1. Здравствуйте, меня зовут Касперович Александр и я работаю в компании Иссофт солюшенз Фронт энд девелопером и занимаюсь доступностью на одном из проектов.
2. Сегодня я хочу вам рассказать о своем опыте работы с доступностью на проекте, где одной из основных технологий является React.
3. Но для начала давайте вспомним о базовых принципах доступности, которые должны знать все и даже React разработчики.
4. Используете контрастные цвет
5. Не полагайтесь только на цвет
6. Проектируйте использование фокусного состояния
7. Используйте лэйблы в формах, не ограничивайтесь использованием плейсходеров
8. Используйте метки и альтернативный текс для описания содержимого
9. Используйте правильную разметку, придерживайтесь семантики
10. Поддерживайте навигацию по клавиатуре
11. Но даже, если мы делаем статическую часть приложения/сайта доступной, зачастую возникают проблемы с обеспечением доступности динамических элементов, а если нам приходиться работать с SPA, где у нас, по сути все приложение - динамически элемент, Возникает не мало вопрос как с этим быть.
12. Сегодня я буду ориентироваться работу с библиотекой, если хотите фреймворком – React, которую частно используется для создания динамических интерфейсов. Т.к. я с ней больше всего работал и в ней можно довольно наглядно продемонстрировать нюансы, которые встречаются и во многих других технологиях.
13. В React используется компонентный подход, что накладывает на разработчика ответственность за создание и взаимодействие компонентов, поэтому нужно сразу закладывать в них возможность работы с aria (Accessible Rich Internet Applications - Доступные многофункциональные интернет-приложения) атрибутами и по возможности делать компоненты восприимчивыми к расширению.
14. Чем же можно вдохновляться или на что ориентироваться при создании компонентов?
15. Как не удивительно, но можно ориентироваться на принципы SOLID, которые больше знакомы Бэк-енд разработчикам, но и во фронте можно найти им применение, они могут помочь сделать структуру расширяемой и избежать многих проблем в бедующем. <https://web-creator.ru/articles/solid>,
16. SOLID — это аббревиатура пяти основных принципов проектирования в объектно-ориентированном программировании — принципы единственной ответственности, открытости / закрытости, подстановки Барбары Лисков, разделения интерфейса и инверсии зависимостей, если вам будет интересно, я на это надеюсь вы можете прочитать про них всех подробнее в интернете или посмотреть разные роли о их применение, но сейчас нам по тебе **обеспечения** **доступности будут интересным самые первые 2 принципа.**
17. **Принцип единственной ответственности** (single responsibility principle ) обозначает, что каждый компонент должен иметь одну обязанность.
18. **О чем нам это говорит? Что каждый компонент должен выполнять какую-то определенную обязанность, да мы можем захотеть сделать что-то максимально универсальное и чтобы все лежало в одном месте, но это будет выглядят порой как объединение молотка с ножем и им будет банально не удобно пользоваться, лучше иметь несколько инструментов (компонентов), которые будут работать хорошо и ими будет удобно пользоваться.**
19. **То же касается и дизайна, рассмотрим кнопочку, красивую, но максимально неудобную для любого пользователя…**
20. **Так же следует пере использовать судействующие компоненты и создавать из комбинации по возможности, что гораздо легче сделать с небольшими компонентами, в противном случаем может понадобиться делать много однообразных и массивных компонентов под каждый специфический случай. Надо ли это большой вопрос?**
21. **Принцип открытости / закрытости** (open-closed principle) декларирует, что  компоненты должны быть открыты для расширения, но закрыты для изменения.
22. **Решить данную задачу помогает Композиция, когда вы создаете более сложные элементы путем некоторого числа оберток базового элемента. Так например можно создать группу полей с легендой для чекбоксов используя обертки, каждая из которых добавляет свою функциональность.**
23. Существуют уже готовые библиотеки, как material-ui, которые хорошо реализованы, и вы можете просто подключить их к своему проекту и все должно быть хорошо, либо прочесть документацию и просмотреть исходный код и на основе библиотеки реализовывать уже свои компоненты, под нужды конкретного проекта.
24. И так на примере SOLID мы понимаем, что можно находить вдохновение и идеи в совершенно разных областях и не только программирования, чтобы постоянно совершенствовать свой проест. Ведь нет ничего идеального, но именно в стремлении к нему мы находим что-то уникальное (Цытата…) эта цитата дает нам поение того, что важен именно сам процесс развития, а не конечная цель.
25. Мы разобрались со структурой компонентов, но теперь, вернемся именно к доступности, через что пользователь взаимодействует с приложением, как правило через Компьютер, планшет или мобильный телефон,
26. Для разных устройств существуют разные вспомогательные устройства, которые иногда работают поразному из-за чего возникают специфические баги. (Слайд читаю)
27. И об этом нужно помнить, когда появляется задача обеспечить доступность приложения, часто работа может сводится к поиску универсального решения для большего числа вспомогательных программ и к само собой расширенному тестированию приложения.
