Geekbrains

*(пример)*

**Разработка веб-приложения книжного интернет-магазина с использованием прогрессивного JavaScript-фреймворка** **Vue**

Программа: Разработчик

Специализация: Frontend-разработка

ФИО: Балахонов Леонид Сергеевич

Город Москва

Год 2024

**Содержание**

1. Введение

2. Компоненты

3. App.vue

4. Home.vue

5. Favorite.vue

6. Card.vue

7. CardList.vue

8. CartItem.vue

9. CartItemList.vue

10. Drawer.vue

11. DrawerHeader.vue

12. Header.vue

13. InfoBlock.vue

14. Заключение

**Введение**

В проекте используется веб-приложение — это программное обеспечение, которое запускается в веб-браузере.

В отличие от стандартных приложений, речь идет о программах, которые способны работать полноценно даже без установки на устройство. Смартфон, планшет или компьютер получает онлайн-доступ к данным, а пользователю не нужно проводить установку в постоянную память — это и есть главная отличительная черта веб-приложений.

Эту разновидность инструментов нельзя путать с сайтами. Веб-приложения интерактивны, пользователи могут совершать в них различные действия: заказать товар или услугу, забронировать билет, оставить комментарий или отзыв, редактировать контент и так далее

Разработка проводилась на основе прогрессивного JavaScript-фреймворка Vue.

В качестве редактора кода используется VS Code.

Что такое Vue.js?

Vue (произносится /vjuː/, примерно как view) — прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов, Vue создавался пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), упрощая интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для разработки сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его в комбинации с современными инструментами и дополнительными библиотеками.

Vue.js изначально разрабатывался быть инкрементально адаптируемым. Это значит, что он может быть интегрирован в проект несколькими способами, в зависимости от требований.

Есть четыре основных способа добавления Vue.js в проект:

1. Импорт CDN пакета на странице
2. Скачивание файлов JavaScript и их самостоятельный хостинг

3. Установка с использованием npm

4. Использование официального CLI для развёртывания проекта, предоставляющего продвинутые инструменты для создания современного рабочего процесса разработки фронтенда (например, горячая перезагрузка, линтинг при сохранении и многое другое).

CDN

Для прототипов и в обучающих целях можно использовать последнюю версию с CDN:

<script src="https://unpkg.com/vue@next"></script>

Скачивание и самостоятельный хостинг

Если нужно избежать использования систем сборки, но нет возможности использовать CDN в production — в таком случае можно скачать соответствующий .js файл и разместить его на собственном веб-сервере. После чего подключать его, с помощью тега <script>, точно также, как и при подключении с CDN.

Можно просматривать и скачивать файлы с CDN, например unpkg (opens new window) или jsDelivr (opens new window).

npm

npm — рекомендованный способ установки при создании больших приложений на Vue. Он прекрасно сочетается с системами сборки, такими как [webpack](https://webpack.js.org/) или [Rollup](https://rollupjs.org/).

# установка последней стабильной версии

$ npm install vue@next

Также Vue предоставляет инструменты для создания [однофайловых компонентов](https://v3.ru.vuejs.org/ru/guide/single-file-component.html) (также именуемых как SFC). Если хотите их использовать, то также потребуется установить @vue/compiler-sfc:

$ npm install -D @vue/compiler-sfc

Обратите внимание, если выполняете миграцию с Vue 2, то @vue/compiler-sfc заменяет используемый ранее vue-template-compiler.

Кроме @vue/compiler-sfc также понадобится подходящий загрузчик SFC или плагин для системы сборки. Более подробную информацию можно найти [в разделе про однофайловые компоненты](https://v3.ru.vuejs.org/ru/guide/single-file-component.html).

В большинстве случаев предпочтительным способом создания сборки с webpack и минимальной конфигурацией будет использование Vue CLI.

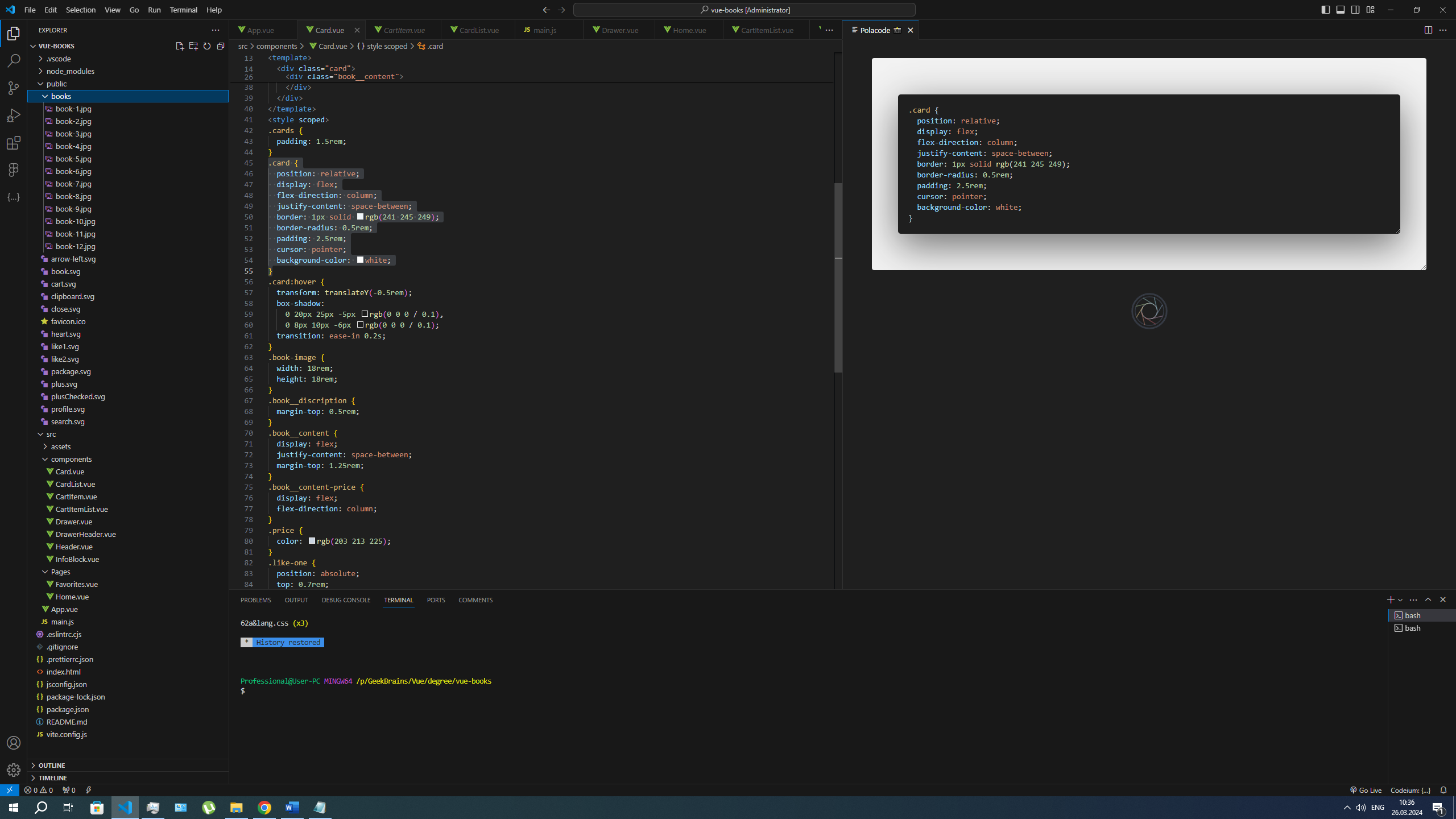
В проекте используется Composition API.

Composition API — это набор функций, которые позволяют создавать компоненты без использования классов и методов this. Это делает код более чистым и понятным. Кроме того, Composition API предлагает дополнительные возможности, такие как реактивные наборы (reactive sets) и наблюдатели (watchers), которые позволяют более гибко работать с данными.

**Структура проекта**

Обычно приложение организуется в виде дерева вложенных компонентов.

Например, может быть компонент для заголовка, боковой панели, зоны контента, каждый из которых содержит другие компоненты для навигационных ссылок, постов и т.д.



В папке components содержатся компоненты используемые в в более высокоуровневых компонетах отраженных в папке Pages таких как Home.vue и Favorites.vue.

В папке public расположена папка books, содержащая изображения для отображения книг, в корне лежат svg-файлы для иконок, используемых в проекте.

**App.vue**



Импортируем методы из библиотеки Vue.

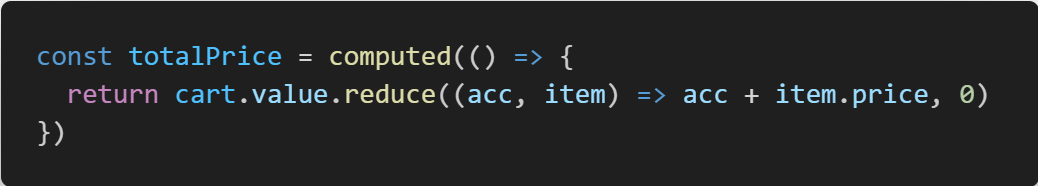
ref. Представьте случай, когда есть отдельное примитивное значение (например, строка) и необходимо сделать её реактивной. Для этого у Vue уже есть метод, который сделает то же самое — ref:

watch API наблюдения является точным эквивалентом свойства наблюдения компонента. требует наблюдения за конкретным источником данных и применяет побочные эффекты в отдельной функции обратного вызова. По умолчанию он также является lazy, то есть обратный вызов вызывается только тогда, когда отслеживаемый источник изменился.

