1. **Что такое React простыми словами?**

React. js — это JavaScript-библиотека от «Фейсбука»\* для удобной разработки интерфейсов, то есть внешней части сайтов и приложений, с которой взаимодействует пользователь.

**2) назовите основные особенности реакт**

К её основным особенностям относится декларативность, компонентный подход, универсальность, использование виртуального DOM и JSX. На Хекслете курс по React входит в профессию «Фронтенд JavaScript». К её основным особенностям относится декларативность, компонентный подход, универсальность, использование виртуального DOM и JSX. На Хекслете курс по React входит в профессию «Фронтенд JavaScript».

React Native позволяет использовать уже имеющуюся логику веб-приложения при создании мобильного приложения. Это значит, что команда разработчиков может использовать тот же код, который был использован в процессе создания сайта вместо того, чтобы начинать с чистого листа.

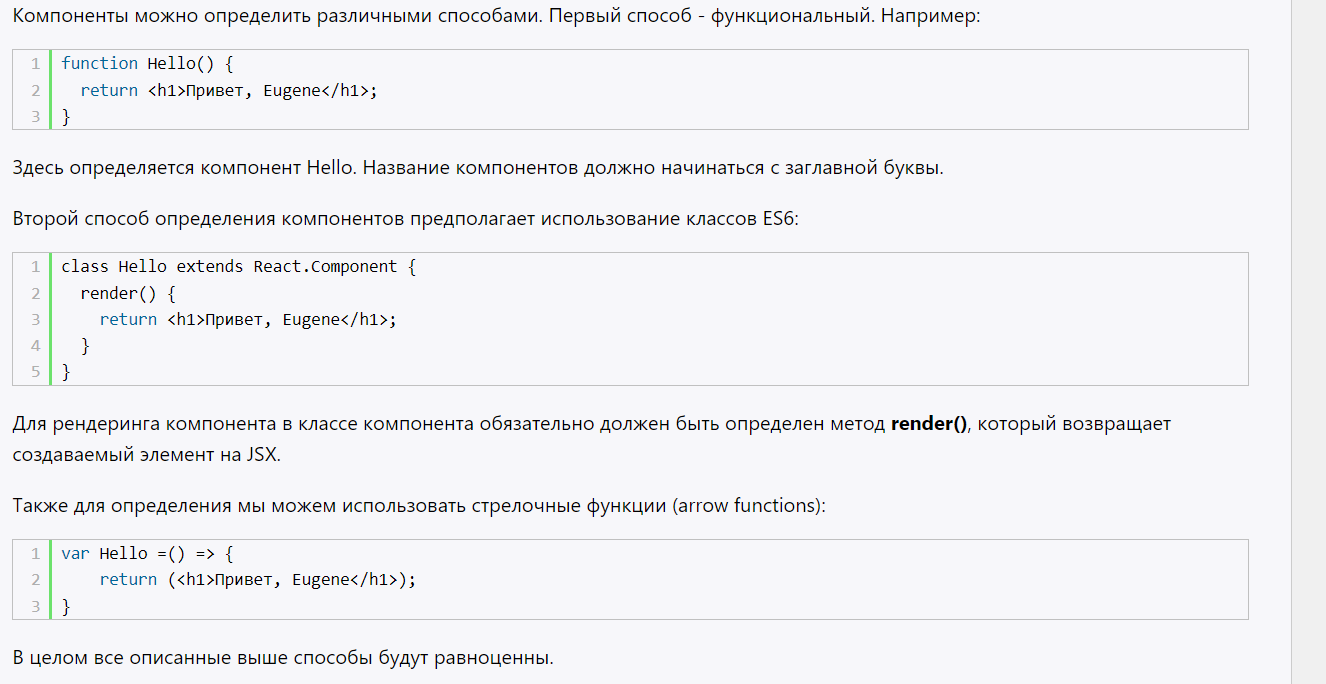
**3) что такое JSX**

JSX — расширение синтаксиса JavaScript. Этот синтаксис выглядит как язык шаблонов, но наделён всеми языковыми возможностями JavaScript. В результате компиляции JSX и вызова React. createElement() возникают простые объекты — «React-элементы».

**4) В чем разница между элементом и компонентом?**

JSX — расширение синтаксиса JavaScript. Этот синтаксис выглядит как язык шаблонов, но наделён всеми языковыми возможностями JavaScript. В результате компиляции JSX и вызова React. createElement() возникают простые объекты — «React-элементы».

**5) как в реакте создаются компоненты**



**6) Когда лучше использовать классовый компонент, а когда**

**функциональный?**

React до версии 16.8:

Классовый компонент - когда необходимо работать с состоянием и использовать методы жизненного цикла компонента;

Функциональный компонент - когда компонент не использует состояние и нужно только "отрендерить" в зависимости от поступивших props;

React с версии 16.8:

Классовый компонент - если не используете хуки;

Функциональный компонент - во всех остальных случаях;

**7) Что такое "чистые" компоненты (Pure Components)?**

Чистые компоненты (pure components) - это компоненты, которые не рендерятся повторно при обновлении их состояния или пропов одними и теми же значениями. Если значение предыдущего и нового состояния и пропов равны, компонент не повторно отрисовывается. Чистые компоненты ограничивают повторный рендеринг, обеспечивая повышение производительности приложения.

