Столыпинский
вестник

Научная статья
Original article
УДК 004

АРХИТЕКТУРНЫЕ РЕШЕНИЯ В БИБЛИОТЕКИ РЕАКТ

ARCHITECTURAL SOLUTIONS IN THE REACT LIBRARY

Яровая Екатерина Владимировна, Магистрант Гродненского государственного университета им Янки Купалы, г.Гродно

Katsiaryna V. Yaravaya, Master's student Yanka Kupala State University of Grodno, Grodno

Аннотация

В статье мы затронем историю создания React. Ключевые изменения, которые внесли особо значительные изменения и которые сделали React таким какой он есть сегодня. Более подробно рассмотрим Fiber и механизмы его работы, узнаем, что делает его таким особенным и почему он так сильно повышает производительность приложений, написанных с помощью библиотеки react. Разберем как fiber создает “виртуальное дерево” и как он взаимодействует с ним при изменении элементов на странице. Рассмотрим хуки, которые используются в функциональных компонентах, как применяются, для чего созданы и узнаем где они не используются.

Annotation

Here we'll cover the history of React's creation. The key changes that made React the way it is today. We take a closer look at Fiber and its operational mechanisms, and learn what makes it so special and why it greatly improves the performance of applications written with react. We'll describe how fiber creates the "virtual tree" and how it interacts with it when elements on the page change. Let's look at the hooks that are used in functional components, how they are used, what they are created for and find out where they are not used.

Ключевые слова: React, JavaScript, Fiber, архитектура приложения

Keywords: React, JavaScript, Fiber, application architecture

Когда появилась библиотека React, большое количество разработчиков отнеслись к ней скептически, но скоро реакт отметит свой десятилетний юбилей. Многие приняли решение с ним работать, мы говорим о таких огромных компаниях, как Twitter Uber и AirBnb - это достаточно большие компании, которые используют React и поняли, что он помогает делать отличные приложения и ускоряет процесс разработки. Библиотека React была создана инженером-программистом Джорданом Уолком. В 2015 году компания Netflix объявила об использовании React, в том же году появилась React Native библиотека, которая предназначена для создания мобильных приложений. В 2015 и 2016 годах появились такие инструменты, как React Router, Redux, и Mobx. Одним из переломных изменений был выпущен React на движке Fiber в 2017 году. Fiber изменяет и ускоряет процесс рендеринга. В 2019 год выходят Hooks, новый способ использования логики с отслеживанием состояния между компонентами.

Fiber - фундаментальное изменение, появившееся в 16 версии. В данном разделе мы и обсудим как он работает, зачем он, и какие дает преимущества. Fiber сфокусирован на анимации и отзывчивости приложения. Что Fiber умеет делать: разделять работу в “чанках” и раздавать приоритет для задач, может

остановить работу и вернуться к ней позже, умеет переиспользовать уже завершенные работы или останавливать ее, если она больше не требуются. Это совершенно другой подход к React от 16 версии. Старая версия React была синхронной и работала как stack (стопка - представьте себе фишки в казино, сложенные стопку), вы можете добавить что-то в stack, убрать из него что то, но ничего не будет работать, пока он не опустеет, работа даже не может быть прервана. Представим себе такую ситуацию, мы вводим что-то в текстовую строку в нашем браузере, и вдруг начинается загрузка каких-то файлов нашим браузером, пока загрузка не закончится в нашем поле, не появится ни одного знака, но как только загрузка прекратится, там отобразятся все знаки, которые мы вводили. Суть, лежащая в основе идеи, заключается в том, что Fiber также представляет собой единицу работы, после выполнения этой работы, она фиксирует, что приводит к видимому изменению DOM (Document Object Model). Все это проходит в два этапа. Первый этап это рендеринг, во время этого процесса react выполняет все асинхронные процессы, во время этого процесса выполняются такие функции как beginWork() и completeWork(). Вторая фаза называется фиксацией функции commitWork(), эта фаза в отличии от фазы рендеринг - синхронная и не может быть прервана. Давайте рассмотрим какие есть свойства у Fiber: всегда отношение один к одному, например к компоненту React или узлу DOM, всего от 0 до 24 типов к чему Fiber может быть привязан, эту нумерацию можно найти в свойстве tag. Другими словами, можно представить веревку, где на одном конце Fiber, а на другом что-то со свойством tag из которого мы можем понять, что это. Еще можно упомянуть о свойстве stateNode с помощью которого можно получить доступ к состоянию. Это очень похоже с архитектурой React элемента, отчасти это правда, так как очень часто основой является React элемент, они даже имеют одни и те же свойства, такие как type и key. Но React элемент каждый раз создается новый, Fiber же максимально переиспользуется. У React и Fiber схожие структуры в виде деревьев. С деревом в React все просто и понятно. Дерево Fiber отличается от

того, что мы себе обычно представляем, у Fiber есть три типа связей child, sibling, return. Представим себе div с вложенными в него трех элементов h1, h2 и h3. Вот div с первым элементом (h1) будет иметь связь child, но div не будет иметь никаких связей с h2, первый элемент после div h1 будет иметь связь с h2 и она будет называться sibling, также h2 и h3 связь sibling и у всех этих трех элементов h1 h2 и h3 в return будет храниться ссылка на родительский div.