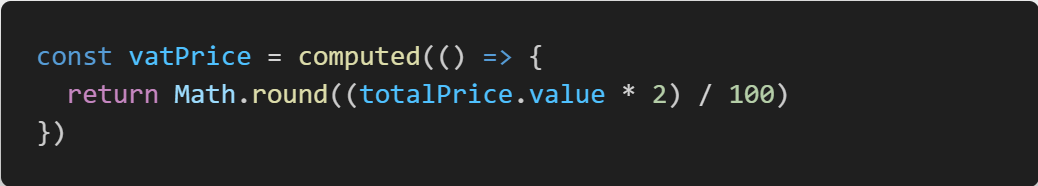
Computed в Vue 3 — это свойства, которые вычисляются динамически на основе других реактивных свойств. Они определяются как функции и ведут себя как свойства данных, но перепроверяются только при изменении их зависимостей.

Provide. Родительский компонент может служить dependency provider для всех своих потомков. Любой компонент в дереве потомков, независимо от его глубины, может inject зависимости, предоставляемые компонентами в его родительской цепочке.

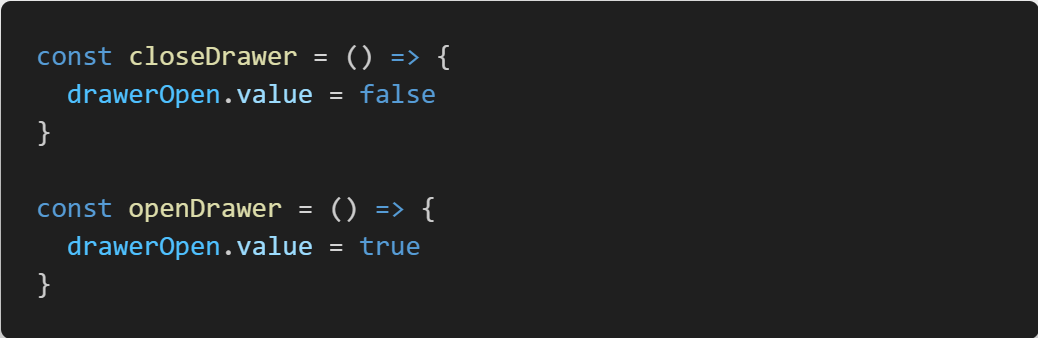
Так же импортируем два компонента Header.vue и Drawer.vue.



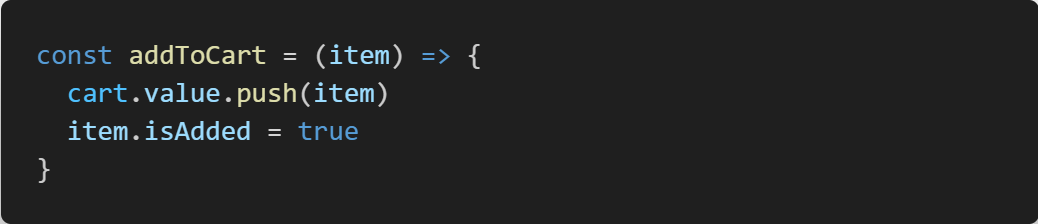
Расчет общей стоимости товаров в корзине. Используется reduce для суммирования элементов.



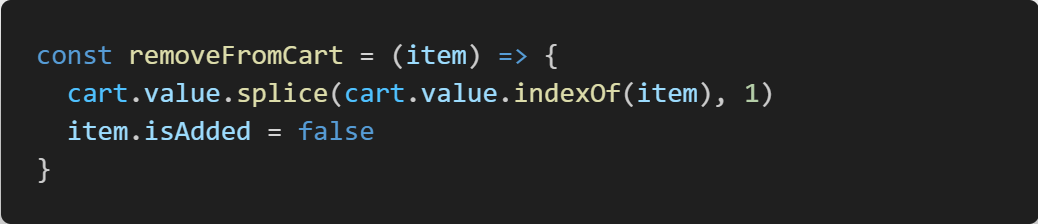
Расчет налога для конкретной общей стоимости товаров в корзине. Math.round используется для округления результата.



Открытие и закрытие корзины по флагу.



Добавление элемента в корзину и изменение флага isAdded в значение true.



Удаление элемента из корзины с помощью метода для массивов splice(). Метод indexOf() возвращает позицию элемента в списке. Флаг isAdded выставляем в значение false.



При использовании watch с флагом deep на объекте вы можете отслеживать изменения во вложенных свойствах объекта. local.storage.setItem() записывает в хранилище данных наши ключ и значение в строчном формате. JSON.stingify статический метод преобразует значение JavaScript в строку JSON.



Родительский компонент имеет опцию provide для предоставления данных, а дочерний компонент имеет опцию inject для использования этих данных.



Шаблон компонента. В проекте применен подход с использованием Vue Router. Для установки используем команду в терминале.

npm install vue-router@4

Компонент <router-view> сообщает Vue Router, где отображать текущий компонент маршрута.



На изображении представлено подробное описание стилей компонента. В данном случае не используется препроцессор.

**Home.vue**

****



Компонент отображения страницы Home. Пройдемся по импортам.

reactive. Другой способ объявить реактивное состояние с помощью API reactive(). В отличие от ref, которая оборачивает внутреннее значение в специальный объект, функция reactive() делает сам объект реактивным.

onMounted(). Регистрирует обратный вызов, который будет вызываться после монтирования компонента.

Компонент считается смонтированным после:

Все его синхронные дочерние компоненты смонтированы (не включая асинхронные компоненты или компоненты внутри деревьев <Suspense>).

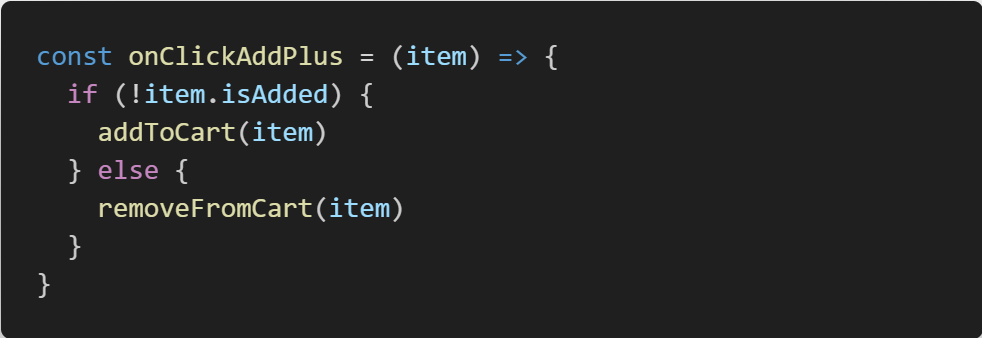
Создано собственное DOM-дерево, которое вставлено в родительский контейнер. Обратите внимание, что это гарантирует, что дерево DOM компонента находится в документе, только если корневой контейнер приложения также находится в документе.

Axios. HTTP-клиент на основе Promise для браузера и node.js

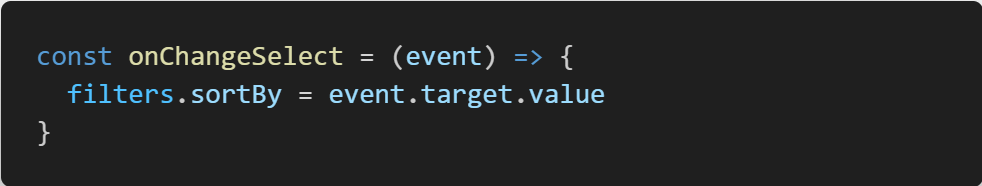
Axios — это простой HTTP-клиент на основе обещаний для браузера и node.js. Axios предоставляет простую в использовании библиотеку в небольшом пакете с очень расширяемым интерфейсом.

debounce. Функция debounce - это обёртка над вашей функцией, которая позволяет отменить и отложить на какое-то время её повторные вызовы разными способами.

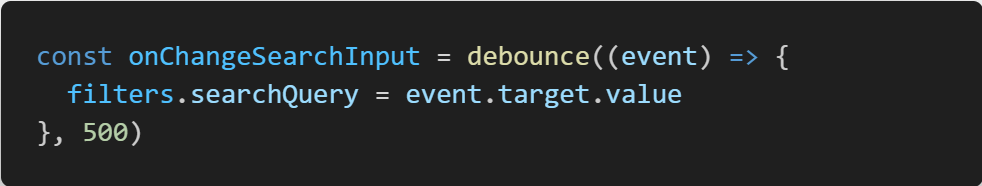
Импортируем компонент CardLIst.