Особенности чистых компонентов

Предотвращают повторный рендеринг компонента, если его состояние и пропы остались прежними.

Неявно реализуют метод shouldComponentUpdate.

state и props сравниваются поверхностно (shallow).

В ряде случаев такие компоненты являются более производительными, чем обычные.

**8) Что такое состояние (state) в React?**

Объект state описывает внутреннее состояние компонента, он похож на props за тем исключением, что состояние определяется внутри компонента и доступно только из компонента.

Если props представляет входные данные, которые передаются в компонент извне, то состояние хранит такие объекты, которые создаются в компоненте и полностью зависят от компонента.

Также в отличие от props значения в state можно изменять.

И еще важный момент - значения из state должны использоваться при рендеринге. Если какой-то объект не используется в рендерниге компонента, то нет смысла сохранять его в state.

Нередко state описывает какие-то визуальные свойства элемента, которые могут изменяться при взаимодействие с пользователем. Например, кнопку нажали, и соответственно можно изменить ее состояние - придать ей какой-то другой цвет, тень и так далее. Кнопку нажали повторно - можно вернуть исходное состояние.

Стоит отметить, что традиционно объект state применялся только в классах-компонентах. В функциональных же компонентах для управления состоянием применяется другая архитектура, основанная на хуках.

**9 Что такое пропы (props) в React?**

Пропы (props) — это свойства, которые передаются в компонент. Пропы были созданы, чтобы обеспечить передачу данных от родительского компонента к дочернему. Пропы всегда неизменны в пределах компонента, получившего их. Если значения пропов изменились (поступили новые пропы из родительского компонента), то происходит перерисовка компонента.

**10 В чем разница между состоянием и пропами?**

В чем разница между пропами и состоянием в Реакте?

Сходства пропов и состояния

Пропы и состояние — это объекты, которые используются внутри компонента.

При изменении пропов или состояния происходит перерисовка (рендеринг) компонента.

Различия пропов и состояния

Пропы поступают в компонент как входные параметры — это механизм передачи данных от родительского компонента к дочернему, а состояние доступно только внутри своего компонента, другие компоненты не имеют к нему доступа.

Пропы неизменны, тогда как состояние внутри компонента может меняться.

**11 Почему не следует обновлять состояние напрямую?**

Почему состояние в компоненте нельзя менять напрямую в Реакте?

Состояние компонента является простым объектом Javascript, поэтому у многих программистов, только начавших изучать Реакт, возникает желание менять его также, как и у объекта в Javascript — просто задать новые значения свойств:

this.state.isEnabled = false;

Однако такой способ не является правильным, так как при этом не отслеживается, что состояние было изменено, и нужно выполнить перерисовку компонента.

Метод setState для обновления состояния

Для изменения состояния в компоненте нужно использовать метод setState(). Он позволяет запланировать обновление состояния. После выполнения метода происходит перерисовка компонента.

Метод принимает на вход первым параметром новое состояние компонента в виде объекта либо функцию, возвращающую этот объект. При этом достаточно указать в методе только изменившуюся часть состояния, а не все состояние компонента целиком. Остальные данные будут взяты из прежнего состояния.

Пример использования setState():

this.setState({

isEnabled: false

});

Вторым параметром можно задать callback-функцию, которая будет выполнена после завершения установки нового состояния и перерисовки компонента, однако вместо callback-функции рекомендуется использовать метод жизненного цикла компонента componentDidUpdate.

Пример использования setState() со 2 параметром:

this.setState({

isEnabled: false

}, () => {

console.log('Элемент больше не доступен');

});

Важно понимать, что setState() не гарантирует мгновенного изменения состояния, несколько вызовов могут быть объединены для улучшения производительности. Поэтому в качестве первого параметра следует использовать функцию, если новое состояние зависит от старого или от пропов. В противном случае данные в состоянии могут оказаться не теми, какие ожидались.

**12 Для чего в setState() используются функции обратного вызова?**

Функция обратного вызова — это функция, которая передается в качестве аргумента другой функции и выполняется позже. Обратные вызовы используются для обработки асинхронного кода, например ожидания извлечения данных с сервера или обработки событий пользовательского ввода.

В React обратные вызовы широко используются для обработки взаимодействия с пользователем и обновления состояния приложения. Например, когда пользователь нажимает кнопку, запускается функция обратного вызова для обработки события клика и соответствующего обновления состояния приложения. import React from 'react';

class MyComponent extends React.Component {

handleClick() {

console.log('Button clicked');

}

render() {

return (

<button onClick={this.handleClick}>Click me</button>

);

}

}

**13 В чем разница между обработкой событий в HTML и React?**

В HTML имя события должно быть в нижнем регистре :

< button onclick = ' activLasers () ' >

Принимая во внимание, что в React следует соглашение CamelCase :

< button onClick = { activLasers } >

В HTML вы можете вернуться, falseчтобы предотвратить поведение по умолчанию:

< a href = ' # ' onclick = 'console.log ("Ссылка была нажата.");

return false

; ' />

Принимая во внимание, что в React вы должны вызывать

preventDefault()явно:

function handleClick(event) {

event.preventDefault()

console.log('The link was clicked.')

