

ПРО

доступ. ночь iOS

Михаил Рубанов



Что такое доступность?

Доступность приложения – это адаптация интерфейса для разных групп людей. Если человек хуже видит (или совсем потерял зрение), не слышит, не двигается, но при этом полноценно пользуется приложением, то оно доступно.

- Не знаете английский, но хочется посмотреть новый сериал на Netflix? Есть субтитры, а часто и полноценный перевод на русский.
- Не можете перехватить телефон одной рукой, чтобы нажать кнопку назад, потому что в другой руке держите ребенка? Можно свайпнуть от края экрана.
- Перед сном читаете книжку? Темная тема не будет слепить.

Это некая «стандартная» доступность для всех людей. Вроде можно прожить и без этих фич, но с ними удобнее. Если каждый из этих примеров усилить, то он поможет людям с проблемами со здоровьем, для кого эти «фичи» становятся жизненно необходимыми.

- Пожилые люди могут пользоваться телефоном без очков, увеличив шрифт, или включив экранный зум.
- Аудиодескрипция вместо субтитров расскажет незрячему, что происходит на экране, а вернуться на экран назад поможет специальный жест в VoiceOver.
- Темная тема поможет людям со светобоязнью.
- Вибрация на часах помогает узнавать о звонке тем, кто не слышит.

А для тех, кто совсем неподвижен, одно лишь движение щеки позволит контактировать с миром: отдавать телефону команды «выбери» и «следующий элемент». Возможно, это поможет кому-то написать книгу об устройстве вселенной.

Я расскажу о самых сложных нарушениях, в которых даже не сразу понятно, как вообще можно пользоваться телефоном с сенсорным экраном. Может показаться, что незрячему удобней пользоваться телефоном с кнопками, но на самом деле мир изменился после выхода iPhone 3GS. Подробней про это можно прочитать в статье [A Timeline of iOS Accessibility: It Started with 36 Seconds](#)

Зачем заниматься доступностью?

Я, как мобильный разработчик, делаю интерфейсы для людей: удобные, красивые и простые. В этой работе много знаний: и законы взаимодействий с интерфейсами, и восприятие информации, и влияние на эмоции. При этом вся индустрия пытается максимально охватить пользователей – многие приложения работают на всю страну или весь мир.

Какие свойства у хорошего интерфейса? Он понятный, отзывчивый, красивый, быстрый и... доступный для всех.

Можно ли назвать качественным недоступный интерфейс? Только в редких случаях. Например, если вашим продуктом пользуется узкий круг людей и вы точно уверены, что они здоровы «как космонавты».

Для популярных приложений картина другая: из миллионов пользователей очень много людей с нарушениями. Иногда ограничения небольшие, иногда серьезные, но все меняет образ жизни. Раньше такие особенности могли создавать непреодолимые препятствия для жизни человека, но сегодня она может быть комфортной, насыщенной и интересной, благодаря технологиям, что есть у каждого человека в кармане.

Самое крутое, что для этого не нужно делать очень много. Айфон уже содержит все нужные инструменты: синтезатор речи, мультитач-дисплей, трекер головы и несколько модулей iOS, обеспечивающих доступность. Вам нужно им только немного помочь.

Почему это полезно для разработчиков?

Адаптация приложения для незрячих помогает понять что такое «интерфейс». Она показывает сложность мира и путей взаимодействия: люди не только жмут на экран, но и могут управлять с клавиатуры, голосом, жестами. Качественный интерфейс всё это обеспечивает.

Превращая графический интерфейс в звуковой лучше понимаешь, как работает наше восприятие. Мы быстро считываем ментальную модель с экрана, а дальше уже работает наше понимание этой модели. Тот, кто умеет работать с информацией на таком уровне может раньше найти проблемы и в графическом дизайне. Глубокое понимание взаимодействия учит делать лучше, толкает вас в правильном направлении и, в итоге, выигрывают все пользователи.

Почему это важно для компаний?

Если продуктом пользуются миллионы, то среди ее пользователей тысячи людей с ограничениями. Для них сервис может быть жизненно важным, но недоступным. Приложения влияют на всю жизнь человека и на то, что он в ней может сделать.

Не всё измеряется цифрами и прибылью. Любая крупная компания несёт ответственность за свое влияние на людей: она может помогать, мешать, создавать праздник, банить политиков. Всё ИТ сейчас отвечает за то, как мы живем.

Адаптируя приложения и сайты, мы даем незрячим людям способ взаимодействовать с миром: они так же заказывают еду на доставку, ездят на такси, оплачивают счета, общаются, читают новости и соцсети, ходят на работу. Не адаптируя приложения, мы забираем кусочек их жизни и функционала.

Приложения и сервис незрячим даже нужнее, чем всем остальным. Представьте, насколько сложно сходить в магазин: идёте с тростью до магазина, выбираете продукты (как?), возвращаетесь обратно. Можно в тысячу раз проще – заказать доставку через приложение. Если приложение адаптировано, то легко узнать состав, ассортимент, указать адрес, оплатить и купить всё, что хотите.

При этом, доступность – это не фича, её нельзя измерить сроком, сделать один раз и никогда больше не возвращаться. Доступность в головах: это набор правил, которые разработчики и дизайнеры должны учитывать, когда создают новый интерфейс.

Доступность – это навык и ему можно научиться

Поддержка доступности – хорошая метрика культуры компании. Все крупные иностранные приложения Эпла, Гугла, Фейсбука адаптированы и работают очень хорошо. Почти все крупные российские – не адаптированы совсем или содержат критичные ошибки.

Создавайте приложения для людей. Для всех.

Приступим?

Версия для незрячих? Нет

В вебе часто можно встретить отдельную версию для незрячих или слабовидящих, в мобиле я такого не встречал. Видимо, никому в голову не приходит написать второе приложение для особого случая, но тему обсудим.

Допустим, мы решили написать приложение специально для незрячего. Первое, что надо понять: а как такую версию писать?

Технически – это адаптация контроллов с помощью `UIAccessibility`. Функционал нужен в том же объеме, что и у обычной версии, при этом неясно, как должен поменяться графический интерфейс. Получается, что есть две версии: одну мы не можем прочитать через `VoiceOver`, вторую не можем нарисовать. Но и там, и там, мы всё равно работаем с `UIKit`.

Крутость технологий доступности в том, что они берут много данных от существующего графического интерфейса. Они выступают над ним неким «вторым слоем», давая новые возможности работы с интерфейсом: воспринимать на слух, управлять голосом или иначе сообщать сигналы, если не можешь коснуться экрана.

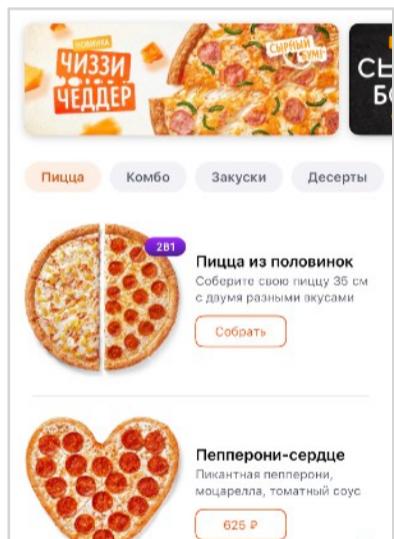
Иногда у доступности не получается правильно понять, что происходит на экране и нужно немного помочь. Это немного кода, зачастую, лишь несколько дополнительных меток, которые позволяют работать `VoiceOver` с приложением как надо: сообщать о выбранном состоянии интерфейса, правильно называть все элементы на экране, поправить порядок чтения элементов.

При этом, крайне редко приходится писать какой-то специальный текст для незрячих. Надо лишь правильно оформить существующие контроллы.

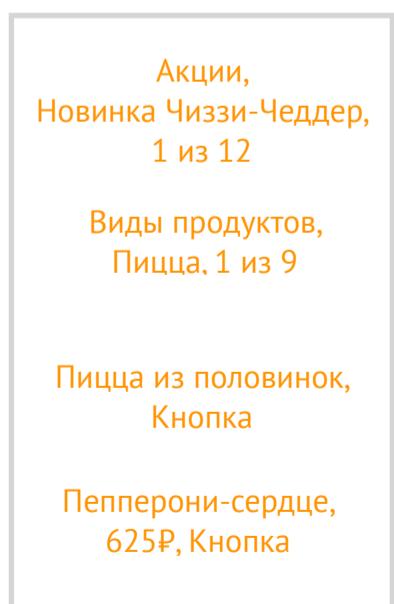
Ментальная модель

Не адаптируя доступность, мы работаем в режиме «анимированного макета с данными», копируя только внешние атрибуты, уверяя себя, что так и надо. Адаптируя VoiceOver, мы работаем с ментальной моделью восприятия информации: это другой уровень понимания интерфейса и его сложности.

