JavaScript and JSON Essentials,Second Edition

JavaScript与JSON从入门到精通（第2版）

[美] 布鲁诺·约瑟夫·德梅洛 等 著　　刘晓雪 译

清华大学出版社

JavaScript 与 JSON 从入门到精通

（第 2 版）

[美] 布鲁诺·约瑟夫·德梅洛　等著
刘晓雪　译

清华大学出版社
北　京

内 容 简 介

本书详细阐述了与 JSON 相关的基本解决方案，主要包括 JSON 简介、JSON 结构、基于 JSON 的 AJAX 请求、跨域异步请求、JSON 调试、构建 Carousel 应用程序、JSON 的替代方案、hapi.js 简介、在 MongoDB 中存储 JSON 文档、利用 JSON 配置任务管理器、实时系统和分布式系统中的 JSON、JSON 用例等内容。此外，本书还提供了相应的示例、代码，以帮助读者进一步理解相关方案的实现过程。

本书既可作为高等院校计算机及相关专业的教材和教学参考书，也可作为相关开发人员的自学教材和参考手册。

北京市版权局著作权合同登记号　图字：01-2019-1270

图书在版编目（CIP）数据

JavaScript 与 JSON 从入门到精通：第 2 版/（美）布鲁诺 • 约瑟夫 • 德梅洛等著；刘晓雪译. —北京：清华大学出版社，2019
书名原文：JavaScript and JSON Essentials-Second Edition
ISBN 978-7-302-53242-2

I. ①J…　II. ①布…　②刘…　III. ①JAVA 语言–程序设计　IV. ①TP312.8

中国版本图书馆 CIP 数据核字（2019）第 131108 号

责任编辑： 贾小红
封面设计： 刘　超
版式设计： 文森时代
责任校对： 马军令
责任印制： 刘海龙

出版发行： 清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者： 三河市国英印务有限公司
经　　销： 全国新华书店
开　　本： 185mm×230mm　　**印　　张：** 10.75　　**字　　数：** 218 千字
版　　次： 2019 年 7 月第 1 版　　**印　　次：** 2019 年 7 月第 1 次印刷
定　　价： 89.00 元

产品编号：082572-01

译 者 序

JSON 和 XML 是较为流行的数据交换格式，在 Web API、NoSQL 数据库、服务端编程语言和客户端框架中都可以看到 JSON 的身影。在不同平台间的传递数据方面，JSON 已成为 XML 强有力的替代者。与 XML 相比，JSON 更加简洁且易于阅读，同时方便检查排错。另外，JSON 更加轻量级，不管是编写、传输，还是解析都更加高效。JSON 在传输过程中采用了压缩技术，因而更加节省带宽。最后，JSON 还得到了众多语言的支持，如 JavaScript、Python、C、C++等主流语言。

本书通过示例展示了 JSON 在 Web 开发中扮演的不同角色。在了解了 JSON 的基础知识后，将在 Angular 5、Node.js、模板嵌入机制的基础上实现 JSON 应用程序。对于服务器端的脚本编程，本书还将尝试实现 Hapi.js（因其可配置的 JSON 架构著称）。此外，本书还将学习如何使用 Kafka 为实时应用程序实现 JSON，以及如何为任务运行器和 MongoDB BSON 存储实现 JSON。本书通过 JSON 格式的案例进行讲解，帮助读者构建快速、可伸缩和高效的 Web 应用程序。

在本书的翻译过程中，除刘晓雪外，王辉、刘璋、张博、刘祎、张华臻等人也参与了部分翻译工作，在此一并表示感谢。

由于译者水平有限，难免有疏漏和不妥之处，恳请广大读者批评指正。

译 者

前　言

JSON 是数据交换的一种标准格式，本书将通过各种示例讨论 JSON 在 Web 开发中饰演的不同角色。在阅读完本书后，读者将会以全新的角度理解应用程序的解决方案和复杂问题的处理方式。

