Инженерный вестник Дона, №11 (2023)
ivdon.ru/ru/magazine/archive/n11y2023/8804

О фреймворках Vue, Svelte, Solid и Lit для разработки клиентских веб-приложений

И.А. Короленко

Yandex Europe B.V. (Amsterdam, Netherlands)

Аннотация: В данной работе рассматриваются фреймворки Vue, Svelte, Solid и Lit, применяемые для создания части Представления клиентских веб-приложений. Изучаются их устройство, предлагаемый подход в разработке клиентских приложений, решаемые и создаваемые им проблемы, сильные и слабые стороны, а также особенности и ограничения в применении.
Ключевые слова: Vue, Svelte, Solid, Lit, Web Components, view frameworks, клиентские веб-приложения, front end, рендеринг.

Введение

Разработка программного обеспечения, решающего проблемы в какой-либо из сфер человеческой деятельности, помимо решения специфичных для конкретной компании и случая применения проблем, состоит также из решения набора схожих технических проблем [1]. Эти проблемы должны быть решены в любой написанной программе, которая будет применена в данной сфере. К примеру, любое клиент-серверное приложение должно иметь интерфейс между клиентской и серверной частью [2], любое веб-приложение нуждается в инструментах удобного взаимодействия с браузерными API (Application Programming Interface) [3], любое SPA (Single Page Application) нуждается в инструменте для рендеринга на клиентской части [4]. Как правило, эти схожие задачи решаются с помощью фреймворков [5].

Фреймворк – это абстракция, которая предоставляет готовые унифицированные инструменты для решения определенного набора задач в процессе разработки [6]. Фреймворки помогают достичь консистентности кода, единообразия используемых парадигм и инструментов.

Целью данной статьи является рассмотрение фреймворков Vue, Svelte, Solid и Lit для построения клиентской части веб-приложений, их внутреннего устройства и особенностей применения на практике.

Vue

Данный фреймворк разработан Эваном Ю. Основные функции аналогичны Angular (шаблоны, привязка модели к шаблону, директивы) и React (разработан компанией Meta, запрещена в России), если вы используете Composition API вместо Options API [7], но декларативны и более просты в применении. Применяется односторонняя привязка данных, но фреймворк также поддерживает двустороннюю привязку данных через v-model [8]. Имеет набор официальных инструментов и поддерживает сторонние библиотеки.

Рабочий процесс выглядит следующим образом:

- создайте файл с расширением vue;
- создайте разделы <template>, <script>, <style>;
- определите компонент в <script> как простой объект JavaScript с ключами name, data, methods, mounted;
- Определите состояние в data и вызывайте его через this[ключ];
- Используйте атрибуты v-for, v-bind, v-on:submit.prevent, v-model, и т. д. для привязки данных к HTML (HyperText Markup Language).

Преимущества:

- очень зрелый и проверенный временем;
- один из самых популярных view фреймворков. Это означает огромное сообщество, которое уже решило все проблемы, с которыми команда столкнется во время разработки;
- имеет официальные пакеты для решения таких проблем, как маршрутизация и управление состоянием;
- поддерживает органическое использование сторонних пакетов.

Недостатки:

- количество проектов, использующих его, меньше, чем у React;
- меньшее количество компонентов и плагинов по сравнению с React;
- сообщество в основном сконцентрировано в Китае, поэтому без знания китайского языка будет сложнее решить некоторые проблемы.

Svelte

Фреймворк разработан Ричем Харрисом в 2016 году. Основные особенности: гибкость, отсутствие виртуального DOM (но при этом рендеринг обычно работает быстрее, чем React [9]), компиляция кода в обычный JavaScript, встроенная реактивность данных, построенные вокруг HTML компоненты, отсутствие boilerplate кода [10].

Рабочий процесс выглядит следующим образом:

- создайте файл с расширением svelte;
- создайте разделы <script>, <main>, <style>;
- внутри <script> просто определите переменные let для состояния, функции для обработчиков и функцию onMount(), импортированную из Svelte;
- в разделе <main> добавьте привязки, используя такие атрибуты, как: on:submit|preventDefault, bind:value, и т. д.

Преимущества:

- относительно зрелый и проверенный в боевых условиях;
- один из самых популярных view фреймворков. Это означает огромное сообщество (хотя и не такое большое, как у React или Vue), которое уже решило все проблемы, с которыми команда может столкнуться во время разработки;
- позволяет писать менее многословный и более естественный (по отношению к простому JavaScript) код по сравнению с другими view

фреймворками. Однако код с фреймворком Lit получается еще более нативный;

- легко освоить, поскольку он очень близок к обычному JavaScript и использует простой HTML и CSS (Cascading Style Sheets).

Недостатки:

- количество проектов, использующих его, меньше, чем у React.

Solid

Фреймворк разработан open source сообществом в 2018 году. Основные особенности: похож на React, использует JSX (JavaScript Syntax Extension), компоненты — это функции, встроенная реактивность данных, сигналы похожи на React Hooks, не имеет виртуального DOM (Document Object Model), но имеет очень быстрый рендеринг, близкий к обычному JS.

Компоненты визуализируются только один раз. Таким образом, вы можете объявить что-то вроде таймера или вызова API прямо в компоненте без необходимости использования useEffect. Благодаря гранулярным обновлениям, фреймворк перерисовывает только те части DOM компонента, которые были фактически изменены.