}

В HTML вам нужно вызывать функцию, добавляя () В то время как в реакции вы не должны добавлять ()с именем функции. (см., например, функцию «activLasers» в первом пункте

**14 Как в коллбеках JSX привязываются методы или обработчики событий?**

Для привязки методов или обработчиков событий в React используется концепция "связывания" (binding) в конструкторе класса. Это нужно для того, чтобы контекст this имел правильное значение внутри методов и обработчиков.

Существует несколько способов связывания функций в React-компонентах:

С помощью метода bind()

this.handleClick = this.handleClick.bind(this);

Используя стрелочную функцию

handleClick = () => {

// Обработка клика

};

Создавая обработчик события внутри метода render():

handleClick() {

// Обработка клика

}

render() {

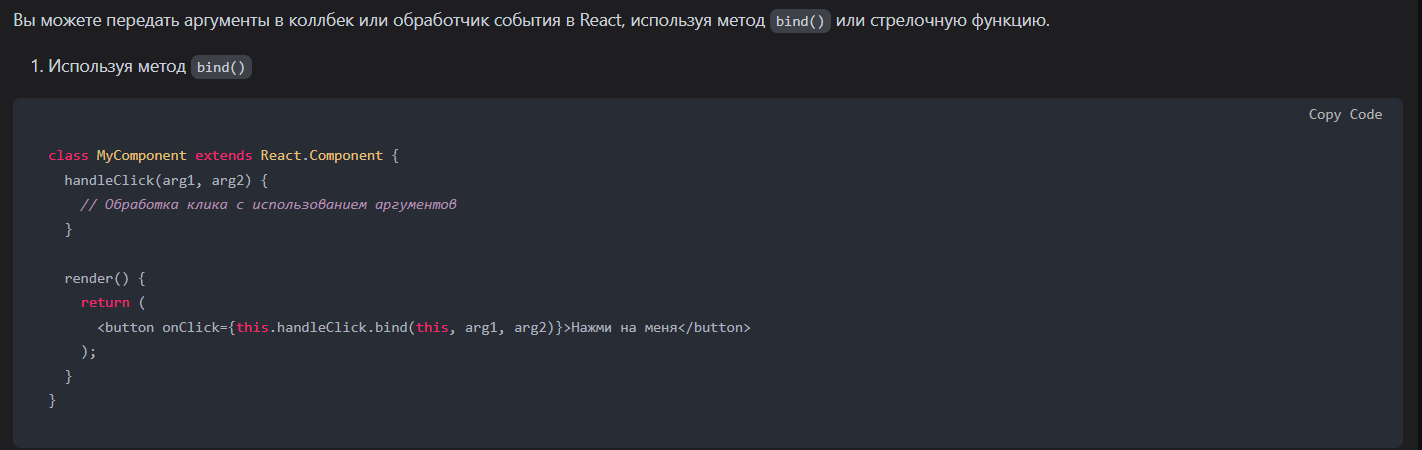
return (

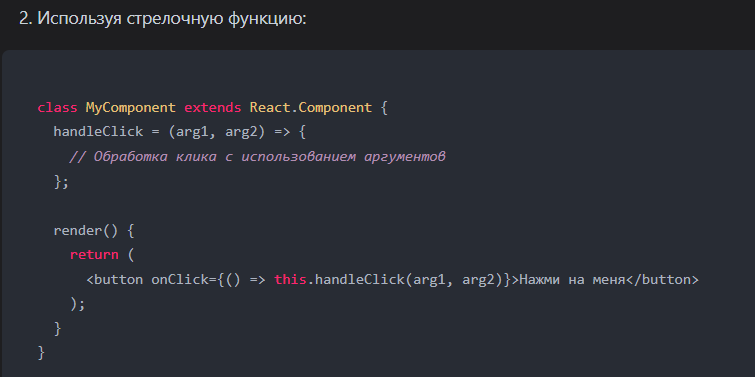
<button onClick={() => this.handleClick()}>Нажми на меня</button>

);