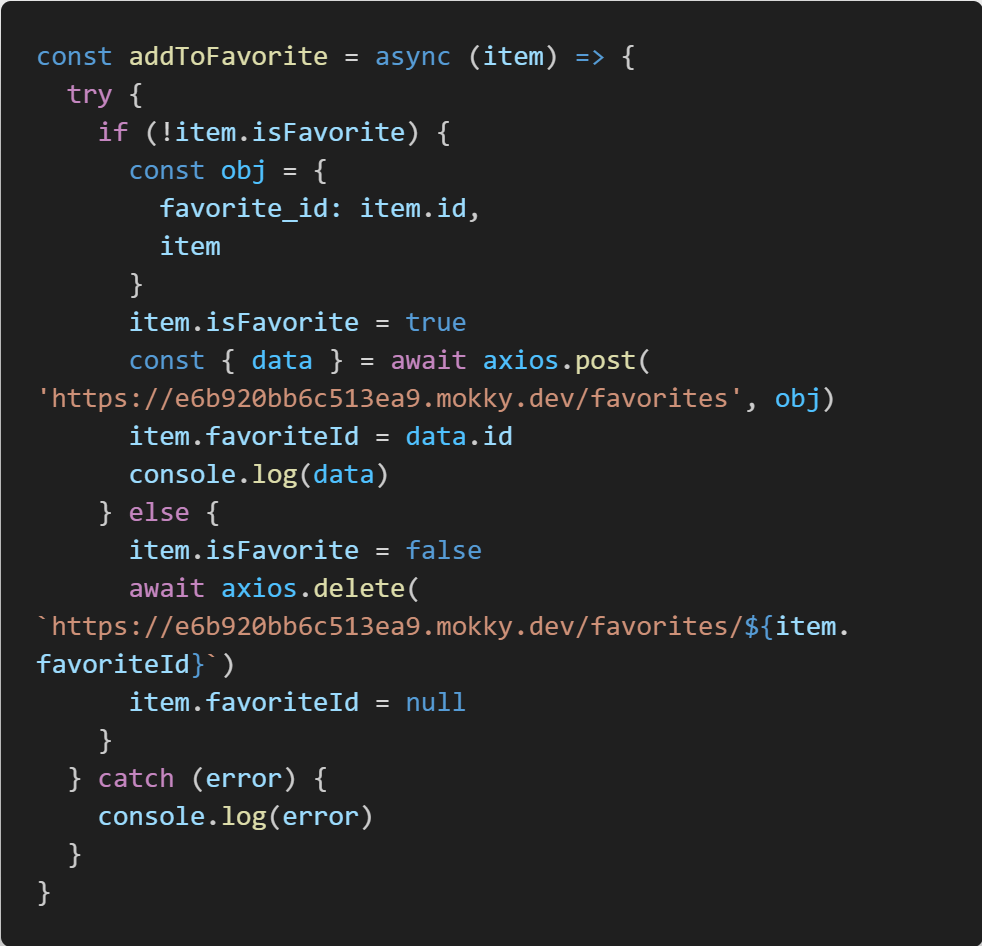
Функция onClickAddPlus добавляет элемент или удаляет его из корзины по флагу isAdded.



Функция записывает значение изменения select из трех option по заголовку, цене по убыванию и цене по возрастанию.



С помощью задержки debounce в пол секунды записывает результат ввода клавиатуры.



addToFavorite используется для добавления и удаления элементов из закладок по значению флага isFavorite.

Async/await

Существует специальный синтаксис для работы с промисами, который называется «async/await». Он удивительно прост для понимания и использования.

У слова async один простой смысл: эта функция всегда возвращает промис. Значения других типов оборачиваются в завершившийся успешно промис автоматически.

Ключевое слово await заставит интерпретатор JavaScript ждать до тех пор, пока промис справа от await не выполнится. После чего оно вернёт его результат, и выполнение кода продолжится.

Конструкция try..catch состоит из двух основных блоков: try, и затем catch.

Сначала выполняется код внутри блока try {...}.

Если в нём нет ошибок, то блок catch(err) игнорируется: выполнение доходит до конца try и потом далее, полностью пропуская catch.

Если же в нём возникает ошибка, то выполнение try прерывается, и поток управления переходит в начало catch(err). Переменная err (можно использовать любое имя) содержит объект ошибки с подробной информацией о произошедшем.

Для хранения данных применяется такой сервис как Mokky.

сервис для создания тестового REST API. С его помощью вы можете легко настроить свой API для авторизации, регистрации, фильтрации, загрузки файлов и хранения данных в формате JSON.

Для добавления закладки закладок применяется метод POST, для удаления DELTE.



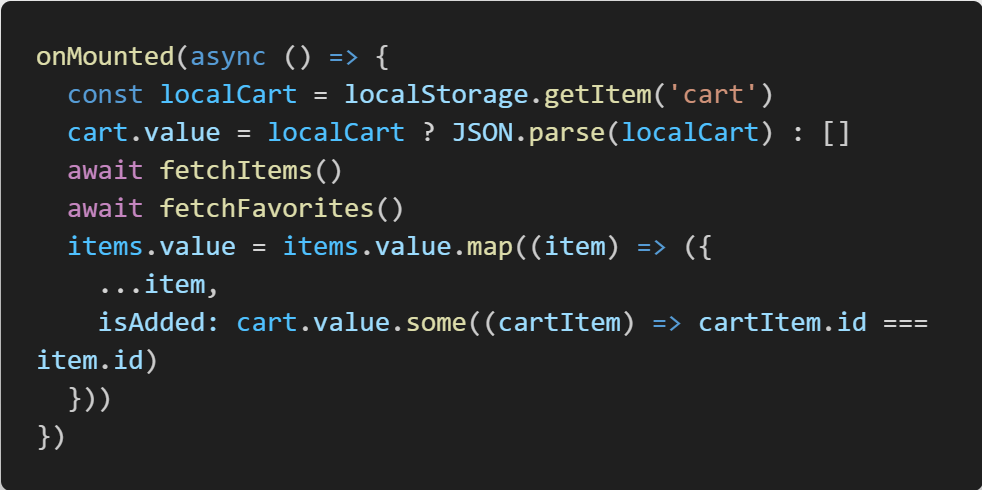
fetchFavorites с помощью метода get получает список закладок.

Метод map() экземпляров Array создает новый массив, заполняемый результатами вызова предоставленной функции для каждого элемента вызывающего массива.

Метод find() экземпляров Array возвращает первый элемент предоставленного массива, который удовлетворяет предоставленной функции тестирования. Если ни одно значение не удовлетворяет функции тестирования, возвращается неопределенное значение.



С помощью функции fetchItems получает всех элементы для отображения на странице. Как и выше примениям метод GET и метод map() экземпляров Array. Принцип работы очень похож на fetchFavorites.



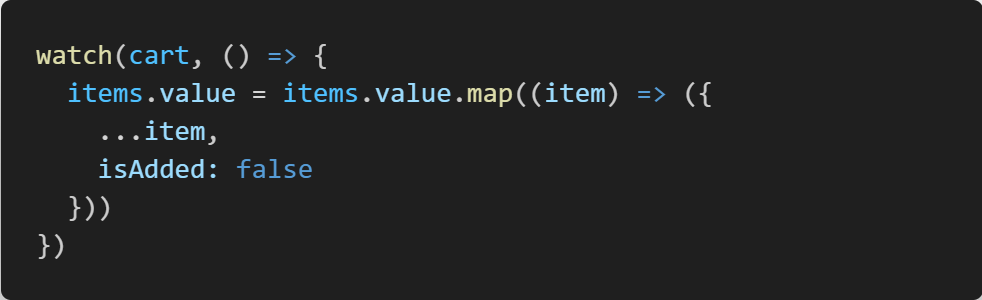
onMounted() хук жизненного цикла обычно используется для выполнения побочных эффектов, требующих доступа к отображаемому DOM компонента, или для ограничения кода, связанного с DOM, клиентом в приложении, отображаемом на сервере.

С помощью localStorage.getItem() получаем данные из локального хранилища по ключу cart данные.

Записываем в переменную cart значение исходя из тернарного оператора, данные из localStorage или пустой массив соответсвенно.

После получаем все элементы, далее закладки.

Метод some() экземпляров Array проверяет, проходит ли хотя бы один элемент массива тест, реализованный предоставленной функцией. Он возвращает true, если в массиве находит элемент, для которого предоставленная функция возвращает true; в противном случае он возвращает false. Он не изменяет массив.



Отслеживаем изменение cart.



Шаблон компонента. Используются дикективы:

v-on. Сокращённая запись: @. Прикрепляет обработчик события к элементу. Тип события определяется аргументом. Выражение может быть именем метода, инлайн-выражением или не указываться при использовании модификаторов.

При использовании на обычном элементе отслеживает только нативные события DOM. При использовании на компонентах отслеживает пользовательские события, которые были сгенерированы в дочернем компоненте.

