Web 开发与设计

Svelte 和 Sapper 实战

[美] R. 马克·沃尔克曼(R. Mark Volkmann) 著

颜 宇 周 轶 王 威 译

清华大学出版社

北 京

R. Mark Volkmann
Svelte and Sapper in Action
EISBN: 978-1-61729-794-6

图书在版编目(CIP)数据

Svelte 和 Sapper 实战 / (美) R.马克·沃尔克曼(R. Mark Volkmann) 著；颜宇，周轶，王威译. —北京：清华大学出版社，2022.1
(Web 开发与设计)
书名原文：Svelte and Sapper in Action
ISBN 978-7-302-59515-1

I. ①S… II. ①R… ②颜… ③周… ④王… III. ①JAVA 语言－程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字(2021)第 230504 号

责任编辑：王　军
封面设计：孔祥峰
版式设计：思创景点
责任校对：成凤进
责任印制：宋　林

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：大厂回族自治县彩虹印刷有限公司
经　　销：全国新华书店
开　　本：170mm×240mm　　印　　张：25.5　　字　　数：588 千字
版　　次：2022 年 1 月第 1 版　　印　　次：2022 年 1 月第 1 次印刷
定　　价：118.00 元

产品编号：090583-01

译 者 序

对于前端开发人员来说，最主要的工作任务就是通过代码为用户呈现优美、生动、体验极佳的页面，而 UI 框架在其中起到了至关重要的作用。纵观最近十几年的前端发展历程，可以简单地将前端技术在 UI 展示领域的发展大致归纳为 3 个阶段。

在第一阶段，开发人员通过浏览器的渲染 API 直接操作 DOM 来完成页面呈现和交互，典型代表就是 jQuery。

在第二阶段，随着页面中内容和交互逻辑逐渐增多，依赖直接调用渲染 API 的方式无法应对复杂的业务场景，此时一些设计模式和理念被引入前端领域，其中最知名的是 MVC 模型。著名的 Backbone 框架就是 MVC 模型的一个实现。通过 MVC 模型，前端开发者初步实现了将数据与渲染解耦合。

在第三阶段，即进入 2010 年后，前端应用场景呈现井喷式增长，终端用户对于页面呈现的内容和交互要求逐渐增加。这推动前端技术在提升用户体验和降低开发复杂度方面的革命性创新，催生了现在大家耳熟能详的三大框架：Angular、React 和 Vue。从此前端进入了新时代！

此处就不赘述这三者的概念了，相信大多数开发者现在都至少使用其中一个框架。同时这三个框架也在不断优化和提升自身；目前，Vue 已经发布了 3.0 版本，React 的 Concurrent、函数式组件、Hooks 也被越来越多地应用到业务中，Angular 在 PWA 方面也颇有建树。那么是不是可以认为我们只需要在这三者之间做出选择就够了呢？

答案是否定的，因为技术始终是用于服务产品的，随着产品领域逐渐细分，涌现了如 Web、智能手机、无线、物联网等多个细分领域，这就要求前端技术能够应对越来越精细的应用场景。要求上述三大框架覆盖所有需求是不现实的，其他技术的出现也是自然而然的了。

Svelte 就是在这样的大环境下应运而生的，主要用来解决页面加载时静态资源大小的问题，通过前置的静态编译过程，最大限度压缩页面运行时需要加载的静态资源。与三大框架相比，使用 Svelte 组件并不需要引入一个完整的框架，因为组件需要的全部代码(可以将其理解为运行时依赖)在编译时都已注入组件中，

这极大地压缩了组件。结合 TreeShaking 技术，通过 Svelte 开发的页面所需的静态资源非常少。

本书系统讲述 Svelte，全面介绍 Svelte 的功能特性及 API。以一个打包行李的应用程序贯穿始终；在每章讲解相关的技术后，会将技术应用于示例，逐步完善。本书附录中介绍了如何为 Svelte 配置 ESLint、Prettier、VSCode、Snowpack，还介绍 REST 服务和 MongoDB 的基础知识。读者可以根据自身需求，优先阅读附录部分或将其作为参考资料。

本书由颜宇、周轶、王威共同翻译，三位译者从事专业前端开发工作均已超过 8 年，都供职于国内的一流互联网公司，有着丰富的前端开发经验以及扎实的技术功底。其中第 1、2、10、11、12、13、14 章以及附录 F、G 的内容，由颜宇翻译；第 15、16、17、18、19、20、21 章以及附录 A、C、E 的内容，由周轶翻译；第 3、4、5、6、7、8、9 章以及附录 B、D 的内容，由王威翻译。在前端开发领域日新月异、蓬勃发展的今天，各种新技术层出不穷；由于译者的自身局限，书中或存纰漏，希望各位读者不吝指正，我们也会及时答复并修改。

最后，在本书翻译过程中清华大学出版社为三位译者提供了热情的帮助。由衷感谢出版社的各位编辑老师，正是他们辛勤的付出和高质量的工作才保证了本书的顺利出版。

作 者 简 介

R. Mark Volkmann 从 1996 年开始就提供软件咨询和培训服务，目前是位于圣路易斯的 Object Computing 公司的合作人。作为一名资深的咨询顾问，Mark 为很多公司提供 JavaScript、Node.js、Svelte、React、Vue、Angular 等方面的帮助，创建并讲授了许多课程，包括 React、Vue、AngularJS、Node.js、jQuery、JavaScript、HTML5、CSS3、Ruby、Java 和 XML。他经常面向圣路易斯地区的用户发表演讲，并出席各种会议，包括 Nordic.js、Jfokus、NDC Oslo、Strange Loop、MidwestJS、No Fluff Just Stuff 和 XML DevCon。Mark 长期撰写各类关于软件开发的文章，这些文章收录在 https://objectcomputing.com/resources/publications/mark-volkmann。

在业余时间，Mark 爱好跑步，已经在 39 个州参加了 49 场马拉松比赛。

关于封面插图

《Svelte 和 Sapper 实战》一书封面插图的标题为 femme Corfiote，即“希腊科孚岛的女人”，选自 Grasset de Saint-Sauveur(1757—1810)所著的 *Costumes de Différents Pays*，该书于 1797 年在法国出版，是一本关于各国服饰的合集。其中每一页插图都是手工绘制和着色的，做工非常精致。Grasset de Saint-Sauveur 的收藏品种类丰富，形象地展示了 200 年前全球不同地区的文化多样性。那时人们说着不同的方言和语言，彼此之间的沟通很少。站在街头或乡村，仅从他们的衣着就很容易辨别出他们来自何处，从事哪种职业以及生活状况如何。

当前，人们的着装已经发生了改变，世界上不同国家和地区的多样性也已经消失了。现在很难区分不同大陆的居民，更不用说区分不同的城市、地区或国家的居民了。也许是我们个人生活的丰富多彩(或更多样化和快节奏)取代了文化的多样性。

在很难区分各种电脑书籍的时代里，Manning 以两个多世纪前世界各地的生活多样性为基础，在本书封面中重现了 Grasset de Saint-Sauveur 的作品，以表现计算机科技的创造性。