}

**15 Как передать аргумент в "коллбек" или обработчик событий?**



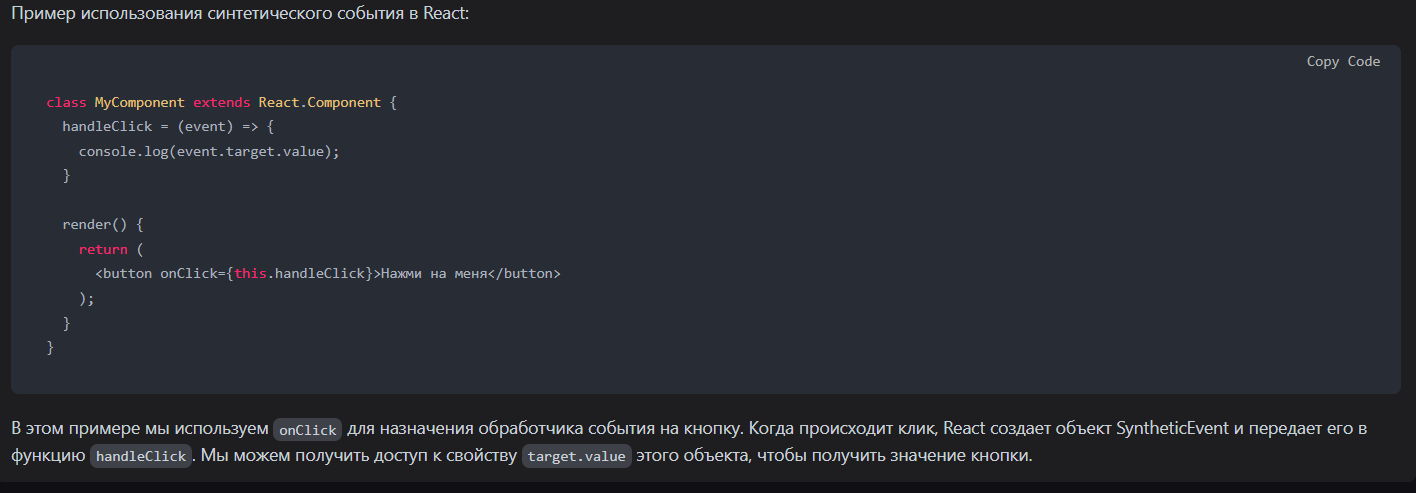


**16 Что такое синтетические события в React?**

Синтетические события в React - это обертка вокруг нативных браузерных событий, которые предоставляют дополнительные возможности и упрощают работу с событиями в React.

React использует систему обработки событий, которая создает объекты SyntheticEvent и передает их в качестве аргументов в обработчики событий. Эти объекты являются кросс-браузерными, что позволяет писать переносимый код без необходимости проверять поддержку каждого браузера отдельно.

Объект SyntheticEvent имитирует стандартное событие DOM, такое как клик или наведение курсора, и содержит дополнительную информацию о событии, такую как координаты мыши, клавиши-модификаторы и прочее.

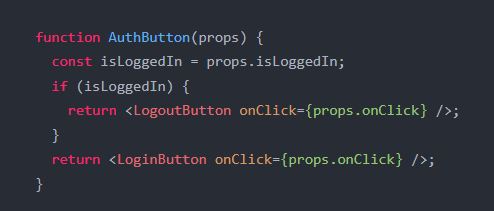


**17 Что такое условный рендеринг?**

Условный рендеринг в React - это процесс рендеринга компонента на основе определенного условия. Когда мы используем условный рендеринг, мы контролируем, когда и как компонент должен отображаться на странице.

Для выполнения условного рендеринга в React, мы можем использовать конструкцию if-else, тернарный оператор (? :), логический оператор && и другие подходы.

Например, если мы хотим отобразить кнопку "Войти" для неавторизованных пользователей и кнопку "Выйти" для авторизованных пользователей, мы можем использовать условный рендеринг:



**18 Что такое проп key и в чем заключаются преимущества его использования?**

key - это специальный атрибут, который используется в React для определения уникальности каждого элемента списка при рендеринге. Когда вы отображаете список элементов в React, каждый элемент должен иметь уникальный ключ key. React использует значения ключей для эффективной перерисовки только тех элементов, которые изменились, вместо полного перерисования всего списка.

При использовании key есть несколько преимуществ:

Улучшенная производительность: при изменении списка React может перерисовывать только измененные элементы, что позволяет добиться лучшей производительности, чем при полной перерисовке всего списка.

Поддержка состояния: key позволяют сохранять состояние компонентов при их перерисовке. Если компоненты имеют уникальный key, React будет использовать существующий экземпляр компонента, если он еще не был размонтирован, что позволяет сохранить его состояние.

Поддержка анимации: key могут быть использованы для создания плавных анимаций при добавлении или удалении элементов из списка.

**19+20 Для чего используются ссылки (рефы, refs)?**

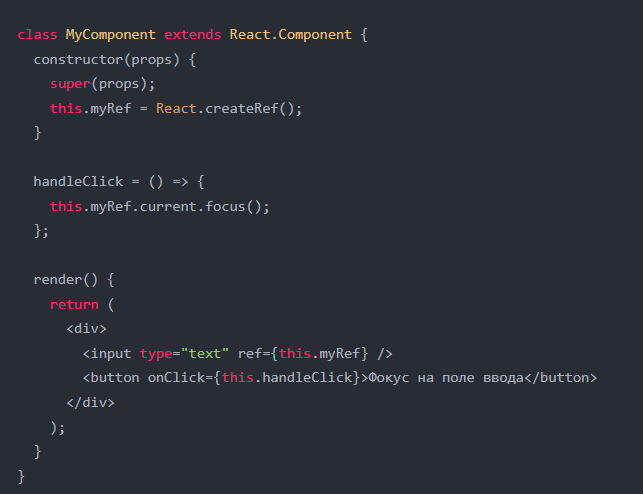
Ссылки (Refs) в React используются для получения доступа к DOM-элементам и компонентам, которые были созданы внутри метода render(). Ссылки позволяют работать с элементами на более низком уровне, обойдя общий поток данных между компонентами.

