Оглавление

Часть 3
Подробнее о Sapper

1 Знакомство с игроками

В этой главе

- ✓ Svelte.
- ✓ Sapper.
- ✓ Svelte Native.

Svelte (https://svelte.dev/) — инструмент для построения веб-приложений на базе JavaScript. Это альтернатива таких веб-фреймворков, как React, Vue и Angular. Как и они, Svelte специализируется на определении компонентов пользовательского интерфейса (UI) и взаимодействий между ними. Каждый UI-компонент представляет собой независимую, пригодную для повторного использования часть более крупного пользовательского интерфейса, которая может проектироваться и реализовываться независимо от других.

У Svelte есть ряд преимуществ перед другими веб-фреймворками.

- В приложениях, построенных с помощью Svelte, получается меньше кода при той же функциональности, что и при использовании других фреймворков.
- Svelte уменьшает размер распространяемого пакета, из-за чего снижается время загрузки в браузере.
- Svelte значительно упрощает управление состоянием как внутри компонентов, так и между ними. (К управлению состоянием относится организация данных, управляющих работой приложения, и реакция на их изменения.)

Sapper (https://sapper.svelte.dev/) — фреймворк, построенный на базе Svelte и предназначенный для создания более сложных веб-приложений. Sapper расширяет Svelte многими функциональными возможностями, включая маршрутизацию страниц, визуализацию на стороне сервера, разделение кода и генерирование статических сайтов. При этом создатели веб-приложений, которым такие возможности не нужны (или которые предпочитают реализовать их другим способом), могут ограничиться использованием только Svelte.

Svelte Native (https://svelte-native.technology/) также строится на базе Svelte. Эта технология интегрирует использование NativeScript для построения мобильных приложений для Android и iOS.

ПРИМЕЧАНИЕ Упоминая в книге различные технологические решения для создания веб-приложений, я буду называть их фреймворками, хотя некоторые авторы предпочитают термин «библиотека».

1.1. ЗНАКОМСТВО СО SVELTE

Зачем нужен еще один инструмент для построения веб-приложений?

Тратить время и усилия на изучение очередного решения стоит только в том случае, если он приносит значительные преимущества. Возможно, вам придется писать меньше кода для достижения того же результата. А может быть, значительно сократится объем работы. Или для нового решения браузеру придется загружать меньше байтов.

Svelte помогает достичь всех этих целей, и не только.

Как и другие фреймворки, Svelte может использоваться для построения целых веб-приложений. Компоненты Svelte могут использоваться в одном приложении либо определяться в библиотеках, совместно используемых многими приложениями. При помощи Svelte можно создавать нестандартные элементы (веб-компоненты), которые могут использоваться в других веб-приложениях, реализованных на базе прочих фреймворков или вообще без какого-либо фреймворка.

Рич Харрис, который ранее работал в The Guardian, а теперь перешел в The New York Times, занялся разработкой Svelte в 2016 году. Ранее он создал веб-фреймворк Ractive (https://ractive.js.org/), который используется в The Guardian и послужил источником вдохновения для некоторых частей Vue. Он также создал упаковщик модулей Rollup как альтернативу для Webpack и Parcel.

Svelte можно перевести как «стройный»; это точно характеризует как синтаксис Svelte, так и размер создаваемых пакетов.

1.1.1. Почему Svelte

У Svelte много преимуществ перед другими веб-фреймворками. Самые важные из этих преимуществ кратко описаны ниже.

Svelte — компилятор

Другие популярные веб-фреймворки включают большие библиотеки, которые используются во время работы программы и обеспечивают поддержку ее особенностей. Но Svelte не библиотека, а компилятор веб-приложений, реализованный на TypeScript.

> **ПРИМЕЧАНИЕ** Компилятор — программа, переводящая код, написанный на одном языке программирования, на другой язык. Обычно компиляторы переводят высокоуровневый язык (такой как Go или Java) на язык низкого уровня (например, машинный код или байт-код).

> **ПРИМЕЧАНИЕ** TypeScript — язык программирования с открытым исходным кодом, который представляет собой надмножество JavaScript; программы, написанные на TypeScript, компилируются на JavaScript. TypeScript добавляет к JavaScript много новых возможностей, самая важная из которых — возможность определения типов переменных и функций. Разработкой и сопровождением TypeScript занимается компания Microsoft.

