12.1.2 可寻址网络

网络使用统一资源定位器（Uniform Resource Locator，URL）为网络上的每个资源提供一个唯一的地址，其中包括了资源的位置和访问方式。URL一般被称为网址或地址。让我们用一个虚拟的旅游网站的URL来说明这些地址的构成，如图12-1所示。这个URL标识了一个网页，该网页包含美国卡罗莱纳州旅游信息。

图12-1 一个URL例子

URL由多个部分组成。URL协议（scheme）标识了访问资源的应用层协议。本例中，使用的协议是HTTP，稍后我们将对其进行详细介绍。冒号（：）表示协议部分结束。两个斜杠（//）后面是URL的授权部分。本例中，授权部分包含了资源所在服务器的DNS主机名：travel.example.com。这里也可以用IP地址。除主机之外的其他信息也可以出现在这一部分，比如用户名（在主机的前面，后面跟@符号）或者端口号（在主机的后面，前面有一个冒号）。接着是URL的路径部分，它指示了资源在网络服务器上的位置。URL路径类似于文件系统路径，用逻辑层次结构来组织资源。本例中，路径/destinations/carolinas表示网站有一组描述旅游目的地的网页，而URL中指定的特定网页是关于卡罗莱纳州的页面。我们可以合理地假设，如果网站有一个页面把佛罗里达州描述为目的地，那么应该会在/destinations/florida找到这个页面。最后，URL的查询部分充当返回给客户端的资源修饰符。本例中，查询指明卡罗莱纳州页面应显示海滩上的位置。URL查询部分的格式和含义随网站的变化而变化。

这个URL包含了大量的信息，所以我们来用通俗易懂的语言重申一下如何读懂它。网站在名为travel.example.com的计算机上运行。它使用HTTP协议，所以当连接到该网站时使用这个协议。这个网站上有名为carolinas

（是一组目的地中的一个）的页面。查询字符串指示页面只显示海滩上的位置。

URL不用包含图12-1所示例子的全部元素。它也可以包含这个例子中没有的一些元素。只含有协议和授权部分的URL是完全有效的，比如http://travel.example.com。在这种情况下，网站提供默认页面，因为没有提供路径。

练习12-1：识别URL的各个部分

对如下URL，请识别出协议、用户名、主机、端口、路径和查询部分。并非所有的URL都包含这些部分。

❑https://example.com/photos?subject=cat&color=black

❑http://192.168.1.20:8080/docs/set5/two-trees.pdf

❑mailto:someone@example.com

网络浏览器一般在其地址栏显示当前URL，如图12-2所示。

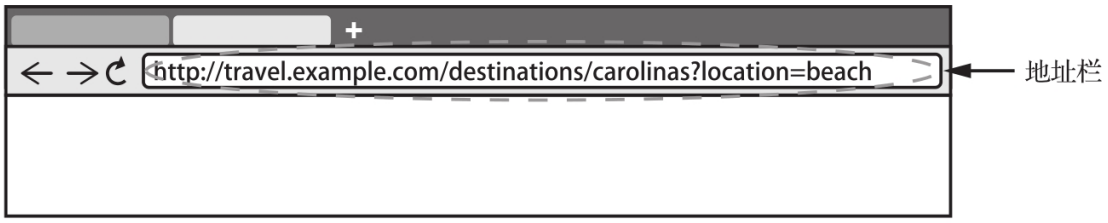

图12-2 地址栏

现在，浏览器通常会在地址栏的URL表示形式中去掉协议、冒号和斜杠。这并不意味着浏览器不再使用URL的这些元素了。浏览器只是尝试为用户做简化。URL显示方式的细节不断地随时间而变化，不同浏览器的行为不同。