Ментальная модель не меняется от способа взаимодействия – хорошая модель прекрасно адаптируется и для звукового интерфейса. У аудио-интерфейса может быть другая скорость считывания, другие жесты взаимодействия, но суть остаётся той же самой. Не делайте отдельные версии для людей с нарушениями – делайте нормально для всех.



Графический интерфейс



Аудио-интерфейс

Сверху акции,
потом виды продуктов,
в конце список

Ментальная модель



Человек

VoiceOver



Как работает VoiceOver

Перед тем как заняться адаптацией приложения, мы должны понять, как незрячие люди пользуются телефоном и что мы должны сделать, чтобы им стало удобней.

Для незрячего экран телефона превращается в сенсорную панель: по ней можно свайпать в разные стороны, тапать на нее одним или несколькими пальцами. В ответ на действия VoiceOver озвучивает текущее состояние интерфейса: на каком элементе мы сейчас находимся, что с ним можно сделать.

Чтобы лучше понять, попробуем пройти всю эволюцию звукового интерфейса с ноля: расскажем о содержимом экрана, отметим элементы, с которыми можно взаимодействовать, подумаем, как ускорить навигацию.

Как работает VoiceOver

Звуковой интерфейс

Ситуация: вам нужно сделать новый экран для приложения, а макета у вас нет, интернета тоже нет. Вы звоните дизайнеру и он вам описывает картинку по телефону. Что он расскажет?

Скорее всего, он перечислит элементы по порядку, даст им название и расскажет об особенностях: заголовок «Пицца», надпись с размером теста, кнопка «купить». Из рассказа вам будет понятен порядок элементов, что они обозначают, в каком состоянии находятся, что с ними можно сделать. Вы готовы работать с этой информацией.

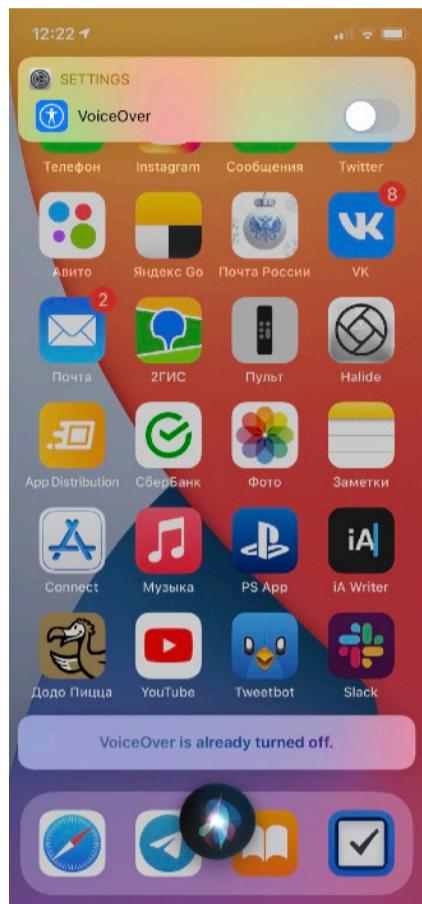
VoiceOver выступает в роли такого друга. Он берёт один элемент, читает его описание, рассказывает о состоянии (выбрано, отключено), говорит какой это тип и что с ним можно делать. В ответ вы можете попросить перейти к следующему элементу, нажать на кнопку или закрыть текущий экран. Команды отдаются разными свайпами на экране.

VoiceOver всеми силами старается уменьшить количество дополнительной работы. Не делайте специальных версий для незрячих, ваша задача лишь помочь VoiceOver назвать всё правильно и обработать ввод. Крутость технологии в том, что работы не так и много: синтез речи уже встроен в iOS, мультитач экрана распознает сложные жесты. Просто подправьте читаемый текст, тип элементов и обработайте несколько дополнительных функций.

Перед адаптацией приложения надо разобраться в деталях работы VoiceOver, научиться ей управлять, посмотреть, как она работает на примере стандартных приложений. Дальше, с пониманием звукового интерфейса, мы приступаем к адаптации своего приложения.

Как работает VoiceOver

Включение



Чтобы понять, как адаптировать интерфейсы, научитесь работать с ними в таком же режиме, как это делают незрячие. Это просто, включите на телефоне VoiceOver и попользуйтесь.

Перед включением важно **научиться отключать**. В самой сложной ситуации вас точно выручит Siri: «Привет Сири, выключи VoiceOver». Чтобы не натолкнуться на выключенную Сири, первое время включайте VoiceOver только через неё.

Включить VoiceOver можно еще несколькими способами:

- через Настройки,
- шорткат для доступности,
- бэк-тап по спинке телефона.

The first screenshot shows the main Settings menu with 'Универсальный доступ' selected. The second screenshot shows the 'Универсальный доступ' settings screen with 'VoiceOver' turned off. The third screenshot shows the 'VoiceOver' settings screen where the toggle switch is turned on.

Настройки

- Основные
- Пункт управления
- Экран и яркость
- Экран «Домой»
- Универсальный доступ**
- Обои
- Siri и Поиск
- Face ID и код-пароль
- Экстренный вызов — SOS
- Уведомления о контакте
- Аккумулятор
- Конфиденциальность
- App Store
- Wallet и Apple Pay

Универсальный доступ

Функции Универсального доступа помогают настроить iPhone для Ваших индивидуальных нужд.

ЗРЕНИЕ

- VoiceOver** Выкл.
- Увеличение Выкл.
- Лупа Выкл.
- Дисплей и размер текста
- Движение
- Устный контент
- Аудиодескрипция Выкл.

ОПОРНО-ДВИГАТЕЛЬНЫЙ АППАРАТ

- Касание
- Face ID и внимание
- Виртуальный контроллер Выкл.
- Управление голосом Выкл.
- Боковая кнопка
- Пульт Apple TV

VoiceOver

VoiceOver произносит названия объектов на экране:

- Коснитесь для выбора объекта.
- Коснитесь дважды для активации выбранного объекта.

