**50 вопросов на React JS собеседование**

[**https://www.youtube.com/watch?v=-cZOdWjFwXw**](https://www.youtube.com/watch?v=-cZOdWjFwXw)

1. **Что такое React**

**Это JS библиотека, разработанная компанией facebook. Пропагандирует компонентный подход, то есть вместо больших полотен кода проще разбить его на простые компоненты и уже их по частям использовать в коде. Подходит для разработки интерфейсов не только в WEB, но и мобильных и десктопных приложений.**

1. **Основные преимущества React**

**React оптимизирует процессы отрисовки интерфейсов. В нем используется специальный jsx синтаксис, помогающей улучшить понимание кода при создании различных шаблонов. Касается к примеру html кода, который мы вписываем в jsx файлы. Хорошо используется с назными фреймворками ведь в конце концов это библиотека. На нем легко писать юнит тесты (программирование, позволяющее проверить на корректность отдельные модули исходного кода программы)**

1. **Ограничения в React**

**Это библиотека и у нее может не быть готовых решения для некоторых случаев. Например валидация форм – проверка корректности введенных в форму данных перед отправкой.**

1. **Что такое JSX**

**JSX синтаксис позволяет писать похожий на html код внутри компонентов тем самым, делая их понятными. Без лишних кавычек и тд. Хотя на деле это работает с помощью функции React Create Element.**

1. **Что такое Virtual DOM в React**

**В JS самая затратная операция это работа с DOM деревом, ну то есть например когда я через document.getElementbyID() пытаюсь обратится к элементу. Получить в скрипте определенный элемент. В React это оптимизировано следующим образом, в js создается объект древовидной структуры, который был точной копией реально DOM дерева. При каких-либо изменениях в React впервую очередь изменяется виртуальный DOM, а это намного быстрее чем с реальным. А затем эти DOM сравнивались и реальной DOM изменялся только в тех местах, где был он не совпадал с виртуальным.**

1. **Что такое Props**

**От английского properties – свойства. Необходимы для интерактивности и связи между родительскими элементами и дочерними, позволяя рендерить различные интерфейсы.**

1. **Что такое State**

**Это обычный объект, который содержит в себе определенные данные. Состояние мутабельны, то есть их можно менять в отличии от пропсов, которые менять нельзя(Нельзя?).**

**Когда мы изменяем State React это вычисляет и рисует новый интерфейс.**

1. **Что такое refs в React**

**От английского reference – ссылки. С помощью этого свойства можно получать ссылки на Dom элементы. Полезно при взаимодействии с анимациями или если есть необходимость иметь доступ к dom элементу из другого элемента в обход quarrySelector.**

1. **Что такое JEST.**

**Впервые слышу. Frame work для проведения юни тестов в том числе и в React приложениях.**

1. **Когда следует использовать класс компоненты, а когда функциональные.**

**Если компонент используется только для отрисовки какого-то интерфейса, то его обязательно нужно сделать функциональным, так потребляется меньше ресурсов. Но если это более сложный компонент, которому нужны доступы до жизненных этапов компонента, то нужно использовать класс компонент (Это легаси фича? Не помню подобного)**

1. **Что происходит, когда вы вызываете SetState.**

**В первую очередь React получает объект, которую мы передаем в функцию setState и соединяет его с прошлым объектом, который был. Затем получается новый state и React создает новый виртуальный DOM. Затем сравнивает его с реальным DOM, находит изменения и меняет его.**

1. **В чем разница между state и props.**

**state структура данных имеющая начальное значение и когда react рисует интерфейс, он основывается на начальном значении. Но в работе приложения state может меняться, но в основном, он меняется от каких-то пользовательских действий. А props это параметры, которые идут в компонент от его родителя. Вроде как однонаправленное состояние, передающееся по древовидной структуре.**

1. **Когда следует делать асинхронные запросы на сервер в React?**

**Какой момент самый подходящий для этого. В классическом представлении React подойдет метод component did mount (Что это).**

**Если сделать запрос сервер раньше, то есть когда React компонент не готов для работы с dom деревом, то есть еще не был “Замаунтин”, то если нам сервер ответит раньше, то мы будем менять состояние еще до того, как html готов, это может вызвать некоторые ошибки. Поэтому лучше подождать, когда React подготовит шаблон для компонента, а уже потом сделаем запрос. Это позволит избежать некоторых ошибок**

