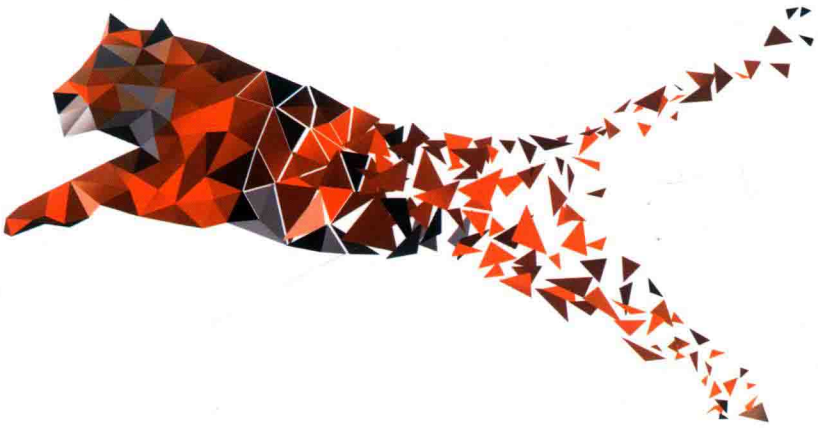
HZ BOOKS
华章IT

基于Angular 2，深入讲解基本概念的背后原理，以及众多优秀的设计模式和编程范式。
摆脱复杂配置，带你从无到有地搭建前端应用，讲解开门见山，语言风趣幽默。

實戰



王芃 编著

Angular From Zero to One

Angular 从零到一

机械工业出版社
China Machine Press

實戰

Angular From Zero to One

Angular 从零到一

王芃 编著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Angular 从零到一 / 王芃编著 . —北京：机械工业出版社，2017.3
（实战）

ISBN 978-7-111-56283-2

I. A… II. 王… III. 超文本标记语言 – 程序设计 IV. TP312.8

中国版本图书馆 CIP 数据核字（2017）第 042166 号

Angular 从零到一

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：吴 怡 责任校对：李秋荣
印 刷：三河市宏图印务有限公司 版 次：2017 年 3 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：16.75
书 号：ISBN 978-7-111-56283-2 定 价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

Preface 前　言

一个大叔码农的 Angular 2 创世纪

作为一个出生于 20 世纪 70 年代的大叔，我在软件这个领域已经摸爬滚打了 16 年，从程序员、项目经理、产品经理，项目总监，到部门管理等各个角色都体验过，深深地了解到这个行业发展的速度之快是其他行业无法比拟的：从编程语言、各种平台、各种框架、设计模式到各类开源工具、组件林林总总，要学习的东西实在太多，因为变化太快。

但万变不离其宗，名词变化虽多，透射的本质其实是趋同的：那就是程序员受不了代码的折磨，千方百计地想让这个工作更简单，更能应对变化。比如说，面向对象编程（Object-Oriented Programming）理念的提出其实是牺牲了部分性能换来代码层次结构的清晰，因此也催生了 C++、Java、C# 等一系列优秀的面向对象编程语言；后来程序员们发现在实际的编程逻辑中，往往不是像对象树那样可以划分得那么清楚。还有一些类似安全、日志等功能其实是撒在系统各个角落的，于是，面向切面的编程（Aspect-Oriented Programming）应运而生。再后来一部分科学家发现现有的编程语言做分析或数据计算实在太麻烦，明明要计算的逻辑很清晰，却要用一大堆的对象封装赋值，函数式编程（Functional Programming）便出现了。最近几年被产品经理逼疯的程序员认为强类型语言改动起来太慢太繁琐，于是动态脚本类语言大行其道。

仔细分析一下，这些语言不是互斥的，其实好的元素都是会被慢慢吸收到各自的语言、平台上面去的。比如 C#、Java 也采纳了函数式编程的一些特点，像 Lamda 表达式；再比如 .NET 和 Java 平台基础上也拥有动态脚本语言，像 .NET 平台上的 IronRuby，Java 平台上的 Scala 等。本书写的 Angular 2 就是在 JavaScript 这种脚本语言基础上引入了 TypeScript，进而可以兼具面向对象编程和强类型语言的优点；引入了依赖性注入（Dependency