UI-компоненты Svelte определяются в файлах `.svelte`. Такие файлы порой содержат смесь JavaScript, CSS и HTML. Например, у компонента могут быть элементы HTML для формы авторизации, CSS для ее стилевого оформления и JavaScript для передачи введенных данных службе аутентификации при нажатии кнопки `Login`.

Компилятор Svelte преобразует файлы `.svelte` в JavaScript и CSS. Плюс такого подхода в том, что новые возможности могут добавляться в Svelte без увеличения размера пакета развертываемого приложения. Компилятор включает код только тех возможностей Svelte, которые действительно используются в приложении.

Компактные пакеты

Пакеты приложений Svelte гораздо меньше, чем у эквивалентных приложений, создаваемых в других веб-фреймворках. Так что приложения Svelte быстрее загружаются браузерами.

> **ПРИМЕЧАНИЕ** В контексте веб-приложений пакеты (bundles) представляют собой файлы JavaScript, которые появляются благодаря объединению, оптимизации и минимизации всего необходимого приложению JS-кода.

В целом в Svelte размер пакета минимальный за счет включения только необходимого кода вместо всей библиотеки фреймворка. Например, у приложения Todo, представленного в главе 2, размер пакета на 13 % меньше, чем у эквивалентного приложения React. Ссылки на версии этого приложения на базе Svelte, React и Vue даны в главе 2.

> **ПРИМЕЧАНИЕ** Все веб-фреймворки включают «встряску дерева», чтобы убрать неиспользуемый код. И все же Svelte оставляет намного меньше кода фреймворка. Например, приложения React должны включать код, который строит представления виртуальных моделей DOM и определяет различия между ними. Использование виртуальных DOM описано ниже.

В опубликованном FreeCodeCamp сравнительном анализе фреймворков *A RealWorld Comparison of Front-End Frameworks with Benchmarks* (2019 update) приводится статистика построения реальных веб-приложений на базе разных веб-фреймворков (http://mng.bz/8pxz). В данном случае для сравнения используется приложение сайта социальных блогов Conduit, сходного с Medium.com.

Размеры приложения со сжатием gzip для некоторых популярных фреймворков:

- Angular + ngrx — 134 Кбайт;
- React + Redux — 193 Кбайт;
- Vue — 41,8 Кбайт;
- Svelte — 9,7 Кбайт.

Очевидно, Svelte показывает блестящие результаты по этому показателю.

Меньший объем кода

Svelte нужно меньше кода для реализации той же функциональности. В тех же сравнительных тестах приводятся следующие данные о количестве строк кода:

- Angular + ngrx — 4210;
- React + Redux — 2050;
- Vue — 2076;
- Svelte — 1116.

Этот показатель важен по многим причинам. Чем меньше объем кода, тем меньше вам придется прикладывать усилий, чтобы в нем разобраться. Также в таком варианте меньше мест, в которых могут скрываться ошибки.

Svelte обеспечивает реактивность без использования виртуальной DOM

Некоторые веб-фреймворки, включая React и Vue, используют виртуальную модель DOM для оптимизации обновления реальной модели DOM в ответ на изменения данных. При изменении состояния компонентов фреймворк строит новую версию DOM в памяти, а затем сравнивает ее с предыдущей версией. К реальной модели DOM применяются только отличия. Хотя такое решение работает быстрее, чем обновление всех данных в реальной модели DOM, на построение виртуальной модели и сравнение ее с предыдущей требуется время.

Реактивность — это способность обновлять DOM в ответ на изменения состояния приложения и компонентов. Для обеспечения реактивности Svelte отслеживает изменения в переменных компонентов верхнего уровня (область видимости которых не ограничивается функциями), влияющих на визуализацию компонентов. Таким образом, вместо перерисовки всего компонента обновляются только те части DOM, которые затрагивают изменения. Это позволяет Svelte выполнять меньший объем работы, по сравнению с другими фреймворками, для синхронизации DOM с состоянием приложения.

Svelte быстро работает

Ознакомьтесь с эталонными тестами Штефана Краузе (Stefan Krause) по ссылке https://krausest.github.io/js-framework-benchmark/current.html. Приложение, используемое в этих тестах, строит таблицу из четырех столбцов и 1000 строк. На этой странице приводятся сравнительные данные по разным фреймворкам и параллельная статистика. Например, выберите варианты angular-v8.0.1-keyed, react-v16.8.6-keyed, svelte-v3.5.1-keyed и vue-v2.6.2-keyed. Вы получите информацию о времени запуска приложения, показанную в табл. 1.1. Результаты доказывают, что Svelte работает достаточно быстро по сравнению с другими вариантами.