图12-3展示了谷歌Chrome（版本77）浏览器在其地址栏显示URL的例子。

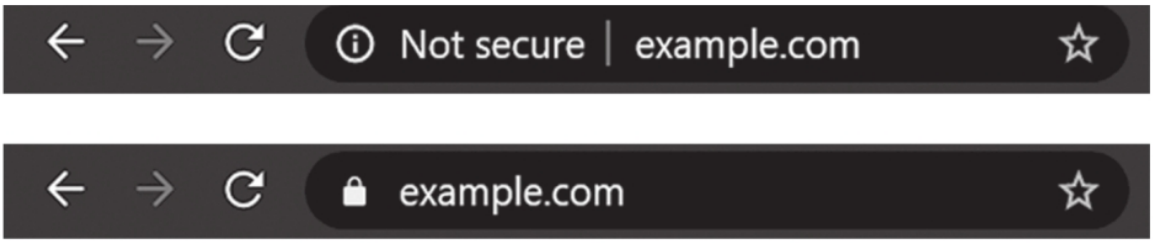

图12-3 Chrome的地址栏

图12-3中上面的图片显示了加载HTTP网站时的地址栏。Chrome没有在地址栏中显示http://前缀。请注意“Not secure”（不安全）的字样。下面的图片显示了加载HTTPS网站时的地址栏。HTTPS是HTTP的安全版本。Chrome省略了https://前缀，但显示了一个锁的图标以表示这是个HTTPS网站。

我们一直在网页的环境下讨论URL，但是URL还可延伸到网络上的其他资源，例如在网页上显示的图像就有自己的URL，脚本文件或XML数据文件也是如此。网络浏览器在其地址栏中只显示网页的URL，但典型的网页通过URL指示各种其他资源，浏览器会自动加载这些资源。

有时候没有必要在URL中包含协议、主机名，甚至完整路径。当URL忽略这些元素中的一个或多个时，它被称为相对URL。相对URL被解释为相对于其被发现的上下文的URL。例如，如果在网页上使用了/images/cat.jpg这样的URL，加载网页的浏览器会假设猫咪照片的协议和主机名与网页自身的协议和主机名是匹配的。

12.1.3 链接网络

网络上的每个资源都有独一无二的地址，URL的这个性质使得一个网络资源很容易引用另一个网络资源。从一个网络文档到另一个网络文档的引用被称为超链接（或简称为链接）。这种链接是单向的，任何网页都能在未获得许可或没有互惠链接的情况下链接到另一个网页。这种网页链接到另一个网页的系统是把“网络”置入万维网的原因。像可以用超链接连接的网页这样的文档被称为超文本文档。

12.1.4 网络协议

网络用超文本传输协议（HTTP）及其安全变体HTTPS进行传输。

1.HTTP

虽然叫这个名字，但HTTP不只可以传输超文本，它还可用于阅读、创建、更新和删除网络上的一切资源。HTTP通常依赖于TCP/IP。TCP保证数据可靠传输，IP处理主机寻址。HTTP自身是以请求-响应模型为基础的。HTTP请求被发送到网络服务器，然后服务器用响应来进行回复。

每个HTTP请求都包含一个HTTP方法，它也被非正式地称为HTTP谓词（verb），它描述了客户端向服务器请求的操作类型。

下面是一些常用的HTTP方法：

❑GET检索资源而不修改它。

❑PUT创建或修改服务器上特定URL处的资源。

❑POST在服务器上创建一个资源，比如现有URL的子资源。

❑DELETE从服务器上移除一个资源。

每个HTTP方法都可以在任何资源上进行尝试，但是托管特定资源的服务器常常不允许在这个资源上使用某些方法。例如，大多数网站都不允许客户端删除资源。那些允许删除的几乎总是要求用户登录到有权删除内容的账户。

典型网站上最常用的方法是GET。当网络浏览器导航到一个网站时，浏览器对被请求网页执行GET。这个网页可能包含对脚本、图像等的引用，那么，浏览器还使用GET方法，在页面完全显示之前获取这些资源。