Injection）这种在强类型语言中被证明非常有用的设计模式；通过引入 Rx，让 JavaScript 拥有了函数式编程的能力。

写这本书的起因很偶然。我们团队以 Android 和 iOS 开发人员为主，前端开发人员只有一个。但在开发过程中我们体会到原生 App 的开发迭代速度比较慢，因此希望以前端开发快速迭代，逻辑和界面摸清楚后再进行 App 开发。我们决定走前端路线后，就开始挑选前端框架，React、Vue 和 Angular 2 我们都尝试了，最终选择 Angular 2 是因为谷歌在 Angular 2 中把多年 Android 开发积累的优秀思想带入了 Angular，使得 Angular 的开发模式太像 App 开发了。有 App 开发经验或者 Java、.NET 开发经验的人可以非常舒服地切入进去。有了选择，我就开始边学习边给开发小伙伴做培训，培训资料也就当成网文发表出来。没想到在网上得到很多网友的支持和鼓励，觉得我边学边写时对一些问题的思考过程和改进过程对大家的学习也很有帮助。而我也在与大家的互动和分享中纠正了对一些概念和模式的认识。互动和分享是最好的学习方式，这也是本书区别于其他“专门教程”的重要一点，我们是一起在学习，一起在思考的。特别感谢简书和掘金等平台的读者，帮我纠正了很多错误认识和笔误等。机械工业出版社的吴怡编辑也正是在网上看到我的文章后，鼓励我结集出书，给我提了很多中肯意见，最终才有此书，非常感谢。

本书分为 9 章，第 1 ～ 7 章中我们从无到有地搭建了一个待办事项应用，但是我们增加了一些需求：多用户和 HTTP 后台。这样待办事项这个应用就变得麻雀虽小五脏俱全。通过这样一个应用的开发，我们熟悉了大部分重要的 Angular 2 概念和实践操作。建议读者按顺序阅读和实践。阅读完第 7 章，基本可以在正式的开发工作中上手了。第 8 章介绍了响应式编程的概念和 Rx 在 Angular 中的应用，可以说，如果不使用 Rx，Angular 2 的威力就折半了，很多原来需要复杂逻辑处理的地方用 Rx 解决起来非常方便。由于 Rx 本身的学习曲线较陡，我们花了很大篇幅做细致的讲解。第 9 章是在第 8 章基础之上，引入了在 React 中非常流行的 Redux 状态管理机制，这种机制的引入可以让代码和逻辑隔离得更好，在团队工作中强烈建议采用这种方案。第 8 章和第 9 章由于学习门槛较高，有的读者可能暂时接受起来有困难，遇到这种情况可以先放下，等到使用 Angular 一段时间后再回头来看。

大家在阅读过程中可能会发现从第 3 章开始起，我们在不断地打磨待办事项这个应用的逻辑，持续地优化。我写这本书其实不仅是为了让大家入门 Angular（类似的书太多了，不需要我再写一本），更多的是想把自己琢磨这些问题、解决这些问题的过程和逻辑与大家分享，把一些好的设计模式和思想介绍给大家，这些模式和思想远比一个框架更有生命力。

本书适合有面向对象编程基础的、掌握一门现代编程语言的读者阅读。如果有 Java、C#、Objective-C 等强类型语言背景，对于本书中介绍的 Angular 各种元数据修饰符接受程度会很高，对于 TypeScript 的类型等也会一点就透。如果有 JavaScript 背景，理解 TypeScript 语法是无障碍的，但强类型的约束和修饰符等概念需要仔细体会。如果使用过 Spring Framework 或者 Dagger2 等 IoC 框架，那么对依赖性注入的概念就再熟悉不过了。

建议学习的同时或之后可以比较一些其他主流前端框架，比如 React 或 Vue，参照后你会发现很多功能其实异曲同工。在读本书的过程中如果发现有错误，希望你可以在书籍源码的 Github 地址（https://github.com/wpcfan/awesome-tutorials）上提问题，我们一起打造一本一直在生长的书。希望年轻的你和大叔的我一起学习，一起面对这个迅速成长的行业！

王芃

2017 年 2 月 11 日

目　录 *Contents*

第 1 章 Chapter 1

认识 Angular

1.1 Angular 2 简介

Angular 2 是 Google 推出的一个跨平台全终端的 Web 前端框架，使用 Angular 2 可以快速开发出适合手机、平板以及 PC 端的前端网页应用。它让开发人员可以采用组件化的方式来编写应用，像 App 开发一样。借助来自 Ionic、NativeScript 和 React Native 中的技术与思想，我们可以使用 Angular 2 构建原生移动应用。