适用读者

如果读者是一名对 JavaScript 或 PHP 开发有着基本了解的 Web 人员，并且希望编写 JSON 数据进而将其与 RESTful API 集成，以创建快速、可伸缩的应用程序，那么，本书将十分适合于您。

本书内容

第 1 章：JSON 简介。将讨论 JSON 的历史及其工作方式和内存中的存储方式。另外，本章还将介绍一些支持 JSON 的、较为流行的编程语言。在本章结束时，还将利用不同的 JSON 数据类型编写一个较为基础的应用程序。

第 2 章：JSON 结构。将利用多种数据类型、多个对象和多维数据进一步丰富 JSON 实现。

第 3 章：基于 JSON 的 AJAX 请求。将探讨基于 JSON 数据的 AJAX 请求，并通过 HTTP 请求传递 JSON 数据，以及处理此类问题的异步技术。

第 4 章：跨域异步请求。介绍跨域的异步调用这一概念。由于数据将在域间进行传输，因而用户有必要了解基于填充（padding）的 JSON 设疑概念，即 JSONP。

第 5 章：JSON 调试。将讨论可用于调试、验证和格式化 JSON 的强大工具。

第 6 章：构建 Carousel 应用程序。实现了 Carousel 应用程序的编程思想，以及应用程序所需的设置项和依赖项，如 jQuery 库和 jQuery Cycle 插件，并使用 Bootstrap 来维护应用程序的基本设计。

第 7 章：JSON 的替代方案。讨论了 JSON 的非 Web 开发实现，如依赖项管理器、元数据存储和配置存储。

第 8 章：hapi.js 简介。介绍在 Hapi 服务器中实现基于 JSON 的配置，并借助于 Hapi

创建 RESTful API。

第 9 章：在 MongoDB 中存储 JSON 文档。讨论 MongoDB，以及 JSON 在 MongoDB 中的使用方式。随后，本章还将介绍如何在 MongoDB 文档上执行不同的操作。

第 10 章：利用 JSON 配置任务管理器。将简要描述 gulp.js 库。Gulp 是一个功能强大的库，主要用于构建任务的管理并提供相关工具。

第 11 章：实时系统和分布系统中的 JSON。通过实现 socket.io 服务器，使读者熟悉 JSON 数据在实时 Web 应用程序中的应用，以及 Apache Kafka。

第 12 章：JSON 中的用例。将讨论一个用例，并考查 JSON 针对不同领域的增强方案，以及移植后 JSON 所提供的各种优点。

阅读方式

如果读者是一名 Web 开发的初学者，可从第 1 章开始阅读，并了解 JSON 中的基础知识。另外，前 5 章简单易懂且便于操作。在后续学习过程中，读者可尝试实现每章所提供的代码片段。

随着时间的推移，读者还可在 StackOverflow 或 GitHub 等论坛上进行讨论，以确保书中的所有问题均已被解决。

软件环境和资源下载

读者可访问 http://www.packtpub.com 并通过个人账户下载示例代码文件。另外，在 http://www.packtpub.com/support 中注册成功后，我们将以电子邮件的方式将相关文件发与读者。

读者可根据下列步骤下载代码文件。

- 利用电子邮件和密码登录或注册我们的网站 www.packtpub.com。
- 单击 SUPPORT 选项卡。
- 单击 Code Downloads & Errata。
- 在 Serach 文本框中输入书名。

当文件下载完毕后，确保使用下列最新版本软件解压文件夹。

- Windows 系统下的 WinRAR/7-Zip。
- Mac 系统下的 Zipeg/iZip/UnRarX。
- Linux 系统下的 7-Zip/PeaZip。

另外，读者还可访问 GitHub 获取本书的代码包，对应网址为 https://github.com/

PacktPublishing/JavaScript-and-JSON-Essentials-Second-Edition。

此外，读者还可访问 https://github.com/PacktPublishing/以了解丰富的代码和视频资源。

最后，读者还可访问 https://www.packtpub.com/sites/default/files/downloads/JavaScriptandJSONEssentialsSecondEdition_ColorImages.pdf 以下载并查看书中的图片。

