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, для того чтобы контент не сливался с фоном.
5. Не полагайтесь только на цвет, добавляете изменение формы или различные орнаменты, бордеры и тд для людей не различающих цвета
6. Проектируйте использование фокусного состояния на проекте
7. Используйте лэйблы в формах, не ограничивайтесь использованием только плейсходеров
8. Используйте лейблы и альтернативный текс для описания содержимого
9. Используйте правильную разметку, придерживайтесь семантики, НЕ используйте по всюду ДИВ
10. И наконец Поддерживайте навигацию по клавиатуре, некоторые пользователи используют ее даже при работе с планшетами или телефонами.
11. Теперь обратим внимание на динамику, ведь даже, если удается сделать 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 – это и есть динамический элемент, Возникает не мало вопросов.
12. Сегодня я буду ориентироваться работу с библиотекой, если хотите фреймворком – React, которую частно используется для создания динамических интерфейсов. Именно на нее Т.к. я с ней больше всего работал и на примере React 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 атрибутами и по возможности делать компоненты восприимчивыми к расширению. aria в свою очередь расшифровывается, как Accessible Rich Internet Applications - Доступные многофункциональные интернет-приложения и о самих атрибутах и правилами работы с ними можно познакомиться на сайте W3C, ссылочку с боку если кому нужно
14. Теперь представим, что вам повезло и вы попали на новый проект. И На самых ранних этапах разработки возникают вопросы к структуре проекта и его компонентам, Чем же можно вдохновляться или на что ориентироваться на старте, чтобы избежать ошибок и патовых ситуаций ?
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 проекта как минимум расширяемой. <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 в одном компаненте, но это будет выглядят порой как объединение молотка с ножем и им будет банально не удобно пользоваться, лучше иметь несколько инструментов (компонентов) (ТЫК), которые будут работать хорошо и ими будет удобно пользоваться и комбинировать для создания чего-то более сложного.
19. То же касается и дизайна, рассмотрим кнопочку, красивую, но максимально неудобную для любого пользователя, тут одна кнопка, один компонент вобрал в себя чекбокс, саму кнопку, так еще и ссылку на кокой-нибудь поясняющий ресурс, Да дизайнер может нарисовать ее действительно хорошо и она будет визуально приятной, но разделение кликабильных зон будет не удобно в пользовании, можно бонально промахнуться и получить не желаемый результат, а что же касается людей, которые пользуются вспомогательными программами… Для них будет какое-то ужасное и не понятное место в вашем приложении и в лучшем случае такой пользователь будет воспринимать такую 1 кнопку как 3 суверенно самостоятельные элемента и будет переключаться по ним внутри 1 кнопки…
20. Так же следует переиспользовать 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. (ТЫК) Надо ли это большой вопрос?
21. Ведь так мы рискуем раздуть парк компонентов, в каждом из которых нужно будет разбираться, а идеале написать для каждого компонента документацию, Короче говоря каждый рискует стать Гомерром симпсоном и не понимать куда бежать и что происходит , далее
22. **Принцип открытости / закрытости** (open-closed principle) декларирует, что  компоненты должны быть открыты для расширения, но закрыты для изменения.
23. 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.
24. 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.
25. И так на примере SOLID мы понимаем, что можно находить вдохновение и идеи в совершенно разных областях, чтобы постоянно совершенствовать свой проест. Ведь нет ничего идеального, но главное само стремление.  Даже Сальвадор Дали писал, *Не бойтесь совершенства. Вам его не достичь. Тем более, что в совершенстве нет ничего хорошего.* эта цитата и дает нам поение того, что важен именно сам процесс развития, а не конечная цель.
26. До этого мы разобрались со структурой компонентов, но теперь, посмотрим на более конкретные нюансы, которые возникают при работе с доступностью, прейдем от структуры к содержимому
27. Первое, что нужно вспомнить – это то что пользователи используют совершенно различные девайсы: компьютеры, планшеты, телефоны и тд.
28. Для разных устройств и даже операционных систем существуют разные вспомогательные программы, например как VoiceOver на IOS или TalkBack на Android которые увы иногда работают поразному из-за чего возникают специфические баги. (Слайд читаю)
29. И об этом нужно помнить, когда появляется задача обеспечить доступность, часто работа может сводится к поиску универсального решения для большего числа вспомогательных программ и к само собой расширенному тестированию приложения на большем числе конечных устройств.
30. 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. Здесь приведен небольшой список атрибутов, который мы рассмотри поподробнее.
31. Начнем пожалуй с ролей, допустим мы хотим создать таблицу не используя сами теги таблицы, а напрмер Дивы, мы воспользуемся ролями для обозначения самой таблицы, далее ее строк и ячеек, после чего нам нужно добиться чтобы заголовок столбца или строки произносился при попадании на относящуюся к нему ячейку. Для этого существуют роли **columnheader и rowheader** (ТЫК) и на войс овере все заработает - заголовок начнет произноситься при попадании в область ячейки столбаца, однако ТолБэк андройда будет игнорировать заголовки из-за чего может понадобиться отказаться от роли заголовка и просто с помощью ариа-лайблов дублировать заголовок для каждой ячейки, такое решение менее красиво, но будет более универсальным и поэтому более предпочтительным, и все из-зо того что определенный скринридер не поддерживает некоторые роли
32. Очередным примером нюанса с ролями может служить создание табов на основе кнопок и листа, но тут решение с ролями для списка и его элементов не будет работать на андройде, а те же роли для тегов ul и butoon не будут работать на АЙОС
33. Возникают подобные ошибки и-за не нужных промежуточных оберток, в данном случае ей был список, который не нес никакой функциональной нагрузки, можно конечно обнулить роль у листа в, но это является не хорошим тоном при разработке тк если вам приходиться обнулять роль, то вы просто некорректно подобрали теги. Вывод – Избегайте большого числа нефункциональных оберток
34. При работе с ариа-лабел лучше метки сразу держать рядом с основам текстом элементов, например название кнопки и рядом сразу описание действия кнопки.
35. Aria-live