从性能角度来看，Angular 会把你的模板转换成代码，针对性地进行高度优化，轻松获得框架提供的高生产率，同时又能保留所有手写代码的优点。Angular 应用通过新的组件路由（Router）模块实现快速加载，提供了自动拆分代码的功能，为用户单独加载所请求视图中需要的那部分代码。

官方还提供了命令行工具（也就是本书采用的 Angular-CLI），屏蔽了大量配置细节，使得我们更专注于代码的开发，让开发人员快速进入构建环节、添加组件和测试，然后立即部署。

与目前比较火的 React 和 Vue.js 相比，Angular 有如下优点：

- 由于 Google 的目的是推出一个完整解决方案，所以官方默认提供的类库（比如 routing、HTTP、依赖性注入（DI）等）非常完整，无需自己选择。React 的一大痛点就是选择太多导致在配置寻找组件和类库的过程中消耗太多精力，当然从另

一方面看这也是其优势，选择众多且自由。

- 官方支持 TypeScript（微软出品，也是 JavaScript 的超集，也是 JavaScript 的强类型版本）作为首选编程语言，使得开发脚本语言的一些问题可以更早更方便地找到。
- RxJS 友好使得响应式编程在 Augular 2 中变得极为容易（Google 开发的框架依赖这么多微软的产品，可见微软的转型还是很成功的）。
- 支持 NativeScript 甚至 ReactNative 等进行原生 Android/iOS 应用开发（React 支持 React Native）。
- 支持服务器端渲染（React 也支持）。

总体来讲，个人认为 Angular 2 更适合从原生 App 开发或后端 Java/.Net 等转型过来开发前端的程序员，因为它的开发模型更接近于传统强类型语言的模式，加上官方内建的组件和类库比较完整，有官方中文网站 https://angular.cn，学习曲线要低一些。有过 Angular 1.x 开发经验的人要注意了，虽然只有一个版本号的差距，但 2.x 和 1.x 是完全不同的，不要奢望 1.x 的应用会平滑迁移到 2.x 上。

有趣的是，Angular 团队内部在准备下一个版本，目前准备叫做 Angular 4（只是计划哈），看起来太恐怖了，一下跳到 Angular 4，那 Angular 3 怎么办，我还要不要学 Angular 2。其实不用慌张，和前面的 1.x 到 2.x 的差距恰恰相反，这次只是版本号跳跃，框架却不会有太大差距，只是因为内部几个组件需要统一版本号，因此跳过了 Angular 3。所以呢，不要慌张，Angular 2 和 Angular 4 不会有本质的差别。

Angular 支持大多数常用浏览器，参见表 1.1。

表 1.1 Angular 2 的浏览器兼容性

Chrome	Firefox	Edge	IE	Safari	iOS	Android	IE Mobile
45 以上	40 以上	13 以上	9 以上	7 以上	7 以上	4.1 以上	11 以上

1.2 环境配置要求

Angular 2 需要 node.js 和 npm，我们下面的例子需要 node.js 6.x.x 和 npm 3.x.x，请使用 node -v 和 npm -v 来检查，Mac 下建议采用 brew 安装 node。由于众所周知的原因，http://npmjs.org 的站点访问经常不是很顺畅，这里给出一个由淘宝团队维护的国内镜像 http://npm.taobao.org/ 。安装好 node 后，请输入 npm config set registry https://registry.npm.taobao.org 来改变默认的 npm 查找包的站点，加快访问和下载速度。Mac 和 Linux

环境可能需要在我们提到的命令前加 sudo。

和官方快速起步文档给出的例子不同，我们下面要使用 Angular 团队目前正在开发中的一个工具 Angular CLI 。这是一个类似于 React CLI 和 Ember CLI 的命令行工具，用于快速构建 Angular 2 的应用。它的优点是进一步屏蔽了很多配置的步骤，自动按官方推荐的模式进行代码组织，自动生成组件 / 服务等模板以及更方便地发布和测试代码。由于目前这个工具还在 beta 阶段，安装时请使用 npm install -g angular-cli@latest 命令。