> **ПРИМЕЧАНИЕ** Термин keyed (ассоциативный) в этом случае означает, что код создает ассоциативную связь между данными и элементами DOM. При изменении данных изменяется связанный с ними элемент DOM. То же касается добавления или удаления элементов массива. Результаты тестов, приведенные в табл. 1.1, относятся к «ассоциативным» реализациям, потому что они лучше показывают, как приложения пытаются повысить эффективность обновления существующих элементов DOM.

Svelte расходует меньше памяти

Снижение затрат памяти — важный аспект при запуске веб-приложений на старых компьютерах или мобильных устройствах, у которых объем памяти для выполнения приложений обычно меньше.

Сайт сравнительного тестирования, упомянутый в предыдущем разделе, опубликовал данные по затратам памяти, изображенные в табл. 1.2. По ним видно, что приложения Svelte обычно используют меньше памяти по сравнению с другими вариантами.

Таблица 1.1. Время инициализации и загрузки в сравнительных тестах

Метрика	svelte-v3.5.1-keyed	vue-v2.6.2-keyed	react-v16.8.6-keyed	angular-v8.0.1-keyed
Время инициализации сценария (время в миллисекундах, необходимое для разбора/компиляции/построения всех сценариев страницы	19,5 ± 2,4 (1.00)	59,6 ± 28,6 (3.06)	55,6 ± 45,2 (2.85)	159,8 ± 8,8 (8.21)
Общий размер в килобайтах — затраты на пересылку по сети (после сжатия) всех ресурсов, загружаемых на странице	145 ± 0 (1.00)	211,2 ± 0 (1,45)	260,8 ± 0 (1,79)	295,5 ± 0 (2.03)

Компоненты Svelte не используют контейнер JavaScript

Файлы .svelte не определяют контейнер JavaScript для компонента. Вместо этого компонент определяется сочетанием элемента `script`, разметки HTML для визуализации и элемента `style`.

Такой подход проще того, который применяется в большинстве других веб-фреймворков. Для определения компонента требуется меньше строк кода, а разработчику приходится учитывать меньше нюансов, относящихся к JavaScript. Например, компоненты Angular определяются классом, компоненты React — функцией или классом, компоненты Vue 2 — объектным литералом, а компоненты Vue 3 — функциями.

Стилизация Svelte имеет ограниченную область видимости

По умолчанию код CSS, определяемый в каждом компоненте Svelte, применяется только к этому компоненту. Тем самым предотвращается случайная утечка правил CSS, определяемых в файле .svelte, и их непреднамеренное применение к другим компонентам.

В прочих фреймворках используется другой подход к стилизации. В Angular у стилей, заданных в свойстве `styles` компонента, по умолчанию область

видимости также ограничивается компонентом. В Vue область видимости стилей ограничивается компонентами только в том случае, если они задаются внутри элемента `style` с атрибутом `scoped`. React не разрешает ограничивать область видимости стилей компонентами, и это одна из причин, по которым решения «CSS-в-JS» так популярны в приложениях React. В Svelte практически нет смысла использовать решения «CSS-в-JS».

Таблица 1.2. Затраты памяти в сравнительных тестах (в мегабайтах)

Метрика	svelte-v3.5.1-keyed	vue-v2.6.2-keyed	react-v16.8.6-keyed	angular-v8.0.1-keyed
Готовая память (затраты памяти после загрузки страницы)	1,9 ± 0 (1.00)	2,1 ± 0 (1.13)	2,3 ± 0 (1.23)	4,8 ± 0 (2.54)
Память в ходе выполнения (затраты памяти после добавления 1000 строк)	3,9 ± 0 (1.00)	7,1 ± 0 (1,81)	6,9 ± 0 (1,76)	9,1 ± 0 (2.34)
Обновление каждой десятой записи для 1k записей (пять циклов) (затраты памяти после нажатия кнопки обновления каждой десятой строки пять раз)	4,3 ± 0 (1.00)	7,5 ± 0 (1,76)	8,0 ± 0 (1,89)	9,5 ± 0 (2.23)
Замена 1k строк (пять циклов) (затраты памяти после нажатия кнопки создания 1000 строк пять раз)	4,5 ± 0 (1.00)	7,7 ± 0 (1,71)	8,9 ± 0 (1,98)	9,9 ± 0,1 (2.20)
Создание/очистка 1k строк (пять циклов) (затраты памяти после нажатия кнопки создания и очистки 1000 строк пять раз)	3,2 ± 0 (1.00)	3,8 ± 0 (1,20)	4,7 ± 0,1 (1,48)	6,6 ± 0 (2.07)

Svelte предоставляет место для глобальных стилей

Svelte предоставляет конкретное место для определения глобальных стилей, которые могут повлиять на любой компонент. Эти стили определяются в файле public/global.css.

