УДК 004.02, 004.5

ПРИМЕНЕНИЕ КОМПОНЕНТНОГО ПОДХОДА ДЛЯ РАЗРАБОТКИ АРХИТЕКТУРЫ FRONTEND-ЧАСТИ СОЦИАЛЬНОЙ СЕТИ

Ширяева Диана Владимировна[1], Ушанкова Мария Юрьевна[2]

[1]*Студент;*
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г. Дубна, ул. Университетская, д. 19;
e-mail: shdv.19@uni-dubna.ru.

[2]*Старший преподаватель;*
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г. Дубна, ул. Университетская, д. 19;
e-mail: ushankova.m.ju@uni-dubna.ru.

Компонентный подход – это подход к разработке программного обеспечения, который заключается в создании независимых компонентов, которые могут быть использованы повторно в различных частях приложения. В статье рассматривается суть компонентного подхода, его преимущества и недостатки, а также фреймворки, построенные на компонентном подходе. Статья также описывает технологии, которые используются при создании компонентов, такие как Templates, Shadow DOM, Custom Elements и HTML imports и объясняет, как они могут быть использованы для создания более эффективных и гибких приложений. В целом, статья является полезным ресурсом для разработчиков, которые хотят улучшать свои навыки и создавать более качественные и масштабируемые приложения.

Ключевые слова: компонентный подход, архитектура *frontend*, социальные сети, *React*, *Vue*, *Angular*.

Для цитирования:

Ширяева Д. В., Ушанкова М. Ю. Применение компонентного подхода для разработки архитектуры frontend-части социальной сети // Системный анализ в науке и образовании: сетевое научное издание. 2023. № 2. С. 21-26. EDN: FELBVK. URL : https://sanse.ru/index.php/sanse/article/view/575.

USING A COMPONENT-BASED APPROACH TO DEVELOP THE FRONTEND ARCHITECTURE OF A SOCIAL NETWORK

Shiryaeva Diana V.[1], Ushankova Maria Yu.[2]

[1]*Student;*
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: shdv.19@uni-dubna.ru.

[2]*Senior teacher;*
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: ushankova.m.ju@uni-dubna.ru.

The component-based approach is an approach for software development that involves the creation of independent components that can be used in various parts of the application. The article discusses the essence of the component-based approach, advantages and disadvantages, as well as frameworks built. The article also describes technologies that are used for creating components, such as Templates, Shadow DOM, Custom Elements and HTML imports and explains how it can be used to create more efficient and flexible applications. In general, the article is a useful resource for developers, who want to improve their skills and create better and scalable applications.

Keywords: component approach, frontend architecture, social network, React, Vue, Angular.

For citation:
Shiryaeva D. V., Ushankova M. Yu. Using a component-based approach to develop the frontend architecture of a social network. *System analysis in science and education*, 2023;(2):21-26 (in Russ). EDN: FELBVK. Available from: https://sanse.ru/index.php/sanse/article/view/575.

Введение

Мир веб-разработки развивается стремительно быстро. На начало 2022 года в мире существует более двух миллиардов сайтов, и каждый день появляется еще более полумиллиона [1]. Есть множество причин такой популярности.

1. Кроссплатформенные приложения. Веб-приложения могут работать на разных операционных системах и устройствах без необходимости создания отдельной версии для каждой из них.
2. Широкое распространение Интернета. Сегодня люди пользуются Интернетом десятки раз за день, что делает веб-приложения более удобными и доступными.
3. Большое количество инструментов и технологий. Существует множество библиотек, фреймворков и инструментов, что делает процесс разработки более быстрым и эффективным.
4. Простота использования.
5. Возможность создания интерактивных и динамических приложений. Такие приложения обеспечивают привлекательный пользовательский интерфейс и улучшают юзабилити.
6. Большое сообщество разработчиков. Имеется очень большое сообщество разработчиков, которые создают новые инструменты, делятся своим опытом и помогают друг другу в решении проблем.