致　谢

许多作者都会在致谢时感谢他们的另一半在写作过程中给予的耐心。在撰写本书的过程中，我也深刻感受到妻子 Tami 作出的牺牲。她慷慨地给予我莫大的鼓励和充足的时间来完成本书。Tami，非常感谢你帮助我完成本书！

感谢 Manning 的组稿编辑 Jennifer Stout，她给予我恰当的修订、建议、鼓励和称赞，这些让我能够坚持完成工作。她的批注，比如“我喜欢这个”“你需要解释为什么需要这样做”，为本书的撰写提供了很大的帮助。

感谢 Manning 的技术编辑 Alain Couniot，他不断指出我没有说清楚，或指出无法令人信服之处。他还总是提醒我尽可能地提及 TypeScript。这些反馈令本书更加完美。

感谢 Manning 的技术编辑 Erik Vullings。他在我的代码示例基础上，提出了很多我从未设想过的功能，并对本书的文字部分给出了很多改进意见。我很钦佩他一丝不苟的态度。

感谢 MEAP 的审稿人 Peer Reynders，他梳理了本书所有的代码示例，指出了许多可以改进的地方。

感谢本书所有的审稿人：Adail Retamal、Amit Lamba、Clive Harber、Damian Esteban、David Cabrero Souto、David Paccoud、Dennis Reil、Gerd Klevesaat、Gustavo Filipe Ramos Gomes、Jonathan Cook、Kelum Senanayake、Konstantinos Leimonis、Matteo Gildone、Potito Coluccelli、Robert Walsh、Rodney Weis、Sander Zegveld、Sergio Arbeo 和 Tanya Wilke。众人拾柴火焰高！

感谢 Object Computing 公司的 Charles Sharp，我之前撰写的大部分书籍他都参与了编辑工作。Charles 在过去 10 多年花费了大量时间帮我提高写作水平。

感谢 Object Computing 公司的 Eldon Ahrold 审阅了本书第 21 章的内容。Eldon 是一位资深的移动和 Web 开发人员，能够从他那里得到意见和建议，我感到非常荣幸。

最后感谢 Ebrahim Moshiri 博士，24 年前他引荐我入职 Object Computing 公司，为我提供了持续学习的空间。如果没有他给我的帮助，我可能永远无法完成本书。

序　言

我是一名具有 37 年经验的专业软件开发人员，作为 Web 开发人员也有 10 年左右的时间了。我参与的项目使用过很多技术和框架，包括直接操作原生 DOM、jQuery、Ruby、Angular 1、React、Polymer、Angular 2+、Vue、Svelte，还有一些我已经记不起来了。

我很看重开发效率，非必要的复杂度会降低开发效率。Svelte 和 Sapper 有很多吸引我的优点，其中最打动我的一点是，与其他 Web 开发技术相比，它们更简单。我使用过很多开发框架，从经验看，Svelte 和 Sapper 可以极大地提高开发效率。

我第一次接触 Svelte 是通过 Svelte 的创始人 Rich Harris 的名为 Rethinking Reactivity 的演讲。这个演讲非常成功，显然我被其中能够降低 Web 开发复杂度的优点所吸引；于是开始进一步研究，首先写了一篇关于 Svelte 的长文，并在 Svelte 的用户群体中演讲，之后开始在会议上发表相关演讲。接下来就应该是撰写本书了。

本书几乎涵盖与 Svelte 和 Sapper 有关的所有方面，还包括了其他一些话题。阅读完本书后，我相信你将可在下一个 Web 项目中使用 Svelte 和 Sapper。

前　言

本书读者对象

《Svelte 和 Sapper 实战》一书面向希望提升开发效率的 Web 开发人员。也许你一直在思考是不是有一种更简单的开发方式来开发 Web 应用程序。恭喜你，答案就在本书。通过大量代码示例，你将学会如何使用 Svelte 和 Sapper 开发 Web 应用程序。

本书面向的读者需要具备一些基本的 HTML、CSS 和 JavaScript 知识。

- 关于 HTML，读者需要熟悉 html、head、link、style、script、body、div、span、p、ol、ul、li、input、textarea 和 select 等元素。
- 关于 CSS，读者需要理解 CSS 语法规则，什么是 CSS 的“级联”，了解基本的 CSS 选择器(包括元素名、类名、id、继承关系)，了解常用的 CSS 属性(包括 color、font-family、font-size、font-style 和 font-weight)以及 CSS 盒模型(content、padding、border 和 margin)。
- 关于 JavaScript，读者需要知道变量、字符串、数组、对象、函数、类、promise、解构、spread 操作符、export 和 import。

如果你发现了关于本书的任何问题，可以在网络上与我们沟通。我希望当你读完本书时，能够发现 Svelte 和 Sapper 的一些与众不同的地方，并在下一个项目中尝试使用它们。

本书的结构：路线图

本书分为四部分，包括 21 章。

第Ⅰ部分介绍 Svelte 和 Sapper。

- 第 1 章主要阐述 Svelte 和 Sapper 的一些过人之处，在结尾部分介绍 Svelte Native，并与其他主流的 Web 框架进行对比。此外，还介绍开发所需的工具。

- 第 2 章将带你使用在线工具(REPL)创建第一个 Svelte 应用程序。通过这种在线方式构建的应用程序可以在线保存，并与他人共享代码，还可将代码导出到本地继续进行开发。此外，还介绍在本地开发 Svelte 应用程序的步骤。

第Ⅱ部分将深入研究 Svelte，并提供大量代码示例供参考。

- 第 3 章介绍如何构建 Svelte 组件，包括其中的逻辑、标签和样式。随后介绍如何使用响应式语句以及模块上下文来管理组件状态。最后，将展示一个自定义组件的示例。
- 第 4 章涵盖 Svelte 的块结构，包括条件逻辑、迭代、promise 异步控制 HTML 标签等。{#if}实现了条件逻辑，{#each}实现了迭代遍历的功能，{#await}实现了 promise 异步功能。
- 第 5 章将带你探索组件之间的通信，包括 props、双向绑定、slot、事件和上下文。
- 第 6 章阐述如何使用 store 共享组件之间的数据。store 有四种类型：可读写、只读、派生和自定义。随后介绍如何使用 JavaScript 类创建 store 以及持久化 store 中的数据。
- 第 7 章展示在 Svelte 组件中操作 DOM 的几种方法，包括插入 HTML，使用“动作”获得 DOM 元素，在 Svelte 重新渲染后使用 tick 函数手动修改 DOM。最后，将展示一个对话框以及实现拖曳功能的示例。
- 第 8 章将讲解 Svelte 的生命周期函数，包括 onMount、beforeUpdate、afterUpdate 和 onDestroy。最后，展示一个基于现有 Svelte 生命周期函数自定义新的生命周期函数的示例。
- 第 9 章演示为 Svelte 应用程序添加页面路由的三种方法：手动路由、哈希路由以及使用 page.js 进行路由。我们将开发一个购物应用程序来演示这三种路由。此外，还可使用 Sapper 实现路由，相关内容将在第 16 章详细介绍。
- 第 10 章探讨在 Svelte 中对于动画的有力支持，详细介绍 svelte/animate、svelte/motion 和 svelte/transition 这三个包，以及在两个列表之间移动列表元素的两种方式：一种方式使用 fade 过渡效果和 flip 动画的组合，另一种方式使用 crossfade 过渡效果。最后讨论如何创建自定义动画以及如何使用过渡事件。
- 第 11 章展示如何调试 Svelte 应用程序。包括@debug 标签、使用 console 方法调试响应式语句以及配套的浏览器调试插件 svelte-devtools。
- 第 12 章演示 Svelte 应用程序的多种测试方法。Jest 和 svelte-testing-library 可用来执行单元测试。端到端的测试可使用 Cypress。Svelte 还为可访问