Существует два способа использования ссылок в React:

Создание ссылок при помощи колбеков:



Создание ссылок при помощи свойства createRef():



**21 Что такое передача ссылки (Forward Ref)?**

Передача ссылки (Forward Ref) - это механизм в React, который позволяет передавать ссылку на вложенный компонент через вышестоящий компонент, чтобы родительский компонент мог иметь доступ к DOM-элементам или экземпляру класса вложенного компонента.

В случаях, когда компонент не имеет своего собственного DOM-элемента, таких как HOC-компоненты, передача ссылок может быть одним из способов предоставления доступа к экземпляру класса или методам дочернего компонента.

Чтобы создать ссылку и передать ее через компонент, можно использовать функцию React.forwardRef(). Эта функция принимает функциональный компонент и возвращает новый компонент, который принимает ref как аргумент и передает его через пропсы в дочерний компонент.

**22 Что лучше использовать, "коллбек-рефы" или findDOMNode()?**

В React рекомендуется избегать использования метода findDOMNode() и вместо этого использовать коллбек-рефы (callback refs).

Коллбек-рефы позволяют получить ссылку на DOM-элемент после того, как он был создан и отрисован компонентом. Таким образом, вы можете получить доступ к DOM-элементу без необходимости использования метода findDOMNode(), который может вызвать проблемы с производительностью и избыточными перерисовками.

**23 Что такое виртуальный DOM?**

Виртуальный DOM (VDOM) - это концепция веб-разработки, которая представляет собой абстракцию реального DOM. Виртуальный DOM является легковесным и быстрым представлением DOM и используется для оптимизации производительности при обновлении содержимого веб-страницы.

В React, например, когда компонент обновляется, React сначала создает новое дерево элементов виртуального DOM на основе измененных данных состояния компонента. Затем React сравнивает новый виртуальный DOM с предыдущим виртуальным DOM и выявляет различия между ними. После этого React выполняет минимальное количество операций для обновления только тех частей реального DOM, которые изменились в соответствии с изменениями в виртуальном DOM.

Таким образом, использование виртуального DOM позволяет уменьшить количество операций на реальном DOM и, следовательно, улучшить производительность приложения. Кроме того, виртуальный DOM помогает разработчикам отделить логику работы со структурой веб-страницы от манипуляций над реальным DOM, что облегчает поддержку и разработку приложения.

**24 Как работает виртуальный DOM?**

Виртуальный DOM (VDOM) работает следующим образом:

Когда компонент React обновляется, React создает новое дерево элементов виртуального DOM на основе измененных данных состояния компонента.

Затем React сравнивает новый виртуальный DOM с предыдущим виртуальным DOM и выявляет различия между ними.

После этого React выполняет минимальное количество операций для обновления только тех частей реального DOM, которые изменились в соответствии с изменениями в виртуальном DOM.

Наконец, React обновляет реальный DOM только в тех местах, где были произведены изменения.

**25 В чем разница между теневым (Shadow) и виртуальным DOM?**

Таким образом, основная разница между теневым и виртуальным DOM заключается в том, что теневой DOM используется для создания изолированных компонентов на странице, а виртуальный DOM - для оптимизации работы с реальным DOM. Кроме того, теневой DOM является частью стандарта Web Components, в то время как виртуальный DOM используется современными библиотеками JavaScript, такими как React, Vue.js, Angular и другие.

**26-27 Что такое React Fiber?**

React Fiber - это переработанная версия алгоритма согласования (reconciliation algorithm) в React, который был представлен в React 16.0. Разработчики React создали Fiber для улучшения производительности и поддержки анимаций в приложениях на React.

Алгоритм согласования - это процесс, который React использует для сравнения предыдущего и текущего состояния компонента и определения минимального количества изменений, необходимых для обновления интерфейса пользователя. Первоначальная реализация алгоритма согласования в React была основана на рекурсивном вызове, что могло приводить к блокировке браузера при работе с большими деревьями компонентов.

React Fiber решает эту проблему с помощью нового подхода к организации работы с алгоритмом согласования. Вместо рекурсивного вызова, React Fiber использует цикл с приоритетами (priority-based loop), позволяющий выполнять части алгоритма согласования поэтапно и прерывать их при необходимости. Такой подход позволяет React Fiber более эффективно использовать ресурсы системы и отрисовывать пользовательский интерфейс более плавно и быстро.

React Fiber также добавляет новые функциональные возможности, такие как отмена операций (откат изменений) и приостановка выполнения согласования (suspense), что упрощает работу с асинхронными операциями и оптимизирует производительность приложений.

Таким образом, React Fiber является важным шагом в развитии React и позволяет разработчикам создавать более эффективные и масштабируемые веб-приложения на базе этой библиотеки.