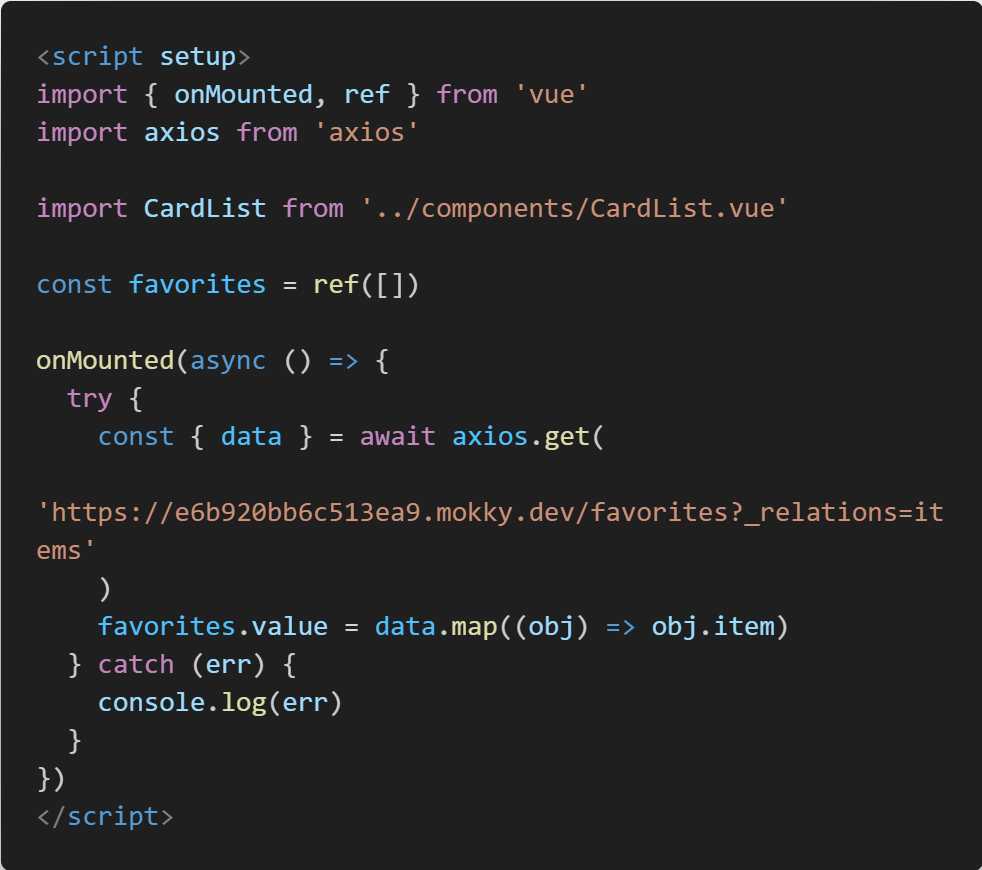
При отслеживании нативных событий DOM в метод единственным аргументом будет передаваться объект события. При использовании инлайн-выражения объект события доступен через специальное свойство $event: v-on:click="handle('ok', $event)".

v-bind. Сокращённая запись: : . Динамически привязывает выражение к одному или нескольким атрибутам или входным параметрам компонента.

При использовании с атрибутами class или style в качестве значений допускаются массивы и объекты. Подробнее можно узнать в разделе руководства по ссылкам ниже.

При привязке входных параметров к дочернему компоненту потребуется их также определить в нём.

**Favorite.vue**

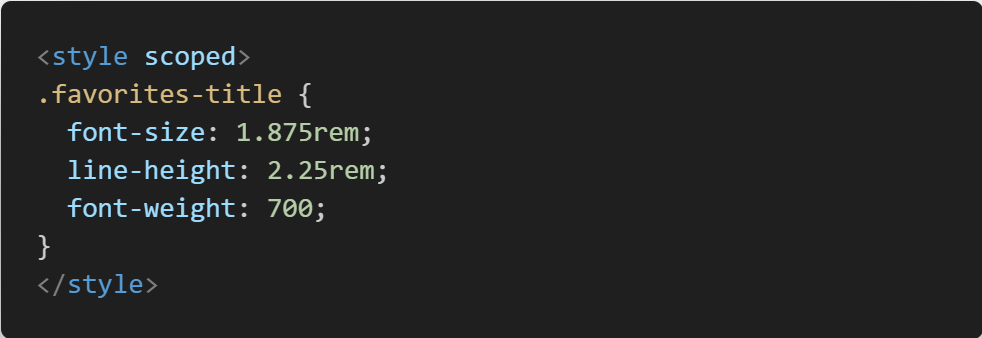


Компонент отображения страницы Favorites. Описание импортов представлено выше.

onMounted(). Регистрирует обратный вызов, который будет вызываться после монтирования компонента, с помощью метода GET получает список закладок. Обрабатываем ошибки с помощью try … catch.



Шаблон предельно прост. Заголовок С помощью компонента CardList отображаем элементы добавленные в закладки.

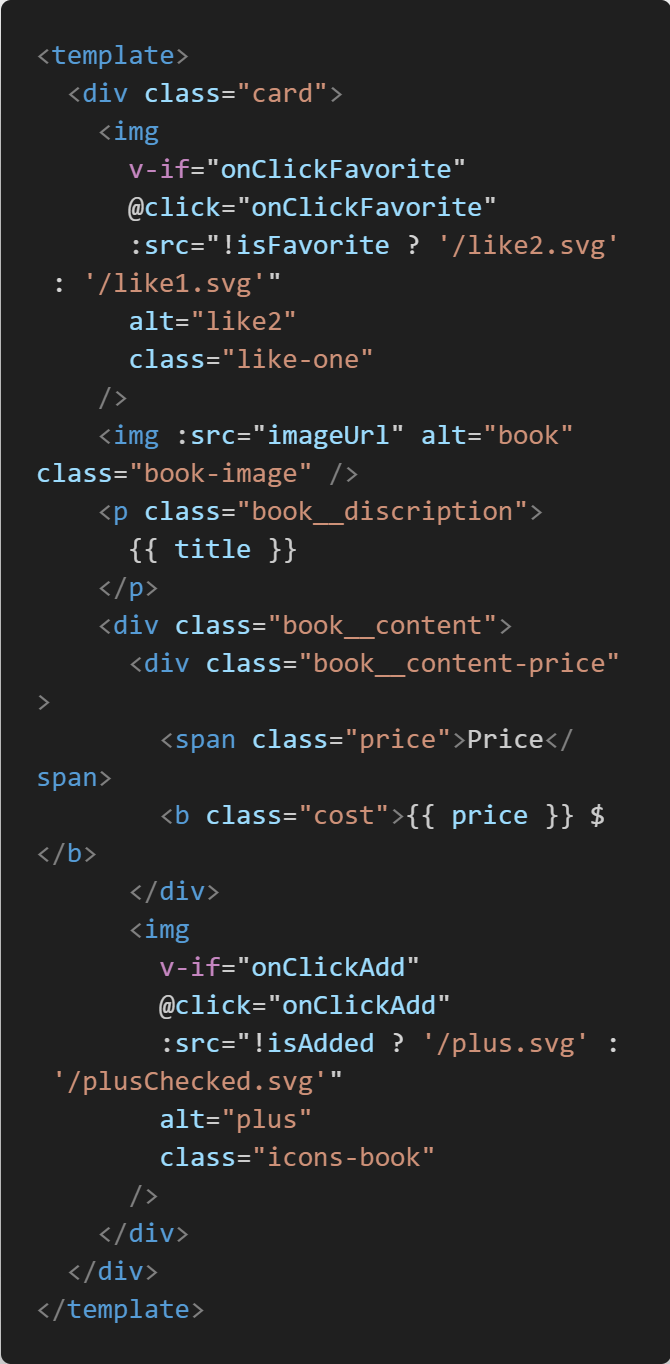


Стили компонента Favorites.

