React использует JSX, это позволяет объединять HTML и JavaScript в одном файле. Компоненты независимы и могут повторно использованы. Virtual DOM делает приложение производительным при обновлении. Состояние позволяет хранить данные в состоянии компонента и передавать их через свойства другим компонентам. Клиент-серверная архитектура позволяет делать клиентскую часть, взаимодействуя с сервером посредством API.

Алгоритмы и паттерны помогают писать эффективный и масштабируемый код. Пробовала использовать сортировку пузырьком, выбором и сортировку слиянием. Из Паттернов использовала Мемоизацию для предотвращения лишних вычислений и ускорения рендера компонента. Наблюдатель пригодился для синхронного отображения товара в корзине и на странице каталога… Но классовый подход я стараюсь не использовать, теперь есть хуки.

При работе с тестированием React компонентов, хранилищ, экшенов, редьюсеров, асинхронных действий и серверных API методов важно использовать соответствующие методы тестирования для каждого уровня абстракции. Ниже приведены основные подходы и инструменты для тестирования различных частей React приложения:

1. Тестирование React компонентов:

- Для тестирования React компонентов часто используется библиотека Jest в сочетании с Enzyme. Jest предоставляет возможности для запуска тестов, а Enzyme облегчает работу с компонентами React, позволяя симулировать события и проверять результаты рендеринга.

- Для тестирования компонентов рекомендуется писать unit-тесты для отдельных компонентов, снимки (snapshots) компонентов и тесты на функциональность.

2. Тестирование хранилищ, экшенов и редьюсеров:

- Для тестирования хранилищ Redux и связанных с ними экшенов и редьюсеров также широко используется Jest. Для тестирования Redux рекомендуется использовать библиотеку Redux Toolkit и официальный плагин Redux DevTools Extension для Chrome.

- Для тестирования экшенов и редьюсеров предпочтительно писать unit-тесты, проверяющие их функциональность и корректность обработки данных.

3. Тестирование асинхронных действий:

- Для тестирования асинхронных действий, таких как запросы к серверу, можно использовать Jest в сочетании с библиотекой Axios или другими инструментами для выполнения HTTP запросов.

- Для тестирования асинхронных действий рекомендуется использовать мокирование (mocking) для имитации запросов и проверки результатов обработки данных.

4. Тестирование серверных API методов:

- Для тестирования серверных API методов можно использовать инструменты, такие как Postman, Swagger или другие инструменты для тестирования API.

- Для автоматизации тестирования серверных API методов можно также использовать инструменты для написания и запуска автотестов, такие как Jest, Mocha, Chai и др.

При работе с тестированием React приложений рекомендуется писать как unit-тесты для отдельных частей приложения, так и интеграционные тесты для проверки взаимодействия между компонентами, хранилищем, асинхронными действиями и серверными API методами. Непрерывное тестирование поможет обнаруживать ошибки на ранних этапах разработки и обеспечит стабильную работу приложения.

Для каждого уровня абстракции используются свои методы тестирования: компоненты библиотекой Jest, хранилища, экшены и редьюсеры Redux Toolkit, Postman и Swagger для API.