**28 Что такое управляемые компоненты?**

Управляемые компоненты - это компоненты пользовательского интерфейса, которые поддерживаются и управляются приложением. Они связаны с моделью данных, которая содержит все необходимые данные для отображения компонента. Когда данные в модели изменяются, управляемый компонент автоматически обновляется, что позволяет создавать динамические пользовательские интерфейсы, реагирующие на изменения данных. Примерами управляемых компонентов могут служить текстовые поля, кнопки, флажки и списки выбора. В ReactJS, например, большинство компонентов являются управляемыми компонентами.

**29 Что такое неуправляемые компоненты?**

В контексте React, термин "неуправляемые компоненты" обычно относится к компонентам, которые не хранят своё состояние внутри себя и не предоставляют пользователю способа управлять этим состоянием. Вместо этого, состояние компонента хранится в другом месте, например, в компоненте-родителе или в глобальном хранилище.

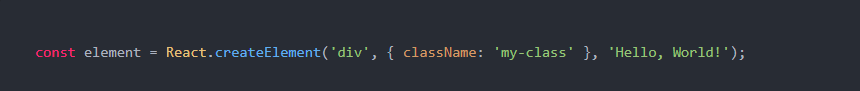
Такие компоненты могут быть полезны в некоторых случаях, например, когда нужно создать компонент для работы со вводом пользователя, который может использоваться в разных местах приложения и не требует передачи и управления состоянием извне. Например, такой компонент может принимать только обработчики событий в качестве свойств (props), а его значение можно получить только через эти обработчики.

Однако, следует учитывать, что такой подход может усложнить отладку и понимание кода, особенно если компонент используется в разных местах приложения. Поэтому, необходимо внимательно оценивать использование неуправляемых компонентов и выбирать подходящую стратегию управления состоянием в каждом конкретном случае.

**30 В чем разница между createElement() и cloneElement()?**

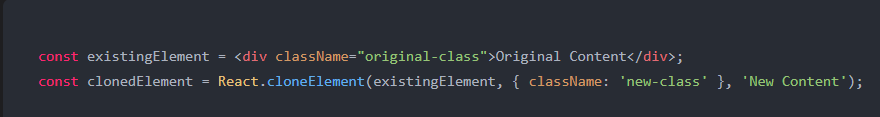
Метод createElement() используется для создания нового элемента в React, а метод cloneElement() используется для клонирования существующего элемента и применения к нему изменений.

createElement() создает новый элемент, заданный типом (например, div, span) и свойствами (props), переданными ему в аргументах. Например:



Этот код создаст новый элемент div с классом my-class и содержимым Hello, World!.

cloneElement(), с другой стороны, позволяет клонировать существующий элемент и добавлять или изменять его свойства. Например:



Этот код склонирует элемент existingElement и изменит его свойство className на new-class, а содержимое заменит на New Content. Обратите внимание, что первый аргумент - это клонируемый элемент, а остальные аргументы - это свойства, которые нужно изменить и/или новый контент для элемента.

**31 Что такое поднятие состояния в React?**

Поднятие состояния (lifting state up) в React означает перемещение состояния из компонента потомка в его родительский компонент. Это позволяет нескольким компонентам совместно использовать и изменять одно и то же состояние.

В React, данные передаются по иерархии компонентов сверху вниз - от родительского компонента к дочернему. Когда некоторые данные нужно разделить между несколькими компонентами на разных уровнях иерархии, возможно поднятие состояния для передачи этих данных от дочернего компонента к родительскому.

При поднятии состояния из дочернего компонента в родительский, родительский компонент получает новое свойство, которое он может передать обратно в дочерний компонент через функцию обратного вызова (callback). Эта функция может быть определена в родительском компоненте и передана в качестве свойства дочернему компоненту. Таким образом, дочерний компонент может изменять состояние, находящееся в родительском компоненте.

**32 Назовите стадии жизненного цикла компонента**

React-компоненты проходят через несколько стадий своего жизненного цикла:

Монтирование (Mounting) - компонент создается и вставляется в DOM дерево.

Обновление (Updating) - компонент обновляется, когда изменяются его свойства или состояние.

Размонтирование (Unmounting) - компонент удаляется из DOM дерева.

Получение ошибок (Error Handling) - если в компоненте происходит ошибка выполнения, React вызывает специальный метод, который позволяет ее обработать.

**33 Назовите методы жизненного цикла в React**

В React есть три метода жизненного цикла компонента, которые называются «методами-шаблонами»: render(), componentDidMount() и componentWillUnmount().

render() вызывается каждый раз, когда компонент обновляется. Он отвечает за рендеринг элементов на странице.

componentDidMount() вызывается после того, как компонент монтируется в DOM-дерево. Этот метод используется для выполнения действий, которые требуют доступа к DOM или другим ресурсам браузера.

componentWillUnmount() вызывается перед тем, как компонент будет удален из DOM-дерева. Он используется для очистки ресурсов, связанных с компонентом (например, отмены запросов на сервер).