IDE 的选择也比较多，免费的有 Visual Studio Code 和 Atom，收费的有 WebStorm。我们这里推荐采用 Visual Studio Code，可以到 https://code.visualstudio.com/ 下载 Windows/Linux/MacOS 版本。需要注意这个可不是 Visual Studio，不是那个庞大的 IDE，别下载错了。

安装完以上这些工具，开发环境就部署好了，下面我们将开始 Angular 2 的探险之旅。

1.3　第一个小应用 Hello Angular

那么现在开启一个 terminal（命令行窗口），键入 ng new hello-angular ，你会看到以下的命令行输出。

```
wangpengdeMacBook-Pro:~ wangpeng$ ng new hello-angular
installing ng2
  create .editorconfig
  create README.md
  create src/app/app.component.css
  create src/app/app.component.html
  create src/app/app.component.spec.ts
  create src/app/app.component.ts
  create src/app/app.module.ts
  create src/app/index.ts
  create src/assets/.gitkeep
  create src/environments/environment.prod.ts
  create src/environments/environment.ts
  create src/favicon.ico
  create src/index.html
  create src/main.ts
  create src/polyfills.ts
  create src/styles.css
  create src/test.ts
```

```
  create src/tsconfig.json
  create src/typings.d.ts
  create angular-cli.json
  create e2e/app.e2e-spec.ts
  create e2e/app.po.ts
  create e2e/tsconfig.json
  create .gitignore
  create karma.conf.js
  create package.json
  create protractor.conf.js
  create tslint.json
Successfully initialized git.
Installing packages for tooling via npm.
```

这个安装过程需要一段时间，请一定等待安装完毕，命令行重新出现光标提示时才算安装完毕。

这个命令为我们新建了一个名为“hello-angular”的项目。进入该项目目录，键入 code 可以打开 IDE 看到如图 1.1 所示的界面。

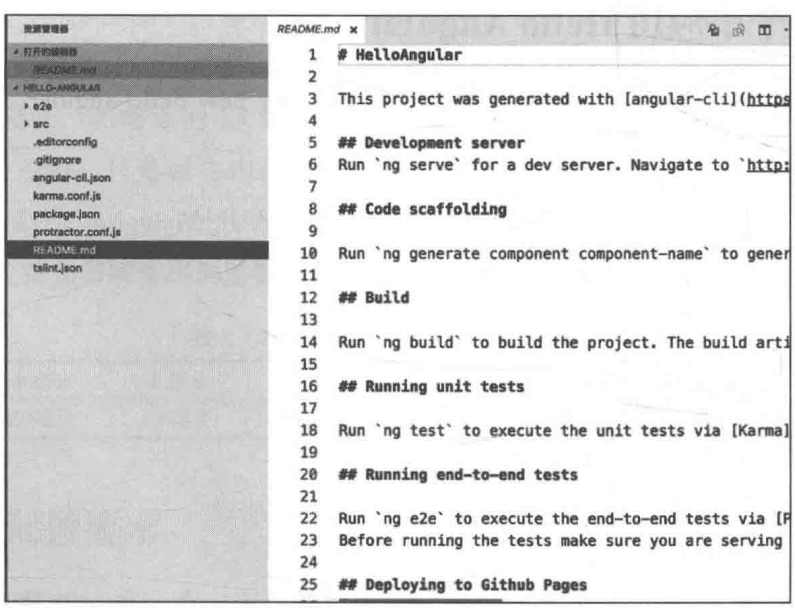


图 1.1　VSCode 管理项目

使用 Mac 的用户可能发现找不到我们刚才使用的命令行 code，需要通过 IDE 安装一下。点击 F1，输入 install，即可看到“在 Path 中安装 code 命令”，选择之后就可以了，如图 1.2 所示。