Тренировка жестов VoiceOver

СКОРОСТЬ РЕЧИ

Речь

Брайль

Распознавание VoiceOver

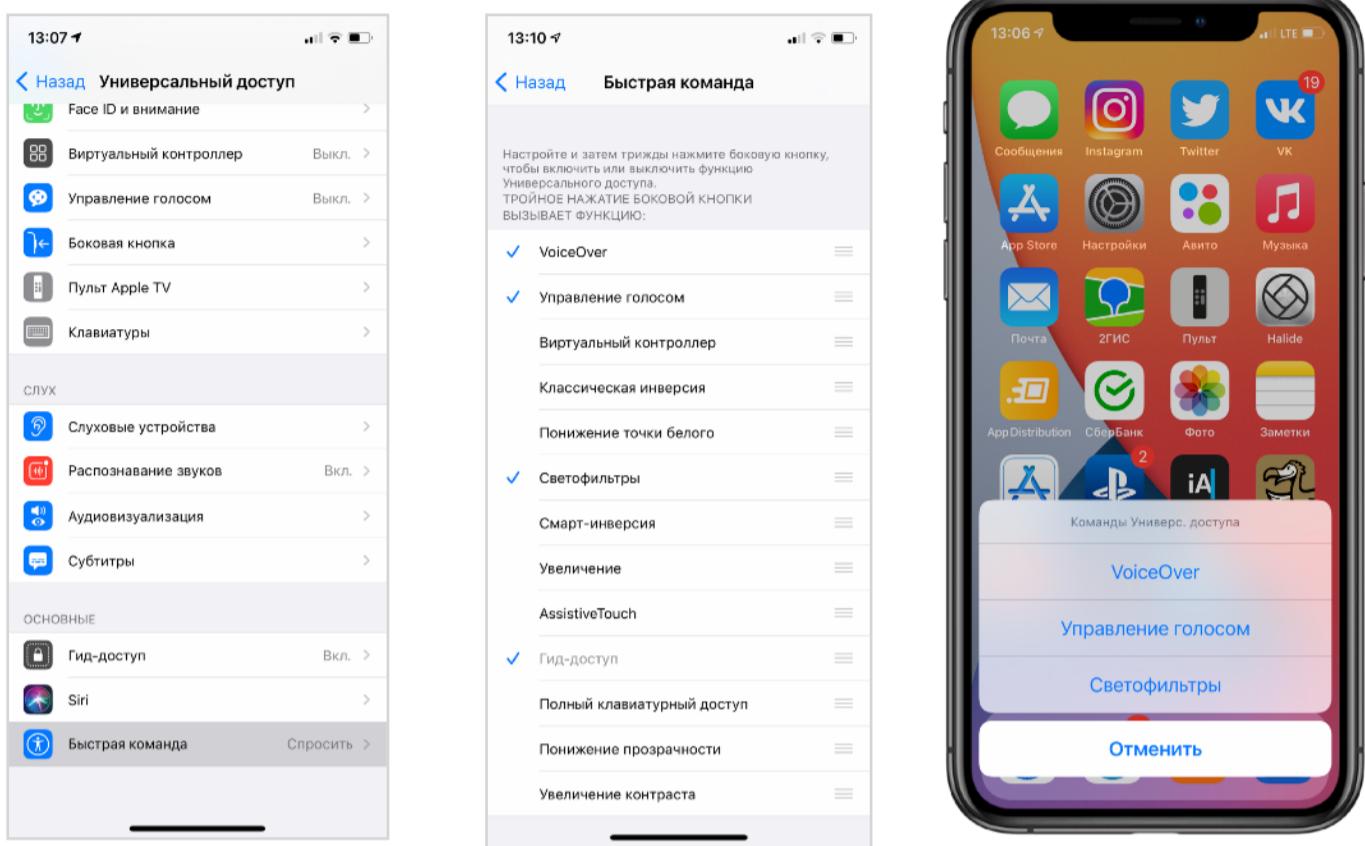
Детализация

Аудио

Команды

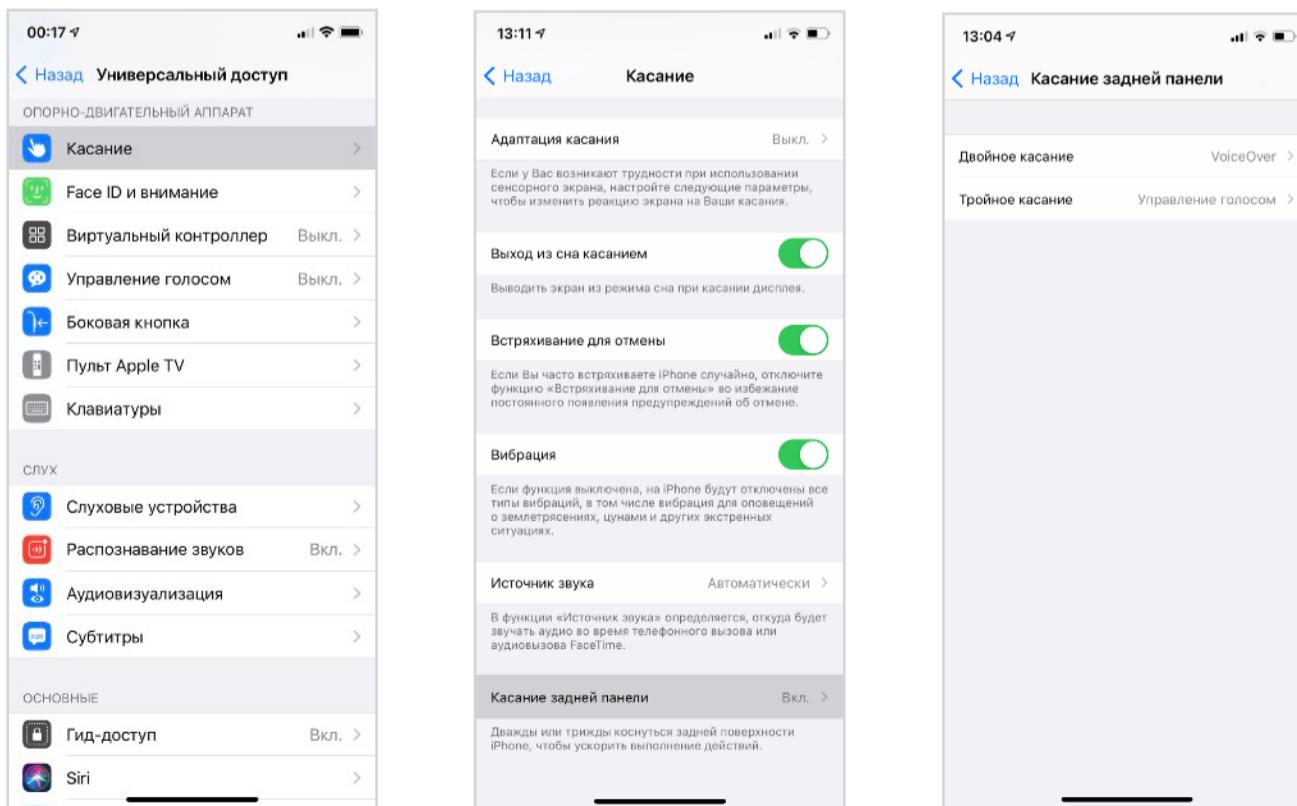
Настройки, VoiceOver, Кнопка-переключатель, вкл., Коснитесь д

Параметры VoiceOver находятся в разделе «Универсальный доступ» в настройках телефона. Незрячие включают высокую скорость речи, им так удобней.



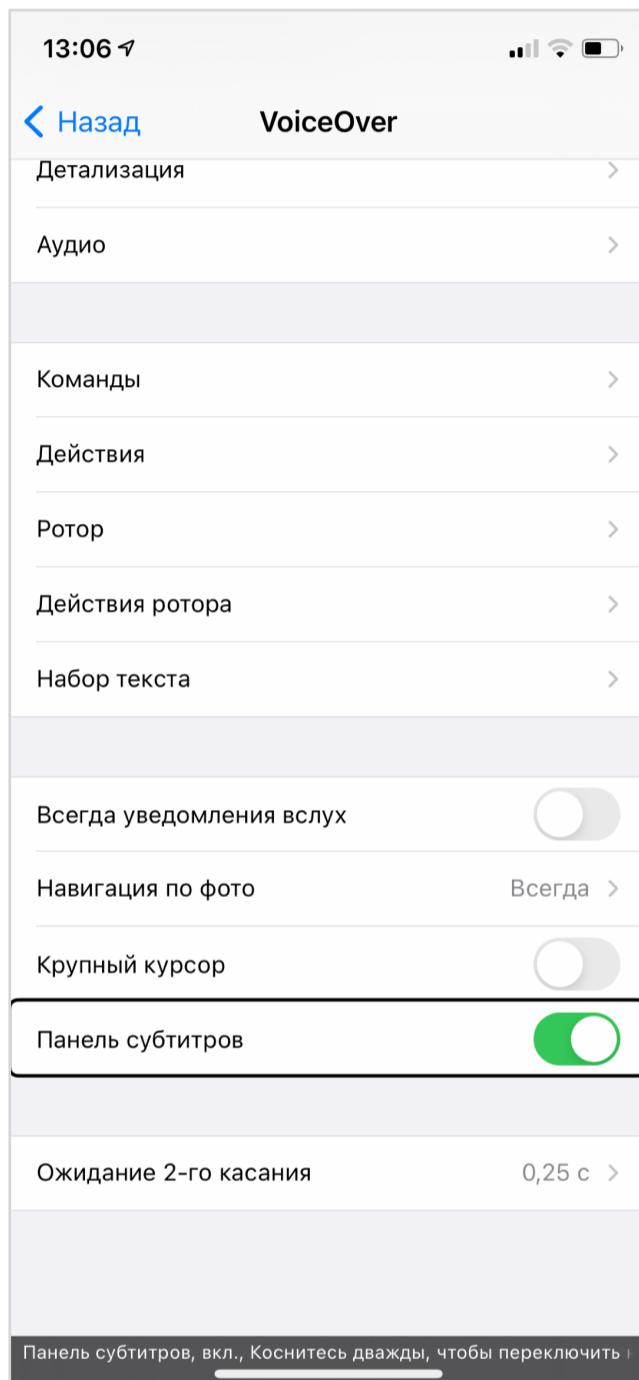
Разные варианты доступности можно активировать по тройному нажатию на кнопку включения. Ищите «Быстрые команды» в конце экрана «Универсальный доступ».

Удобно настроить включение VoiceOver на двойной (или тройной) тап по спинке телефона. Я пользуюсь этим способом чаще всего.