性提供了一些测试手段，如果你想进行额外的可访问性测试，可采用 Lighthouse、axe 和 WAVE。最后，可以使用 Storybook 展示和操作测试组件。

- 第 13 章将带你探索如何部署 Svelte 应用程序，包括手动部署一个 HTTP 服务器，以及如何使用 Netlify、VercelNow 和 Docker。
- 第 14 章主要介绍 Svelte 的其他一些知识点，包括表单验证、CSS 变量、使用“特殊元素”创建 Svelte 组件库，以及利用 Svelte 组件生成 Web Components。

第Ⅲ部分将深入研究 Sapper。Sapper 是一个基于 Svelte 的 Web 应用程序开发框架。

- 第 15 章将使用 Sapper 重构第 9 章中的购物应用程序，这将是你的第一个 Sapper 应用程序。
- 第 16 章将全面介绍 Sapper。首先介绍 Sapper 应用程序的结构，之后是 Sapper 的一些功能，包括页面路由、页面布局、预加载、预请求以及代码分割。
- 第 17 章将探索 Sapper 的服务器路由，通过服务器路由，我们的项目就不仅是 Web 应用程序的客户端了，还具备了提供 API 服务的能力。你将学会创建、查询、更新、删除(CRUD)等一系列服务是如何实现的。
- 第 18 章展示如何将 Sapper 应用程序部署成一个静态站点。对于那些采用 HTML 作为页面展示载体的应用程序来说，这非常有用。最后，将带你一起实现一个类似的应用程序，其中包括两个页面，一个是石头剪刀布的游戏，另一个是我家的狗。
- 第 19 章描述如何使用 service worker 实现 Sapper 的离线功能。将介绍以下内容：多种缓存策略；Sapper 内置 service worker 的一些细节，包括 install、activate 和 fetch 等 service worker 事件；如何启用 HTTPS。最后，带你一起体验 Sapper 的离线功能是如何发挥作用的。

第Ⅳ部分将不局限于 Svelte 和 Sapper。

- 第 20 章将带你探索对于高级语法的预处理技术，包括 Sass、TypeScript 和 Markdown，并提供这些预处理技术对应的示例。
- 第 21 章将介绍 Svelte Native 以及如何使用 Svelte 和 NativeScript 来开发 Android 端和 iOS 端的移动应用程序。我们将利用 REPL 创建两个在线的 Svelte Native 应用程序，使用 REPL 的好处是并不需要在计算机中安装任何软件。同时将提供一个示例来详细解释显示、表单、动作、对话框、布局和导航等组件的实现细节，以及如何为 Svelte Native 组件添加样式。最后，介绍 NativeScript UI 组件库，并使用其中的组件 RadSideDrawer

创建一个示例应用程序。

最后一章结束后我们的学习并没有告一段落！还有七个附录在等着你。

- 附录 A 整理与 Svelte、Sapper 和 Svelte Native 相关的资料的链接。
- 附录 B 介绍如何使用 Fetch API 请求 REST 服务。
- 附录 C 介绍在第 17 章中使用过的 MongoDB 数据库。
- 附录 D 介绍如何配置和使用 ESLint 来检查应用程序中的问题。
- 附录 E 介绍如何配置和使用 Prettier 来格式化 Svelte 和 Sapper 应用程序中的代码。
- 附录 F 介绍在使用 VS Code 开发 Svelte 和 Sapper 应用程序时所用到的几种插件。
- 附录 G 介绍如何使用 Snowpack 构建 Svelte 应用程序。Snowpack 与传统的编译工具(如 Webpack、Rollup 和 Parcel)相比，是一种更高效的构建 Web 应用程序的工具。

在本书中，我们将开发一个 Travel Packing 应用程序。本书中的大部分章节都围绕这个应用程序展开讨论，并以它为基础添加对应的功能。

对于 Svelte 新手来说，应该首先按照顺序读完本书的第 1～8 章，这八章涵盖了 Svelte 的核心理念。之后可以根据兴趣和需要有选择地进行阅读。当然，如果你有使用 Svelte 的经验，那么可以根据兴趣从本书的任何章节开始阅读。

关于代码

可扫描封底二维码来下载相关代码。

本书中包含了很多用于演示的源代码，既有通过编号列举出来的，也有与正文混排在一起的。上述两种源代码会被格式化为等宽字体。有一些代码还会被特意**加粗**以强调其与之前章节中代码的区别，比如当为之前的代码添加一段新功能，新功能的代码就会被加粗。

有些情况下，源代码已经被重新格式化过了；为适应本书印刷的排版，额外增加了换行符，并重新设计了缩进。然而在极少数情况下，换行符和缩进也无法解决排版混乱的问题，为此会增加续行标记(➥)来调整排版。此外，代码的注释会被从代码清单中删除，而是在代码清单外的其他地方标注出来，以强调注释的重要性。

其他在线资源

附录 A 列出了一系列在线资源。其中的大多数都与 Svelte 和 Sapper 有直接关

系，但也有一些涵盖了适合于所有 Web 开发的内容。

关于彩图

正文中有时提到界面颜色，由于本书是黑白印刷，将无法显示彩色。请读者在实际操作过程中从计算机屏幕上查看；另外，也可扫描封底二维码下载彩图。

目　录

第 I 部分　起步

欢迎开始《Svelte 和 Sapper 实战》学习之旅！首先我们会聊聊为什么 Svelte 和 Sapper 如此吸引人，之后会学习 Svelte Native，并将 Svelte 与其他流行的 Web 框架进行对比。此外，还会带你深入了解开发 Svelte 所需的工具。我们还会使用在线工具 REPL 构建第一个 Svelte 应用程序。使用 REPL 构建的应用程序可以被保存并共享给其他开发人员，也可将代码下载到本地继续开发。

第 1 章

初识Svelte

本章内容：

- Svelte
- Sapper
- Svelte Native

Svelte(https://svelte.dev/)是一个基于 JavaScript 的用来开发 Web 应用程序的工具，可以将其视为 React、Vue、Angular 等 Web 框架的替代品。与这些框架一样，Svelte 专注于 UI 组件和交互，每个 UI 组件均是独立的、可复用的。Svelte 可将复杂的 UI 界面拆分成多个独立的组件，单独设计和开发每一个组件。

与其他 Web 框架项目相比，Svelte 具备以下优点：

- 实现同样的功能，Svelte 所需的代码更少。
- 打包后的代码更小，有利于减少页面加载时间。
- 无论是组件内部还是组件之间的状态通信，用 Svelte 管理都更简单(状态管理器主要用来管理应用程序的数据，并驱动应用程序响应数据的变化)。