Этот атрибут используется для определения важности изменений, которые произошли в элементе.

То есть значения данного атрибута отражают то, насколько срочно и быстро вспомогательным технологиям нужно сообщить пользователям об этих изменениях. У атрибута есть три значения: off, polite и assertive.

• off (значение по умолчанию) — указывает на отсутствие приоритета, поэтому такие изменения не объявляются.

• polite — обозначает низкий уровень приоритета. Используется в случаях, когда в области происходят изменения, которые вспомогательным технологиям не нужно объявлять моментально. Скринридеры делают перед таким объявлением паузу, не прерывают текущие задачи и ждут, пока пользователь перестанет взаимодействовать с интерфейсом.

• assertive — указывает на наивысший уровень приоритета. О таких изменениях будет объявлено сразу же, а изменения с более низким приоритетом встанут в очередь и будут объявлены позже.

1. Простыми примерами в СПА могут служить оповещения пользователя о новых сообщениях или событиях как в вк, инстограмм и тд.
2. Далее идет aria-atomic. Этот атрибут необязательный и влияет на то, в каком объёме вспомогательные технологии объявят об изменениях: это будет весь контент целиком или только его изменившаяся часть.

У атрибута есть всего два значения — false и true.

false (значение по умолчанию) — значение, при котором вспомогательные технологии сообщат только об изменениях.

true — при этом значении объявляется весь контент, включая изменившуюся часть.

Можно использовать например, для приложения с тестами по английскому языку, где с помощью радио кнопки выбирается слово, которое нужно вставить в предложение, после чего с помощью атрибута aria-atomic выбирается, будет прочитываться пользователю все предложение со вставленным словом или только само вставленное слово.

1. Можно использовать для счетчика длинны, веса и тд., где к значению добавляется размерность, и с aria-atomic= true, пользователю будет прочитываться например 120 сантиметров, а не только само число.
2. Aria-busy

aria-busy="" даёт вспомогательным технологиям знать, обновляется ли сейчас содержимое элемента или нет.

У aria-busy="" есть два значения — false и true.

false (значение по умолчанию) — при этом значении вспомогательные технологии не ждут, пока изменения завершаться.

true — это значение говорит вспомогательным технологиям, что им нужно подождать, пока элемент не закончит изменяться, после чего они могут собрать все изменения и сделать объявление. То есть во время обновления контента пользователи скринридеров, например, не смогут читать эту обновляющуюся часть.