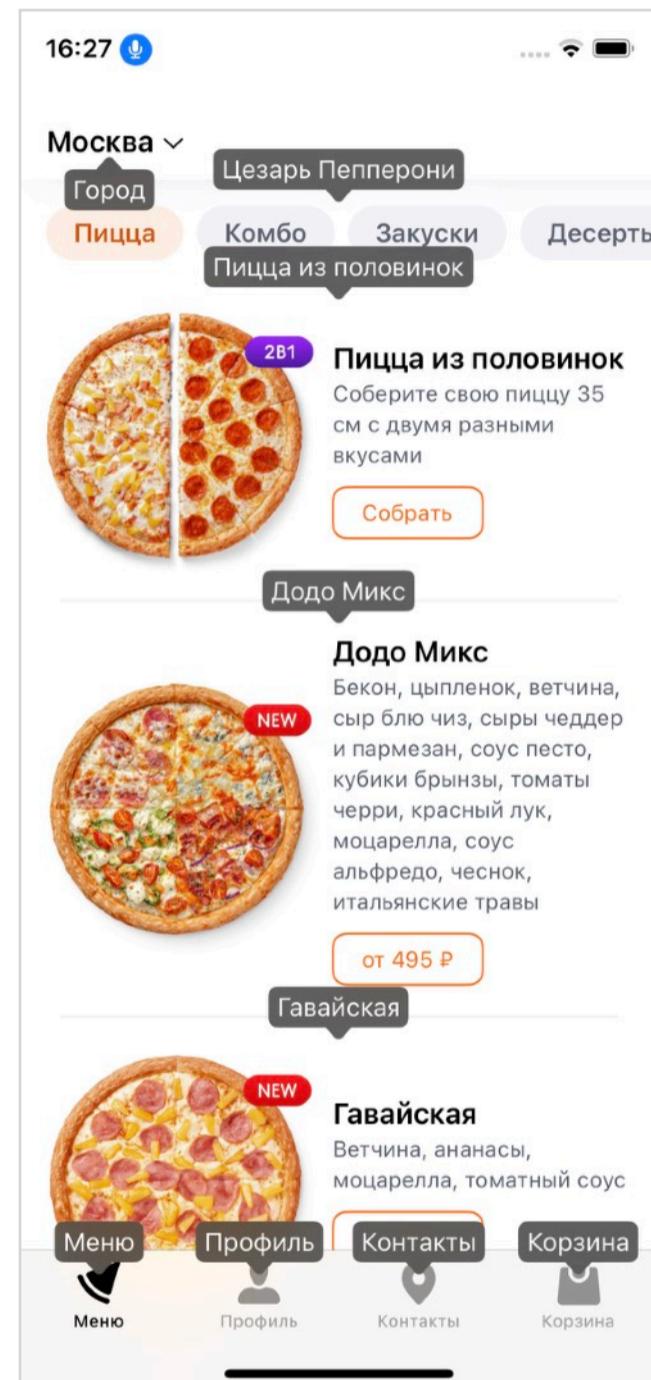


Для отладки удобно не слушать VoiceOver, а смотреть текст, который он читает. Включив субтитры, вы увидите описание текущего элемента.

Описания можно смотреть и через Voice Control – он покажет все элементы за раз, но про него в отдельной главе.



Панель с субтитрами



Voice Control

Панель субтитров, вкл., Коснитесь дважды, чтобы переключить

Пример субтитров

Как работает VoiceOver

Навигация

При включении VoiceOver на экране появится черная рамка – это фокус. Фокус перемещается по элементам интерфейса от свайпов по экрану и читает описание нового элемента сразу после перемещения.

У такого интерфейса сразу несколько следствий:

- Работать можно только с одним элементом за раз.
- Все действия с экрана передаются элементу в фокусе.
- Не надо прицеливаться на элемент. Свайп в любом месте экрана будет влиять на элемент в фокусе.
- Если хотите прочитать описание ещё раз – тапните по экрану.

Свайпните вправо, чтобы перейти к следующему контролю, элементы переключаются в порядке чтения. Свайп влево переключит фокус на предыдущий элемент.

Контролы на экране

Фокус на первой надписи

Контролы на экране

После свайпа вправо фокус переключился в порядке чтения

Если фокус дошел до последнего элемента в строке, то по свайпу вправо он сам переключится на следующую строку. Всё ровно так, как мы читаем текст.

Контролы на экране в две строки

Эта система логична, но в мобилке есть один нюанс: чаще всего в строку помещается только один элемент, поэтому при свайпе вправо VoiceOver переключается на следующую строчку.

Контролы в мобиле

Начальное положение фокуса

Выходит, что чаще всего свайп вправо переключает на следующий элемент *вниз*, а свайп влево – на предыдущий элемент *вверх*. Непривычно, но теперь вы знаете почему так.

Контролы в мобиле

По свайпу вправо фокус переходит на следующую строчку

Перемещать фокус можно не только свайпами, но и «касанием»: водите палец по экрану, а VoiceOver будет читать элемент, который находится под вашим пальцем. Свайпами пользуются намного чаще, но изучение касанием – важный помощник в повседневных задачах.

Оба способа будут упоминаться дальше в книге, поэтому запомните их официальное название: **навигация смахиванием** и **изучение касанием**.

Действия

Одиночный тап по экрану выполняет самое важное действие – **читает описание элемента**. Используется он не часто, ведь описание читается при смене фокуса автоматически.

Нажмите на кнопку тапнув дважды в любом месте экрана – действие передастся элементу в фокусе.

Попробуйте двойным тапом открыть программу. Если у вас айфон «с челкой», то она закрывается привычным свайпом снизу, только чуть медленнее: почувствуйте лёгкую вибрацию, после этого отпустите палец и программа свернётся.

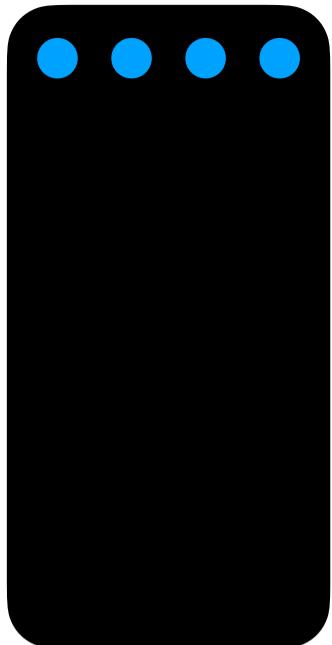
Прокрольте экран свайпнув тремя пальцами в нужном направлении. Полистайте главный экран айфона со значками.

Если хотите погрузиться во взаимодействие незрячих людей с интерфейсами, то **отключите экран**, тапнув трижды тремя пальцами. Включается тоже так.

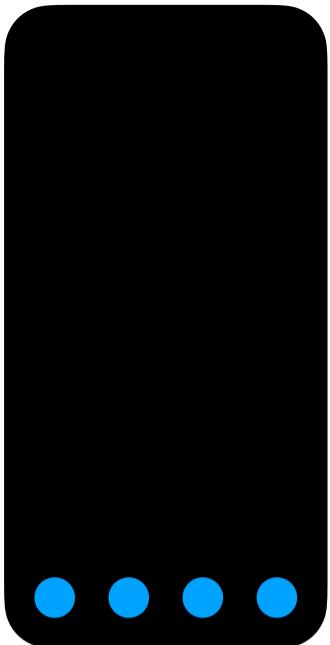
Примечание. В некоторых случаях ещё работает вертикальный свайп, но про это расскажу позже в главах «Элемент регулировки», «Контекстные действия» и «Ротор».

При знакомстве с новой программой, часто нужно **прочитать все элементы** на экране. Если свайпнуть двумя пальцами вверх, то VoiceOver прочитает все элементы с начала экрана, а если свайпнуть вниз – то начиная с текущего.

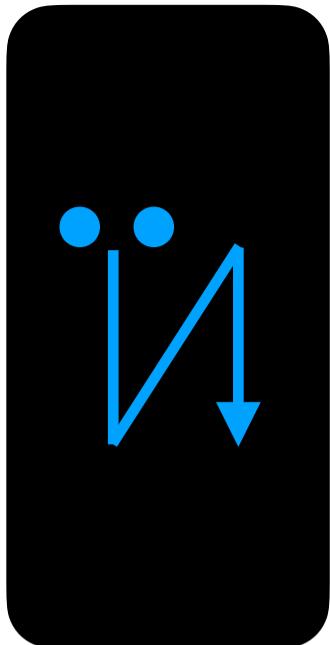
Мы познакомились с несколькими базовыми жестами, но у VoiceOver их намного больше. Несколько популярных можно посмотреть на следующей странице, а полный список [в гайде](#).