Кроме того, в React есть еще несколько методов жизненного цикла, которые используются для изменения состояния компонента, например componentDidUpdate().

**34 Что такое компонент высшего порядка (Higher Order Component,**

HOC)?

Компонент высшего порядка (Higher Order Component, HOC) - это функция в React, которая принимает один компонент и возвращает другой компонент. Новый компонент, возвращаемый HOC, может обеспечивать дополнительную функциональность или поведение для переданного ему компонента.

HOC позволяют создавать логику, которую можно переиспользовать между различными компонентами. Они используются для абстрагирования и переноса необходимого состояния и методов из одного компонента в другой. При этом, они не изменяют свойства и методы переданного компонента напрямую.

Например, HOC может добавлять общие функции каких-то операций, таких как аутентификация, логирование или маршрутизация, к любому компоненту, переданному этому HOC.

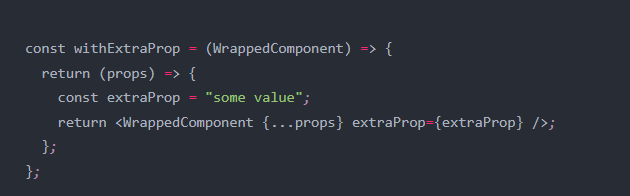
Пример использования HOC:



**35 Как в HOC-компоненте создаются прокси для пропов?**

HOC-компоненты в React используются для обертывания компонентов с целью добавления дополнительной функциональности. При этом, прокси для пропс можно создать несколькими способами.

Один из способов - использование оператора spread (...) для передачи всех пропсов из HOC-компонента внутрь обернутого компонента:



В этом примере, мы создаем HOC-компонент withExtraProp, который добавляет новый пропс extraProp к оборачиваемому компоненту. Мы используем оператор spread для передачи всех остальных пропсов из HOC-компонента внутрь обернутого компонента.

**36 Что такое контекст (Context)?**

Контекст (Context) - это механизм в React, который позволяет передавать данные глубоко вниз по иерархии компонентов без явной передачи пропсов через каждый промежуточный компонент.

Контекст может быть использован для передачи таких данных, как текущая тема оформления, авторизационные данные пользователя, локализация и другие данные, которые необходимы в разных частях приложения.

Для создания контекста используется метод createContext() из библиотеки React. Он возвращает объект, который содержит провайдер и потребитель контекста.

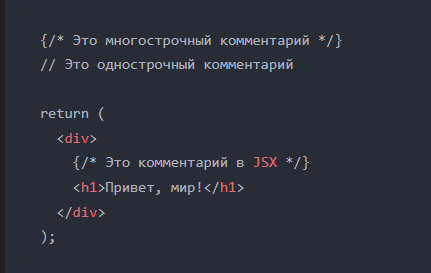
Провайдер контекста используется для установки значения, которое будет доступно всем компонентам потомкам, независимо от глубины вложенности. Потребитель контекста используется для чтения этого значения из компонента.

**37 Что такое проп children?**

children - это особый проп в React, который может содержать вложенные компоненты или элементы между открывающим и закрывающим тегами компонента.

38 Как выглядят комментарии в React?

Комментарии в React выглядят также, как и в обычном HTML коде, то есть заключаются в /\* \*/ или // в зависимости от того, является ли комментарий многострочным или однострочным. Однако, если вы хотите написать комментарий в JSX, то его нужно заключить в фигурные скобки, как показано ниже:



Обратите внимание, что комментарии в JSX должны находиться внутри элемента или компонента, а не быть самостоятельными.

**39 Для чего используется ключевое слово super с аргументом props в конструкторе?**

Ключевое слово super с аргументом props в конструкторе используется при создании классовых компонентов в React для вызова конструктора родительского класса и передачи ему объекта props.

Если мы создаем классовый компонент, который расширяет другой класс, например React.Component, то мы должны вызвать конструктор родительского класса в конструкторе дочернего класса. Для этого применяется ключевое слово super, которое вызывает конструктор родительского класса.

Аргумент props передается родительскому конструктору в качестве первого аргумента. Обычно, в конструкторе родительского класса выполняются какие-то начальные настройки, и переданный объект props может быть использован в этих настройках.

**40 Что такое согласование?**

Согласование (англ. "reconciliation") - это процесс, который React выполняет при обновлении состояния компонентов и дерева виртуального DOM для сравнения текущего состояния с новым и определения минимального набора изменений, которые необходимо внести в реальный DOM.

При каждом обновлении состояния или пропсов компонента React создает новое дерево элементов виртуального DOM, которое должно быть согласовано со старым деревом. Для этого React использует алгоритм согласования, который определяет, какие элементы должны быть обновлены, удалены или добавлены в реальный DOM.

Алгоритм согласования работает по принципу сравнения двух деревьев элементов и определения разницы между ними. React анализирует каждый элемент в деревьях и сравнивает их типы, ключи и свойства. Если элементы различаются, React обновляет соответствующий элемент в реальном DOM.

Если элементы имеют одинаковый тип, ключ и свойства, то React переиспользует уже существующий элемент вместо создания нового. Это позволяет избежать лишних операций обновления DOM и улучшить производительность.

