УДК 004.4’22

Матвеев В. С.
Магистрант 2 курса, ИВТМ-18,
факультет фундаментальных и
гуманитарных дисциплин
Санкт-Петербургского Горного университета

СОВРЕМЕННЫЕ ФРЕЙМВОРКИ ДЛЯ РАЗРАБОТКИ ИНФОРМАЦИОННЫХ СИСТЕМ НА ПРИМЕРЕ ПЛАТФОРМЫ VUE

В данной статье представлено описание платформы Vue, используемой для создания пользовательских интерфейсов. Vue спроектирован таким образом, что его можно применять постепенно. Основную библиотеку легко подобрать и интегрировать с другими библиотеками или существующими проектами, но так же подходит для использования в сочетании с современными инструментами и вспомогательными библиотеками.

Ключевые слова: Vue, интерфейс, приложение, платформа веб-разработки

Matveev V. S.
2nd year master's student, IVTM-18,
department of fundamental science and
Humanities
Saint Petersburg Mining University

MODERN FRAMEWORKS FOR DEVELOPING INFORMATION SYSTEMS ON THE EXAMPLE OF THE VUE PLATFORM

This article describes the Vue platform used for creating users' interfaces. Vue is designed so that it can be applied gradually. The main library is easy to select and integrate with other libraries or existing projects, but it is also suitable for use in combination with modern tools and auxiliary libraries.

Keywords: Vue, interface, application, web development platform

Фреймворки (Frameworks) представляют собой платформы веб-разработки, которыми руководствуются разработчики при создании веб-приложений высокого качества, используя один язык программирования.

Важное значение при изучении компонентов поля веб-приложений для разработчиков играют их рамки, общей целью которых является содействие развитию. Таким образом, результат достижения данной цели и является отличительной чертой. В последние годы была отмечена высокая потребность во фронтенд-фреймворках, ориентированных на веб-разработку. Для построения веб-приложения существует множество фреймворков. Казалось бы, что выбор фреймворка – это довольно простая задача, но реализовать его в целом приложении и тем более в ИС сложно.

Рассмотрим платформу Vue, разработанную Evan You, бывшим сотрудником Google, которая является наиболее быстро растущим JavaScript-фреймворком. Он описывается как перцептивный, быстрый MVVM (Model–View–ViewModel), предназначенный для создания интерактивных интерфейсов. Первый публичный релиз данного фреймворка состоялся в феврале 2014 года. Платформа Vue.js – это прогрессивная структура, которая может быть использована для создания пользовательских интерфейсов. В отличие от других монолитных каркасов, Vue спроектирован с нуля, что делает возможным его постепенное применение. Его основная библиотека ориентирована только

на определенный слой представления, и ее легко сочетать и интегрировать с другими библиотеками или уже существующими проектами.

В то же время, Vue идеально подходит для «накачки» сложных одностраничных приложений при условии использования в сочетании с современными инструментами и вспомогательными библиотеками.

Многие компании, например китайские, такие как: Alibaba, Baidu, Xiaomi, Sina Weibo и др. широко используют Vue. Он входит в ядро Laravel и PageKit. Не так давно GitLab, система управления репозиториями, тоже перешла на Vue.js.

В конце сентября 2016 года вышел в релиз Vue.js 2.0. Данная платформа обладает еще большими возможностями. Она была создана с упором на производительность: используется виртуальный DOM, поддерживается серверный рендеринг, возможность использовать JSX и т.д. Хотя сейчас данный фрейворк поддерживается только сообществом, но даже на уровне продуктов таких гигантов, как Google и Facebook (Angular 5-6 и React 15) выглядит достойно и постепенно догоняет их по популярности.

Основными концепциями Vue являются:

- Конструктор;
- Компоненты;
- Директивы;
- Переходы.

Работа с Vue.js начинается с создания нового инстанса new Vue: el – элемент, за которым следит Vue. В template выбран (либо прописан инлайново) элемент, куда Vue будет рендерить. В data хранится текущее состояние инстанса, а метод computed предоставляет нам вычисляемые свойства.

В methods можно выделить следующие кастомные методы и методы жизненного цикла Vue:

- **beforeCreate** — смотрит данные и инициализирует события;
- **created** — смотрит, есть ли el или template. Если есть, то рендерит в них, если нет, то ищет метод render;
- **beforeMount** — создает vm.$el и заменяет им el;
- **mounted** — элемент отрендерен.

