1. Что такое React?

React это библиотека для создания пользовательских интерфейсов. Основная логика реакт сосредоточена в ядре. А за отрисовку пользовательского интерфейса отвечает react-dom или react-native. Может работать в браузере, может на Android и Native.

1. Опишите алгоритм согласования.

Алгоритм «согласования» - это то, как принимается решение о повторном рендеринге компонента. React использует концепцию виртуального дерева элементов. Поэтому он быстрый. При каждом изменении состояния приложения React использует «снимок», который участвует в анализе изменений. Алгоритм согласования рассматривает не сами элементы, но их отдельные атрибуты. React не перестраивает компонент целиком, если в этом нет необходимости. Например, при использовании динамического className. Но при переключении компонента на другой с помощью условной конструкции произойдет полная перестройка, даже если компоненты обладают одинаковыми дочерними элементами. Атрибут key используется, чтобы указать, какие компоненты и в какой момент времени остаются стабильны (не изменяются). Обновляются только те части, в

которых произошли изменения.

1. Что такое JSX?

Это расширение языка JavaScript, позволяющее писать HTML-подобный код. Это по сути <div> и что в нем, он преобразуется в JS и отдается браузеру. Браузер JSX понимать не может, нужно преобразование в JS. Запрещено использовать зарезервированные слова, такие, как class и for. Используются className и htmlFor. Можно вставлять значение переменной или JS выражение (сумма).

1. Как с помощью React отрисовать список?

JSX позволяет использовать возможности JS. Отрисовывать списки можно с помощью метода массива map. Необходимо преобразовать каждый элемент массива в необходимый объект TSX. Чтобы React мог правильно отслеживать изменения, происходящие с элементами списка, необходимо передавать им атрибут key, уникальный для каждого элемента.

1. Что такое синтетическое событие?

Это кроссбраузерные обертки над стандартными событиями, которые обеспечивают одинаковую работу событий во всех браузерах. Например, чтобы обработать событие change, необходимо написать соответствующий обработчик и передать его в качестве атрибута onChange.

1. Что такое пропсы (props)?

Это данные, доступные только для чтения, которые передаются от родительского компонента дочернему компоненту. Они используются для настройки поведения и внешнего вида компонента. Пропсы похожи на аргументы функции или атрибуты HTML, но они специфичны для компонентов React. Пропсы передаются компоненту как объект, и к ним можно получить доступ в компоненте, используя ключевое слово **props**. Пропсы могут использоваться для передачи данных любого типа, включая строки, числа, логические значения, объекты и даже функции. Пропсы также можно использовать для передачи других компонентов в качестве дочерних.

1. Что такое состояние?

State (**состояние**) в React – это объект простого JS, позволяющий отслеживать данные компонента. Состояние позволяет компоненту “запоминать” информацию, подобную вводимой пользователем. Например, компонент формы может использовать state для сохранения входного значения, в то время как компонент галереи изображений может использовать state для сохранения индекса выбранного изображения.

Чтобы добавить состояние компоненту, используется перехватчик:

* useState объявляет переменную состояния, которую вы можете обновлять напрямую.
* useReducer объявляет переменную состояния с логикой обновления внутри функции-редуктора.

Состояние нужны, чтобы React понял, когда перерендерить.

1. Как передать данные из дочернего компонента в родительский?

С помощью callback handlers.

1. Что называется двусторонним связыванием?

При изменении поля ввода меняется состояние, и при изменении состояния меняется значение поля ввода. Например, пользователь меняет значение input, событие изменения попадает в обработчик, там новое значение устанавливается в качестве значения состояния, и мы видим обновленное значение.

1. Опишите жизненный цикл компонента.

Компонент создается, затем добавляется в DOM, получает пропсы и удаляется из дерева.

1. Что такое хуки?

Это технология, которая перехватывает вызовы функций. Функции, начинающиеся с use, называются перехватами. useState это встроенный перехват, предоставляемый React. Можно написать свои собственные хуки, объединив существующие.