Svelte упрощает управление состоянием

Управление состоянием приложений и компонентов значительно упрощается в Svelte по сравнению с другими фреймворками. В этом отношении вносят свой вклад такие средства, как контекст, хранилища и контекст модулей, — далее в книге они рассматриваются более подробно.

Svelte поддерживает двустороннее связывание данных

Svelte также упрощает связывание значений элементов форм с переменными компонентов. К элементам форм относятся элементы `input`, `textarea` и `select`. Переменные высокого уровня в файлах .svelte представляют состояние компонента.

При изменении значения связанной переменной значения соответствующих элементов формы автоматически обновляются. Когда пользователь изменяет значение связанного элемента формы, автоматически обновляется значение соответствующей переменной.

Svelte упрощает анимацию

В Svelte реализована встроенная поддержка эффектов анимации. Добавить анимацию в приложение на удивление просто. Благодаря этой простоте анимация широко используется, а это улучшает взаимодействие пользователя с приложением.

Среди примеров анимаций в приложении Todo — проявление новых и растворение удаленных элементов задач. Если элементы задач организуются в иерархические списки, анимация может использоваться для плавного вывода элемента из текущей категории и перемещения его в новую.

Svelte повышает доступность

Svelte выдает предупреждения о возможных проблемах доступности. Например, помечаются элементы `img` без атрибута `alt`. Благодаря этому приложения Svelte, скорее всего, будут доступны для пользователей, взаимодействующих с браузером с применением вспомогательных средств.

1.1.2. Новый подход к реактивности

В контексте веб-приложений *реактивностью* называется способность DOM обновляться автоматически в ответ на изменения данных (то есть состояния). Сравните с электронными таблицами: изменение значения одной ячейки может привести к изменению других. Это происходит в случае, если значение ячейки вычисляется по формуле, в которой задействованы значения других ячеек.

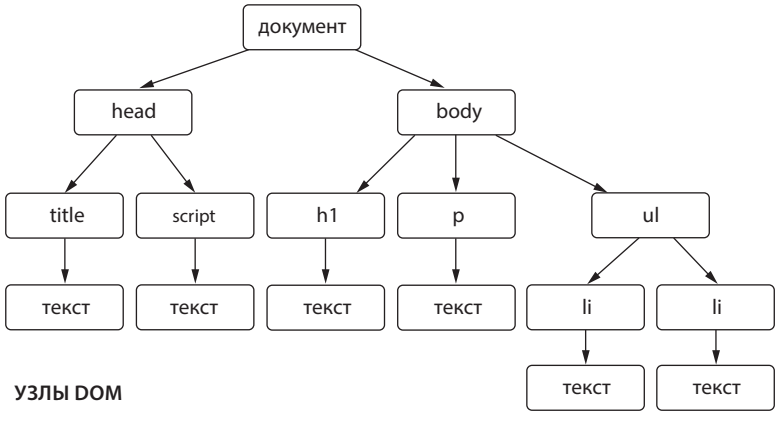
В Svelte реактивность реализуется проще, чем в других фреймворках. Svelte поддерживает уникальный способ управления состоянием компонента, который основывается на отслеживании переменных верхнего уровня (см. раздел 3.9). Также Svelte упрощает совместное использование данных состояния разными компонентами.

HTML DOM

Модель HTML DOM обеспечивает представление веб-страницы в памяти. Модель выглядит как дерево объектов JavaScript, представляющих узлы. Один объект JavaScript представляет весь документ; он содержит ссылки на другие объекты DOM, представляющие элементы страницы. Объекты DOM содержат методы, которые можно вызывать для получения информации об узле, добавления дочерних узлов, регистрации слушателей событий и т. д. Изменение модели DOM приводит к изменению изображения в браузере.

Пример документа HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Page</title>
  </head>
  <script>
    // Для исследования DOM с консоли DevTools ...
    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>
```



УЗЛЫ DOM

На этой схеме изображены узлы DOM, созданные браузером для представления документа HTML в памяти.