При изменении состояния:

- **beforeUpdate** — снова рендерит VDOM и сравнивает с реальным DOM-ом, применяет изменения;
- **updated** — изменения отрендерены;
- **beforeDestroy** — полный демонтаж watchers, внутренних компонентов и слушателей событий;
- **destroyed** — вызывается, когда выполнение операции останавливается.

Графически работу с Vue можно представить на рисунке 1.

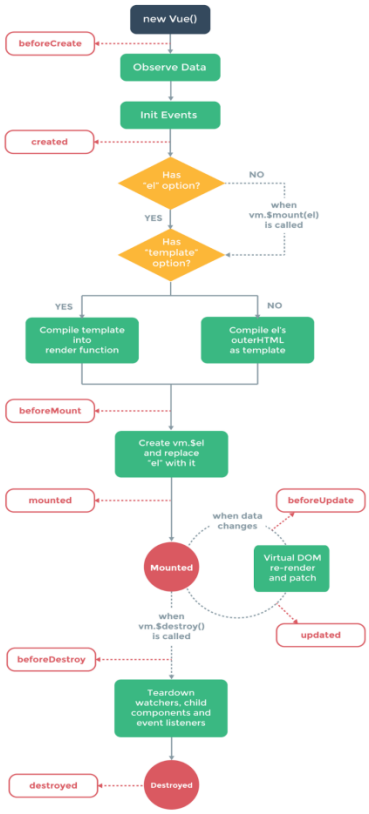

Рис. 1 Схема работы на платформе Vue

Директивами в этом фреймворке являются специальные атрибуты для добавления элементам html дополнительной функциональности.

Рассмотрим некоторые встроенные директивы:

- V-bind – динамически связывается с одним или несколькими атрибутами.
- v-cloak – прячет «усатые» выражения, пока не подтянулись данные
- v-if – условие для рендера элемента
- v-else – обозначает “else блок” для v-if
- v-for – циклично проходит массив объектов
- v-model – связывает состояние с input элементом
- v-on – связывает слушателя события с элементом
- v-once – рендерит элемент только вначале и больше не следит за ним
- v-pre – не компилирует элемент и его дочерние элементы
- v-show – переключает видимость элемента, изменяя свойство CSS display
- v-text – обновляет textContent элемента.

Все Vue-директивы имеют префикс “v-”. В директиву передается какое-то значение состояния, а в качестве аргументов могут быть атрибуты html или события.

Компоненты помогают расширить основные html элементы и внедрить переиспользуемый код. По сути, компоненты – это повторно используемые части UI.

На этапе проектирования мы разбиваем наше приложение на независимые части и получаем древовидную структуру компонентов как это представлено на рисунке 2.