1. Какой хук позволяет работать с жизненным циклом компонента?

`useState` — инициализация и обновление состояния.

`useEffect` — работа с побочными эффектами после рендеринга.

`useContext` — доступ к глобальным данным через контекст.

`useReducer` — управление сложным состоянием с помощью редьюсера.

`useCallback` — мемоизация колбэков.

`useMemo` — мемоизация значений.

`useRef` — управление ссылками на DOM-элементы и сохранение изменяемых данных без ререндера.

`useLayoutEffect` — идентичен `useEffect`, но срабатывает синхронно после всех изменений DOM.

1. Что такое фрагменты

Это компонент **React**, который помогает нам возвращать несколько дочерних элементов без использования дополнительных DOM узлов. Если мы хотим вернуть группу элементов, то нужно обернуть их в общий тег без имени. **React** компонент должен возвращать только один элемент, иначе будет ошибка.

1. Как использовать вложенные в компонент элементы (children)?

Элементы, помещаемые внутрь компонентов, передаются им как пропс. Из пропсов можем извлечь props.children.

1. Расскажите про хук useRef.

**useRef** это хук, который позволяет не обращаться к тегу по id, возвращающий изменяемый ref-объект. Его свойство current инициализируется переданным аргументом. Чтобы связать элемент с ref-объектом, используется атрибут ref. Если надо передать ref дочернему компоненту, его необходимо обернуть в forwardRef.

1. Как произвести условный рендеринг

В React нет специального синтаксиса для записи условий. Вместо этого используются те же методы, что и при написании обычного кода JavaScript: оператор if, тернарный оператор, логический &&.

1. Опишите механизм работы с редьюсерами (reducers).

useReducer это React-хук аналог useState, который позволяет реализовать более сложную логику управления состояниями. Начинается с функции-редьюсер, которая реализуется вне компонента. Она получает на вход состояние и действия, и в зависимости от них возвращает новое состояние. Редьюсер передается в хук useReducer первым параметром. Вторым параметром в хук передается начальное состояние. Редьюсер вызывается и в зависимости от типа action будет менять состояние данных, которые передаем.

Т.е. мы не просто передаем состояние, а передаем объект, на основании которого редьюсер будет совершать какие-то действия.

1. Для чего нужны хуки useCallback и useMemo?

**useCallback** принимает на вход функцию и массив зависимостей. Он позволяет вернуть меморизированную версию принятой функции, которая изменяется только при изменении зависимостей. Такой подход позволяет не создавать функцию при каждом обновлении компонента, избежать ситуаций с бесконечным циклом useEffect (когда эффект при создании обновляет состояние, изменение заставляет обновиться компонент и пересоздать эффект, который меняет состояние).

**useMemo** в отличие от useCallback возвращает результат выполнения полученного коллбэка и обновляет каждый раз при изменении зависимостей, что позволяет избежать лишних вычислений.

1. Назовите типы компонентов в React

Изначально были только классовые компоненты. Это классы с обязательным методом render(), возвращающим JSX. Наследует React.Component. Доступ к пропсам имеет через this. Классовые считаются устаревшими.

С появлением хуков стали использоваться функциональные компоненты. Хуки принесли реактивность функций.

Есть еще компоненты высшего порядка, который принимают компонент на вход и возвращают компонент на выход, но с расширенными функциональными возможностями.

Еще подварианты: Глупые компоненты – это функциональные компоненты без состояния. Умные компоненты – компонент класса, который управляет своим состоянием. Презентационные компоненты часто называют функциональным компонентом без состояния, который принимает параметры (пропсы) и отображает пользовательский интерфейс. Контейнерный компонент - это компонент класса, который предоставляет данные и поведение презентационным или другим контейнерным компонентам. Контейнер выполняет выборку данных, а затем отображает соответствующий подкомпонент.

1. Перечислите варианты работы со стилями в React.

Через импорт простого css файла. Через задание inline стилей. Можно импортировать классы из файла css.

import classes from ‘./App.module.css’

Можно установить библиотеку styled-components. Она пользуется шаблонными литералами. import styled from ‘styled-components’.