每个HTTP响应都包含描述服务器响应的HTTP状态码。每个状态码都是一个3位数，其中最高有效数表示响应的一般类别。在100范围内的响应是信息。在200范围内的响应表示成功。在300范围内的响应表示重定向。在

400范围内的响应表示客户端出错——客户端没有正确地形成请求。在500范围内的响应意味着服务器遇到了错误。

下面列出了一些常用的HTTP状态码：

❑**200** 成功。服务器可以满足请求。

❑**301** 永久移动。浏览器应把请求重定向到响应中指定的其他URL。

❑**401** 未经授权。需进行身份验证。

❑**403** 被禁止的。用户不能访问被请求的资源。

❑**404** 未找到。服务器没有找到被请求的资源。

❑**500** 内部服务器错误。服务器上发生了意外情况。

HTTP理解起来相当容易。它用人类可读的文本描述请求和响应。请求的第一行包括HTTP方法、资源的URL和请求的HTTP版本。下面是一个例子：

```
GET /documents/hello.txt HTTP/1.1
```

这个例子只表示客户端请求服务器使用HTTP 1.1向其发送/documents/hello.txt的内容。在请求行的后面，HTTP请求通常包含提供了更多请求信息的请求头字段和一个可选的消息体。

同样，HTTP响应也使用了简单的文本格式。第一行包括HTTP版本、状态码和响应短语。下面的例子是一个HTTP响应的第一行：

```
HTTP/1.1 200 OK
```

在这个响应示例中，服务器指示状态码为200，响应短语为OK。就像HTTP请求一样，响应也可能包含响应头值和消息体。图12-4提供了一个更加详细，但仍然简化的HTTP请求与响应的例子。