1. **В чем смысл специального атрибута key?**

**React требует, чтобы мы добавляли его в списки однотипных элементов, которые мы рендерим. В настоящем DOM дереве этого атрибута нет. Этот атрибут позволяет менеджерить состояние каждого элемента в массиве. Это требуется React, на случай если какой-то элемент будет удален и чтобы был удален сразу этот элемент, без полной итерации массива. Конечно можно использовать индекс, как уникальное значение, но лучше все же id.**

1. **Что значит компонент mounted?**

**Это значит, что компонент уже вмонтирован в dom дерево. То есть шаблон компонента был подготовлен и помещен в реально html и он готов к работе.**

1. **Разница между контролируемым и неконтролируемым компонентом.**

**У контролируемого компонента есть State за которым мы следим. Допустим есть input и на каждое изменение input вызывается функция setState или используем хук.**

**В неконтролируемом компоненте к примеру, также присутствует State, но мы его не обрабатываем. Эм…, например, textArea, где мы можем что-то вписать, но не взаимодействовать со state. Через тот же Ref как я понял. А в первом случае через onChange например**

1. **Что такое фрагменты.**

**В Реакт нельзя возвращать не обёрнутую структуру с определенным содержанием. Можно вернуть скажем один условный параграф, но если это два параграфа, то необходима обертка. Например, Div, но если нужно без него, то есть тег - <React.Fragment> </React.Fragment>. Но гораздо удобнее использовать просто пустой тег, кажется это тоже самое <></>. Синтаксис JSX требует какой-то родительский элемент, но он будет пустым если использовать фрагмент.**

1. **Как React обрабатывает пользовательские события.**

**Во-первых, для того, чтобы оптимизировать приложение, React всегда добавляет одно событие, для корневого элемента. React не добавляет много обработчиков кликов к примеру, он добавляет один обработчик кликов и дальше уже понимает, по какому элементу был сделан клик или другое событие. Это оптимизирует работу с событиями.**

**Во вторых, когда мы получаем некоторый объект событие, то React, на самом деле, предлагает специальный класс, специальная обертка под названием SyntheticEvent. К примеру в React есть обработчик onMouseLeave, когда настоящий это mouseout. Оборачивая все события в одну обертку, мы получаем общий API который можем использовать независимо от браузера и т.д.**

1. **Что такое Redux**

**Этот вопрос не совсем относится напрямую к React. Но часто встречается на собесах по React. React это скажем инструмент для визуализации, мы через него скажем так, рисуем. Но когда речь о каком-то сложном приложении, то все же нужен некоторый абстрактный слой для работы с данными.**

**Redux это библиотека, которые если что, не привязана только к React, но это решение позволяющее работать с потоком данных, который идет отдельно от самого приложения. И с помощью такой библиотеки как react-redux, мы можем связывать конкретно React и Redux. Отображение и модель данных. Еще одна особенность Redux, в нем используется один объект для состояние всего приложения. Состояния описываются в одном объекте, в разных полях и уже на основе этого состояния рисуются разные интерфейсы.**

1. **Основные этапы жизненного цикла компонента**

**Сначала процесс инициализации в конструкторе, устанавливаются props и state.**

**Затем процесс Mounting, в других библиотеках этот этап может называться этапом рендеринга, сначала идет метод componentWillMount, React сообщает, что сейчас будет происходить процесс рендера. Затем происходит рендер, React берет JSX и преобразует его в шаблон. И после вызывается componentDidMount, оповещающий о готовности компонента, после чего можно, например, делать асинхронные запросы.**

**Updation. Этап, когда приходят новые свойства в компонент или меняется state. С пропсами так: сначала метод componentWillReceiveProps, далее если реализуем метод shouldComponentUpdate у него есть 2 вариации, если вернуть true, то будут те же этапы, что и во время цикла mounting, а если вернуть false, то рендеринг не будет происходить, про этот метод узнаю отдельно.**

**Для state точно также, но без первого метода componentWillReceiveProps.**

**И последний из основных жизненных циклов, это unmounting. Процесс уничтожения компонентов. Метод compnentWillUnmount. В нем обычно идет очистка, обнуление State и ссылок, слушателей и тд.**

1. **В SetState можно передавать объект или функцию. В чем разница и что лучше использовать?**

**По умолчанию в setState передается только объект. При этом также можно передать функцию. Но эта функция, принимает в себя всегда предыдущее состояние, то есть state. И на основе предыдущего state можно сформировать новый. Зачем это нужно. Дело в том, что props и state в React работают асинхронно. Поэтому, когда в метод setState передается объект, это не гарантирует, что вы всегда будете работать с предыдущим состоянием. Если хочется быть уверенным, что работа идет с предыдущим состоянием, то нужно передавать функцию.**