Согласование - это одна из ключевых возможностей React, которая позволяет эффективно обновлять компоненты и дерево виртуального DOM. Благодаря ему React может реагировать на изменения состояния и пропсов компонентов быстро и безопасно для производительности.

**41 Как определить состояние с помощью вычисляемого свойства?**

Вычисляемое свойство (computed property) - это способ определения свойства объекта в JavaScript, который позволяет использовать выражение для вычисления значения свойства. В React вычисляемые свойства могут быть использованы для определения состояния компонентов.

Для определения вычисляемого свойства в компонентах React используйте синтаксис {} при определении свойства состояния:



В этом примере мы создаем классовый компонент MyComponent, в котором определяем два свойства состояния: counter и doubledCounter. Свойство doubledCounter является вычисляемым и определяется как удвоенное значение свойства counter.

При обновлении компонента React автоматически пересчитывает значение вычисляемых свойств, так что мы можем использовать их в методе render для отображения данных.

Обратите внимание, что при определении вычисляемого свойства мы можем использовать значения других свойств, определенных в объекте состояния. Однако мы не должны обращаться к this.state напрямую извне конструктора, так как это может привести к ошибкам в работе компонента. Вместо этого мы можем использовать предыдущее состояние в методе setState.

**42 Какая распространенная ошибка приводит к вызову функции при каждом рендеринге?**

Частой ошибкой, приводящей к вызову функции при каждом рендеринге, является передача функции в компонент без использования колбэка. В этом случае функция будет вызываться каждый раз при изменении состояния или свойств компонента, что может привести к нежелательным побочным эффектам и ухудшить производительность приложения.

Например, если в компоненте необходимо обрабатывать события кликов на кнопке, то следует передавать колбэк-функцию в свойство onClick элемента button, а не саму функцию:



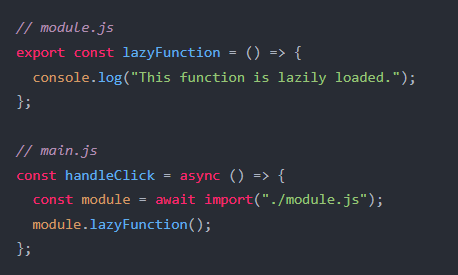
В первом случае функция handleClick будет вызываться при каждом рендеринге компонента, даже если пользователь не кликает на кнопку. Во втором случае функция будет вызываться только при клике на кнопку.

**43 Поддерживают ли "ленивые" (lazy) функции именованный экспорт?**

Да, "ленивые" (lazy) функции в поддерживают именованный экспорт.

"Ленивый" (lazy) импорт и экспорт позволяют отложить загрузку модуля до момента, когда он действительно будет нужен. Это может ускорить инициализацию приложения и уменьшить его объем.

Пример именованного экспорта ленивой функции:



Здесь мы экспортируем именованную ленивую функцию lazyFunction из модуля module.js, а затем импортируем ее в асинхронной функции handleClick в файле main.js. При вызове функции module.lazyFunction() произойдет ленивая загрузка модуля module.js.

Обратите внимание, что поддержка ленивых функций не является стандартной функцией JavaScript, она предоставляется некоторыми средами выполнения JavaScript, такими как браузер или Node.js.

**44 Почему в React используется className вместо атрибута class?**

Классы в JavaScript являются зарезервированным словом, что может привести к конфликтам и неожиданному поведению при использовании атрибута class. В React для указания классов компонентов используется атрибут className, который является эквивалентом атрибута class из HTML.

Также, использование className вместо class в React соответствует принципу разделения логики и представления (Separation of Concerns). Компоненты React вместе с их стилями обычно содержат логику и представление одновременно, и использование className позволяет отделить CSS-классы от логики компонента.

Наконец, использование className также соответствует подходу BEM (Block Element Modifier), который рекомендует использовать префиксы для имён классов, чтобы избежать конфликтов при наследовании стилей между элементами.

**45 Что такое фрагмент (Fragment)?**

Фрагмент (Fragment) в React - это компонент, который позволяет группировать список дочерних элементов без создания дополнительных узлов в DOM.

**46 Почему фрагменты лучше, чем div?**

Фрагменты (Fragments) в React лучше подходят для группировки дочерних элементов, поскольку они не создают дополнительных уровней DOM-узлов, как это происходит при использовании обычных тегов div или span.

Когда мы группируем элементы в React с помощью тегов div или span, React создает новые узлы в DOM для каждого такого тега. Это может привести к избыточной нагрузке на память и производительность при работе с большим количеством элементов.

При использовании фрагментов React создает только один узел в DOM, который содержит все дочерние элементы, объединенные вместе. Это делает код более быстрым и эффективным.

Кроме того, фрагменты могут быть полезными, когда вы хотите вернуть из компонента несколько элементов без обертки в дополнительный тег. Например, если у вас есть компонент, который должен отображать два текстовых блока, то вы можете использовать фрагменты, чтобы вернуть два дочерних элемента без оборачивания их в общую обертку.