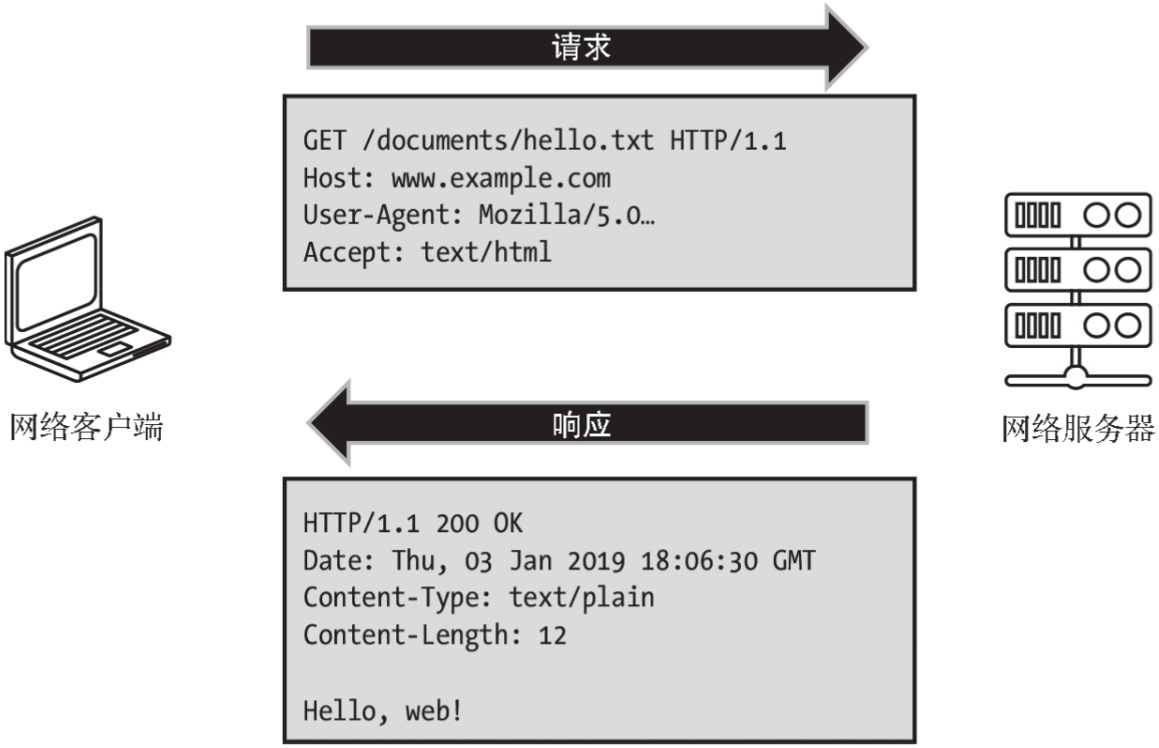

图12-4 一个简化的HTTP请求与响应

注意

请参阅设计36查看HTTP网络流量。

2.HTTPS

HTTP的一种安全变体是HTTPS（HyperText Transfer Protocol Secure，超文本传输协议安全），它通常在网络上用于加密通过互联网发送的数据。加密是把数据编码成不可读格式的过程。解密是加密的逆过程，它使得加密数据再次变得可读。密码算法利用被称为密钥的秘密字节序列对数据进行加密和解密。由于密钥可以保密，因此算法自身可以是众所周知的。

HTTP使用两种加密方式。对称加密使用单个共享密钥来加密和解密消息。非对称加密使用两个密钥（一个密钥对）：公钥用于加密数据，私钥用于解密数据。非对称加密允许自由共享公钥，这样任何人都可以加密并发送数据，但私钥只能与接收并解密数据的受信任方共享。

如果没有HTTPS，那么网络流量会以“明文”的方式传送，意思是没有加密，在传输过程中能被恶意方拦截或修改。HTTPS有助于降低这些风险。有了HTTPS，整个HTTP请求都是加密的，包括URL、请求头和请求体。HTTPS响应也是一样的，它也是完全加密的。HTTPS接受HTTP请求并用传输层安全（Transport Layer Security，TLS）协议对其进行加密。过去曾使用过类似的名为安全套接字层（Secure Sockets Layer，SSL）的协议，但由于安全问题，这个协议已经被弃用，转而被TLS取代。当我们说起HTTPS时，我们的意思是指使用TLS加密的HTTP。

当HTTPS会话开始时，客户端用client hello消息连接到服务器，该消息包含它希望进行安全通信的细节。服务器用server hello消息进行响应，该消息确认通信将要如何发生。服务器还会发送一组被称为数字证书的字节，其中包括了服务器用于非对称加密的公共加密密钥。然后，客户端检查服务器的证书是否有效。如果有效，客户端就用服务器的公共密钥加密字节串，然后把加密后的消息发送给服务器。服务器用自己的私钥解密这些字节。服务器和客户端都使用之前交换的信息来计算用于对称加密的共享密钥。当客户端和服务器都有了共享密钥后，在会话期间，这个密钥会被用于加密和解密所有在客户端和服务器之间的通信。

HTTPS以前只在有限的情况下使用，如针对处理特别敏感信息的网站。但是，网络正在进入这样一种状态，即HTTPS是常态，而不是例外。人们越来越相信HTTPS的安全和隐私优势对于大多数网络流量而言都是有意义的。Google鼓励这种变化，它在Chrome中把HTTP网站标记为“不安全”，并把HTTPS的存在作为其搜索引擎的积极信号，帮助提高HTTPS网站的Google搜索排名。

注意

请参阅设计37在本地网络上设置一个简单的网络服务器。

12.1.5 可搜索的网络

对于许多人来说，典型的网络入口是搜索框。用户在其浏览器中输入一些搜索词并查看出现的内容，而不是导航到特定的URL。浏览器的设计鼓

励这样做，因为浏览器通常也将地址栏作为搜索框。实际上，在用户想要访问特定网站时，他们也常常对该网站进行搜索，然后点击生成的链接，而不是在地址栏中输入完整的URL。这种设计增强了浏览器的可用性，即使它模糊了URL和搜索词、浏览器和搜索引擎之间的区别。