1. **В чем разница между Презентационным и Контейнер компонентом.**

**Презентационный компонент, демонстрирует как все должно выглядеть. Обычно работает на пропсах и просто рисует интерфейс. А контейнер компонент, говорит, как все должно работать и взаимодействовать. Обычно в нем содержится различная информация по state, по поведению. Обычно он раздает различные свойства.**

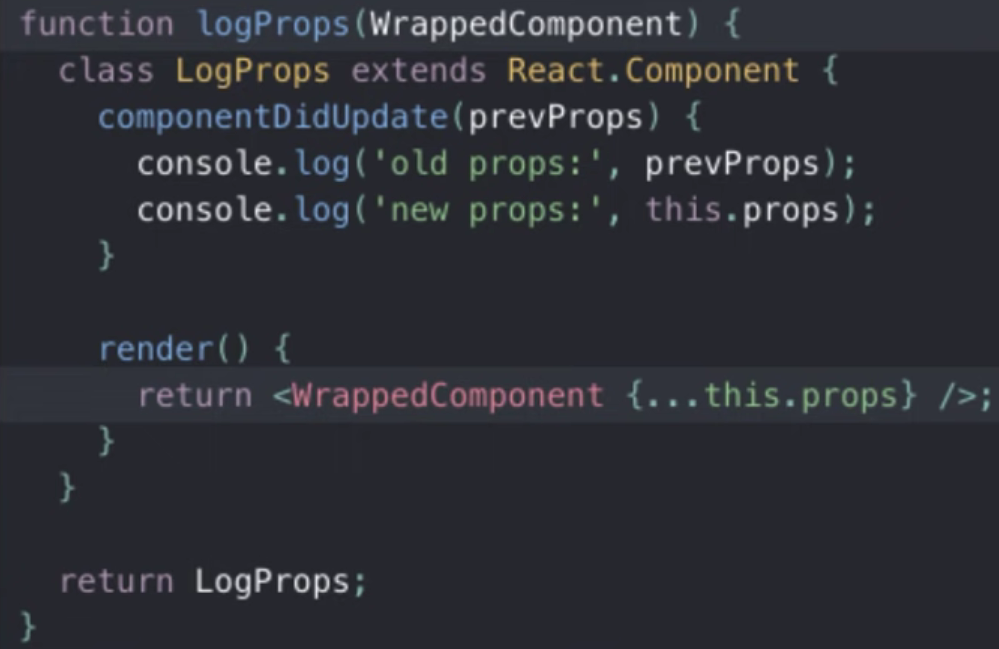
1. **Что такое Context.**

**В React компоненты, могут быть вложены друг в друга. Представим, что в главном родительском компоненте содержится состояние, которое важно какому-то глубоко зарытому дочернему элементу. В классическом React пришлось бы через каждый дочерний компонент пробрасывать это состояние через props. Однако теперь для решение этой задаче лучше использовать context. Сначала назначаем, например, provider из корневого элемента. И в value передаем объект, с его конфигурацией. Состояния, функции, что захотим.**

**И далее обращаемся к любому вложенному компоненту используем тот, же самый компонент контекст, но уже вместо provider используем consumer. Там мы получаем value, который будет содержать то, что мы передавали в provider.**

1. **Что такое Higher-Order компоненты.**

**Это концепт, не относящийся исключительно к React, это скорее такой паттерн. Это такие функции, который как input принимают в себя какой-то компонент, а уже после добавляет в него какой-то функционал. На выходе возвращают тот же самый компонент, но уже с расширенными возможностями. Например, ниже есть изображение, на котором создана обертка, в которую можно поместить компонент и при использовании его в таком виде, при каждом обновлении он будет выводить логи своих свойств.**



1. **Что делает shouldComponentUpdate и почему он важен.**

**Он служит для оптимизации рендеринга React. Возвращает true или false, допустим в компоненте поменялось какое-то состояние, но оно не влияет на отображение компонента, в таком случае более целесообразно не обновлять компонент. Также в React существует класс pureComponent, который автоматически реализовывает метод ShouldComponentUpdate. При использовании функциональных компонентов это функция ReactMemo.**