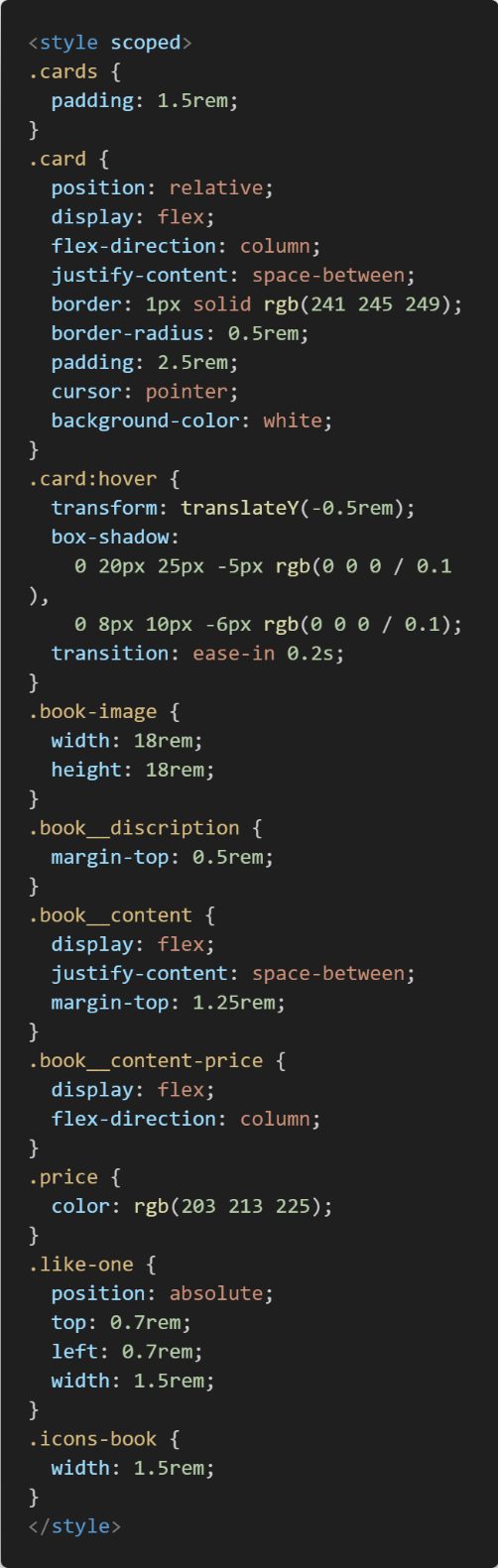
**Card.vue**



Компоненты Vue требуют явного объявления свойств, чтобы Vue знал, какие внешние свойства, переданные компоненту. Для каждого свойства в синтаксисе объявления объекта ключом является имя свойства, а значением должна быть функция-конструктор ожидаемого типа.

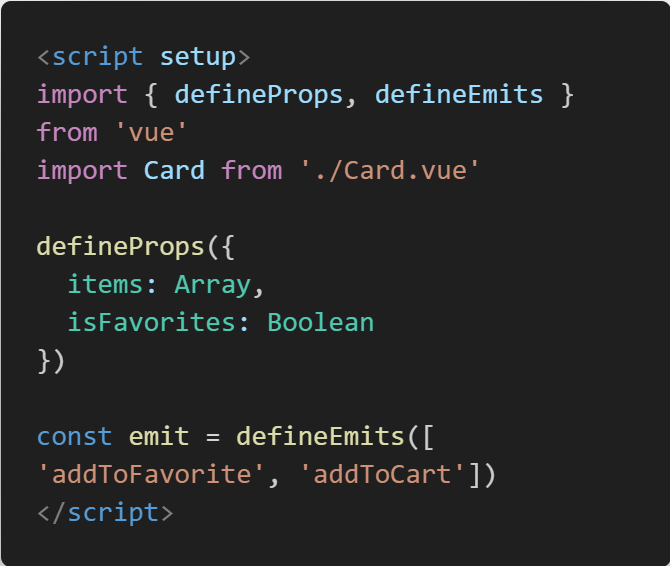


Шаблон компонента Card. По мимо уже упомянутых директив Vue, используется v-if для отрисовки блока по условию. Блок будет отображаться только в случае, если выражение директивы возвращает значение, которое приводится к true. Наиболее простой способ связывания данных — текстовая интерполяция с использованием «Mustache»-синтаксиса (двойных фигурных скобок)



Описание стилей компонента.

**CardList.vue**



Импортируем компонент Card для отображения контента на странице.

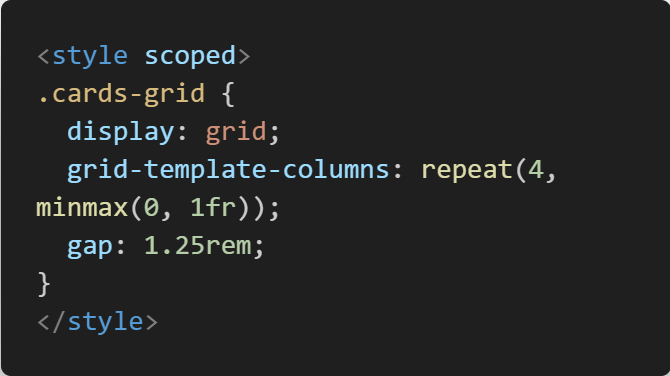
Можно определить события которые генерирует компонент с помощью опции defineEmits. Опция также принимает объект, что позволяет разработчику определить валидаторы для аргументов, которые передаются вместе с генерируемым событием, аналогично валидаторам при определении defineЗrops.



Для анимации используется AutoAnimate

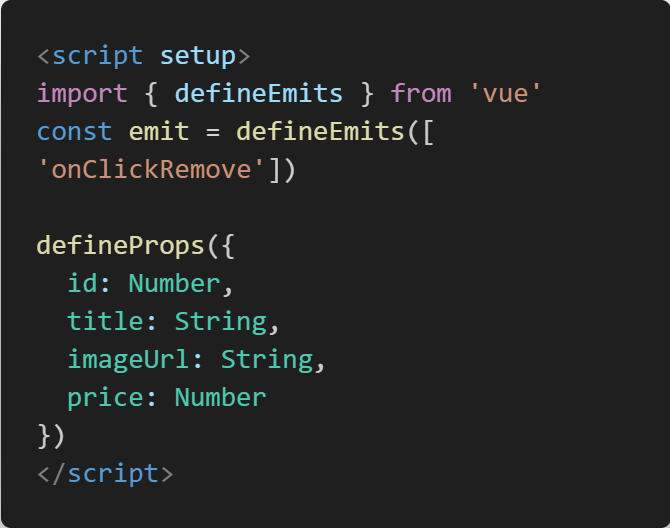
AutoAnimate — это простая в использовании утилита анимации, не требующая настройки, которая добавляет плавные переходы в ваше веб-приложение. Вы можете использовать его с React, Solid, Vue, Svelte или любым другим приложением JavaScript.

Шаблон компонента по мимо уже перечисленных директив используем v-for для отрисовки списка элементов на основе массива данных. У директивы v-for специальный синтаксис: item in items, где items — исходный массив, а item — ссылка на итерируемый элемент массива. Внутри блока v-for доступны все свойства из области видимости родителя. Также может быть второй опциональный параметр у v-for с индексом текущего элемента.

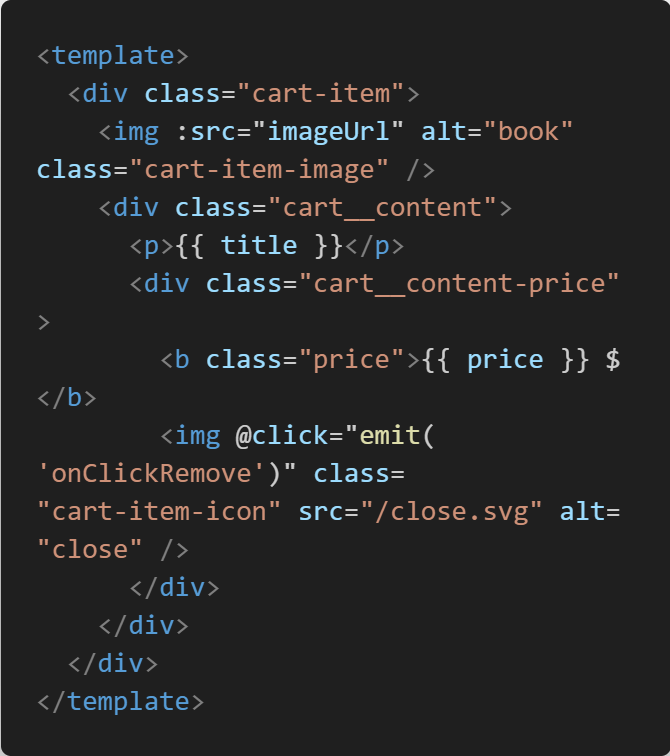


Описание стилей компонента CardList.