尽管网络搜索很流行、也很有用，但搜索功能并不是网络的原生功能。对于搜索应如何工作，并没有一个标准规范。这意味着作为网络关键功能之一的搜索依赖于非标准的专有搜索引擎。在撰写本书时，Google主导着网络搜索领域，尽管还有其他很好的搜索引擎，但它们的全球使用量只是Google的一小部分。

12.2 网络语言

任何能以文件形式保存的内容都可以托管到网络上。例如，网络服务器可以托管Excel文件集合，它们可以从网站下载并用Excel打开。但是，网络浏览器并不只是一个下载文件以便在其他应用程序中打开文件的工具。网络浏览器不仅能下载内容，还能显示网页。这些网页可以是简单的文档，也可以是交互式网络应用程序。为了实现这一点，浏览器要理解三种用于构建网站的计算机语言：

❑**超文本标记语言**（HyperText Markup Language，HTML）定义网页结构。换句话说，它定义了网页上的内容。例如，HTML可以指定网页上存在的一个按钮。

❑**级联样式表**（Cascading Style Sheet，CSS）定义网页的外观。换句话说，它定义了网页的样子。例如，CSS可以指定上面说的按钮宽30个像素，颜色为蓝色。

❑**JavaScript** 定义网页的行为。换句话说，它定义了网页的功能。例如，在单击按钮的时候，可以使用JavaScript将两数相加。

这三种语言一起使用便可创建网络的内容。值得注意的是，网络浏览器也能够呈现一些其他的数据类型，特别是某些图像、视频和音频数据，

但我们不对此进行详细讨论。现在，让我们深入研究一下这三种基本网络语言：HTML、CSS和JavaScript。

12.2.1 用HTML构造网络

HTML是一种描述网页结构的标记语言。请注意，HTML不是编程语言。编程语言描述计算机应执行的操作，而标记语言描述数据的结构。对于HTML来说，所讨论的数据表示网页。一个网页可以包含各种元素，比如段落、标题和图像。下面的例子用HTML描述了一个简单的网页。

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>A Cat</title>
  </head>
  <body>
    <h1>Thoughts on a Cat</h1>