Интересен также подход фреймворка к shared state. Вы можете создать shared state, просто переместив вызов createSignal из компонента в функцию и импортировав объект данных и функцию обновлениях данных (setSomething) в несколько компонентов.

Рабочий процесс выглядит следующим образом:

- создайте файлы с расширениями jsx и module.css;
- в jsx файле создайте функцию App и определите состояние путем вызова функции createSignal;
- добавьте функцию onMount (импорт из «solid-js») и функции-обработчики в виде простых функций JS;
- добавьте return функции в формате JSX, как в React.

Преимущества:

- рабочий процесс и API очень похожи на React;
- легче работать, чем с React. Благодаря гранулярным обновлениям, реактивности данных и другим особенностям.

Недостатки:

- не так популярен, как React. Это означает меньшее сообщество и больше вещей, которые нужно исправить самим.

Lit

Фреймворк разработан компанией Google в 2019 году. Основные особенности: предоставляет простой API для использования Web Components, использует литералы шаблонов JavaScript для создания шаблонов, имеет одностороннюю привязку данных.

Фреймворк существенно полагается на Web Components, поэтому его преимущества и недостатки сводятся к таковым у Web Components.

Преимущества Web Components:

- независимость от фреймворка. Веб-компоненты работают с любым фреймворком (React, Vue, Angular, и т.д.), и, в том числе, вообще без фреймворка. Это очень важно, поскольку это свойство делает веб-компоненты действительно гибкими;
- стандарт HTML. Поскольку веб-компоненты используют нативные спецификации HTML, а также обычные CSS и JavaScript, они моугт похвастаться встроенной поддержкой современных браузеров;
- легко переиспользовать. Использование веб-компонентов упрощает работу с несколькими проектами или экосистемами с различными стеками технологий, где вам необходимо совместно использовать или переиспользовать компоненты;
- никаких сложных зависимостей. Преимущество, которое предоставляют веб-компоненты – это возможность подключить конкретный

пользовательский элемент без импорта сложных зависимостей в проект. Это большая разница по сравнению с популярными фреймворками.

Недостатки веб-компонентов:

- время загрузки. Загрузка, обработка, регистрация и отображение веб-компонентов может занять несколько миллисекунд, поскольку кастомные элементы регистрируются в JavaScript. В промежутке времени кастомный элемент остается без стилей или скрытым;

- неоптимизированы для SEO (Search Engine Optimization). Веб-компоненты пока не оптимизированы для SEO, что является проблемой для многих разработчиков, а веб-сайты, которые их используют, могут страдать от неэффективной индексации. Впрочем, есть библиотеки, которые решают эту проблему;

- совместное использование стилей компонентами. Преимущество инкапсуляции веб-компонентов также может рассматриваться, как недостаток, поскольку оно не позволяет компонентам совместно использовать код стилей.

Заключение

Использование подходящего фреймворка является необходимой частью процесса разработки комплексных веб-приложений. Данная статья рассматривает фреймворки Vue, Svelte, Solid и Lit, предназначенные для создания части «Представления клиентских веб-приложений», включая их внутреннюю структуру, сильные и слабые стороны, а также особенности и ограничения использования. Из представленной информации можно заключить, что фреймворки Vue и Svelte представляют собой отличный выбор при разработке большинства комплексных клиентских веб-приложений. К применению фреймворков Solid и Lit, несмотря на их существенные сильные стороны, на данный момент стоит подходить с

осторожностью ввиду значительно меньшего сообщества и широты использования в комплексных проектах.

Литература (References)

1. Fayad M., Johnson R. Domain-specific Application Frameworks: Frameworks Experience by Industry. Hoboken, New Jersey: Wiley, 1999. 704 p.

2. Benatallah B., Casati F., Toumani F. Web service conversation modeling: a cornerstone for e-business automation. IEEE, 2004. URL: ieeexplore.ieee.org/document/1260703.

3. Blokdyk G. Web framework A Complete Guide. Brendale, Australia: 5starcooks, 2018. 270 p.

4. Mikowski M., Powell J. Single Page Web Applications: JavaScript end-to-end. Shelter Island, New York: Manning, 2013. 432 p.

5. M. Fayad, Schmidt D., Johnson R. Building Application Frameworks: Object-Oriented Foundations of Framework Design. Hoboken, New Jersey: Wiley, 1999. 688 p.

6. Riehle D. Framework Design: A Role Modeling Approach. riehle.org, 2000. URL: riehle.org/computer-science/research/dissertation/diss-a4.pdf.

7. Wildermuth S. Vue’s Composition API. Code Magazine, 2020. URL: codemag.com/Article/2011041/Vue%E2%80%99s-Composition-API.

8. Kilonzi F. Two-Way Data Binding in Vue.js Using V-Model. Sweetcode, 2020. URL: sweetcode.io/two-way-data-binding-in-vue-js-using-v-model.

9. Kugell A. Svelte vs React: Which Framework for Your Project? Trio, 2023. URL: trio.dev/react/resources/svelte-vs-react.

10. Degni R. Svelte: Why Is It an Innovation to Javascript Frameworks? Codemotion, 2022. URL: codemotion.com/magazine/frontend/javascript/svelte-why-is-it-an-innovation-to-javascript-frameworks.