Также быстро развивается и *JavaScript* – язык программирования, который используют разработчики для создания интерактивных веб-страниц. Впервые за много лет главной тенденцией в развитии *JavaScript*-экосистемы становится стандартизация [2]. Все элементы системы, от стиля до тестирования, – перешли на компонентный подход.

Компонентный подход – это подход к разработке программного обеспечения, в котором приложение строится из независимых компонентов, каждый из которых выполняет определенную функцию и может быть использован многократно в различных местах приложения.

Такой подход подходит для всех типов приложений, так как значительно увеличивает скорость разработки и уменьшает дублирование кода. Но стоит отметить, что компонентный подход становится просто незаменим, когда речь идет о приложениях с большим количество одинаковых элементов, например, корпоративные порталы или социальные сети.

Социальные сети состоят из множества одинаковых элементов таких, как посты, профиль пользователя, комментарии, сообщения, чаты и т.д. Поэтому выделение и разработка компонентов являются первым шагом при разработке архитектуры *frontend*-части.

Веб-компоненты

Веб-компоненты (*web-components*) – это технология, которая позволяет создавать переиспользуемые компоненты для веб-приложений. Они позволяют разрабатывать собственные пользовательские элементы, которые могут быть использованы в различных приложениях без необходимости повторного создания.

К веб-компонентам относятся следующие стандарты: пользовательские элементы (*custom elements*), теневой *DOM* (*shadow DOM*), шаблоны (*templates*) и *HTML*-импорты (*HTML imports*). Чаще всего они используются все вместе, потому что по отдельности они слабо полезны.

Шаблоны – специальный тег *<template>*, предназначенный для хранения шаблонов. Данный тег может содержать в себе любую разметку, которая будет читаться наравне с остальными тегами, но при это не будет выполняться. Это значит, что разметка не будет добавлена в *DOM*-дерево.

Спецификация теневого *DOM* позволяет изолировать стили и поведение компонента от остальной части страницы. Это помогает избежать конфликты между различными компонентами.

Стандартная модель *DOM* предполагает, что все потомки элемента доступны через *childNodes*, то есть их можно найти через *querySelector()*. Но есть возможность отображать не то, что находится в *DOM*-дереве, а какую-либо другую разметку. Для этого можно добавить теневой *DOM* с помощью метода *attachShadow()*. После этого вместо обычного *DOM*-дерева у элемента появляется сразу три других: *Shadow DOM* (теневой *DOM*), *Light DOM* (то, что раньше являлось обычным *DOM*-деревом) и *Flattened DOM* (объединение двух предыдущих, то, что человек действительно видит на экране).

Пользовательские элементы – это возможность создавать новые *HTML*-теги с произвольными именами и поведением, например, *<player src=""></player>* или *<map lat="34.86974" lon="-111.76099" zoom="7"></map>*.

Используя все остальные стандарты, можно отделить визуализацию от логики, но на практике гораздо удобнее собирать все исходники в одном месте. Так, *HTML*-импорт позволяет создать один *HTML*-документ, который содержит все ресурсы компонента (разметку, стили, скрипты).

Компоненты быстрые, оптимизированные, удобные, и поддерживаются всеми популярными браузерами (см. рис. 1).