    <p>This is a cat.</p>
    <img src="cat.jpg" alt="cat photo">
  </body>
</html>
```

你会看到许多项目被包含在小于号（<）和大于号（>）中。它们被称为HTML标签，是一组用于定义HTML文档各部分的文本字符。例如， 是用于指示段落开始的标签。相应的 是用于指示段落结束的标签。注意结束标签中的斜杠，它把结束标签与开始标签区别开来。HTML元素是页面的一部分，以开始标签开头，以结束标签结尾，在这两个标签之间的是内容。例如，这是一个HTML元素： This is a cat. 。实际上，并非所有元素都需要结束标签。例如，用于表示图像的img元素就不需要结束标签。你可以在前面的HTML代码示例中看到这一点。

图12-5展示了示例HTML是如何在网络浏览器中呈现的。

图12-5 我们的网页示例在网络浏览器中的呈现

文档中特意没有包含如何显示的信息，所以浏览器对标题和段落使用了默认字体和字号。在本例中，浏览器还默认显示为白底黑字——同样，这也没有在文档中指定。由于这个HTML例子没有包含样式信息，因此不同的浏览器可以选择略微不同的样式来呈现这个网页。

让我们更细致地来看看HTML代码示例。HTML文档的第一行声明该文件是HTML文档：<!DOCTYPE html>。在这下面，HTML文档被构造成有父元素和子元素的树形结构。标签<html>是顶级父标签，所有内容都被包含在<html>和</html>之间。你可以把这两个标签解释为“HTML从这里开始”以及“HTML在这里结束”。这是有道理的——HTML文档中的所有内容都应该是HTML！<html>标签还包含一个名为lang的属性，它把本文档的语言标识为en，即英文的代码。

html元素有两个子元素：head和body。包含在<head>和</head>之间的元素描述了文档，而包含在<body>和</body>之间的元素则构成了文档

的内容。

在我们的例子中，head包含了两个元素：描述文档编码所用字符集的meta元素和title元素。浏览器通常会在页面的标签栏上显示title文本，并在用户添加书签或收藏时把它当作默认名称。搜索引擎在显示结果时使用title文本。出于这些原因，网络开发人员给他们的网页赋予有意义的标题是很重要的。

本例中的body（正文）包含一个<h1>标签，它用于标题元素。标题标签从<h1>到<h6>都可用，其中h1用作各部分标题的最高级别，h6为最低级别。标题的后面是段落，用<p>标签来指示，段落的后面是用<img>表示的图像。注意，图像自身的字节数没有出现在HTML中。相反，<img>标签只是用相对URL（cat.jpg）引用了一个图像文件。要完整加载这个网页，浏览器需要发出单独的HTTP请求来下载这个图像。本例中，这个图像URL只是一个文件名，表示它正驻留在与文档本身相同的服务器和路径中。如果图像驻留在别的地方，就要使用带路径或服务器名的URL。<img>标签还有一个alt属性，它提供了描述该图像的替代文本。这是在图像无法呈现的情况下使用的，例如，当使用纯文本浏览器或大声朗读页面内容的屏幕阅读器时。

你可能已经注意到前面的HTML代码用缩进来显示页面上各种元素的嵌套。例如，标签<h1>和<p>是同一缩进级别的，这表示它们是<body>标签的子元素。这在网络开发中是常见做法，用于改善HTML的可读性，但它不是必需的。实际上，在HTML文档中，除空格符或制表符之外的空白都不重要！我们可以移除所有额外的空格符、制表符和换行符，把HTML代码都留在一行中，文档还是会以相同的方式在浏览器中显示。网络浏览器会忽略多余的空白，所以在网页上分隔元素只对开发人员有帮助。

注意

请参阅设计38让本地网站返回一个用HTML构造的文档，而不是简单的文本。

到目前为止，我们介绍的HTML元素仅占网络浏览器能识别的全部元素的一小部分。我们不会在这里详尽地介绍HTML，网上有很多很好的资料可供参考。HTML规范以前由两个组织维护：W3C（万维网联盟）和WHATWG（网络超文本应用技术工作组）。上一个从W3C获得“推荐”状态的主要HTML版本是HTML5。2019年，这两个组织一致认为，持续进行的HTML标准开发将主要由WHATWG处理，它被称为HTML Living Standard，这个标准会一直被维护。

现代浏览器试图同时支持当前版本和旧版本的HTML，因为许多网络内容都是按照早期HTML标准编写的。过去，浏览器引入了非标准的HTML元素，其中的一些最终标准化了，而另一些则不再使用且失去了支持。网络浏览器开发人员必须在创新和坚持标准之间取得平衡，并且还要支持有时在网络上发现的、不那么完美的HTML。网络浏览器在不断发展，不同的浏览器有时会以不同的形式呈现相同的内容。这就意味着网络开发人员要定期在多个浏览器上测试他们的工作，以确保行为一致。

12.2.2 用CSS设计网络样式

在前面的HTML例子中，我们使用标签描述文档的结构，但这些标签不能传递任何关于应如何显示文档的信息。这是有意为之的，我们希望保持结构与样式各自独立。两者之间的分离使得相同的内容在不同的环境中能以不同的样式予以呈现。例如，在较大的PC屏幕上和较小的移动屏幕上，大多数网络内容应以不同的形式呈现。

级联样式表（CSS）是用于描述网页样式的语言。样式表是规则列表。每个规则描述了应用于页面特定部分的样式。每个规则都包含一个选择器，它指示页面上的哪些元素要应用样式。“级联”一词是指对同一个元素应用多个规则的功能。让我们来看个例子：