Sapper(https://sapper.svelte.dev/)是一个基于 Svelte 的开发框架，能帮助开发人员创建功能更强大的 Web 应用程序。Sapper 在 Svelte 之上封装了更多功能，包括页面路由、服务端渲染、代码分割以及静态站点生成。如果开发人员不需要上述这些功能，或者想自己实现这些功能，那么完全可以只使用 Svelte 进行开发。

Svelte Native(https://svelte-native.technology/)与 Sapper 一样，也构建在 Svelte 之上。结合使用 NativeScript 和 Svelte Native，可开发出 Android 端和 iOS 端的移动应用程序。

注意　在本书中提到创建 Web 应用程序的一些技术时，有人喜欢称这些技术为“库”，而我更倾向称之为“框架”。

1.1　Svelte 介绍

首先问一个问题：我们是否真的需要另一种工具帮我们构建 Web 应用程序？

只有这种工具能够带来足够多的收益，我们才应该去学习如何使用它。比如，用更少的代码实现同样的功能，运行时的性能更优，浏览器需要下载的代码更小。

是的！Svelte 为我们带来了上面这些好处，甚至更多。

与其他框架类似，Svelte 可以创建一个完整的 Web 应用程序。Svelte 组件可以在独立的 Web 应用程序中使用，也可以被抽象成一个库在多个应用程序中复用。还可以利用 Svelte 创建自定义元素(Web Components)，自定义元素可以与其他 Web 框架结合使用或者脱离任何 Web 框架单独使用。

Rich Harris 从 2016 年起开始开发 Svelte，之前曾经创建了 Ractive Web 框架(https://ractive.js.org/)。Vue 中的部分实现也受到了 Ractive 的启发。此外，Rich Harris 还创建了 Rollup 模块打包工具，作为 Webpack 和 Parcel 的替代方案。

单词 Svelte 的意思是“苗条”，恰当地表明了 Svelte 本身的大小，也表明由 Svelte 构建的包都非常小。

1.1.1　为什么选择 Svelte

与主流的 Web 框架相比，Svelte 有很多优势，下面总结其最重要的几个优势。

1. Svelte 是一款编译器

其他主流的 Web 框架中都包含大型的运行时环境，以便支持这些框架的主要功能。Svelte 并不是一个运行时环境，而是一个使用 TypeScript 实现的 Web 应用程序的编译器。

注意　所谓编译器指在代码层面将一种编程语言转换成另一种编程语言的软件。比较典型的场景是将高级语言(如 GO 或 Java)转换成低级语言(如机器码或字节码)。

注意　TypeScript 是一个开源的编程语言，是 JavaScript 的超集。由 TypeScript 开发的程序将被编译为 JavaScript。TypeScript 在 JavaScript 的基础上增加了很多功能，其中最重要的功能就是为 JavaScript 增加了类型。TypeScript 由微软开发并维护。

Svelte UI 组件在以.svelte 为后缀的文件中定义。这些文件囊括了 JavaScript、CSS 以及 HTML。以一个实现了登录功能的 Svelte 组件为例，组件中会包括登录表单的 HTML 元素、与之对应的 CSS 样式以及 JavaScript 代码。JavaScript 实现了单击登录按钮时向登录验证服务发送数据的功能。

Svelte 编辑器会将.svelte 文件编译成 JavaScript 和 CSS 代码。这种做法有很多好处，其中一个好处是即使 Svelte 增加了新功能，已经部署了的应用程序也不会受到任何影响，代码的大小不会发生变化。Svelte 编译器在构建时仅会打包应用程序中实际引用的 Svelte 代码。

2. Svelte 打包后变得更小

对于同样的应用程序，与其他 Web 框架项目相比，Svelte 打包后的体积会小得多。这意味着浏览器下载 Svelte 应用程序会更快。

注意 在 Web 应用程序领域里，“包”是一个应用程序需要的所有 JavaScript 代码在经过编译、优化和压缩处理后生成的文件。

Svelte 仅处理引用到的代码，而不会将整个框架都打包。这使得 Svelte 应用程序的包非常小。

注意 现在，主流的框架都内置了 tree shaking 功能，以消除未引用的代码。但 Svelte 在消除方面更加彻底。

FreeCodeCamp 发布的“A Real World Comparison of Front-End Frameworks with Benchmarks(2019 update)”(http://mng.bz/8pxz)报告中，对一些主流框架进行了比较。通过 RealWorld App 分别使用不同框架创建了一个类似 Medium.com 的社交博客站点：Conduit。

下面列出使用不同框架构建并经 gzip 压缩后生成的包的大小。

- Angular + ngrx：134 KB
- React + Redux：193 KB
- Vue：41.8 KB
- Svelte：9.7 KB

Svelte 的优势一目了然。

3. Svelte 在未使用虚拟 DOM 的情况下实现了响应式设计

React 和 Vue 这类 Web 框架，在数据发生变化时，都使用虚拟 DOM 技术更新真实的 DOM，这对性能有很大帮助。当组件状态发生改变时，框架首先会在内存中构建一个新版本的 DOM，之后将其与前一个版本的 DOM 进行对比。只有这两个版本存在差异时，才会更新真正的 DOM。这种方式会比每次都更新真实 DOM 的效率高得多，但也有一个问题：创建虚拟 DOM 并将其与之前的版本进行比较，这个过程也需要一定的执行时间。

响应式设计指的是响应应用程序或者组件状态变化而更新 DOM 的能力。Svelte 为了实现响应式设计，会跟踪影响各个组件渲染的顶层组件变量的变化，并且仅更新受影响的那部分 DOM，而不是更新整个组件。这样做令 Svelte 与其他框架相比在保持 DOM 和状态一致性方面更加出色。

4. Svelte 性能更好

Stefan Krause 在 https://krausest.github.io/js-framework-benchmark/current.html 中给出了一份

性能测试报告。通过渲染一个 4 列 1000 行的表格来测试性能。你可以选择希望对比的框架并查看统计数据。比如，选择"angular-v8.0.1-keyed""react-v16.8.6-keyed""svelte-v3.5.1-keyed"和"vue-v2.6.2-keyed"，对应用程序的启动速度进行测试，结果如图 1.1 所示。Svelte 的性能明显优于其他框架。

名称	svelte-v3.5.1-keyed	vue-v2.6.2-keyed	react-v16.8.6-keyed	angular-v8.0.1-keyed
脚本启动时间 解析/编译/执行页面中所有脚本需要的时间，单位为毫秒	19.5±2.4 (1.00)	59.6±28.6 (3.06)	55.6±45.2 (2.85)	159.8±8.8 (8.21)
脚本文件的大小，单位为KB 页面中所有压缩处理后的静态资源在网络传输过程中使用的流量	145.7±0.0 (1.00)	211.2±0.0 (1.45)	260.8±0.0 (1.79)	295.5±0.0 (2.03)

图 1.1　测试启动时间和加载时间

注意　表头中的 keyed 单词表示框架会在数据和 DOM 元素之间创建一种联系。当数据发生变化时，与之关联的 DOM 也会更新。向数组中添加数据或从其中删除数据将引起 DOM 元素的添加和删除。图 1.1 中的每一个参与测试的框架都标注了 keyed，因为采用这种方式更新 DOM 非常普遍，也更高效。