28. Наконец подходим именно к обеспечению доступности, сколько же всего нужно учесть перед конечной разработкой, но это поможет предугадать проблемы и решить их на ранних этапах. Чаще всего разработчики при работе с доступностью встречаются с ариа атрибутами, особенно если речь идет о динамических интерфейсах.
29. Начнем пожалуй с ролей, пример таблицы, (оговариваю специфика на разных скрин ридерах)
30. Иногда возникают проблемы с ролями (пример табов)
31. Возникают подобные ошибки и-за не нужных промежуточных оберток, можно обнулить роль конечно у листа в примере, но некоторые считают это не хорошим тоном при разработке и должно применяться в самых безвыходных ситуациях. Лучше избегать нефункциональных оберток!
32. При работе с лейблами ариа-лабел лучше метку держать рядом с основам текстом
33. F
34. F
35. F
36. F
37. F
38. F
39. F
40. Программное управление фокусом
41. Работа с событиями мыши (ипользование клавиатуры)
42. Теперь обсудим подходы по достижению доступности. как бы мы не хотели сделать альтернативный текст везде, чтобы слабовидящие пользователи могли оценить все визуальные эффекты, нужно придерживаться достаточности, а не избыточности. Иногда переизбыток вторичной информации (слишком подробной) наоборот может вредить пользователю и сбивать его столку,
43. а также замедлить взаимодействие с приложением, не зря самая часто используемая настройка скринридеров – это скорость чтения. Перед тем как добавить какие-либо метки нужно удостовериться, что текст не дублируется, вспомним про слово “кнопка” оно как правило уже по умолчанию произносится исходя из тегов или назначенных ролей и указывать еще раз, что это кнопка будет уже ошибкой.
44. Помимо самого удобства пользователей, использующих скринридер, избыточность может мешать пользователям вообще не использующих режимы доступности, особенно если различные метки генерируются динамически с множеством проверок, которые могут замедлять работу всех пользователей. Поэтому следует по возможности использовать то, что уже реализовано для обычного пользователя и не перегружать приложение дополнительным функционалом. И аккуратно внедрять то, что нам требуется, ища возможность никаким образом не ухудшить работу приложения для всех пользователей.
45. Одним способом решения проблемы является создание специального режима, который включается по желанию пользователя, и модульно подгружается на устройство пользователя, но это конечно же довольно трудоемкое решение, которое редко кто-то использует тк если вы первоначально придерживаетесь семантики и привязываете альтернативный текст к основному контенту, то вы избежите множества багов в будущем и вам не придется переписывать огромное количество кода, и думать о том, что вместо исправлений реально проще сделать отдельный режим, чтобы все не ломать и не перестраивать весь проект. Поэтому этот способ я рассматриваю только в том случае, если к вам попал уже написанный проект и написанный не очень качественно, но в идеале и если заказчику не жалко денег и времени, лучше переписать до нормального состояния.
46. Целесообразность создания тимы доступности? В рамках проекта можно выделить небольшую команду для обеспечения доступности продукта. И это обусловленно оптимизацией работы всей команды, ведь довольно сложно группе более 10 человек помнить все стандарты, правильно их применять и контролировать уровень доступности каждому члену команды проблематично, поэтому можно и иногда нужно выделять отдельную небольшую группу именно под решение задач связанных с доступностью и это снимет большей груз и все будут четко понимать свои задачи и смогу сосредоточиться на них, подобное разделение когда-то произошло и с тестировщиками. Чисто теоретически разработчики могут самостоятельно все тестировать, но согласитесь эта не лучшая практика.

но часто эта команда занимается исправлением багов, даже если разработка проекта происходит внутри компании(

1. лучше конечно чтобы команда доступности давала свои рекомендации на этапе разработки нового фунциона, а не просто после выката каждой фичи им приходилось переделывать фичи и подправляла косяки своих коллег – это не оптимальное использование рабочего времени. Если у вас на проекте есть такая группа разработчиков, то следует их привлекать на самых начальных этапах разработки, чтобы не увязнуть в переписывании. Тима доступности должна заниматься исправлением багов (куда же без них), но максимально стараться предупреждать ошибки, но для этого их нужно привлекать к встречам и обсуждениям новых фич.
2. Одной из задач такой тимы может быть написание документации и рекомендации в рамках конкретного проекта и заказчика, чтобы мелкие правки проделывались по стандартам и не привлекать доступников для каждого изменения внутри проекта. По возможности и по своему усмотрению ставьте членов доступности в ревьюеры пул реквестов. Не маловажный нюанс.
3. Тима доступности должно довольно тесно общаться со всеми членами проекта и иногда даже с дизайнерами, редко, но бывает что невозможно сделать доступными отдельные прихоти дизайнеров. Поэтому их работа - это сплошной поиск компромиссов и им придется быть хорошими дипломатами и от части тестировщиками. Всем же остальным нужно набраться доброжелательности, чтобы не придушить кого-то из них ненароком ))
4. Общайтесь и уважайте труд и время друг друга.
5. Конец () ссылки и тд.