1. Расскажите про компоненты высшего порядка (Higher-Order Components).

Это функции (HOC Higher-Order Components), принимающие компонент и возвращающие новый компонент. HOC является одним из способов переиспользования логики.

const hoc = (MyComponent) => (props) => {

return (

<div>

<MyComponent {...props}>

{props.children.toUpperCase()}

</MyComponent>

</div>

)

}

1. Опишите технику Render Props..

Рендер пропсы позволяют производить повторное использование. Компонент с рендер-пропсом принимает функцию, которая возвращает Реакт-элемент, и вызывает ее, предоставляя необходимые значения.

1. Как работать с порталами?

Порталы – это технология рендеринга компонентов вне родительского компонента. Используются, когда необходимо визуально отделить компонент от родителя, например, модальные окна, подсказки. Создаются с помощью ReactDOM.createPortal(child, container), где child – Реакт-элемент, фрагмент или строка, container – DOM-элемент, куда будет внедряться child.События внутри портала будут распространяться на предков в дереве Реакт, даже если эти элементы не являются предками в дереве DOM.

1. Расскажите про асинхронную загрузку компонентов.

Не всегда у пользователя хорошее интернет-соединение. Время загрузки напрямую зависит от размера пакета. Чтобы ускорить загрузку, можно разбить пакет на части и подгружать их по мере необходимости. Для реализации асинхронной подгрузки используется специальная функция Lazy. Lazy позволяет разделить бандл на части. Чтобы уведомить о процессе загрузки пользователя, React предоставляет компонент Suspense. Он принимает ленивые компоненты в качестве дочерних, а также заполнитель Loading, который будет отображаться во время загрузки lazy.

1. Для чего нужны компоненты Error Boundaries?

**Error Boundaries** – компоненты, предоставляющие механизмы отлавливания и обработки ошибок, возникающих во время работы компонентов. На данный момент могут быть только классовыми. Для отлова ошибок используется метод жизненного цикла componentDidCatch. На вход он принимает объекты Error и ErrorInfo.

**Не обрабатывают** ошибки обработчиков событий, ошибки в асинхронном коде, ошибки SSR, ошибки самих Error Boundaries.

1. Что такое PropTypes?

**PropTypes** – способ проверки типов пропсов во время выполнения. Требует загрузки пакета prop-types. Чтобы задать для компонента правила валидации пропсов, необходимо задать свойство propTypes. Теперь, если передать параметры не того типа, в консоли появится ошибка.

**Context API**

1. Что такое Context API?

Это интерфейс, который позволяет передавать состояние всей иерархии компонентов, обернутых в специальный компонент-провайдер, без необходимости явной передачи в виде пропсов. Создается с помощью функции createContext(). В качестве параметра createContext() принимает значение по умолчанию и возвращает объект контекста. Context API входит в Реакт.

1. В чем заключается проблема Prop Drilling?

Любое Реакт-приложение обладает вложенностью. Проблема в том, что пользователь находится в самом верху иерархии. Информация передается по дереву компонентов. Например, если приложение приветствует пользователя при входе в систему. Это случай prop drilling.

Prop Drilling описывает передачу данных через несколько вложенных дочерних компонентов с целью доставки этих данных в глубоко вложенный компонент. Проблема заключается в ненадобности передаваемых данных промежуточным компонентам. Это создает проблемы производительности и переиспользования. Context API – вариант решения ситуации.

1. Как создать контекст?

Функцией createContext(). В качестве параметра принимает значение по умолчанию и возвращает объект контекста.

1. Как передать контекст?

Каждый Context объект предоставляет компонент Provider, который позволяет дочерним компонентам, использующим этот контекст, подписаться на его изменения. Provider используется как любой другой компонент и принимает в качестве пропса value начальное значение контекста. Все потомки-потребители Provider будут перерисовываться каждый раз, когда меняется значение value, переданное в Provider.

1. Как принять контекст?