```
p {
   font-family: Arial, Helvetica, sans-serif;
   font-size: 11pt;
   margin-left: 10px;
   color: DimGray;
}

h1 {
   font-family: 'Courier New', Courier, monospace;
   font-size: 18pt;
   font-weight: bold;
}
```

这个例子为段落（p）元素和标题（h1）元素定义了样式规则。当这个CSS应用于网页时，该网页上的所有段落都使用指定字体，字号为11磅，左边距为10个像素，文本为灰色。同样，h1标题使用指定的加粗字体，字号为18磅。注意，font-family是字体列表，不是单个字体。这意味着网络浏览器应该尝试找到匹配的字体，从最左边的字体开始一直向右，直到找到匹配的字体为止。并不是每个客户端设备都安装了首选字体，指定多个字体可增加可用匹配字体的机会。

你可以通过多种方法把样式表应用于网页。其中的一个方法是在网页的style元素中包含CSS规则。例如：

```
<style>p {color: red};</style>
```

这不是理想方法，因为样式和结构在这里是密切相关的。更好的方法是在单独的文件中指定CSS规则并将其托管到网络上。这种方法让HTML和CSS完全分开，并且允许多个HTML文件使用相同的样式表。通过这种方式，我们可以改变CSS规则，并将它一次性应用到多个页面。HTML head部分中的单个元素可以被应用一个CSS文件的样式表规则，如下所示（style.css是要应用的CSS文件的URL）：

```
<link rel="stylesheet" type="text/css" href="style.css">
```

如果把这个样式表应用到示例猫咪网页上，我们会看到标题和段落文本的改变，如图12-6所示。

图12-6 应用了CSS的示例网页

注意

请参阅设计39用一些CSS更新猫咪网页。

这个CSS例子很简单，但是CSS还允许实现更高级的样式。如果你对在网上找到的各式各样的视觉样式都很熟悉，那么你就已经看到了CSS在实际应用中的力量。

12.2.3 用JavaScript编写网络脚本

网络最初被设想为通过超文本文档共享信息的一种方式。HTML赋予我们这种能力，CSS赋予我们控制这些文档表现形式的方法。但是，网络进化成了一种交互内容的平台，JavaScript成为实现交互的标准手段。JavaScript是一种编程语言，它能让网页响应用户行为并以编程方式执行各种任务。使用JavaScript，网络浏览器不仅成了文档阅读器，还成了一个完整的应用程序开发平台。

JavaScript是解释语言：在交付给浏览器之前，它不会被编译成机器码。网络服务器以文本形式托管JavaScript代码，浏览器下载该代码并在运行时进行解释。也就是说，一些浏览器使用在运行时编译JavaScript的即时（Just-In-Time，JIT）编译器，从而提高了性能。有些开发人员在部署其JavaScript脚本之前会将其缩小，删除空格和注释——通常这会减小脚本。缩小JavaScript脚本可以缩短网站的加载时间。缩小脚本与编译脚本不同，缩小后的文件仍然是高级代码，而不是编译后的机器码。

JavaScript的语法类似于C语言以及其他从C借鉴的语言（比如C++、Java和C#）。但是，相似只是表面，因为JavaScript与这些语言有很大的不同。不要让名称迷惑了你：JavaScript与Java没什么关系。这个语言是面向对象的，但本质上依赖的是原型而不是类。也就是说，是现有的对象而不是类充当了其他对象的模板。

JavaScript用浏览器提供的页面表现形式与HTML网页进行交互，这种形式被称为文档对象模型（Document Object Model，DOM）。DOM是页面元素的分层树结构，可以通过编程进行修改。在DOM中更新一个元素会导致浏览器更新显示的网页上的元素。JavaScript包含处理DOM的方法，通过这些方法，JavaScript代码既可以响应页面上发生的事件（比如单击一个按钮），也可以改变被呈现页面的内容。

让我们来看一个简单脚本的部分内容，这个脚本与我们的示例页面进行交互。脚本在每次单击（或在触摸屏上单击）猫咪照片时就向页面的段落添加文本Meow!。