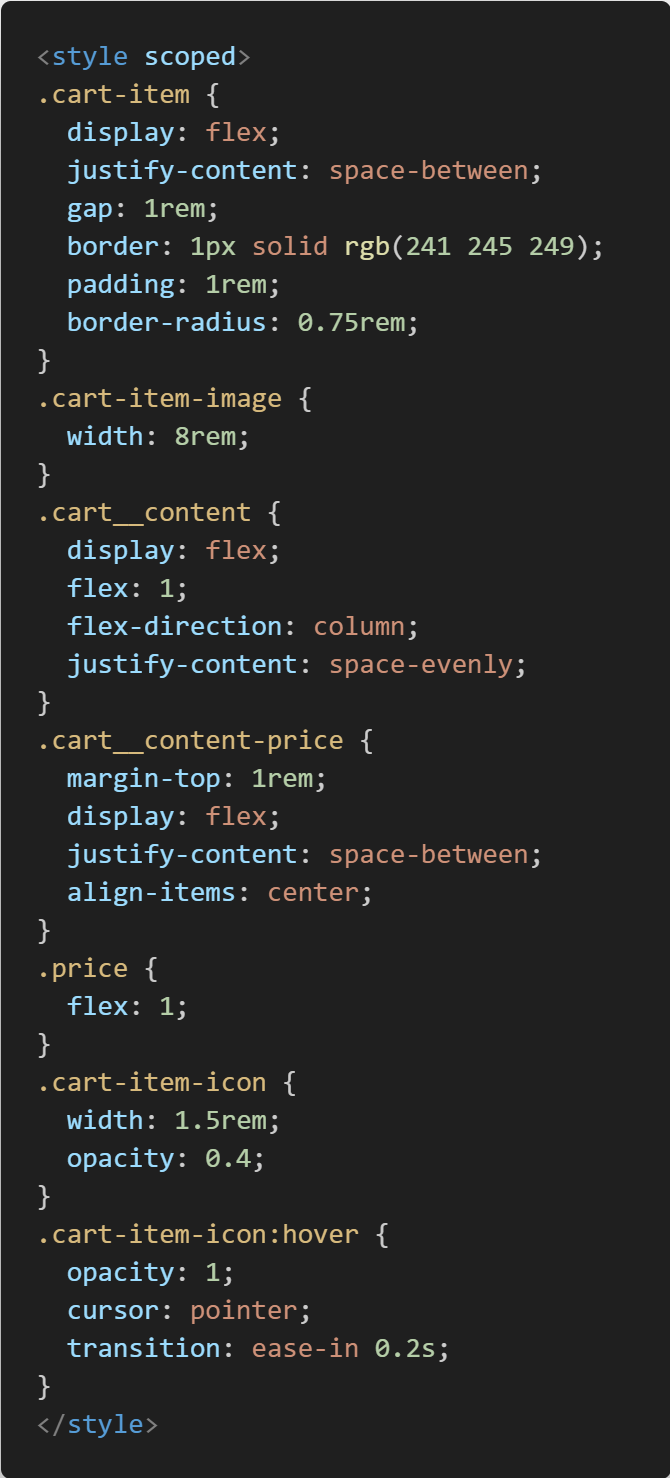
**CartItem.vue**



Импорт, emits, props.



Шаблон компонента CartItem.



Описание стилей компонента.

**CartItemList.vue**



Импортируем CartItem для отображения контента. inject для данных от родительского компонента.

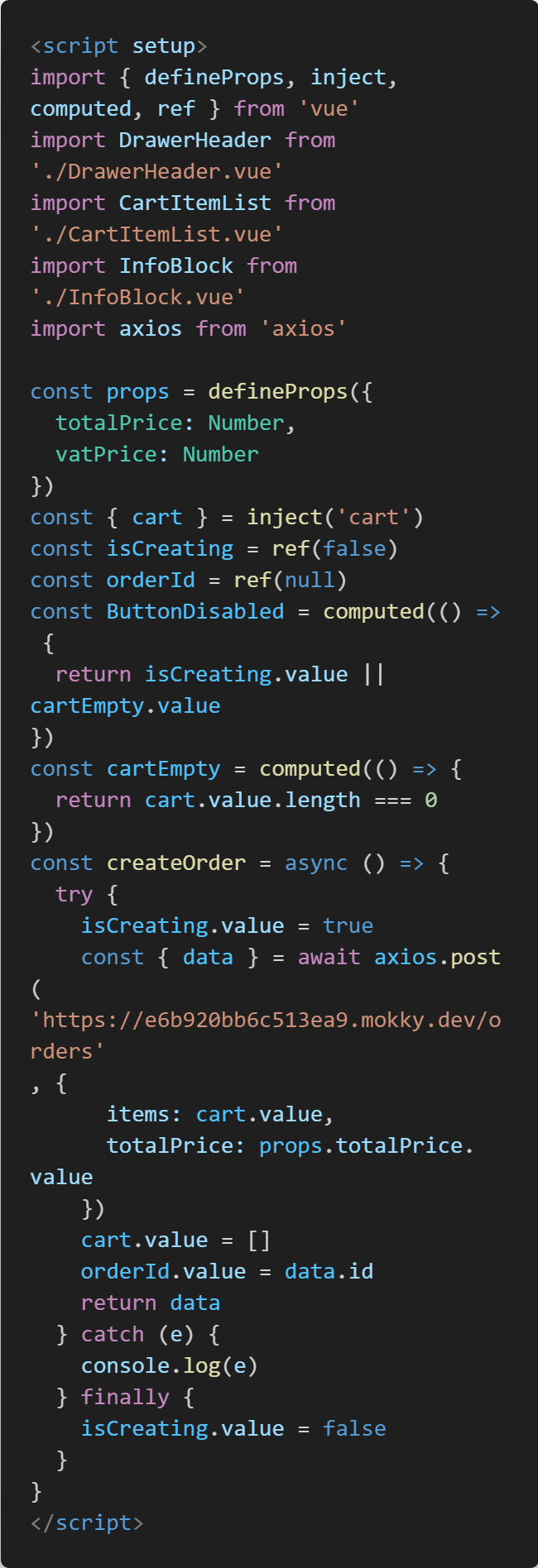


Шаблон компонента CartItemList.

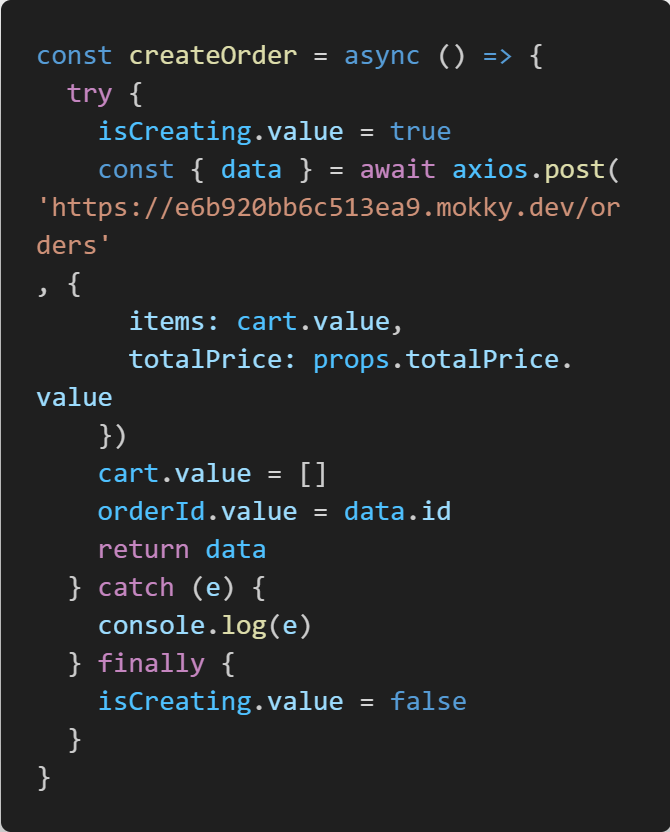


Описание стилей компонента.