1. Атрибут имеет смысл применять, когда на странице происходит автообновление контента. И обо всех изменениях в элементе нужно сообщить только один раз. Это полезно в случаях, когда в приложении есть например статистика, которая обновляется в режиме реального времени
2. Aria-labelledby / aria-describedby

Два схожих атрибута, которые связывают контент (метку) с элементом (картинка, форма и тд) через id.

Но все же у них имеются отличия, Aria-labelledby используется как небольшое описание элемента, довольно краткое и может принимать в себя несколько id меток в порядке чтения, в свою очередь aria-describedby может иметь достаточно большим текст и является дополнительным описанием, которое будет прочитываться в самом конце и как правило после большое паузы.

1. Можно использовать для привязки описания к картине, а так же как дополнение к альтернативному тексту, с помощью aria-describedby
2. aria-hidden Атрибут отвечает за видимость элемента для вспомогательных технологий. aria-hidden="true" говорит о том, что элемент не должен быть видимым. Этот атрибут должн применяться только к элементам, которые мы хотим скрыть и не являются интерактивными
3. . В СПА довольно часто приходиться прибегать к этому атрибуту, чтобы пользователь не попадал скринридером на не нужные или невидимые элементы.
4. Далее Несколько нюансов не связанных с ариа атрибутами, например В SPA пользователь может не понимать переключился ли экран после нажатия на кнопку перехода, для решения такой и подобных проблем можно программно ставить фокус на заголовок экрана сразу же после его отрисовки, вот приведен пример кода, который позволит манипулировать фокусов в React приложении, пример взят из документации самого React.
5. Еще один не маловажный нюанс – это работа с событиями, как правило эвенты реализовываются с помощью события click объекта window и Такой подход хорош для тех, кто использует мышь, тачпад или другое координатное устройства, однако для пользователей, работающих только с клавиатурой, он может стать причиной нарушения работы приложения, например при работе с выпадающим списком при попытке перехода к следующему элементу с помощью клавиши Tab. Причиной этого является то, что в данной ситуации объект window никогда не получит событие click. В итоге мы имеем заблокированную функциональность и не можем продолжать работы с приложением. Такие проблемы можно решать используя обработчики сопутствующих событий, например, onBlur или onFocus и с помощью этих событий уже можем контролировать поведение списка. Это решение будет работать и для клавиатуры, и для любого координатного устройства.
6. И вот мы, вроде закончили работу, у нас якобы все доступно для пользователей с ограниченными возможностями, но… (ТЫК) Теперь нужно подумать, а не избыточно ли часть того, что мы сделали, не будет ли пользователь перегружен информацией, которая ему в принципе и не нужна и только будет замедлять его взаимодействие с приложением, и вспомнить что..
7. Wcag регламентирует доступность для ВСЕХ пользователей, в том числе пользователей, которые не используют, а может и не слышали про вспомогательные технологии, возможно, логика которую вы добавили может мешать пользователям вообще не использующих режимы доступности, особенно если различные метки генерируются динамически с множеством проверок, которые могут замедлять загрузку приложения и не только. Поэтому следует по возможности использовать то, что уже реализовано, привязываться к готовой логике и не перегружать приложение дополнительным функционалом. И аккуратно внедрять то, что нам требуется для дступности, ища возможность никаким образом не ухудшить работу приложения для всех пользователей.
8. Одним способом решения проблемы может 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 делали ваш проект расширяемым, то вы избежите множества багов в будущем и вам будет гораздо легче дополнить функционал проекта, и не придется переписывать огромное количество кода, и думать о том, что вместо исправлений проще сделать отдельный режим, чтобы все не ломать и не перестраивать весь проект. Но есть еще и этические вопросы к данному подходу, не будет ли создание отдельного режима или версии для пользователей с ограниченными возможностями дискриминацией. Довольно трудный вопрос, поэтому от данной идеи лучше отказаться.
9. Подытожим! Сегодня мы с вами повторили базовые принципы доступности, рассмотрели подходы к организации структуры компонентов, разобрали специфические нюансы работы со скринридерами, динамическими элементами и SPA в целом.
10. И Наконец, Придерживайтесь семантики, не используйте только дивы; Тщательно прорабатывайте структуру компонентов; Тестируйте, много тестируйте и на большем количестве устройств; Используйте ариа атрибуты
11. И конечно же Общайтесь, делитесь знаниями.
12. Говорите о доступности
13. Спасибо за внимание
14. 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 ))