Использовать передаваемый контекст позволяет специальный хук useContext(). В качестве параметра он принимает объект контекста и возвращает его текущее значение.

Альтернативный способ подписки на контекст – компонент Consumer, в качестве параметра children он принимает функцию с атрибутом, являющимся текущим значением контекста, возвращающую JSX элемент.

1. Назовите проблемы Context API.

Проблема **переиспользования** компонента, зависимого от контекста, за пределами этого контекста, приведет к ошибке, если у контекста не задано дефолтное значение.

**Производительность** – все компоненты, использующие контекст, перерисовываются каждый раз при изменении состояния этого контекста. Возможна масса излишних перерисовок.

**Надо помнить о трех вещах:**

действительно ли в данной ситуации использование контекста необходимо,

каждый контекст должен отвечать за свое конкретное состояние

Использовать контекст надо максимально близко к компонентам, которые от него зависят.

1. Опишите технику композиции.

Это способы, как обойтись без контекста. **Первый вариант композиции – обертка** родительского компонента вокруг нескольких дочерних.

**Второй вариант – передача компонента в другой** в качестве параметра. При таком подходе передача необходимого состояния происходит при передаче зависимого от него компонента в родительский компонент.

В обоих случаях состояние из родительского компонента в дочерний передается напрямую.

**MobX**

1. Что такое MobX?

Это простая, масштабируемая и мощная библиотека управления состоянием. Подобно React, который использует виртуальный DOM для рендеринга элементов пользовательского интерфейса в наших браузерах, уменьшая количество мутаций DOM, MobX делает то же самое, но в состоянии нашего приложения.

1. Назовите основные концепты MobX.

**State** – состояние, относится ко всей модели приложения и может представляться различными типами данных

**Actions** – действия, служат для управления состоянием приложения

**Derivations** – производные. Производными называется все, что получено из состояния, приложения. Бывают 2 видов: reactions и computed values.

1. Опишите поток данных MobX.

MobX использует однонаправленный поток данных. Действия изменяют состояние, которое, в свою очередь обновляет зависимые представления.

1. В чем отличие реакций (reactions) и вычисляемых (computed ) значений?

**Вычисляемые** свойства выводятся из текущего состояния приложения и обязательно возвращают значение.

**Реакции** – эффекты, происходящие при изменении состояния приложения.

1. Можно ли изменять состояние в производных? Почему?

**Производными называется все, что получено из состояния, приложения**. Бывают 2 видов: reactions и computed values.

Производные обновляются автоматически и атомарно.

По умолчанию все производные обновляются синхронно.

Вычисляемые свойства обновляются лениво. Любое вычисляемое свойство не будет обновляться до тех пор, пока не понадобится.

**Вычисляемые свойства не должны менять свое состояние.**

1. Как с помощью MobX создать хранилище?

Специальной функцией makeObservable(). Она делает переданный объект наблюдаемым и отслеживает изменение его полей. Вторым параметром принимается набор аннотаций полей объекта. Каждое поле может являться observable, computed или action.

Другие функции: makeAutoObservable – аналог makeObservable, но с автоматическим выведением типов наблюдаемых свойств.

extendObservable – используется для введения новых наблюдаемых свойств в observable объект.

1. Назовите функции, используемые для создания реакций.

Autorun – принимает функцию и использует ее каждый раз при изменении присутсвующих в ней наблюдаемых свойств.

Reaction – первым параметром принимает функцию, описывающую наблюдаемые свойства, вторым параметром принимается эффект, который будет выполняться всякий раз при изменении описанных первой функцией свойств.

When – превым параметром принимает функцию предиката, вторым – эффект, который вызывается при возвращении первой функцией true

1. Как “подружить” MobX с React?

Написать пользовательский интерфейс. Потом установить библиотеку mobx-react-lite. Она предоставляет функцию observer. Она принимает в качестве аргумента функциональный компонент, возвращает его observable версию, обновляющуюся при каждом изменении используемых в компоненте observable свойств. Оборачиваем зависимые от хранилища MobX компоненты в observable.