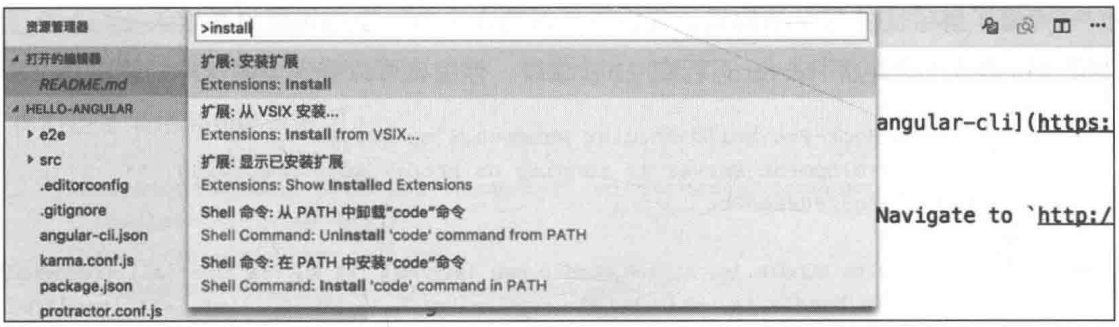


图 1.2　Mac 用户需要安装命令行

项目的文件结构如下，日常开发中真正需要关注的只有 src 目录：

```
├── README.md              -- 项目说明文件（Markdown 格式）
├── angular-cli.json       -- Angular-CLI 配置文件
├── e2e                    -- 端到端（e2e）测试代码目录
│   ├── app.e2e-spec.ts
│   ├── app.po.ts
│   └── tsconfig.json
├── karma.conf.js          -- Karma 单元测试（Unit Testing）配置文件
├── package.json           -- node 打包文件
├── protractor.conf.js     -- 端到端（e2e）测试配置文件
├── src                    -- 源码目录
│   ├── app                -- 应用根目录
│   │   ├── app.component.css      -- 根组件样式
│   │   ├── app.component.html     -- 根组件模板
│   │   ├── app.component.spec.ts  -- 根组件单元测试
│   │   ├── app.component.ts       -- 根组件 ts 文件
│   │   ├── app.module.ts          -- 根模块
│   │   └── index.ts               --app 索引（集中暴露需要给外部使用的对象，方便外部引用）
│   ├── assets                     -- 公共资源目录（图像、文本、视频等）
│   ├── environments               -- 环境配置文件目录
│   │   ├── environment.prod.ts    -- 生产环境配置文件
│   │   └── environment.ts         -- 开发环境配置文件
│   ├── favicon.ico    -- 站点收藏图标
│   ├── index.html     -- 入口页面
│   ├── main.ts        -- 入口 ts 文件
│   ├── polyfills.ts   -- 针对浏览器能力增强的引用文件（一般用于兼容不支持某些新特性的浏览器）
│   ├── styles.css     -- 全局样式文件
│   ├── test.ts        -- 测试入口文件
│   ├── tsconfig.json  -- TypeScript 配置文件
│   └── typings.d.ts   -- 项目中使用的类型定义文件
└── tslint.json        -- 代码 Lint 静态检查文件
```

大概了解了文件目录结构后，我们重新回到命令行，在应用根目录键入 ng serve 可以看到应用编译打包后 server 运行在 4200 端口。你应该可以看到下面这样的输出：

```
wangpengdeMacBook-Pro:hello-angular wangpeng$ ng serve
** NG Live Development Server is running on http://localhost:4200. **
Hash: 0c80f9e8c32908aad0be
Time: 8497ms
chunk  {0} styles.bundle.js, styles.bundle.map (styles) 184 kB {3} [initial] [rendered]
chunk  {1} main.bundle.js, main.bundle.map (main) 5.33 kB {2} [initial] [rendered]
chunk  {2} vendor.bundle.js, vendor.bundle.map (vendor) 2.22 MB [initial] [rendered]
chunk  {3} inline.bundle.js, inline.bundle.map (inline) 0 bytes [entry] [rendered]
webpack: bundle is now VALID.
```

打开浏览器输入 http://localhost:4200 即可看到程序运行成功啦！如图 1.3 所示。

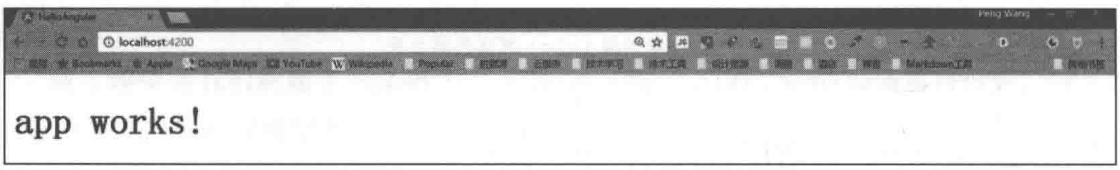


图 1.3　第一次运行应用

自动生成的太没有成就感了是不是，那么我们动手改一下吧。保持运行服务的命令窗口，然后进入 VSCode，打开 src/app/app.component.ts 修改 title，比如：title = 'This is a hello-angular app';，保存后返回浏览器看一下吧，结果已经更新了，如图 1.4 所示。这种热装载的特性使得开发变得很方便。