5. Svelte 占用内存更少

对于一些较旧的电脑或移动设备来说，它们的可用内存往往很少，这意味着在这些设备上，Web 应用程序占用的内存越少越好。

上一节的测试站点也可用来衡量内存使用情况，图 1.2 展示了测试结果。从其中可以发现，与其他框架相比，Svelte 应用程序通常占用更少的内存。

6. Svelte 不依附于任何 JavaScript 容器

.svelte 文件不会为组件定义任何 JavaScript 容器，取而代之的是，组件由一个 script 元素、一段用于渲染的 HTML 以及一个 style 元素组成。

这样的方式比其他大多数 Web 框架都简洁，定义一个组件的代码量更少，需要考虑的 JavaScript 概念也更少。比如 Angular 使用类来定义组件，React 中的组件则是由函数或者类定义的，Vue2 中组件由一个对象定义，而在 Vue3 中又改为由函数定义组件。

名称	svelte-v3.5.1-keyed	vue-v2.6.2-keyed	react-v16.8.6-keyed	angular-v8.0.1-keyed
页面加载后占用的内存 页面加载后的内存占用情况	1.9±0.0 (1.00)	2.1±0.0 (1.13)	2.3±0.0 (1.23)	4.8±0.0 (2.54)
页面运行时占用的内存 在页面中添加 1000 行数据后的内存占用情况	3.9±0.0 (1.00)	7.1±0.0 (1.81)	6.9±0.0 (1.76)	9.1±0.0 (2.34)
查找并更新1000行数据中的第10行 **(连续执行5次)** 更新页面中的第 10 行数据，并将上述操作连续执行 5 次后的内存占用情况	4.3±0.0 (1.00)	7.5±0.0 (1.76)	8.0±0.0 (1.89)	9.5±0.0 (2.23)
重复添加1000行数据 **(连续执行5次)** 在页面中添加 1000 行数据，并将上述操作连续执行 5 次后的内存占用情况	4.5±0.0 (1.00)	7.7±0.0 (1.71)	8.9±0.0 (1.98)	9.9±0.1 (2.20)
新建/删除1000行数据 **(连续执行5次)** 在页面中先添加 1000 行数据，之后删除这 1000 行数据，将上述操作连续执行 5 次后的内存占用情况	3.2±0.0 (1.00)	3.8±0.0 (1.20)	4.7±0.1 (1.48)	6.6±0.0 (2.07)

图 1.2 测试内存占用情况(MB)

7. Svelte 样式是非全局的

Svelte 组件中定义的样式默认仅对当前组件生效。这意味在一个.svelte 文件中定义的样式不会意外“泄露”从而污染其他组件的样式。

不同框架对于样式的处理方式是不同的。在 Angular 中，默认情况下组件中定义的样式仅对当前组件生效。在 Vue 中，只有当样式被标记为 scoped 之后才会只在当前组件中生效。React 并不提供控制样式作用范围的功能，这也是 CSS-in-JS 会在 React 中流行的原因。而对于 Svelte 来说，CSS-in-JS 并没有什么用处。

8. Svelte 可以定义全局样式

Svelte 明确指定了文件 public/global.css 用来定义全局样式，在其中定义的样式会影响所有组件。

9. Svelte 管理状态更简单

与其他框架相比，Svelte 极大地简化了应用程序和组件的状态管理，提供了上下文、store 以及模块上下文来管理状态。本书将详细介绍这三种方式。

10. Svelte 支持双向数据绑定

使用 Svelte 能够很轻松地将表单控件中的某一个值与组件进行绑定。表单控件包括 input 元素、textarea 元素和 select 元素。.svelte 文件中定义的顶层变量会自动成为该组件的状态。

当绑定的变量发生变化时，其所关联的表单控件中的值会自动更新。相反，当用户操作一个绑定的表单控件，更新了它的值，其所关联的值也会自动更新。

11. Svelte 中实现动画效果更简单

Svelte 内置了对各种动画的支持，为应用程序添加动画效果变得格外简单。这会提高开发人员使用动画的动力，从而提供更好的用户体验。

TODO 应用程序中展示了一些动画的示例，包括添加代办事项的淡入效果，删除代办事项的淡出效果。如果在分类列表中维护了代办事项，那么代办事项在不同分类之间移动也可以加上动画效果，使整个过程更加平滑。

12. Svelte 更关注无障碍访问

Svelte 在运行时环境会针对无障碍访问方面的错误进行检查，并给出警告。例如，如果一个 img 元素没有设置 alt 属性，则会被警告。对于一些特殊人士来说，在 Web 浏览器中使用 Svelte 应用程序会更容易。

1.1.2　重新思考响应式设计

在 Web 应用程序中，响应式设计指的是当数据(状态)发生改变时，DOM 会自动更新的一种技术。这与我们常用的电子表格类似，当一个单元格中的内容是由另一个单元格的值计算而得，那么当作为计算基数的单元格中的内容发生变化时，这个单元格中的内容会随之变化。

与其他框架相比，使用 Svelte 实现响应式设计更容易。Svelte 使用一种独有的方式管理组件中的状态，这种方式依赖于对组件顶级变量的监听(3.9 节中会详细介绍)。在 Svelte 的不同组件之间共享状态也非常简单。

HTML DOM

一个 Web 页面在内存中被解析为一个 HTML DOM 节点树，每个节点是一个 JavaScript 对象。其中的一个 JavaScript 对象表示整个页面文档，它持有页面中其他 DOM 对象的引用。每一个 DOM 对象都有相应的方法，调用这些方法能够获得节点的信息，为节点添加子节点、注册事件等。修改 DOM 对象，浏览器的显示也会发生变化。

一个简单的 HTML 文档如下：

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Page</title>
  </head>
  <script>
    // For exploring the DOM from the DevTools console ...
```

