就加上 `--inspect` 参数。如果忘了这个参数，还能不能调试呢？

回答是可以的。首先，正常启动脚本。

```
$ node app.js
```

然后，在另一个命令行窗口，查找上面脚本的进程号。

```
$ ps ax | grep app.js

30464 pts/11   Sl+    0:00 node app.js
30541 pts/12   S+     0:00 grep app.js
```

上面命令中， app.js 的进程号是 30464 。

接着，运行下面的命令。

```
$ node -e 'process._debugProcess(30464)'
```

上面命令会建立进程 30464 与调试工具的连接，然后就可以打开调试工具了。

还有一种方法，就是向脚本进程发送 SIGUSR1 信号，也可以建立调试连接。

```
$ kill -SIGUSR1 30464
```

七、参考链接

- Debugging Node.js with Google Chrome, by Jacopo Daeli
- Debugging Node.js with Chrome DevTools, by Paul Irish
- Last minute node debugging, by Remy Sharp

（完）

文档信息

- 版权声明：自由转载-非商用-非衍生-保持署名（创意共享3.0许可证）
- 发表日期： 2018年3月20日

相关文章

- **2021.09.29:** JavaScript 侦测手机浏览器的五种方法

  > 有时候，前端网页需要知道，用户使用的是手机浏览器还是桌面浏览器。

- **2021.01.20:** 剪贴板操作 Clipboard API 教程

> 一、简介 浏览器允许 JavaScript 脚本读写剪贴板，自动复制或粘贴内容。

- **2020.12.28:** Fetch API 教程

> fetch()是 XMLHttpRequest 的升级版，用于在 JavaScript 脚本里面发出 HTTP 请求。

- **2020.09.15:** 轻松学会 React 钩子：以 useEffect() 为例

> 五年多前，我写过 React 系列教程。不用说，内容已经有些过时了。

Weibo | Twitter | GitHub

Email: yifeng.ruan@gmail.com