Ранее упоминалось, что fiber – это одна единица работы, но что же такое эта работа, изменение состояния, функция жизненного цикла, изменение DOM — это все является работой. Работа может быть разделена на куски или запланирована, к примеру работа с высоким приоритетом может планироваться на будущее, а ресурс затратная разделяться на несколько кусков. Низкий приоритет выполняется в самую последнюю очередь.

Вернемся к Fiber дереву, на самом деле, там два дерева, одно называется current и второй workInProgress. Current дерево отображается на экране пользователя. При выполнении, работа реакт изменяет workInProgress дерево и после как он все сделал просто переключается на отображение workInProgress и изменяет ему статус на current дерево.

В 16.8 версии React появляется Hooks. Hooks позволяют использовать функциональность и особенности React без написания классовых компонентов. С помощью hooks мы можем управлять жизненными циклами компонентов с помощью useEffect, состоянием для этого используем useState и многим другим, к примеру useRef или useContext или даже можем писать собственные хуки. Но есть ряд ограничений, где хуки не должны применяться. Например нельзя использовать хук useState, который предназначен для хранения состояния в циклах и (или) в условиях, а все потому что state – это просто массив и хук useState обращается к этому массиву по индексу, по этой причине эти индексы вызывая в цикле или в условии, могут быть разными при каждом рендеринге страницы, и это приведет к тому, что каждый хук будет использовать некорректное состояние. Второй, наиболее популярный хук – это useEffect,

который принимает в себя функцию и массив зависимостей, если массив зависимостей будет пустой, то useEffect выполнится только единожды при инициализации, если же нет, то он будет отрабатывать каждый раз, когда любой из элементов в массиве будет изменяться, но если массива не будет вообще, то хук будет выполнять свои функции каждый раз когда вызывается компонент в котором используется хук. Хуки должны соответствовать двум очень важным правилам, первое – это композиция, это означает, что хуки не конфликтуют друг с другом, второе правило – это отладка(debugging), область с ошибкой должна легко определяться. Также можно упомянуть о двух весьма популярных хуках: это useMemo и useCallback. useCallback возвращает мемоизированную версию колбека и по схожести работы с useEffect версия изменится только при изменении любого элемента в массиве зависимостей. Примерно по такому же принципу работает useMemo, только useMemo вернёт результат работы этой функции и пересчитает его при изменении зависимостей, предназначеных для оптимизации тяжелых для ресурсов компьютера вычислительных действий, если не добавлять массив зависимостей функция будет выполняться при каждом рендеринге.

Немного новинок в 18 версии React, возможно они будут для кого-то полезны и ожидаемы. Из хуков нам добавят возможность вручную оптимизировать работу нашего приложения, раздавать приоритет для рендренга. И дополнительная оптимизация при внедрении стилей CSS во время рендеринга, но это скорее будет полезно для создания библиотек. Будут предоставленны новые API и какие то части будут выводиться из эксплуатации и будут оповещать разработчиков, что они пользуются “диприкейтед” функциями.

Как вы уже поняли из выше перечисленного, хуки – это достаточно простой и понятный инструмент, который представляет из себя функцию. И еще один важный момент, хуки работают только в функциональных компонентах, с помощью их можно написать почти безклассовый продукт, за одним

исключением и это исключение – ErrorBoundry, он все еще обязует нас работать с классовым компонентом React.

Литература

1. Мардан Азат, React быстро. Веб-приложения на React, JSX, Redux и GraphQL (2019)
2. Алекс Банкс и Ив Порчелло, Learning React: Modern Patterns for Developing React Apps (2020)
3. Ари Лернер, Нейт Мюррей, и Энтони Аккомаццо, Fullstack React: The Complete Guide to ReactJS and Friends (2017)
4. Frank W. Zammetti, Modern Full-Stack Development (2020)
5. Адам Хортон и Райан Вайс, Разработка веб-приложений в ReactJS (2022)

Literature

1. Mardan Azat, React Fast. Web Applications in React, JSX, Redux and GraphQL (2019)
2. Alex Banks and Yves Porcello, Learning React: Modern Patterns for Developing React Apps (2020)
3. Ari Lerner, Nate Murray, and Anthony Accomazzo, Fullstack React: The Complete Guide to ReactJS and Friends (2017)
4. Frank W. Zammetti, Modern Full-Stack Development
5. Adam Horton and Ryan Weiss, Developing Web Applications in ReactJS (2022)

Для цитирования: *Яровая Е.В.* АРХИТЕКТУРНЫЕ РЕШЕНИЯ В БИБЛИОТЕКИ РЕАКТ// Научный сетевой журнал «Столыпинский вестник» №5/2022