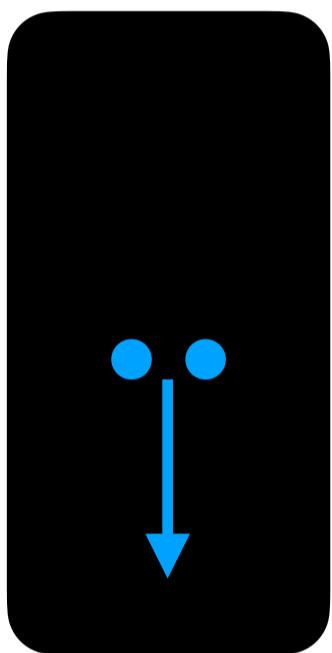
К первому элементу



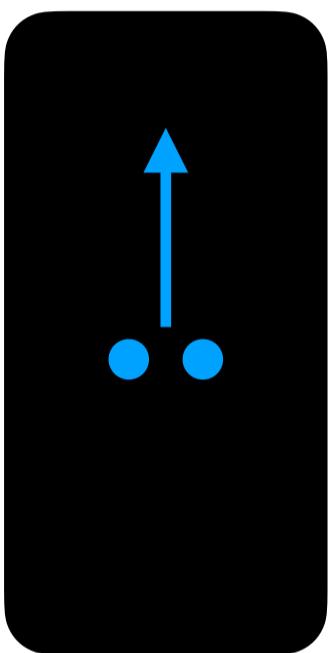
К последнему



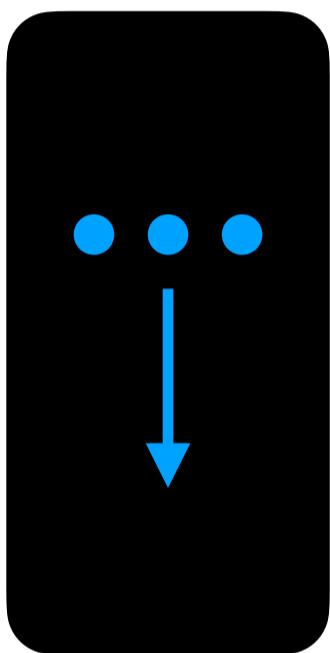
Выйти с экрана



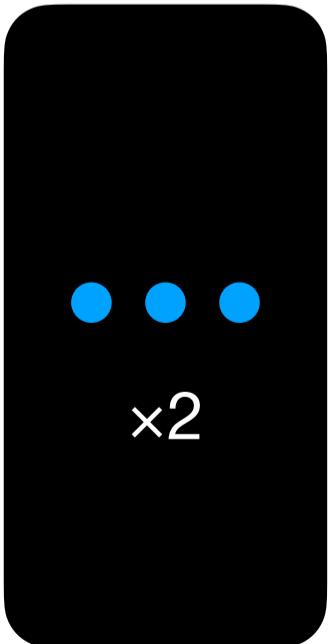
Читать все с текущего



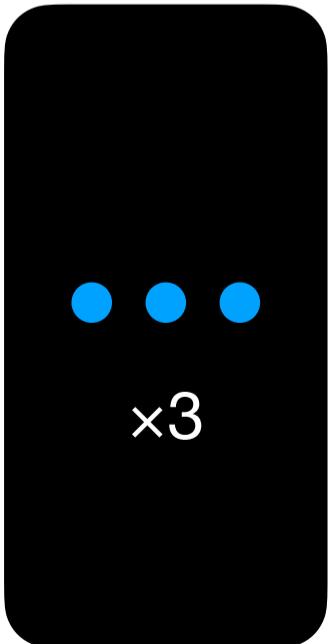
Читать все с первого



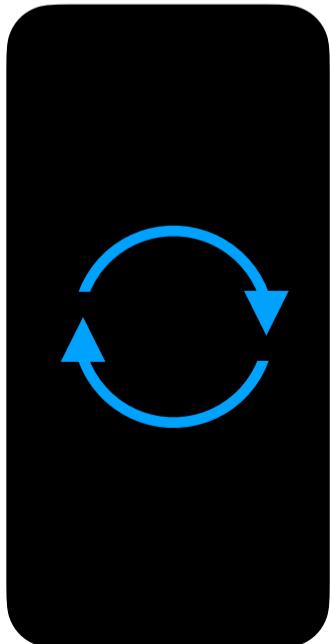
Свайп вниз



Заглушить
VoiceOver



Отключить экран



Ротор

Как работает VoiceOver

До и после

Чтобы понять разницу в значимости адаптации, посмотрим на интерфейс так, как им пользуются незрячие люди. Возьмём экран, который разработчик просто сверстал и ничего не делал для доступности, а потом сравним с тем, как этот же экран воспринимается после адаптации.

Я запускаю приложение и оказываюсь на каком-то экране. Чтобы понять, что на нём, свайпаю двумя пальцами вверх и VoiceOver читает все элементы на экране. Я могу пройтись по ним вручную, но это долго, ведь на экране их больше 22.

Прочитайте. Понимаете на каком экране находитесь? На какие элементы можно нажать?

Москва, кнопка

В ресторане, выбрано, 2 из 2.