```
    window.onload = () => console.dir(document);
  </script>
  <body>
    <h1>My Page</h1>
<p>I like these colors:</p>
    <ul>
      <li>yellow</li>
      <li>orange</li>
    </ul>
  </body>
</html>
```

图 1.3 显示了浏览器中的 HTML 文档在内存中是如何转化为 DOM 节点树的。

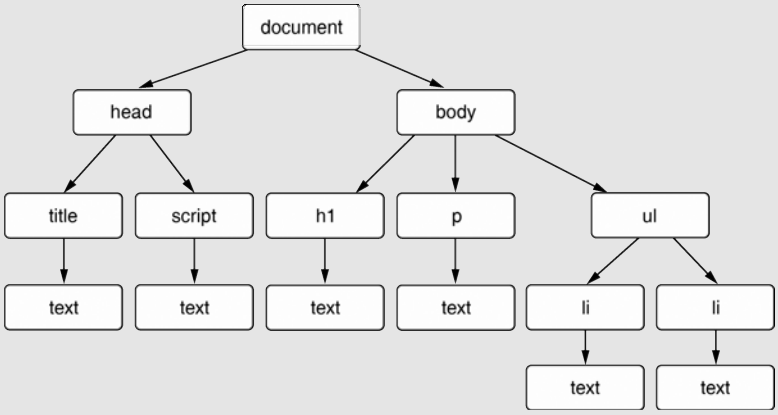

图 1.3 DOM 节点树

Rich Harris 曾经在多个场合发表了“重新响应式设计”的演讲，其中清晰地讲述了创造 Svelte 的动机(www.youtube.com/watch?v=gJ2P6hGwcgo)。演讲中最重要的十个观点都与 Svelte 的特性有关：

(1) 实现真正的响应式编程，就像我们在电子表格中所看到的那样。当一个值发生改变时，其他值也随之改变。

(2) 避免使用虚拟 DOM。Rich 认为：“作为工程师，我们应该拒绝所有的低效”。

(3) 开发更少的代码。对于开发人员来说代码越少越好，对于性能也是如此。Rich 认为：“要提高性能，唯一可靠的方式是精简代码”。

(4) 为无障碍访问相关的问题提供警告。

(5) 样式仅在在组件内部生效，防止样式泄露和全局污染。

(6) 识别没有引用的 CSS 样式，并删除它们。

(7) 使用 CSS 提升动画性能，令添加过渡和动画效果更加方便。

(8) 当为框架添加了一些新功能时，如果这些新功能并没有被引用，应用程序包的大小将不会增加。

(9) 基于 Svelte 创建的 Sapper 为开发人员提供了页面路由、代码分割、服务端渲染等一系列功能。

(10) 可使用 Svelte Native 代替 React Native 开发移动端应用程序。

1.1.3 Svelte 的缺点

使用 Svelte 可以开发几乎所有的 Web 应用程序。然后在一些特殊场景下，开发人员可能会考虑使用其他框架。

Svelte 使用 TypeScript 实现，但并不支持使用 TypeScript 定义组件，需要额外的配置和工具才能在 Svelte 中使用 TypeScript。Svelte 正在解决这个问题。

注意 第 20 章将介绍 Svelte 预处理器，以便支持在.svelte 文件中使用 TypeScript；还将介绍如何使用命令行工具 svelte-check 检查.svelte 文件中的错误，包括 TypeScript 错误。附录 F 讨论了 VS Code 插件 Svelte for VS Code，该插件基于 Svelte Language Server 检查.svelte 文件中的错误，包括 TypeScript。

Svelte 对 IE 浏览器的兼容性并不好。需要额外的 polyfill 以便支持在 IE11 中运行 Svelte 应用程序(https://github.com/sveltejs/svelte/issues/2621 和 https://mvolkmann.github.io/blog/topics/#/blog/svelte/supporting-ie11/)。对于 IE 11 之前的 IE 浏览器，Svelte 不提供任何支持手段。幸运的是，现在大部分 Web 应用程序都不需要运行在 IE 浏览器上。

一些流行的 Web 框架(如 React 和 Vue)可在组件中定义不同的方法，用于创建组件中不同的部分，而 Svelte 做不到这一点。在 Svelte 中，所有 HTML 都是与 JavaScript 解耦的。这样做的一个缺点是，如果我们想按模块方式管理 HTML，将不得不为每个独立的 HTML 创建.svelte 文件。尽管如此，这些为管理 HTML 而创建的.svelte 文件中并没有繁复的样板代码，创建起来很容易。

主流的 Web 框架因其存在的时间较长，第三方支持比 Svelte 要好得多；比如各种各样的组件库。Svelte 在这方面的第三方支持目前较少，但数量在持续增加。

你需要在 Svelte 的优点和缺点之间进行衡量，仔细思考得失。不过我有信心，当你体验过 Svelte 后，你会跟我一样觉得它值得一试。

1.1.4 Svelte 原理

图 1.4 展示了 Svelte 编译器的组成部分。用户界面(UI)组件定义在.svelte 文件中。组件由 JavaScript 代码(定义在 script 元素中)、CSS 样式(定义在 style 元素中)以及用于渲染的 HTML 组成。

.svelte 文件中可以引用其他.svelte 文件，并将其作为子组件使用。比如，对于管理待办事项的应用程序来说，最重要的组成部分是 TodoList 组件，TodoList 组件中引用了 Todo 组件，每个 Todo 组件表示一个单独的待办事项，由复选框、待办详情和删除按钮组成。

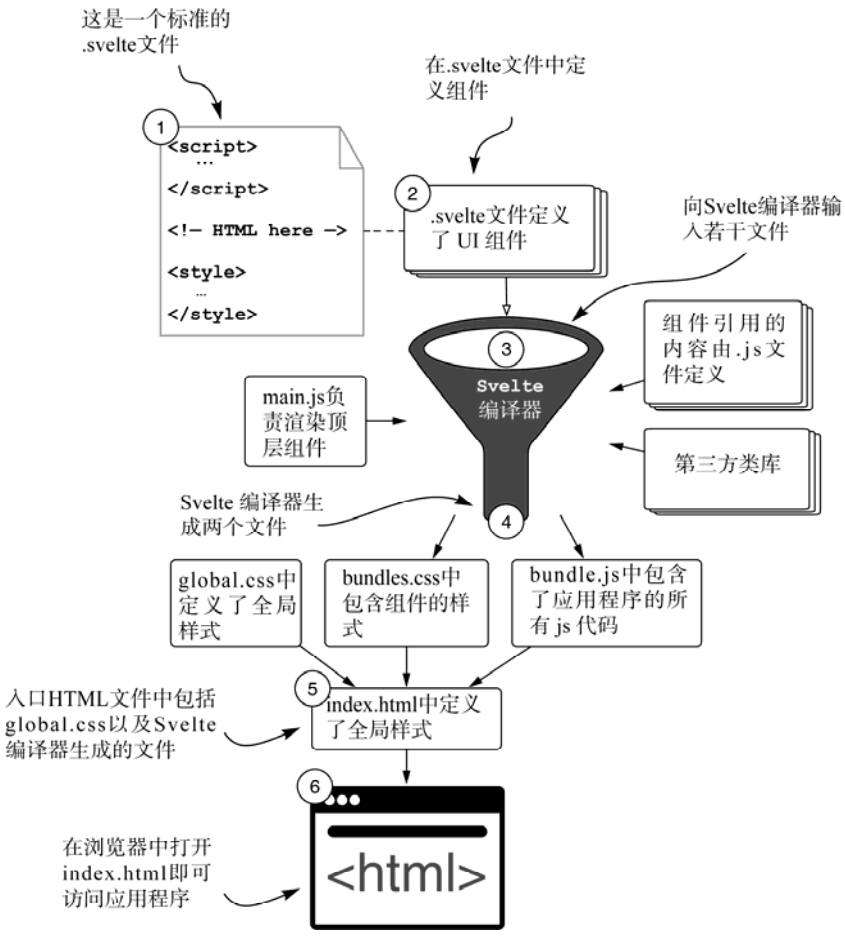

图 1.4 Svelte 编译器的输入与输出

.svelte 文件还可以引用.js 文件和第三方库(通常从 npm 获得)导出的对象，调用其中定义的函数，或者读取其中的变量。比如，在 Svelte 应用程序中可以引用流行的 Lodash 库，使用 Lodash 提供的各种函数。

Svelte 编译器将所有代码编译为一个 bundle.js 文件和一个 bundle.css 文件，这两个文件分别包含应用程序的 JavaScript 代码和 CSS 代码。Svelte 是按需构建的，因此 bundle.js 文件中仅包含 Svelte 框架里真正用到的那部分代码。

global.css 文件中定义了全局样式。index.html 是页面入口文件，其中引用了 global.css 以及 Svelte 编译出的包文件。当用户从浏览器中访问应用程序时，会直接加载 index.html。

Svelte 编译器还会生成文件 bundle.css.map 和 bundle.js.map，保存了压缩后的代码与源代码之间的映射关系。通常调试工具(如浏览器内置的调试工具)会读取这些文件，帮助开发人员更方便地调试。

1.1.5　Svelte“消失”了？

对于大多数 Web 框架，开发人员编写的代码和框架的代码会被打包在一起，浏览器会加载所有代码从而运行应用程序。如果框架实现了很多功能，那么即使开发人员只使用了其中一小部分功能，整个框架也会被打包。

有开发人员发现在使用 Svelte 框架时，一旦应用程序被构建后，Svelte 就“消失”了。此处“消失”指的是经过 Svelte 编译的代码仅包含 Svelte 框架中的一部分代码，而有些代码不见了。

node_modules/svelte 目录下的 JavaScript 文件定义了 Svelte 的主要功能。其中 internal.js 中包括核心功能，目前大概有 1400 行代码。其他一些额外功能被定义在 easing.js、motion.js、register.js、store.js 和 transition.js 中。