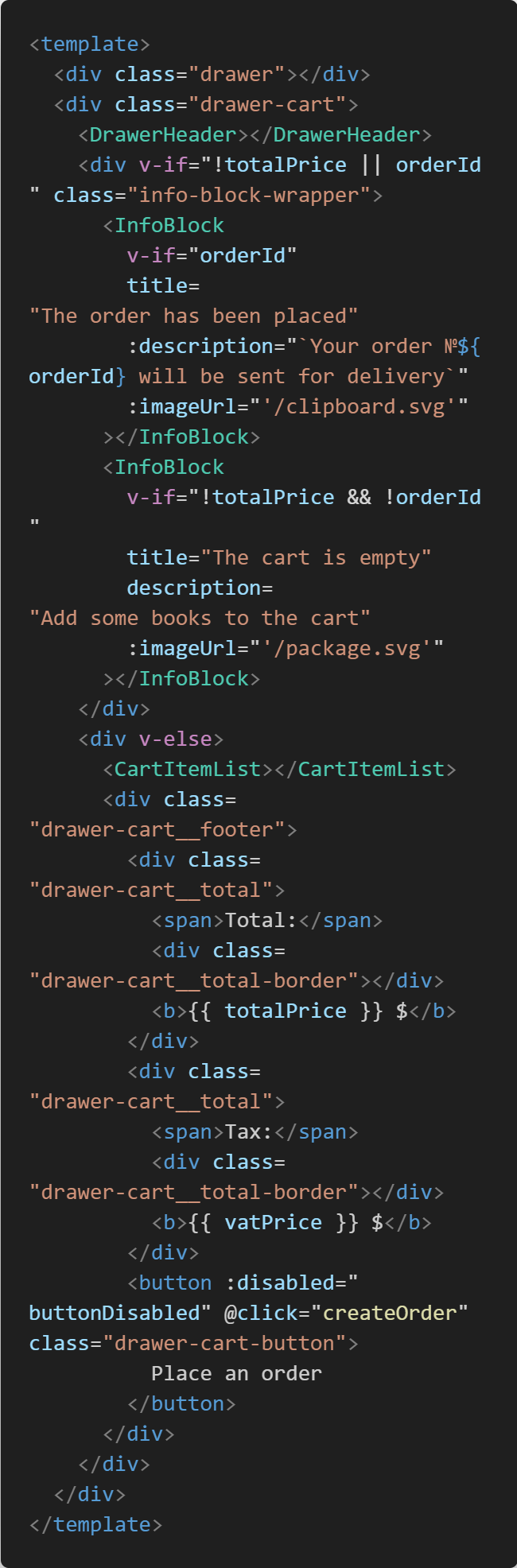
**Drawer.vue**



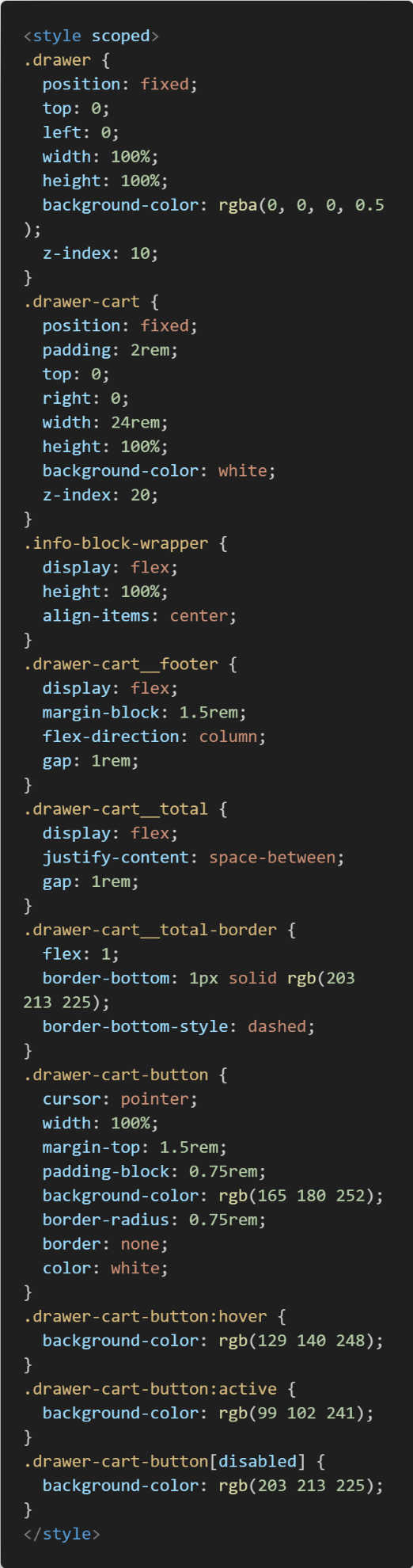
Импортируем компоненты DrawerHeader, CartItemList, InfoBlock



Функия createOrder применяется для создание заказа в корзине. Выставляем флаг isCreating в значение true. С помощью метода POST отправляем данные. Обрабатываем ошибки с помощью try … catch. В конце используем finally для изменения флага isCreating в значение false.

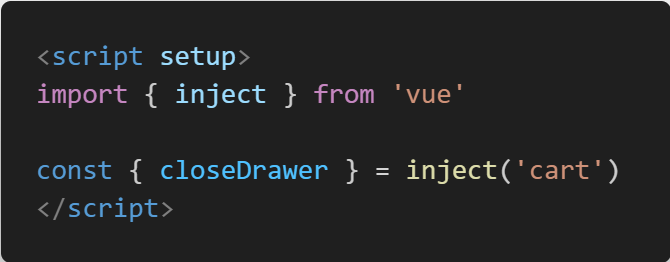


Шаблон компонента Drawer.

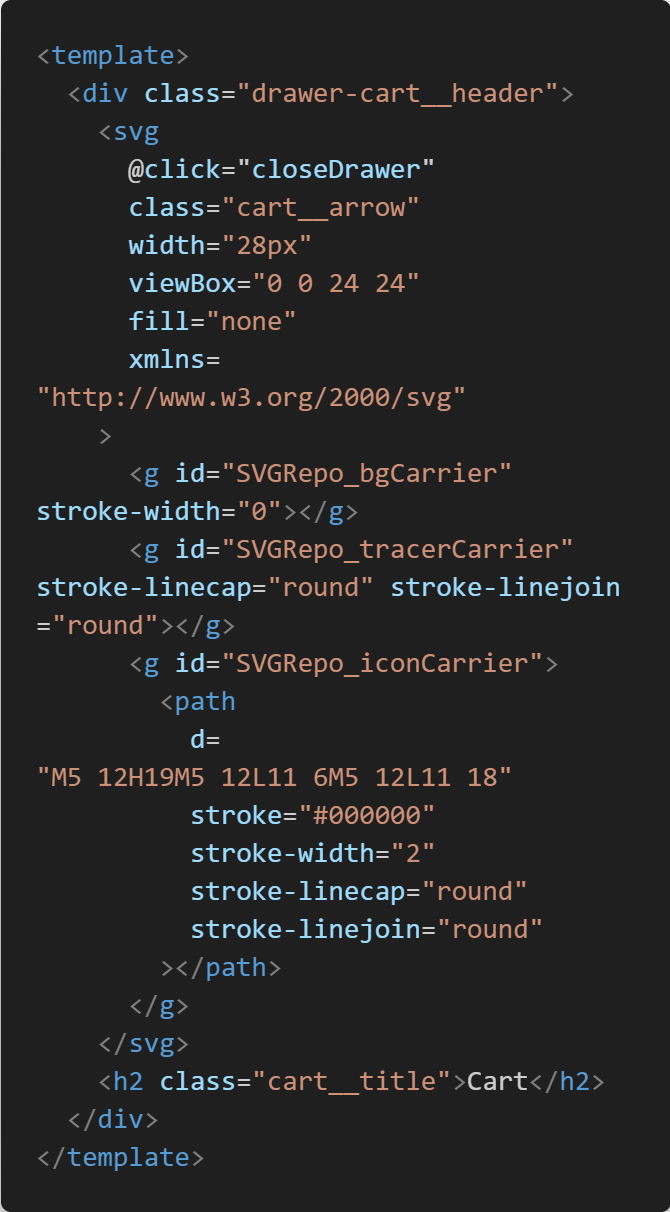


Описание тилей компонента.

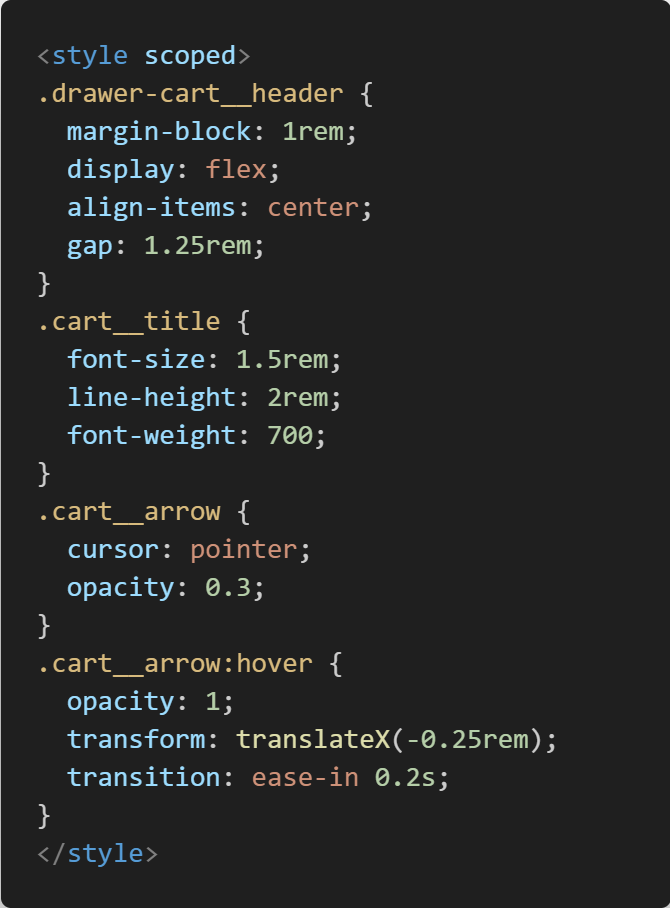
**DrawerHeader.vue**



Скрип компонента.

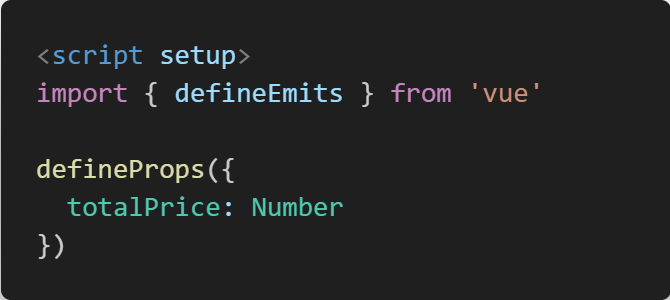


Шаблон компонента.