本书约定

本书通过不同的文本风格区分相应的信息类型。下面通过一些示例对此类风格以及具体含义的解释予以展示。

代码块如下所示。

```
for(let j=0;j<designationCount;j++){
  designations+= `, ${data_json[i].designation.title[j]}`;
}
```

当某个代码块希望引起读者的足够重视时，一般会采用黑体表示，如下所示。

```
const http = require('http');
const port = 3300;
http.createServer((req, res) => {
  res.writeHead(200, {
    "Content-Type": "application/json"
  });
  res.write(JSON.stringify({
    greet : "Hello Readers!"
  }));
  res.end();
}).listen(port);
console.log(`Node Server is running on port : ${port}`)
```

命令行输入或输出则采用下列方式表达。

```
$ mkdir test-node-app
$ cd test-node-app
$ npm init
```

图标则表示较为重要的说明事项。

图标则表示提示信息和操作技巧。

读者反馈和客户支持

欢迎读者对本书的建议或意见予以反馈。

对此，读者可向 feedback@packtpub.com 发送邮件，并以书名作为邮件标题。若读者对本书有任何疑问，均可发送邮件至 questions@packtpub.com，我们将竭诚为您服务。

若读者针对某项技术具有专家级的见解，抑或计划撰写书籍或完善某部著作的出版工作，则可访问 www.packtpub.com/authors。

勘误表

尽管我们在最大程度上做到尽善尽美，但错误依然在所难免。如果读者发现谬误之处，无论是文字错误抑或是代码错误，还望不吝赐教。对此，读者可访问 http://www.packtpub.com/submit-errata，选取对应书籍，单击 ErrataSubmissionForm 超链接，并输入相关问题的详细内容。

版权须知

一直以来，互联网上的版权问题从未间断，Packt 出版社对此类问题异常重视。若读者在互联网上发现本书任意形式的副本，请告知网络地址或网站名称，我们将对此予以处理。关于盗版问题，读者可发送邮件至 copyright@packtpub.com。

问题解答

若读者对本书有任何疑问，均可发送邮件至 questions@packtpub.com，我们将竭诚为您服务。

目　录

第 1 章　JSON 简介

JSON（JavaScript 对象表示法）是一种非常流行的数据交换格式，最先由 Douglas Crockford 爵士发布。根据 Douglas Crockford 所言，JSON 通常以对象表示法而存在。另外，Douglas Crockford 并非发明了 JSON，但他首次制定了 JSON 规范并设计了 JSON，以便将其用作标准格式。

本章主要涉及以下主题。

- 什么是 ISON?
- 利用 JSON 实现简单的 Hello World 程序。
- JSON 中的数据类型。
- JSON 所支持的各种语言。
- PHP 和 Python 简介。

对于第一次听说 JSON 这一术语的所有初学者来说，下面的各个小节将帮助您了解 JSON。

1.1　数据交换格式 JSON

对于客户端和服务器间的数据交换，当定义 JSON 时，我们可以说这是一种基于文本的、轻量级的、具有人类可读的数据格式。JSON 源于 JavaScript，与 JavaScript 对象非常相似，但独立于 JavaScript。另外，JSON 独立于任何语言，但所有流行的编程语言均对 JSON 数据格式提供了支持，其中包括 C#、PHP、Java、C++、Python 和 Ruby。

JSON 可用于 Web 应用程序的数据交换行为。考查如图 1.1 所示的简单的客户端-服务器架构。假设客户端为浏览器，并向服务器发送 HTTP 请求；随后，服务器按预期处理请求并提供响应结果。