稍后将介绍如何通过 npm run dev 或 npm run build 命令编译 Svelte 应用程序，编译成功后会在 public 目录中生成 build.js 和 build.css 文件。应用程序中使用的与 Svelte 有关的代码会复制到 bundle.js 文件的最上方。对于中小型应用程序(如待办事项应用程序)来说，大约有 500 行代码左右。

Svelte 代码并非“消失”了，只不过与其他框架相比，Svelte 代码很少而已。

1.2　Sapper 介绍

Sapper 是一个基于 Svelte 的框架。它包括 Svelte 的所有功能，并额外添加了其他许多功能。当然，这些功能也可由开发人员在开发 Svelte 应用程序时自行实现，但使用 Sapper 更便捷。

Sapper 这个名字有两个含义。首先 Sapper 是“Svelte app maker”的缩写。其次在英语中 Sapper 的含义是：负责修筑和修葺道路、桥梁，埋设和清除地雷的士兵。从某种意义上讲，这也恰当描述了 Sapper 与 Svelte 之间的关系。

Sapper 由 Svelte 的作者 Rich Harris 创建和维护，并且有很多贡献者共同参与其中。

1.2.1　为什么选择 Sapper？

下面列出 Sapper 中新增的功能，后续章节将详细介绍这些功能。

- Sapper 提供了页面路由。页面路由将应用程序的“页面”与 URL 绑定在一起，并定义了如何在 HTML 中描述页面之间的导航。Sapper 的页面路由完全由目录结构和文件命名约定所决定，与那些需要通过额外类库配置路由的方式相比，这种方式更容易理解和实现。
- Sapper 支持页面布局。页面布局为应用程序中的页面提供了一个通用布局。比如，大部分页面可能包括一个通用的 header 区域、footer 区域以及 nav 区域。通过页面布局可将通用代码从这些页面中抽离出来。
- Sapper 提供服务端渲染(SSR)。当用户访问页面时，服务端渲染能将生成 HTML 的工作从浏览器端转移到服务端。这使得浏览器在 JavaScript 被下载之前就能将页面渲染出来，从而提供更好的用户体验。同时，服务端渲染还能提供更好的 SEO 能力。应用程序的每

一个会话中，第一次访问应用程序时会执行服务端渲染，之后这个会话内其他对应用程序的访问则不会触发服务端渲染。

- Sapper 支持服务端路由。利用服务端路由，我们可在原本仅支持客户端的项目中增加基于 node 的 API 服务。这对于那些全栈工程师来说非常方便。服务端路由是一个可选功能，开发人员也可以选择不开启此功能，转而使用其他任何一种技术实现应用程序所需要的 API/REST 服务。

REST

REST 即描述性状态传递(Representational State Transfer, REST)，首次提出是在 2000 年 Roy Fielding 的博士论文中。REST 仅仅是一种风格，而不是一种标准或 API。

下面列出 REST 背后的主要理念。

(1) 软件通过提供资源标识符和所需的媒体类型向服务请求“资源”。资源标识符可以是一个 URL，可以通过Ajax(https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX)进行请求。

(2) 在资源的“描述”被返回后，其中包括一个字符串序列以及描述它的元数据。比如，可以是 JSON、HTML Header 中的键值对或者一个图片等。一个描述可包括其他资源的标识符。

(3) 组件在获得请求的返回后，将根据返回的数据“转移”到另一种“状态”。

通常，REST 使用 HTTP 作为请求和返回的载体。HTTP 中规定了几个动词表示请求的方法：POST、GET、PUT 和 DELETE，这些动词映射到 CRUD 操作中，分别表示了创建、查询、更新和删除。

- Sapper 支持代码分割。代码分割技术使得只有在访问页面时才会下载这个页面所需的 JavaScript 代码。与无论访问什么页面都下载全部代码的方式相比，代码分割的性能更优秀。
- Sapper 支持预请求。预请求根据鼠标的轨迹来预测用户将访问的下一个页面，从而提供更快的页面加载速度。每个页面链接都可以配置预请求。当用户操作鼠标并停留在某个页面链接上，Sapper 将开始下载这个页面所需的 JavaScript，同时可以发起对于 API 服务的调用，获取页面需要的数据。
- Sapper 支持生成静态站点。在构建静态站点时，会访问 Svelte Web 应用程序，提前创建应用程序的页面。这种方式生成的网站性能极佳，因为所有页面都是预先渲染好的。

采用这种方式创建的站点，并不一定要完全的静态化。站点仍可包括 JavaScript 代码，并利用 Svelte 的响应式来更新 DOM。

- Sapper 支持离线使用。在没有网络的情况下，Sapper Web 应用程序使用 service worker 在一定程度上保证应用程序的可用性。离线功能是通过缓存指定的文件以及请求返回来实现的，当没有网络时会使用之前缓存的版本。
- Sapper 支持端到端的测试。Sapper 应用程序支持使用 Cypress 模仿用户操作应用程序的行为进行测试，包括登录、跳转到不同页面、输入文字、单击按钮等。

1.2.2　Sapper 的工作方式

Sapper 应用程序中的每个“页面”都被定义在 src/routes 目录下的.svelte 文件中。这些页面中使用的组件同样被定义在.svelte 文件中，存放在 src/components 目录中。页面之间的跳转通过 HTML 的锚元素(<a>标签)实现，只需要将锚元素的 href 属性设置为要跳转的目标页面的名称即可。

Polka 是一个服务端的库，它是基于 Node 开发的，拥有与 Express 几乎相同的 API，但在性能和大小方面更有优势。默认情况下，Polka 会为 Sapper 应用程序提供服务端的相关功能。如有必要，开发人员也可很容易地用 Express 替换 Polka。

如前面讲到的，Svelte 编译器会生成文件bundle.js 和 bundle.css，分别包括了整个应用程序的 JavaScript 代码和 CSS 样式。而 Sapper 与 Svelte 的处理方式不同，会为应用程序的每个页面生成.js 和.css 文件。这些文件存放在__sapper__/build/client 目录中。一旦页面加载，只有与这个页面相关的 JavaScript 和 CSS 才会被下载。

1.2.3　Sapper 适用的场景

从 Svelte 切换到 Sapper 非常简单，只需要在单独的目录下创建一个新项目即可。稍后将介绍如何通过 npx 命令初始化 Sapper 项目。

Sapper 应用程序默认开启了上述所有功能，并不需要添加额外的库。