Москва, улица Наметкина, 13Б, кнопка

Чиззи чеддер

Сырный бортик

Какао с маршмеллоу

3 за 999 ₽

Втройне сырная

Работать с нами

Пицца

Комбо

Закуски

Десерты

Напитки

Другие товары

Пицца из половинок

Соберите свою пиццу за 35 см с двумя разными
вкусами

Собрать, кнопка

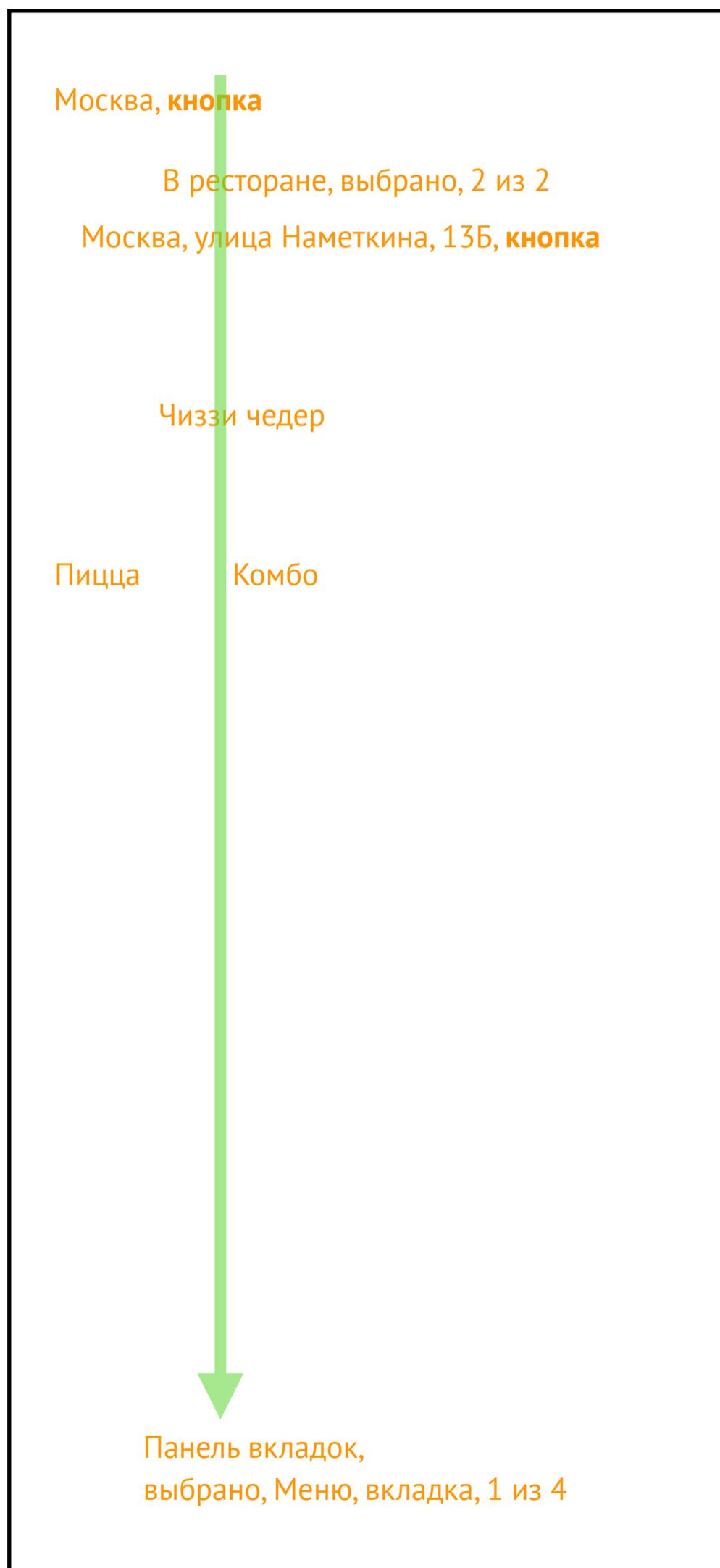
Пепперони-сердце

Пикантная пепперони, моцарелла, томатный соус

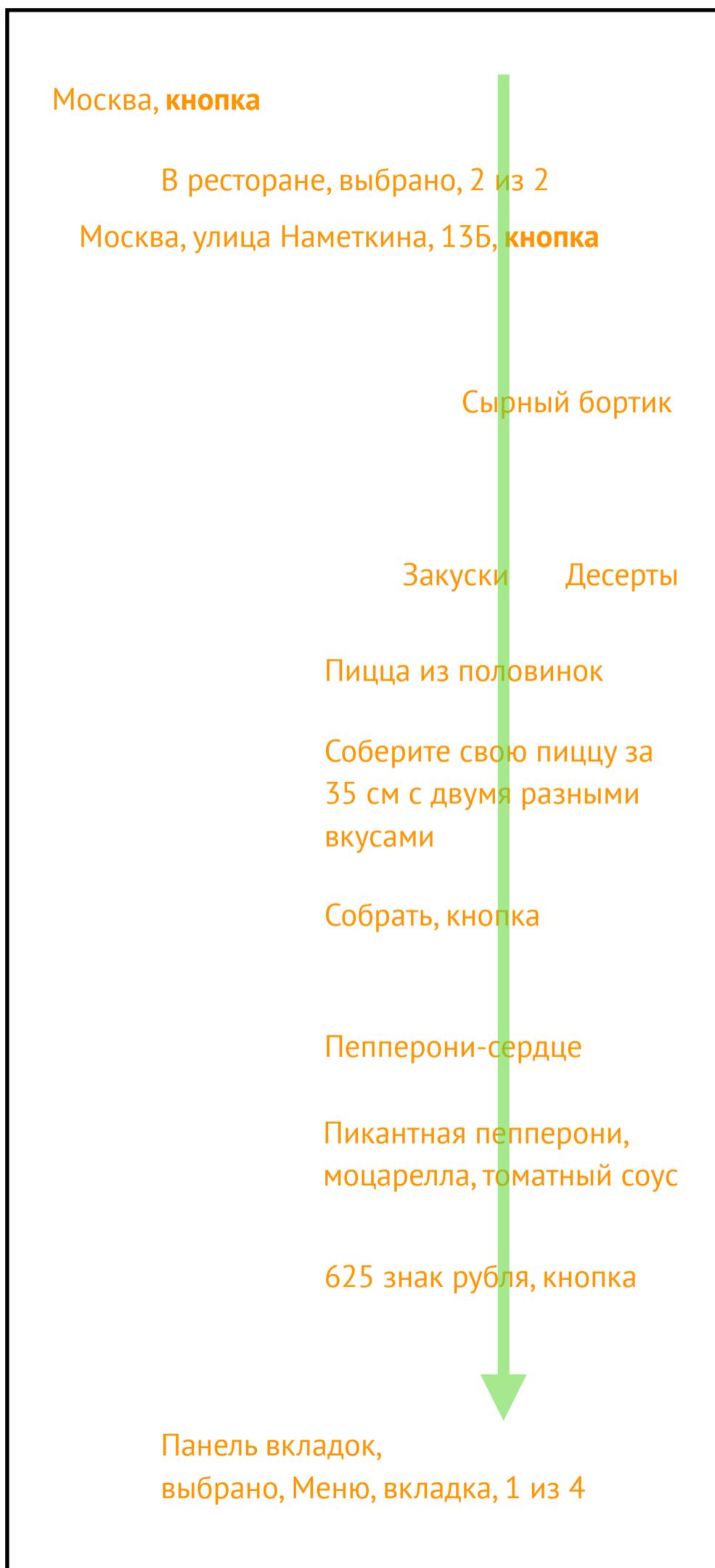
625 знак рубля, кнопка

Панель вкладок, выбрано Меню, вкладка 1 из 4

Элементов очень много. Если я проведу пальцем по левой половине экрана, то тоже получится странно: элементов вдруг стало очень мало.



Если провести по правой половине экрана, то найдутся новые элементы. Вопросов к экрану всё равно много: что такое чиззи чеддер и сырный бортик? Что такое пицца-комбо-закуски-десерты? Я могу на них нажать? Что произойдет?



Москва,
кнопка

Тип заказа,
В ресторане, 2 из 2
элемент регулировки

Адрес
Москва, улица Наметкина, 13Б,
кнопка

Акции,
Чиззи чеддер, 1 из 7,
элемент регулировки

Раздел меню
Пицца, 1 из 7,
элемент регулировки

Меню

Пицца из половинок,
Соберите свою пиццу за 35 см с двумя
разными вкусами,
кнопка

Пепперони-сердце, 625 рублей
Пикантная пепперони, моцарелла,
томатный соус
кнопка

Панель вкладок,
выбрано, Меню, вкладка, 1 из 4

Теперь сравним с адаптированной
версией этого экрана.

Контролов немного, у всех указан
тип, понятно как
взаимодействовать с ними,
описание сгруппировано и
интонация разная (выделил
курсивом и жирным шрифтом).

Кнопки можно нажать, тапнув по
экрану дважды, элементы
регулировки можно менять
вертикальными свайпами,
подписаны области на экране (тип
заказа, акции, меню, панель
вкладок).

По такому тексту можно понять, что
перед нами экран меню. При этом,
мы не сделали ничего сложного:
только дали всем элементам тип,
немного подправили описание и
сгруппировали элементы.
Понятность и удобство для
незрячего выросло в разы.

Количество элементов сильно
сократилось с 22 до 8, при этом
перемещаться между ними можно
как свайпами, так и касанием, в
любом случае, это будет удобно.

Так графически выглядит экран меню. Много элементов и видов контроллов, но легко понять из каких блоков он состоит. Сравните, насколько этот экран похож на текстовую версию по восприятию. Ментальная модель одна, а представления разные.

18:02 1

Москва ▾

На доставку В ресторане

Москва, Главный офис Додо, Ленинска... >

Цыпленок
БЛЮ ЧИЗ
НОВИНКА

СЫРНЫЙ БУМ!

Пицца Комбо Закуски Десерты

2в1

Пицца из половинок
Соберите свою пиццу 35 см
с двумя разными вкусами

Собрать

Пепперони-сердце
Томатный соус, пикантная
пепперони, моцарелла

625 ₽

Меню Профиль Контакты Корзина



Помните, в неадаптированной версии было
22 элемента?

После адаптации 22 элемента – это
четыре с половиной экрана.

Как работает VoiceOver

Анатомия фокуса

Для работы VoiceOver нужен контрол, на котором можно сфокусироваться. Как он его находит?

В первую очередь, мы обращаемся к корневому вью и спрашиваем: есть ли у него **доступные элементы**? Чтобы ответить на вопрос, `UIView` реализует протокол `UIAccessibilityContainer` и может пройтись по всем своим дочерним элементам, чтобы спросить у них, доступны ли они. Маркер простой: `isAccessibilityElement` должен быть `true`.

Если элемент недоступен, то, может быть, он работает как **контейнер** для других доступных контроллов? Тогда надо у всех по очереди спросить, есть ли у них элементы. Вложенность может быть большой, но в конце только два варианта: либо сама `UIView` доступна, либо дочерних элементов больше нет совсем.

Представим, что мы нашли доступный элемент и решили вокруг него нарисовать рамку. Для рамки нужны **координаты и размер элемента**, причем в координатах экрана, чтобы можно было по касанию на дисплей попробовать найти этот элемент среди иерархии.

Доступный контрол мы нашли и обвели. Теперь расскажем о нём.

Описание хранится в `accessibilityLabel`. Если у элемента есть какое-то **значение**, то оно хранится в `accessibilityValue`, оно читается после короткой паузы и немного другим голосом. Это создает динамику в речи VoiceOver.

У контрола может быть одно из стандартных свойств:

- **тип** – например, надпись или кнопка;
- **состояние**: обычное, выбранное, отключенное;
- или **особое свойство**: часто обновляется, начинает проигрывать медиа и т.п.

Обычно, свойство добавит текст к описанию элемента и как-то меняет поведение работы VoiceOver.

Вы поняли, что это за элемент, что это кнопка и вы хотите на неё **нажать**. Нажать кнопку можно двойным тапом, он вызовет специальный метод `accessibilityActivate()`.

Для кнопки этот метод эмулирует касание, при этом нажимает **на точку активации**, которая указана в `accessibilityActivationPoint`. Обычно это центр кнопки.

Если после нажатия открывается новый экран, то он **оповещает** VoiceOver о своем появлении и ставит фокус на свой первый элемент. Цикл повторяется.

Алгоритм простой, но его сила в том, что он может работать со всеми интерфейсами приложений и только в играх нужно придумывать что-то особенное. Другие технологии работают похоже: Voice Control возьмет ту же информацию, чтобы показать какие кнопки можно активировать голосом, Switch Control использует тот же фокус, просто дает другой способ управления фокусом. Если появится интерфейс управления через нервные импульсы, то работать он тоже будет через `UIAccessibility`, зуб даю.

Ваша задача – дополнить интерфейс данными, чтобы помочь алгоритму правильно сработать и превратить ваш графический интерфейс в звуковой.

Когда ломается доступность

Доступность ломается от любого действия.

- Убрали текст у кнопки? Теперь VoiceOver не знает как её назвать.
- Сделали горизонтальную карусель, чтобы уменьшить количество элементов на экране? Получили кучу элементов в VoiceOver.
- Разместили 3 надписи в ячейке? Теперь фокус встает на каждой по отдельности.
- Уменьшили `.alpha` у кнопки, чтобы показать что она отключена? VoiceOver вас не понял и не говорит, что кнопка недоступна.

Нет такого способа верстать интерфейс, который бы не ломал доступность, потому что доступность – это обогащение графического интерфейса информацией о его структуре.