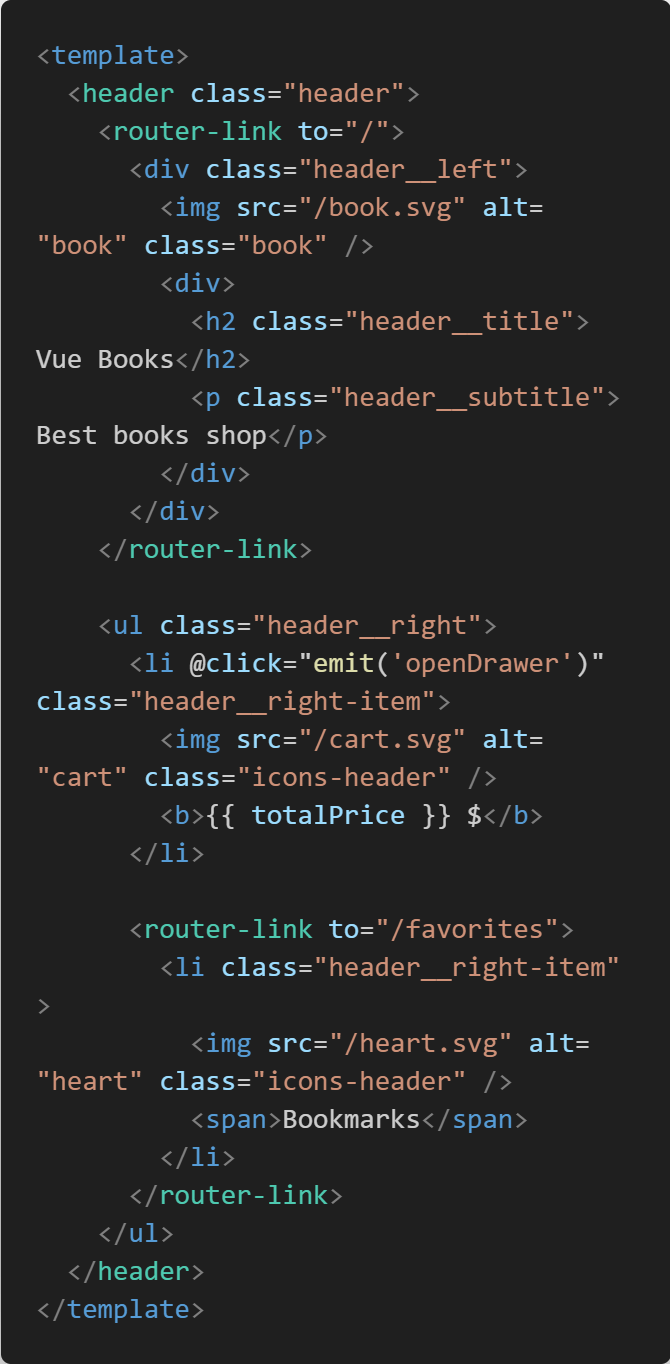


Стили компонента.

**Header.vue**

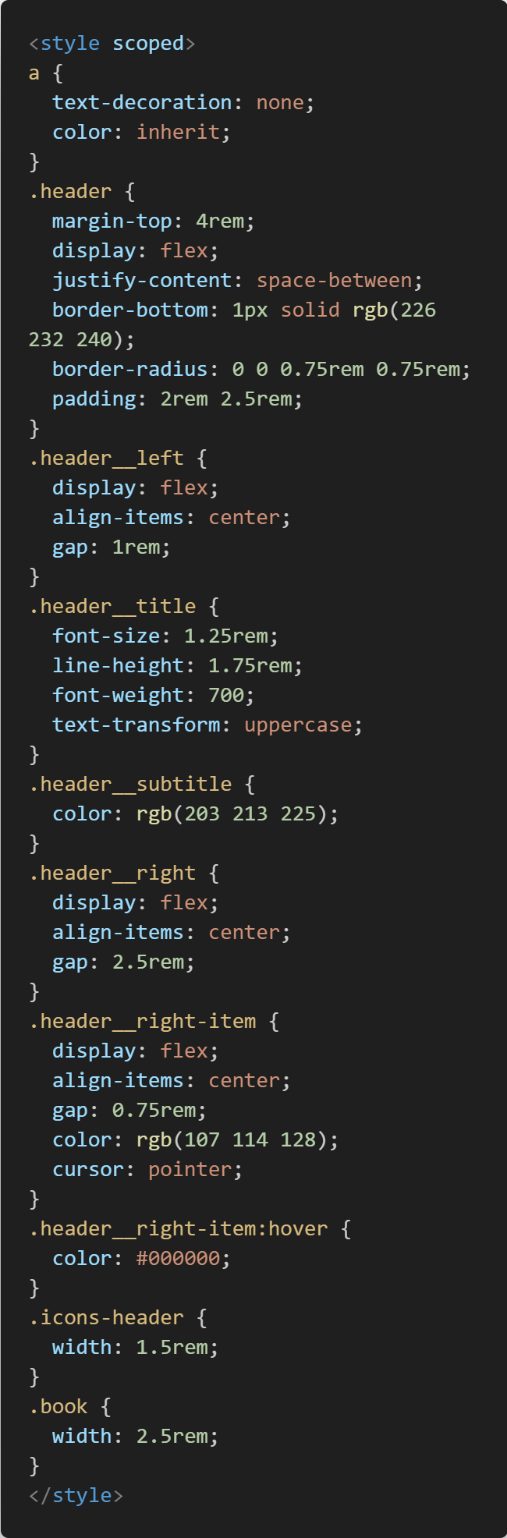


Скрипт компонента.



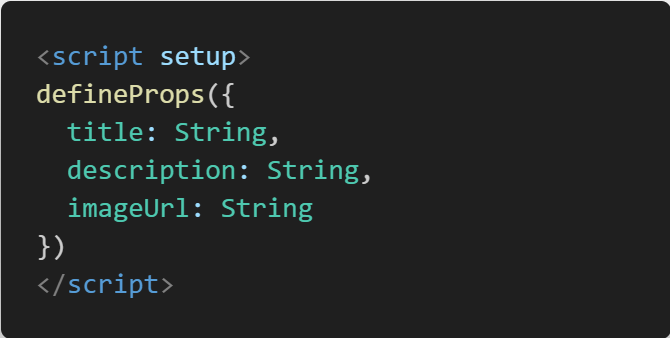
Шаблон компонента.

<router-link> — это компонент предназначенный для навигации пользователя в приложении с клиентской маршрутизацией. Путь назначения указывается входным параметром to. По умолчанию компонент рендерится в тег <a> с корректным значением href, но это можно изменить входным параметром tag. Кроме того, ссылка автоматически получает активный класс CSS при переходе на путь назначения.

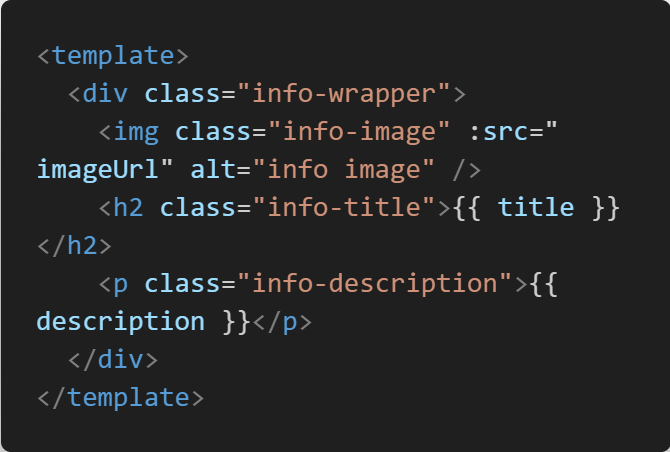


Описание стилей компонента.

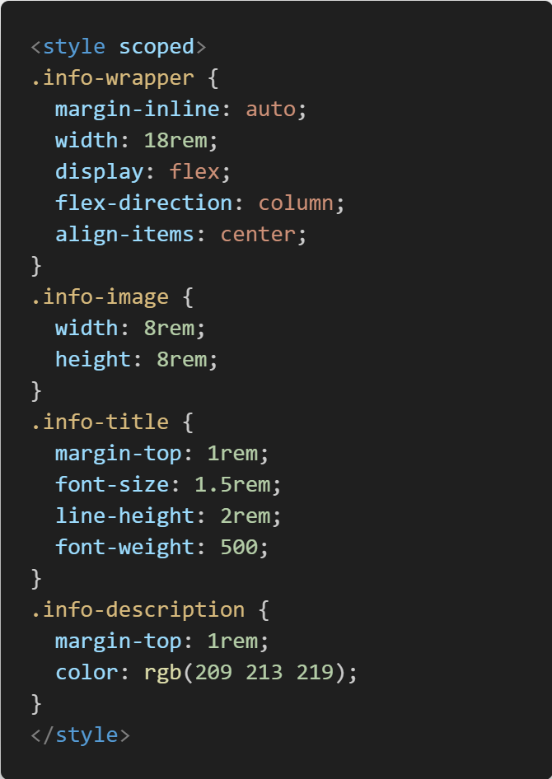
**InfoBlock.vue**



Скрипт компонента



Шаблон компонента.



Стили компонента

**Заключение**

В ходе разработки фронтенд-проекта применялся фраймворк Vue с передачей данных между компонентами, роутингом (Vue Router), запросами к бэкенду c использованием Mokky, анимацией. Mokky - это сервис, который позволяет вам легко создавать тестовое API для ваших пет-проектов, генерировать пользовательские данные и манипулировать этими данными с помощью готовых RESTApi. При создании приложения применялся Composition API.