如果你认可 Sapper 实现的这些功能，并且恰好开发的应用程序又需要这些功能，那么建议采用 Sapper 而不是 Svelte。比如，对于大部分应用程序来说，Sapper 提供的路由方法已经足够满足一般场景了，但开发人员也可能在路由策略方面有额外的需求，此时他们可以选择使用 Svelte，自行实现页面路由功能。

1.2.4　Sapper 不适用的场景

在本书编写时，Sapper 尚未发布 1.0.0 版本，这意味着在后面的升级中，现有功能可能会有破坏性改动。考虑到这点，有些开发人员可能等到 Sapper 发布正式版本才会在生产环境中使用。

尽管如此，鉴于 Sapper 提供了很多非常有用且优秀的功能，建议可直接使用 Sapper，而不是在 Svelte 应用程序重新开发这些功能。

1.3　Svelte Native 介绍

NativeScript(https://nativescript.org/)是一个用于创建 Android 和 iOS 移动应用程序的框架，由 Telerik 创建和维护。Telerik 于 2014 年被 Progress Software 收购。

NativeScript 使用 XML 语法(包括自定义元素)、CSS 和 JavaScript 构建应用程序。既可以使用 Angular、React、Svelte 或者 Vue 之类的框架开发，也可在不使用任何框架的情况下直接开发。

NativeScript 应用程序并没有采用 Web view，而是使用移动端的原生组件。从这个角度看，NativeScript 类似于 React Native。通过插件访问移动端设备提供的所有 API。

Svelte Native(https://svelte-native.technology/)建立在 NativeScript 之上，开发人员能够使用 Svelte 开发移动应用。Svelte Native 对 NativeScript 的 API 加了一层封装，确保对 NativeScript 未来版本的兼容性。

第 21 章将详细介绍如何使用 Svelte Native 应用程序。

1.4 Svelte 与其他框架对比

Svelte 与现在主流的框架存在一些不同之处，让我们一起看看都有哪些差异。

1.4.1 Angular

与 Angular 相比，同样的功能，Svelte 的代码更“苗条”。

Angular 更受 Java 和 C#开发人员的偏爱，他们更习惯编写大量类和使用依赖注入。

Angular 应用程序大量采用了 effects，并使用 RxJS 和 ngrx/store 之类的库，这使得 Angular 的学习成本更高。

1.4.2 React

React 中使用 JSX(JavaScript XML 的简写)描述组件。JSX 是 JavaScript 语法的扩展，与 HTML 类似。React 将 JSX 转化为可以调用的函数以生成 DOM 节点。尽管 JSX 与 HTML 类似，但存在一些不同点。有些开发人员不喜欢 JSX，因为必须将 JSX 与 JavaScript 代码混合在一起。而 Svelte 组件的 JavaScript、CSS 和 HTML 虽然定义在同一个文件中，但三者之间是互相分离的。

React 中还使用了虚拟 DOM，虚拟 DOM 存在的问题前面已经讲过了，此处不再赘述。

React 的学习成本越来越高，所要掌握的概念越来越多。比如，hooks 就是一个很大的变动，此外还有其他很多知识需要学习。在 React 中处理状态有很多方式，包括 this.setState、useState、Redux 等。Suspense 和 Concurrent 模式也即将发布。相对来说，Svelte 更容易掌握。

1.4.3 Vue

在 Vue 2 中使用对象描述一个组件。对象中包括了组件的很多内容，包括从外部传入的 props、computed 属性(基于其他数据执行计算)、data(组件状态)、watch 函数(监听状态变化)以及 methods(如事件处理函数)。Vue 2 的组件中会经常使用 this，一些开发人员觉得这样会令代码很难被理解，也令代码更加冗余。在 Vue 3 中，又改用函数描述组件了。此外，Vue 也使用了虚拟 DOM，在前面的章节中已经介绍过虚拟 DOM 的问题，此处不再赘述。

对于 Svelte 来说，通过响应式语句，组件中所有的内容可以使用普通的 JavaScript 变量和函数描述(第 2 章将详细介绍这方面的内容)。

1.5　开发工具

本书的读者应该具备熟练运用 HTML、CSS 和 JavaScript 开发 Web 应用程序的能力，熟悉基本的 HTML(包括表单元素)、CSS 语法(包括盒模型)和 JavaScript 语法(包括 ECMAScript 2015 及后续版本增加的一些新功能)。

在开始使用 Svelte 和 Sapper 之前，你只需要安装最新的 Node.js。可从 https//nodejs.org/下载最新的 Node.js。

注意　正如 Node.js 网站(https://nodejs.org/)上所介绍的那样，“Node.js 是一个基于 Chrome v8 JavaScript 引擎的 JavaScript 运行时环境”。Node.js 能够帮助你使用 JavaScript 开发多种应用程序，包括 HTTP 服务这类网络应用程序，以及代码检查工具、代码格式化工具、Svelte 编译器等工具型应用程序。

Node.js 提供了 node、npm 和 npx 命令。

- 使用 node 命令可以启动本地服务、本地测试应用程序，还可运行与开发有关的其他任务，比如代码检查、代码格式化和测试任务等。
- npm 命令可以为项目安装依赖。
- npx 命令可以用来创建新的 Svelte 和 Sapper 项目。

注意　虽然尚未发布正式版本，但还是推荐一个开发 Svelte 的工具，即 svelte-gl(https://github.com/sveltejs/gl)。svelte-gl 与 three.js(https://threejs.org/)类似，但它是专门为 Svelte 而开发的。可将对 3D 场景的描述作为输入参数传入 svelte-gl，输出则是 3D 场景对应的 WebGL 代码，并在应用程序中渲染出来。相关示例可访问 http://mng.bz/lG02。

与你现在所用的框架相比，使用 Svelte 和 Sapper 开发应用程序的简单程度会令你惊叹不已。

1.6　小结

- Svelte 是一个构建 Web 应用程序的工具，是主流 Web 框架(如 React、Vue 和 Angular)的替代品。
- Svelte 是 Web 应用程序的编译器，而非库。
- Svelte 有很多优秀的特性，例如，所需开发的代码更少、生成的包更小(这将缩短应用程序的启动时间)并且简化了状态管理机制。对于开发人员来说，Svelte 非常具有吸引力。
- Sapper 在 Svelte 的基础上添加了页面路由、服务端渲染、代码风格以及静态站点生成等功能。
- Svelte Native 是基于 Svelte 的 React Native 的替代方案，可用来开发 Android 和 iOS 移动应用程序。