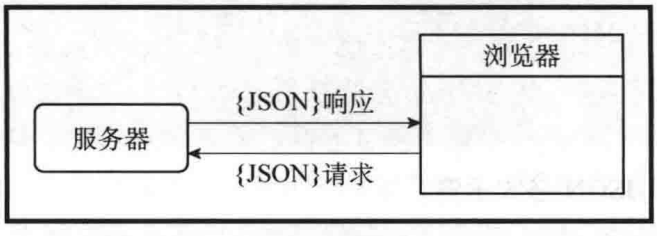

图　1.1

在上述双向通信中，所用的数据格式表示为一个序列化的字符串，且括号中包含了键-值对组合——这便是 JSON。

在 JSON 之前，XML 曾被视作所选的数据交换格式。XML 解析需要客户端的 XML DOM 实现来接收 XML 响应，然后使用 XPath 查询响应以访问和检索数据。这一过程较为烦琐，数据查询须在两级上执行：首先是服务器端，其间，数据通过数据库被查询；其次是在客户端使用 XPath。相比之下，JSON 则不需要特定的实现，浏览器中的 JavaScript 引擎则负责处理 JSON 的解析工作。

在通过网络连接发送数据时，XML 消息通常比较繁重和冗长，并占用大量带宽。一旦检索到 XML 消息，该消息需要加载至内存中以被处理。下面分别考查 XML 格式和 JSON 格式的 students 数据传输。

下列代码展示了 XML 格式示例。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This is an example of student feed in XML -->
<students>
  <student>
    <studentid>101</studentid>
    <firstname>John</firstname>
    <lastname>Doe</lastname>
    <classes>
      <class>Business Research</class>
      <class>Economics</class>
      <class>Finance</class>
    </classes>
  </student>
  <student>
    <studentid>102</studentid>
    <firstname>Jane</firstname>
    <lastname>Dane</lastname>
    <classes>
      <class>Marketing</class>
      <class>Economics</class>
      <class>Finance</class>
    </classes>
  </student>
</students>
```

下列代码展示了 JSON 格式示例。

```
/* This is an example of student feed in JSON */
{
```

```
"students": {
  "0": {
    "studentid": 101,
    "firstname": "John",
    "lastname": "Doe",
    "classes": [
      "Business Research",
      "Economics",
      "Finance"
    ]
  },
  "1":{
    "studentid": 102,
    "firstname": "Jane",
    "lastname": "Dane",
    "classes": [
      "Marketing",
      "Economics",
      "Finance"
    ]
  }
}
```

需要注意的一点是，当与 JSON 相比时，XML 消息尺寸较大，而此处仅展示了两个记录。实时传入数据至少包含数千个记录。需要注意的另一点是，由服务器生成并通过互联网传输的数据量已经很大，而 XML 的冗长使得数据量变得更大。在移动设备时代，智能手机和平板电脑日益流行，在速度较慢的网络上传输大量数据将会导致页面加载缓慢、死机、较差的用户体验和访问量的降低。目前，JSON 已经成为首选的互联网数据交换格式，以避免前面提到的各类问题。由于 JSON 用于在互联网上传输序列化的数据，因此我们需要了解其 MIME 类型。

图 1.2 显示了请求数据如何发送至之前提到的客户端-服务器架构中。

多用途互联网邮件扩展（Multipurpose Internet Mail Extensions，MIME）类型是一种互联网媒体类型，这是由两部分构成的内容标识符，进而在互联网上传输。MIME 类型通过 HTTP 请求和 HTTP 响应的 HTTP 头文件传递。MIME 类型可视为服务器和浏览器间的内容类型通信。总体来说，MIME 类型将包含两个或多个部分，并提供与数据类型（在 HTTP 请求或 HTTP 响应中发送）相关的浏览器信息。JSON 数据的 MIME 类型表示为应用程序/json。如果 MIME 类型头文件没有通过浏览器发送，它会将传入的 JSON 视为纯文本。

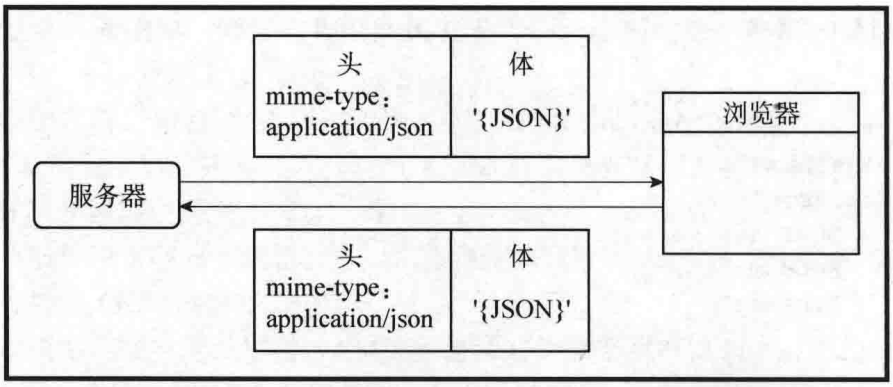

图　1.2

现在，content-type 键（派生自 mime-type 自身）可用于数据头中。

1.2　基于 JSON 的 Hello World 程序

前述内容介绍了 JSON 的基本知识，接下来像往常一样编写一个 Hello World 应用程序，对应的代码片段如下所示。

