1. Можно ли создать несколько контекстов в одном приложении?

Ответ:

Да, в React-приложении вы можете создать несколько контекстов и использовать их одновременно. Контекст предоставляет способ передачи данных через дерево компонентов без явной передачи пропсов через каждый промежуточный компонент.

Каждый контекст в React представлен отдельным экземпляром createContext().

2. Можем ли мы влиять на контекст из вложенных компонентов?

Ответ:

Да, вы можете влиять на контекст из вложенных компонентов, используя Provider для обновления значения контекста и Consumer для получения доступа к этому значению внутри компонентов.

3. Как выбрать компонент, в котором будет определяться контекст?

Ответ:

Выбор компонента, в котором будет определяться контекст, зависит от вашей конкретной структуры приложения и потребностей. В общем случае, компонент, определяющий контекст (также называемый "поставщиком" контекста), должен быть родительским по отношению к компонентам, которым требуется доступ к этому контексту.

4. Возможно ли создать контекст, который доступен только в части приложения?

Ответ:

Да, вы можете создать контекст, который доступен только в определенной части приложения. Для этого вам нужно создать и определить контекст внутри нужного компонента или компонентов.

При создании контекста с помощью createContext в React, вы можете передать начальное значение контекста, которое будет использоваться по умолчанию. Это начальное значение будет доступно только в том контексте, который был создан в определенном компоненте, и его изменения не будут отражаться в других частях приложения.

5. На каком этапе жизненного цикла компонента лучше всего запросить данные с сервера?

Ответ:

Метод `componentDidMount` считается самым удачным местом для первого запроса данных с сервера

6. Вернет ли сообщение об ошибке Fetch API, если код ответа 401 (ошибка авторизации)?

Ответ:

Да, при использовании Fetch API, если сервер возвращает код ответа 401 (Unauthorized) или любой другой код ошибки, обещание (Promise) возвращаемое методом fetch будет перейти в состояние отклонено (rejected) и будет сгенерировано исключение (error). Вы сможете обработать это исключение с помощью метода .catch().

7. Каким еще способом можно делать запросы к API кроме fetch?

Ответ:

- Библиотека Axios: Axios предоставляет удобный интерфейс для выполнения HTTP-запросов и широко используется в React-приложениях. Она поддерживает множество функций, таких как установка заголовков, обработка ошибок и междоменные запросы. Чтобы использовать

Axios, вам нужно установить его с помощью npm или yarn, а затем импортировать его в вашем компоненте и использовать для выполнения запросов.

- Библиотека jQuery AJAX: Если вы уже используете jQuery в своем проекте React, вы можете воспользоваться его функциональностью AJAX для выполнения запросов к API. jQuery AJAX предоставляет методы для отправки запросов GET, POST, PUT, DELETE и других
- Библиотека Fetch API Polyfill: Если вам нужна поддержка старых браузеров, которые не поддерживают Fetch API, вы можете использовать полифилл, такой как whatwg-fetch. Это полифилл, который эмулирует функциональность Fetch API в старых браузерах, позволяя вам использовать fetch в своем коде.

8. Зачем нужен CORS?

Ответ:

CORS (Cross-Origin Resource Sharing) - это механизм, который позволяет веб-приложениям делать запросы к ресурсам (например, API) на других доменах. Без него браузеры применяют политику Same-Origin, запрещающую такие запросы по умолчанию. CORS позволяет серверу указать, какие домены могут получать доступ к его ресурсам, обеспечивая безопасность и контроль доступа