УДК 004.05

М.А. Толстых
магистрант, УО БГУИР,
г. Минск, Беларусь

ПОСТРОЕНИЕ ДИЗАЙН СИСТЕМЫ КОМПОНЕНТОВ REACT

Аннотация
Данная статья посвящена построению дизайна React компонентов.

Ключевые слова
Дизайн программного обеспечения, frontend, React, программирование.

Дизайн системы – это высокоэффективный способ обеспечить стандартизированный, согласованный внешний вид и ощущение на всех экранах и устройствах. Это, в свою очередь, помогает уменьшить путаницу пользователей, повысить удовлетворенность и помочь вашему бренду создать лояльную аудиторию.

Для большинства ведущих компаний системы проектирования реализованы в виде многоразового набора компонентов в React, Vue, Angular. Это имеет смысл, поскольку позволяет разработчикам совместно и повторно использовать компоненты для создания различных приложений, что повышает скорость разработки, а также согласованность пользовательского интерфейса.

Есть несколько способов сделать это. Одним из них является сервис Bit [1] для обмена готовыми компонентами.

Он оптимизирует разработку компонентов, повторное использование, обновление и документацию, чтобы компоненты системы проектирования могли использоваться для создания приложений.

Инструмент CLI Bit [2] помогает разрабатывать, изолировать, создавать версии и публиковать компоненты в проекте. Он также управляет изменениями компонентов (с автоматизированным управлением зависимостями) в различных приложениях.

Рассмотрим, как этот современный инструмент может помочь решить несколько основных проблем при построении вашей системы компонентов React:

- разработка и публикация компонентов;
- документирование и переиспользование компонентов;
- управление состоянием и обновления компонентов.

Много написано о построении многоразовых компонентов React. Кроме того, существуют отличные рекомендации по созданию повторно используемых компонентов в React.

Компоненты могут быть связаны своим контекстом различными способами: зависимостями, построением конфигурационных элементов, стилями, состояниями и т.д.

Наиболее важной частью многоразовых компонентов является изоляция.

Основная функциональность Bit - изолировать компоненты от проекта.

Для этого он помещает каждый компонент в "Контейнер" под названием "Капсула". Каждый контейнер автоматически включает все файлы компонентов, зависимости и конфигурации. Bit оптимизирует этот процесс для вас почти полностью.

Он автоматически определяет файл и зависимости каждого компонента. И, это помогает вам узнать о других видах связи, чтобы вы могли улучшить.

Рассмотрим пример работы. Bit отсканировал папку и идентифицировал компоненты. Он проанализировал код каждого компонента, чтобы решить, есть ли у него какие - либо дополнительные зависимости, такие как внешние файлы пакетов. Затем каждый компонент помещают в капсулу.

В любое время команда bit status позволяет просмотреть состояние всех компонентов в проекте. Это поможет узнать, успешно ли все компоненты изолированы, обновлены ли версии и проходят ли они сборку и тестирование изолированно. Это очень полезно для разработки многоразовых компонентов так как помогает узнать о связи между каждым компонентом и проектом.

Поскольку Bit.dev является центром повторного использования компонентов, он также предоставляет домашнюю документацию по компонентам с автоматически извлеченными ссылками на API (react doc gen) [3], визуальными примерами, результатами тестирования и т.д.

При работе со многими компонентами очень важно иметь возможность найти нужные компоненты.

Для этого Bit.dev предоставляет функцию поиска, построенную для компонентов. Он позволяет выполнять поиск по меткам компонентов, контексту и даже размеру бандла.

Из bit.dev каждый компонент становится автономным для использования одним из двух способов:

- установите его с помощью npm или yarn непосредственно из реестра bit.dev;
- используйте bit import для источника компонента в другом проекте.

Система проектирования — это больше, чем библиотека. Это больше, чем цвета ваших компонентов. Это постоянно растущий и постоянно развивающийся источник базовых компонентов, из которых состоит весь опыт работы с продуктом.

Традиционные библиотеки заставляют выпускать новую версию всей библиотеки при каждом обновлении компонента. Это вредит пользователям библиотеки.

С помощью облачной библиотеки можно просто обновить один компонент. Кроме того, можно убедиться, что этот компонент обновляется в каждом приложении.

Для обновления компонента в Bit.dev необходимо опубликовать новую версию. Это означает использование bit tag для переноса версии, использование bit status для просмотра зависимых компонентов, которые также требуют переноса, и публикацию обновления [4].

Теперь доступно обновление только для этого компонента. Пользователям, не использующим компонент, не придется без причины обновлять всю библиотеку.

Если доступна новая версия компонента, пользователи проектов могут импортировать ее в свои проекты с помощью bit import и обновить компонент.

Bit дает вам возможность создавать, выпускать, организовывать и повторно использовать компоненты практически в любом масштабе. Интегрированный с GitHub, он обеспечивает контроль и управление процессом внедрения и обновления компонентов в приложениях.

Список использованной литературы:

1. Component Sharing. [Электронный ресурс]. URL: https: // www.bit.dev (Дата обращения 05.05.2020).

2. CLI Docs. [Электронный ресурс]. URL: https: // docs.bit.dev / docs / apis / cli - all (Дата обращения 06.05.2020).

3. Генерация документации ля React. [Электронный ресурс]. URL: https: // github.com / reactjs / react - docgen (Дата обращения 06.05.2020).

4. Версирование Bit [Электронный ресурс]. URL: https: // docs.bit.dev / docs / tag - component - version (Дата обращения 07.05.2020).