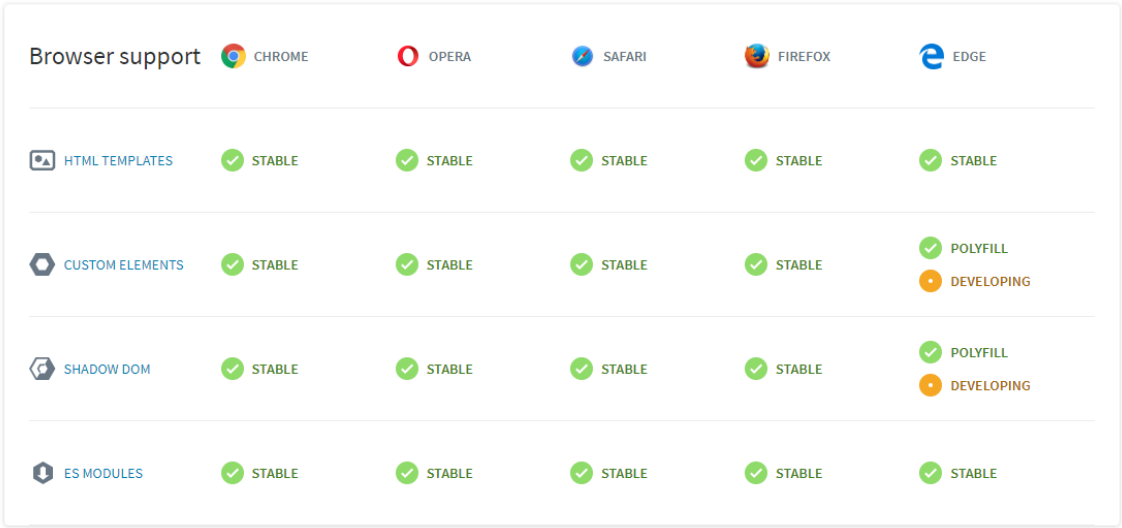

Рис. 1. Поддержка компонентов браузерами

Таким образом, мы получаем модульные, переиспользуемые и легко поддерживаемые компоненты. Они обеспечивают простой и единообразный способ создания пользовательского интерфейса приложений.

Компонентный подход

Суть этого подхода заключается в том, что при разработке приложений в первую очередь создают компоненты. При этом они должны быть как можно более независимыми друг от друга. Это означает, что разработчик не просто создает набор компонентов, он еще и реализует так называемую дизайн-систему компонентов пользовательского интерфейса.

Компоненты можно создавать в рамках самого приложения и в формате отдельного проекта – в виде библиотеки компонентов.

Применение компонентного подхода предусматривает разработку компонентов в виде первого шага работы над проектом и подразумевает, что компоненты стремятся делать подходящими для многократного использования. Это позволяет применять их при создании других приложений.

У данного подхода есть свои преимущества и недостатки, которые представлены в таблице 1.

Табл. 1. Преимущества и недостатки компонентного подхода

Преимущества	**Недостатки**
Повторное использование кода	Горизонтальный рост папок в *components*
Улучшение модульности	Снижение производительности, если компоненты не оптимизированы и не эффективно используют ресурсы
Упрощение процесса тестирования	Необходимость дополнительного этапа проектирования архитектуры взаимодействия компонентов
Упрощение процесса масштабируемости	
Легкость поддержки приложения	

Несмотря на некоторые недостатки, компонентный подход является эффективным и популярным подходом в современной разработке приложений, и может значительно упростить и ускорить процесс разработки и поддержки приложения.

Фреймворки, построенные на компонентном подходе

Сегодня каждый *JavaScript*-фреймворк адаптирован к компонентному подходу. И это касается не только самых крупных (*React*, *Angular*, *Vue*), но и *Ember*, *Dojo*, *Mithril* – все они используют компоненты как ключевую идею по части пользовательского интерфейса.

Можно долго спорить какой фреймворк *React*, *Angular* или *Vue* лучше, но нельзя отрицать, что каждый из них пользуется высоким спросом и успешно применяется при веб-разработке. Это подтверждают данные, представленный в *Google Trends* за последние 5 лет (см. рис. 2). Синяя, красная и желтая линии представляют *React*, *Vue.js* и *Angular* соответственно.