图 1.4　第一次小修改

1.4　第一个组件

现在，为我们的 App 增加一个 Component 吧，在命令行窗口输入 ng generate component login --inline-template --inline-style 。顾名思义，参数 generate 是用来生成文件的，参数 component 是说明我们要生成一个组件，login 是我们的组件名称，你可以自

己想个其他有意思的名字。后面的两个参数是告诉 angular-cli：生成组件时，请把组件的 HTML 模板和 CSS 样式和组件放在同一个文件中（其实分开文件更清晰，但第一个例子我们还是采用 inline 方式了）：

```
wangpengdeMacBook-Pro:blog wangpeng$ ng generate component login --inline-
  template --inline-style
installing component
  create src/app/login/login.component.spec.ts
  create src/app/login/login.component.ts
wangpengdeMacBook-Pro:blog wangpeng$
```

是不是感觉这个命令行太长了？幸运的是 Angular 团队也这么想，所以你可以把上面的命令改写成 ng g c login -it -is ，也就是说可以用 generate 的首字母 g 来代替 generate，用 component 的首字母 c 来代替 component，类似的 --inline-template 的两个词分别取首字母变成 -it。

angular-cli 为我们在 login 目录下生成了两个文件，其中 login.component.spec.ts 是测试文件，我们这里暂时不提。另一个 login.component.ts 就是我们新建的 Component 了。

Angular 提倡的文件命名方式是这样的：组件名称 .component.ts ，组件的 HTML 模板命名为：组件名称 .component.html，组件的样式文件命名为：组件名称 .component.css。建议读者在编码中尽量遵循 Google 的官方建议。

我们新生成的 Login 组件源码如下：

```
import { Component, OnInit } from '@angular/core';

//@Component 是 Angular 提供的装饰器函数，用来描述 Compoent 的元数据
// 其中 selector 是指这个组件的在 HTML 模板中的标签是什么
//template 是嵌入（inline）的 HTML 模板，如果使用单独文件可用 templateUrl
//styles 是嵌入（inline）的 CSS 样式，如果使用单独文件可用 styleUrls
@Component({
  selector: 'app-login',
  template: '
    <p>
      login Works!
    </p>
  ',
  styles: []
})
export class LoginComponent implements OnInit {
  constructor() { }
```

```
  ngOnInit() {
  }
}
```

这个组件建成后我们怎么使用呢？注意上面的代码中 @Component 修饰配置中的 selector: ' app-login '，这意味着我们可以在其他组件的 template 中使用 <app-login></app-login> 来引用我们的这个组件。

现在我们打开 src/app/app.component.html 加入我们的组件引用：

```
<h1>
  {{title}}
</h1>
<app-login></app-login>
```

保存后返回浏览器，可以看到我们的第一个组件也显示出来了，如图 1.5 所示。

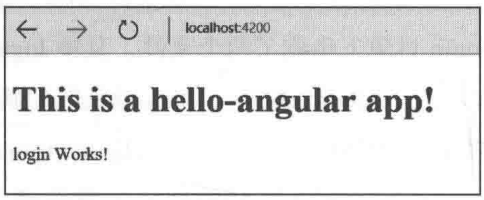


图 1.5　第一个组件的显示

1.5　一些基础概念

有了前面的例子，就可以粗略介绍一些 Angular 的基础概念了，这些基础概念在后面的章节中会更详细地讲解。

1.5.1　元数据和装饰器

Angular 中大量地使用了元数据。元数据是什么呢？**元数据**的定义是这样的：元数据是用来描述数据的数据。

天啊，这什么意思啊？没关系，我们来看一个例子，大家都在电脑上有文件浏览器，随便选择一个文件，我们可以右键选择这个文件的属性看一下，如图 1.6 所示。

我们看到文件本身其实就是一个二进制格式的数据，而在文件的属性中我们又发现了对此数据的描述，包括：文件的种类、文件的大小、文件的位置、创建时间、修改时