1. **Что такое store в Redux.**

**Это по сути обычный JS объект, содержащий в себе состояние нашего приложения. У него есть небольшое количество методов, позволяющих взаимодействовать с этим состоянием. Например getState, получение состояния, dispatch – изменяющий состояние в state, subscribe позволяющий подписываться на обновление.**

1. **Что такое action в Redux**

**Action – js объект, у которого должно быть обязательное поле type, которое должно быть в строковом значении и в нем, прописывается что конкретно этот action делает. То есть вызывается функция dispatch и в нее отправляется объект action, где реализован type. Далее идем в Store и меняем по циклу состоянием State.**

1. **Что нельзя делать в методе render.**

**Render должен быть обычной функцией, которая не меняет никакие состояния, то есть это должна бы pureFunction. Если делать setState в рендер, то приложение войдет в рекурсию, чего важно избежать (Эм… упс). Это запрещено, в нем мы просто должны возвращать JSX.**

1. **Какие типы middleware есть в redux для работы с асинхронностью.**

**Мидлвары — это функции, которые последовательно вызываются во время обновления данных в хранилище.**

**Методы redux thunk, rexux promise, redux saga.**

1. **Что такое Pure Components**

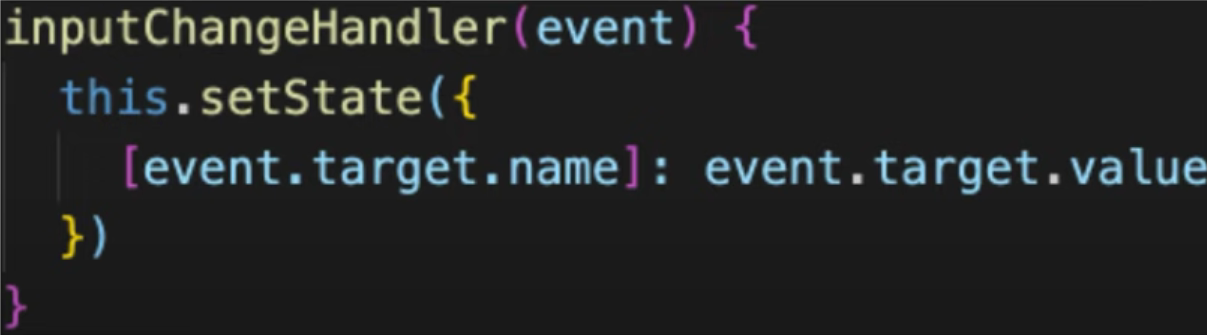
**По умолчанию, в React наследуются от React компонента и реализуют класс. Но если наследоваться просто от компонента, а от pure компонента, то React автоматически реализует за нас метод shouldComponentUpdate, тем самым оптимизируя его.**

1. **Почему не стоит изменять state напрямую**

**Почему нельзя написать, this.stateLoading = false? Все дело в том, что если напрямую так прописать значение, то не будет происходить ререндеринга компонента, теряется смысл React. Поэтому и вызывают метод setState.**

1. **Как изменить state используя динамический ключ.**

**Допустим имеется n-е количество инпутов. То писать обработчик для каждого input, это не эффективно. Поэтому можно воспользоваться свойствами ES6 и использовать динамические ключи в state. Вот пример**



1. **Что такое Error Boundaries в React**

**Это React компонент, у которого может быть реализован жизненный этап, который называется componentDidCatch. Допустим есть потенциально проблемный код. Этот проблемный JSX можно обернуть. И тогда при проблеме с рендером, вместо краша всего приложения на этом компоненте сработает этот эффект. И вместо того, чтобы возвращать JSX от сломанного дочернего элемента, можно вернуть что-то не сломанное.**



1. **Что такое React Hooks.**

**Концепт, позволяющий писать приложение на React не используя класс компоненты. Был введен относительно недавно. Функциональные компоненты работают быстрее, чем классовые компоненты. Хуки это функции, которые позволяют взаимодействовать со state, обновлениями, ререндерингом и тд.**

1. **В чем разница между useRef и createRef**

**Функция createRef может быть использована в компонентах, которые наследуются от React компонента и возвращает всегда новую ссылку. При этом ее нельзя использовать в функциональных компонентах. А функция useRef создает объект, который не будет изменяться в процессе рендеринга функционального компонента. То есть при его использовании, всего получаем объект, который был создан при первом рендеринге. А также если нужно будет изменить это состояние, если это референция, то не будет происходить процесса рендеринга в компоненте.**

1. **Что такое useState.**

**38 мин**