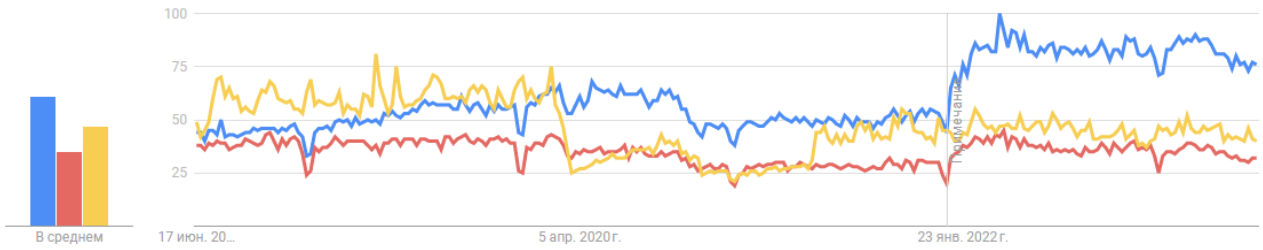

Рис. 2. Спрос в веб-поиске на фреймворки

Все эти фреймворки активно используются популярными компаниями, например, *Angular* используют *Google*, *Microsoft Office* и *Upwork*, *ReactJS – Twitter*, *Amazon Prime*, *Codesandbox*, *Uber* и *Pinterest*, а *Vue.js – Xiaomi*, *Wizz Air*, *Adobe*, *Behance* [3].

У каждого фреймворка есть свои плюсы и минусы, поэтому выбор во многом зависит от цели. Вот несколько отличий, на которые можно ориентироваться при выборе.

Используем *Angular*, когда:

- имеем дело с большим сложным проектом;

- нужна надежная масштабируемость;
- есть время изучить *Typescript*.

Используем *Vue.js*, когда:

- необходимы высокопроизводительные приложения;
- не хватает квалифицированных разработчиков;
- достаточно времени, чтобы ознакомиться с новыми технологиями.

Используем *React*, когда:

- проекты имеют простой интерфейс;
- проекты, требующие высочайшего уровня масштабируемости;
- проекты с короткими сроками.

Повторное использование компонентов

Концепция компонентного подхода в *frontend* разработке открывает невероятные возможности для повторного использования уже написанных компонентов в новых проектах. Инструменты наподобие *Bit* позволяют делать это.

Bit – проект с открытым исходным кодом, предназначенный для сбора ваших компонентов и их удобного переиспользования. Можно мгновенно подключать в ваше приложение любой строительный блок, созданный ранее (вами или тысячами других разработчиков по всему миру).

Если вы захотите доработать ваш компонент, то можете делать это прямо из проекта. *Bit* подтянет все изменения, и другие проекты автоматически с ними синхронизируются. Вы получаете полный контроль над изменениями исходного кода и зависимостями вашего элемента.

Благодаря *bit.dev* вы можете легко наладить обмен компонентами между членами одной команды, обеспечивая тем самым однообразие вашего приложения. Концентратор предоставляет все необходимое для удобной совместной работы: от отличного поиска до полной поддержки *CI/CD* (непрерывная доставка и интеграция кода).

Так, *bit* дает разработчику широкие возможности по созданию, тестированию, повторному использованию компонентов, позволяет дорабатывать их где угодно – независимо от того, где именно они были созданы.

Заключение

Таким образом, можно сделать вывод, что компонентный подход – это эффективный тренд в веб-разработке, потому что он упрощает разработку, ускоряет процесс создания приложения, улучшает модульность и повышает качество программного продукта.

Кроме того, данный подход делает приложение более гибким и масштабируемым, позволяя добавлять новые компоненты или изменять существующие без необходимости изменения всего приложения.

И, наконец, данный подход позволяет повысить эффективность разработки, так как разработчики могут использовать готовые компоненты вместо того, чтобы писать их с нуля. Это уменьшает время и усилия, затраченные на разработку, и позволяет сосредоточиться на более важных задачах.

Список источников

1. Чуранов Е. Интернет в России в 2022 году: самые важные цифры и статистика // *WebCanape* : [веб-студия]. – ООО «Твинс», 2008–2023. – Дата публикации: 20.02.2022. – URL: https://www.web-canape.ru/business/internet-v-rossii-v-2022-godu-samye-vazhnye-cifry-i-statistika/.

2. Ecma International утвердила ECMAScript 2022: что в ней нового? // Хабр : [сайт]. – Habr, 2014-2023. – Дата публикации: 09.07.2022. – Перевод, автор оригинала: Dr. Axel Rauschmayer. – URL: https://habr.com/ru/articles/676032/.

3. React vs Angular vs Vue.js – What to choose in 2021? // medium.com : [сайт]. – Medium, 1998-2023. – Дата публикации: 16.03.2021. – URL: https://medium.com/techmagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2021-b91e028fa91d.