Но есть много свойств у стандартных элементов, про которые мы можем не знать. Например, стандартные ячейки таблицы адаптированы для VoiceOver, но если вы решили сделать свои, или ячейку для `UICollectionView`, то доступность потерянется.

Это не значит, что нельзя делать самодельные контролы – делайте. Просто адаптируйте. Для адаптации сделайте 4 шага, для каждого будет целый раздел:

- подписать,
- упростить,
- поправить навигацию,
- проверить сценарий.

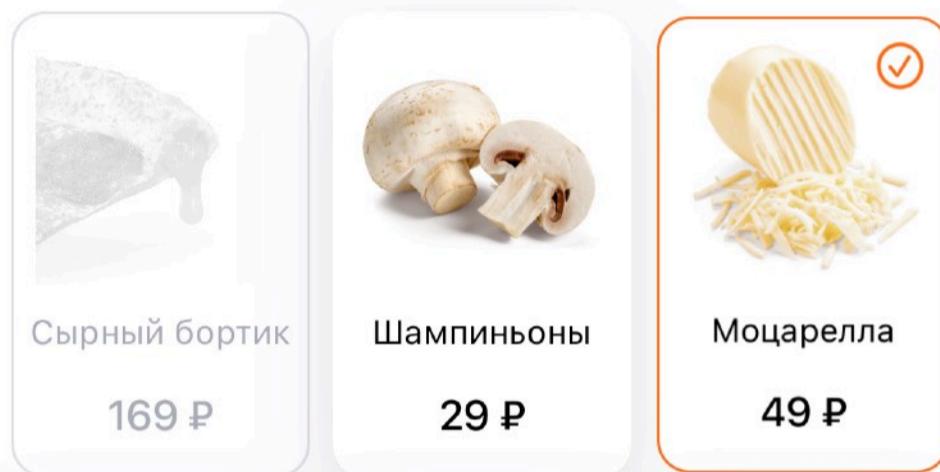
Как работает VoiceOver

Accessibility tree

Доступность в iOS обеспечивается протоколом `UIAccessibility`. Он работает с любыми объектами, не обязательно, чтобы они были элементами интерфейса.

```
@interface NSObject : UIAccessibility
```

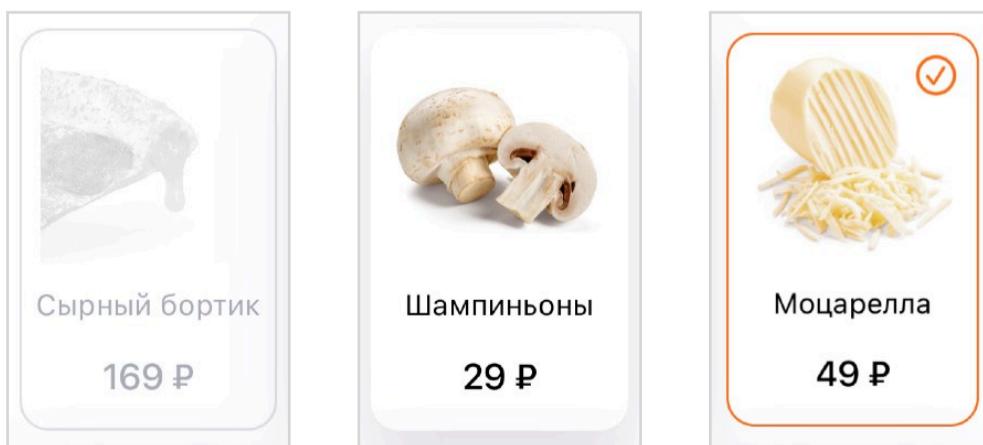
Добавить в пиццу



Чаще всего VoiceOver запрашивает доступность элементов у иерархии `UIView`. Уже из этих данных он создает фокус и накладывает его поверх интерфейса.

Разберем устройство VoiceOver на примере добавок к пицце. Посмотрим, как мы воспринимаем интерфейс, его техническую сторону и где помочь VoiceOver.

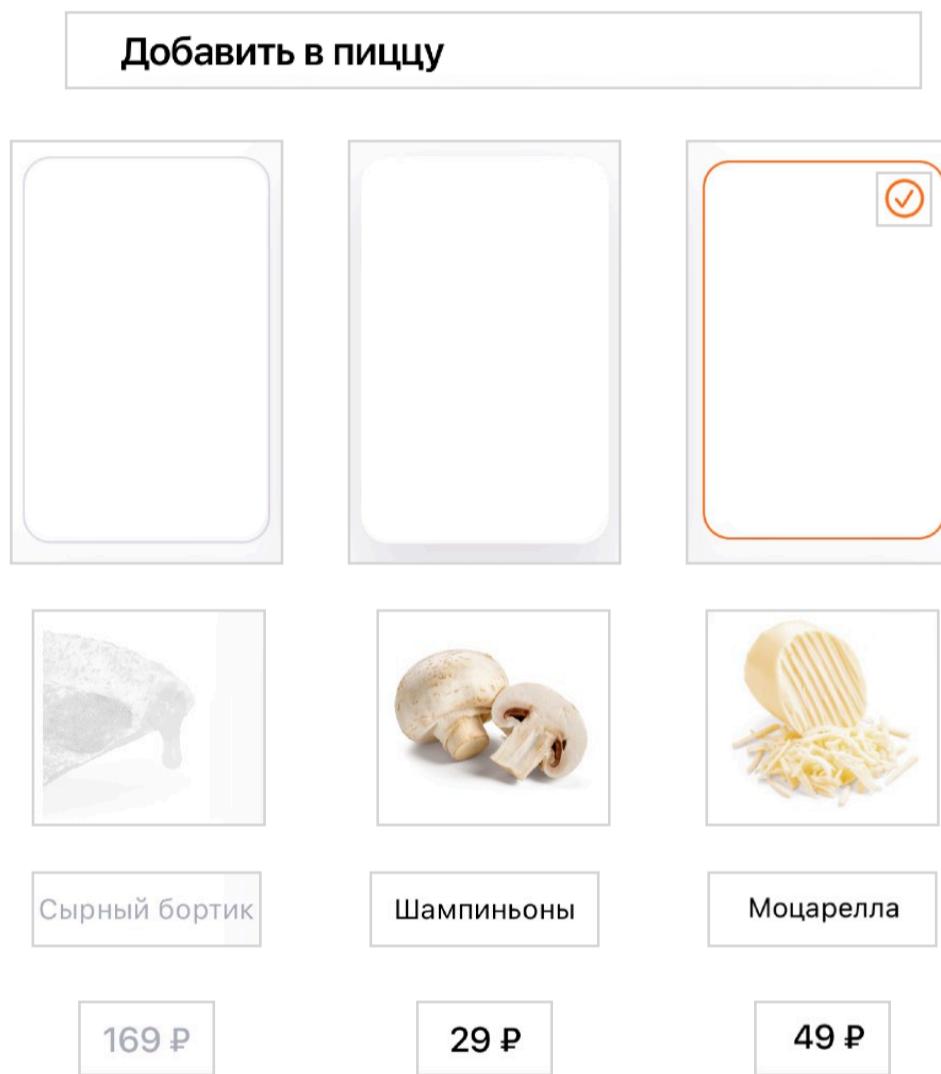
Добавить в пиццу



Для зрячего человека на экране 4 логических элемента: заголовок и три ячейки, на которые можно нажать.

Для iOS на экране 13 элементов:

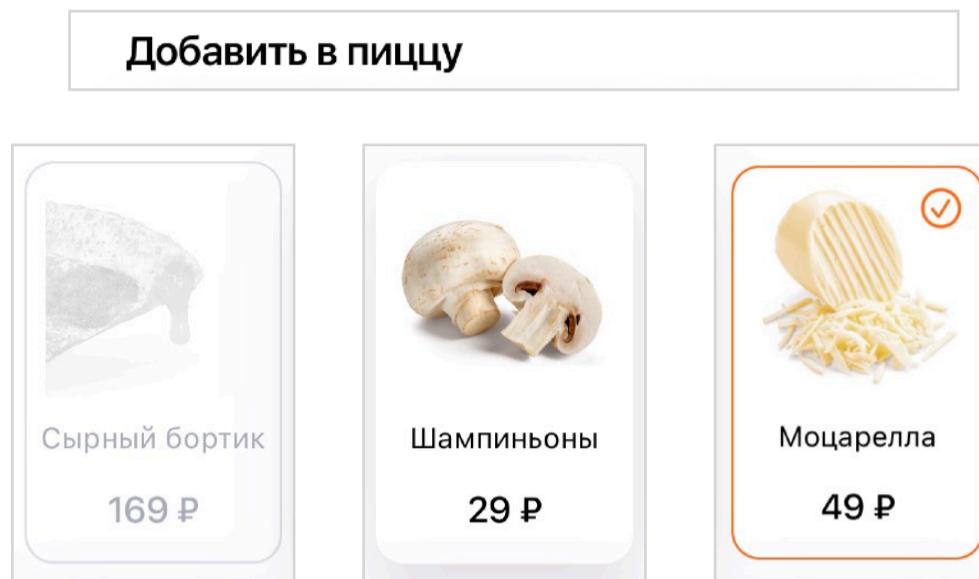
- заголовок,
- 3 ячейки-контейнера с разными состояниями,
- 3 картинки,
- 3 названия,
- 3 цены.