```
<!DOCTYPE html>
<html>
  <head>
    <title>Test Javascript</title>
    <script type="text/javascript">
      let hello_world = {"Hello":"World"};
      alert(hello_world.Hello);
    </script>
  </head>
  <body>
    <h2>JSON Hello World</h2>
    <p>This is a test program to alert Hello world!</p>
  </body>
</html>
```

当在浏览器中被调用时，上述程序将在屏幕上显示 Hello World。

此处使用了新的 ECMAScript 标识符 let，其作用域与一般的变量声明标识符 var 有所不同。前者的作用域是最近的函数块，而后者的作用域则是最近的封闭块。对此，读者可访问 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let 以了解更多信息。

此处应特别关注<script>标签之间的脚本，如下所示。

```
let hello_world = {"Hello":"World"};
alert(hello_world.Hello);
```

此处首先生成一个 JavaScript 变量，并利用 JavaScript 对象初始化该变量。类似于从 JavaScript 对象中检索数据，我们也可采用键-值对检索数值。简单地讲，JSON 表示为一个键-值对集合。其中，每个键表示为数值在计算机上的存储位置的引用。下面让我们思考一下，如果全部工作是分配 JavaScript 对象，为何我们需要使用 JSON。答案在于，JSON 是一种完全不同的格式，而不像 JavaScript 那样是一种语言。

JSON 键和值必须用双引号括起来。如果任何一个都包含在单引号中，则将会得到一条错误消息。

接下来快速浏览一下 JSON 和 JavaScript 对象之间的相似性和差异。如果打算创建一个与之前 hello_world JSON 示例类似的 JavaScript 对象，对应代码如下所示。

```
let hello_world = {"Hello":"World"};
```

此处的差别在于，键并未包含于双引号中。由于 JSON 键定义为一个字符串，因而可对其使用任意有效的字符串。例如，可在键中使用空格、特殊字符和连字符，则这在常规的 JavaScript 对象中均是无效的，如下所示。

```
let hello_world = {"test-hello":"World"};
```

当在键中使用特殊字符、连字符或空格时，我们在对其进行访问时须格外小心，如下所示。

```
alert(hello_world.test-hello); //doesn't work
```

上述 JavaScript 语句无法正常工作的原因在于，JavaScript 不接受包含特殊字符、连字符或字符串的键。因此，需要通过某种方法检索数据，在这个方法中，我们将 JSON 对象作为一个包含字符串键的关联数组来处理，如下所示。

```
alert(hello_world["test-hello"]); //Hurray! It work
```

二者间的另一个差别在于，JavaScript 对象可在内部携带函数，而 JSON 对象则无法携带任何函数。下列示例设置了一个属性 getFullName，其中包含了一个函数并在被调用时显示名称 John Doe。

```
{
  "studentid": 101,
  "firstname": "John",
```