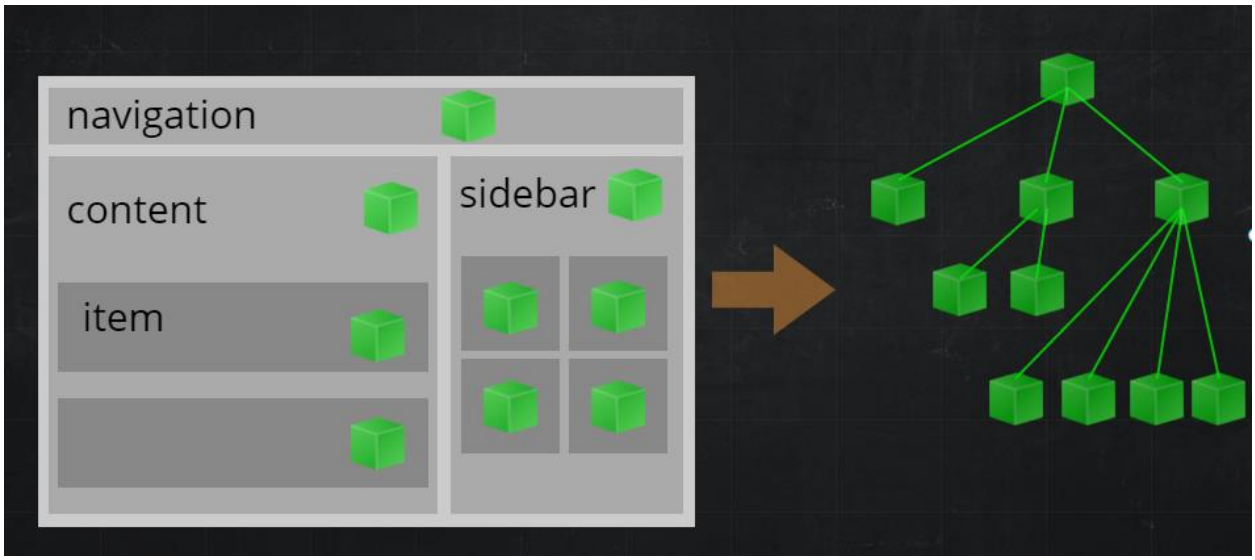

Рис.2 Структура компонентов в Vue.js

В Vue.js нет особых требований к именам компонентов, но лучше придерживаться правил W3C насчет кастомных компонентов, то есть буквы нижнего регистра и разделения через дефис.

Коммуникация между vue-компонентами осуществляется по принципу “Props in, Events out”. То есть от родительского элемента к дочернему информация передается через пропсы, а обратно – вызываются события.

Также во Vue.js есть так называемые однофайловые компоненты. При создании файла с расширением .vue туда вписывают стили, шаблон и логику. Причем писать можно на любом удобном препроцессоре (SASS, Stylus, PostCSS, Jade, ...) и языке, компилирующимся в JS (CoffeeScript, TypeScript).

Vue предоставляет различные способы применения анимационных эффектов, когда элементы отрисованы, обновлены или удалены из DOM. Они включают в себя инструменты для:

- автоматического применения классов для CSS-переходов и анимаций

- интеграции сторонних библиотек для CSS-анимаций, таких как Animate.css
- использования JavaScript для манипуляции DOM-ом
- интеграции сторонних JavaScript библиотек для анимаций, таких как Velocity.js

На платформе реализована возможность управления состоянием через паттерн Vue.js.

Этот паттерн является библиотекой управления состоянием для приложений на Vue.js. Он предоставляет централизованное общее состояние для всех компонентов в приложении и правила, обеспечивающие предсказуемое изменение состояния.

На рисунке 3 представлено приложение на Vue+Vuex со следующими частями:

- Состояние (State), которое является единственным источником данных для компонентов.
- Vue-компоненты (Vue-components), которые по сути являются лишь декларативным отображением состояния.
- Действия (Actions), которые отлавливают событие, которое произошло, собирают данные с внешних API и запускают нужные мутации.
- Мутации (Mutations) — единственная часть, которая может изменять состояние и, получив данные от Actions, применяет их на состоянии.

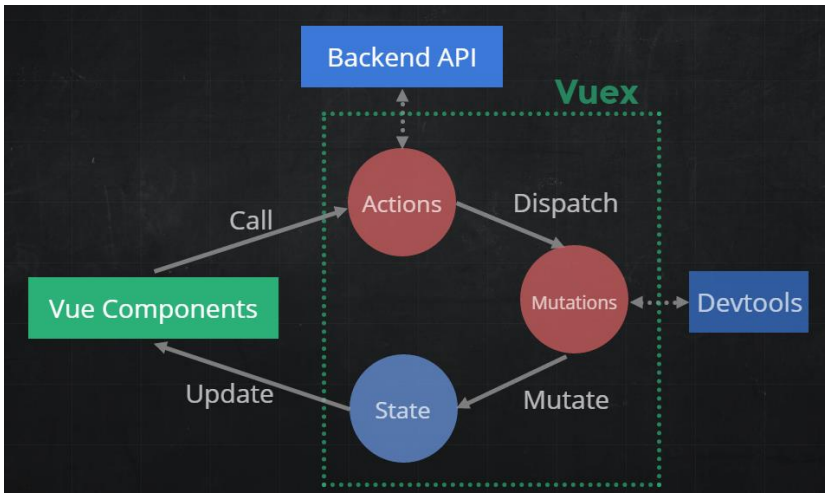

Рис. 3 Библиотека управления на Vue+Vuex

Таким образом, как было показано на примере проведенного анализа фреймворка, используемого для создания современных ИС и КИС, выбор платформы для разработчиков должен основываться, прежде всего, на понимании общих принципов взаимодействия в процессе организации профессионального подхода к соответствующей разработке.

Список использованных источников

1. Эрик Хэнчетт Vue.js. в действии. Профессиональная литература: Издательский дом «Питер» ,2018. 336 с.
2. Full Stack Vue.js with Firestore URL: **https://medium.com/vue-mastery/full-stack-vue-js-with-firestore-62e2fe2ec1f3** (дата обращения 28.05.2020)