VoiceOver уже имеет ряд встроенных правил, по которым она пытается правильно прочитать интерфейс. Например, она не фокусируется на контейнерах других элементов, поэтому на ячейке она останавливаться не будет, картинки по умолчанию недоступны и VoiceOver их тоже не видит. А вот надписи и цены VoiceOver считает отдельными элементами и незрячему придется свайпать между ними раз за разом, чтобы прочитать.

Ещё VoiceOver ничего не расскажет про структуру экрана: он не знает, что «добавить в пиццу» это заголовок, что недоступно или уже добавлено в пиццу, и что на ячейку можно нажать. Мы понимаем это визуально, VoiceOver – нет.

Задача разработчика – подсказать VoiceOver как правильно читать элементы, какие у них свойства, как сгруппировать и что с контролами можно делать. По сути, нужно вернуться к первой ментальной модели.



В процессе мы будем разбирать, как применить все правила так, чтобы VoiceOver не был многословен, но рассказывал о всём нужном.

Адаптированная версия состоит из четырех элементов, их описание будет такое:

Добавить в пиццу, заголовок

Сырный бортик, 169 рублей, недоступно, кнопка

Шампиньоны, 29 рублей, кнопка

Выбрано, Моцарелла, 49 рублей, кнопка

После добавления ингредиента, VoiceOver ещё и скажет о новой цене всей пиццы.

Всё одновременно кратко и понятно. Незрячие слушают VoiceOver на очень большой скорости, поэтому текст похож на звуковые маячки, а не на поставленную речь диктора.

Как работает VoiceOver

Практика

- Включи VoiceOver. Для выключения можно дать команду Сири, проверь заранее, что она работает.
- Настрой шорткат на быстрый доступ через кнопку включения или на тап по спинке телефона.
- Попользуйся стандартными приложениями iOS, посмотри как они читают кнопки, их названия и типы.
- Научись сворачивать приложения, открывать панель с оповещениями.
- Попробуй отключить экран и ориентироваться только на звук.
- Попробуй попользоваться своим приложением, отметить непонятные места, где нет подписей, где много контроллов и где сломался фокус.
- Получится ли выполнить основной сценарий в твоем приложении?

На этом этапе стоит записывать все непонятные места в вашем приложении, позже вы поймете, как их починить.

Если нет своего приложения, то возьми любое популярное российское. Пример хорошей адаптации можно посмотреть [в приложении Додо Пиццы](#).

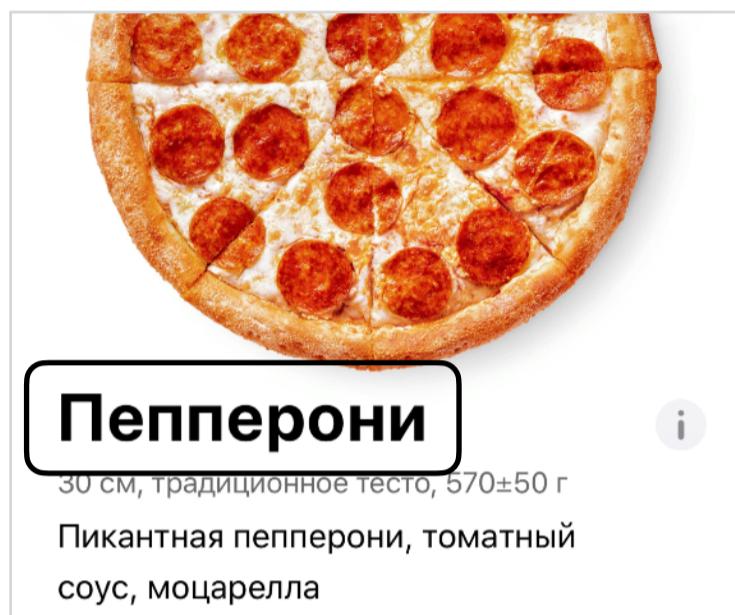
Подписать

Первая проблема доступности, с которой сталкиваются незрячие – элементы подписаны неправильно, или не подписаны совсем. Из-за этого непонятно, что с ними делать и что случится, если попробовать их нажать.

Это самая простая, но самая распространенная проблема. Подписав название элемента и указав его тип, можно исправить половину проблем доступности.

Подписать

Надписи



Текстовая надпись – это базовый элемент доступности.

Обычные подписи уже адаптированы для VoiceOver: их размер задаст фрейм фокусу, а текст в лейбле – это текст самой надписи.

Если вы рисуете текст через CATextLayer, то информация о доступности теряется.

Восстановить её можно самостоятельно, указав параметры. Разберем такой случай и посмотрим, что UILabel делает за нас.

Для начала, отметим, что элемент доступен, а значит на него можно поставить фокус. Если значение будет `false`, то VoiceOver попытается найти доступный элемент среди дочерних.

```
isAccessibilityElement = true
```

Затем подписать элемент, для описания возьмем текст в элементе. Этот текст VoiceOver прочитает, когда фокус попадет на контрол.

```
accessibilityLabel = text
```

В конце нужно задать фрейм элемента: так фокус будет виден на экране и можно будет навести касанием. Фокус нужен не только для незрячих и VoiceOver, но и для Switch Control.

```
accessibilityFrame = frameInScreenCoordinates
```

Готово, доступность в самом простом виде заработала.



Для хорошей доступности надписи нужно проверять, ведь VoiceOver читает ровно тот текст, который ему дали. Если на входе «30 см», то он и прочитает «тридцать сэмэ», а надо «тридцать сантиметров».

Чаще всего это встречается в ценах, когда красивое «30 ₽» превращается в прямолинейное «тридцать знак рубля».

Текст может быть длинным, просто соедините его через запятую, VoiceOver учтет это и даст правильную интонацию.

Починить можно дополнительным форматированием: например, создайте структуру, которая будет хранить и видимый текст, и специальный текст для VoiceOver.

```
struct AccessibleText {  
    let visibleText: String  
    let accessibleText: String  
}
```

Использовать можно напрямую, или завернуть в какой-нибудь красивый экстеншн для UILabel.

```
titleLabel.text = title.visibleText  
titleLabel.accessibilityLabel = title.accessibleText
```

Множественное число

Текст будет читаться намного лучше, если вы его просклоняете: 1 рубль, 2 рубля, 5 рублей. В каждом языке разные способы работы с числами. В английском их два, в русском 3. Все варианты можно описать в файле Localizable.stringsdict. Подробнее про формат файла [в документации](#).

▼ %d рублей	Dictionary	(2 items)
NSStringLocalizedFormatKey	String	%#@Variable@
▼ Variable	Dictionary	(6 items)
NSStringFormatSpecTypeKey	String	NSStringPluralRuleType
NSStringFormatValueTypeKey	String	d
zero	String	%d рублей
one	String	%d рубль
few	String	%d рубля
other	String	%d рублей

В коде строку с локализацией нужно вызвать так:

```
String(format:  
    NSLocalizedString("%d рублей", comment: ""),  
    price)
```

Еще в таком файле можно использовать не только ключ `NSStringLocalizedFormatKey`, но и `NSStringVariableWidthRuleType`. С его помощью можно задать отличающийся текст для разной ширины элементов, например, сокращать текст для маленьких размеров экрана.

▼ Телефон	Dictionary	(1 item)
▼ NSStringVariableWidthRuleType	Dictionary	(2 items)
320	String	Тел.
375	String	Телефон

```
let screenWidth = IntUIScreen.main.bounds.width  
let bundle: Bundle = Bundle()  
(NSLocalizedString("Телефон",  
    bundle: bundle,  
    comment: "Back button title") as NSString)  
.variantFittingPresentationWidth(screenWidth)
```

Подписать

Кнопки

Цезарь

Средняя 30 см, традиционное тесто, 640.0 г
Свежие листья салата айсберг в конверте,
цыпленок, томаты черри, сыры чеддер и
пармезан, моцарелла, сливочный соус, соус
цезарь

[Убрать ингредиенты](#)

Теперь разберем, как работает доступность кнопок. Кнопка с текстом читается так же как и надпись, но в конце добавляется подпись «кнопка»:

[Убрать ингредиенты, кнопка](#)

Описание типа элемента находится в конце текста, чтобы подсказать, что можно сделать после того, как услышите описание. Добавлять текст «кнопка» не нужно, за вас это сделает iOS, вам нужно лишь указать, что этот элемент является кнопкой, для этого поставьте трейт .button.

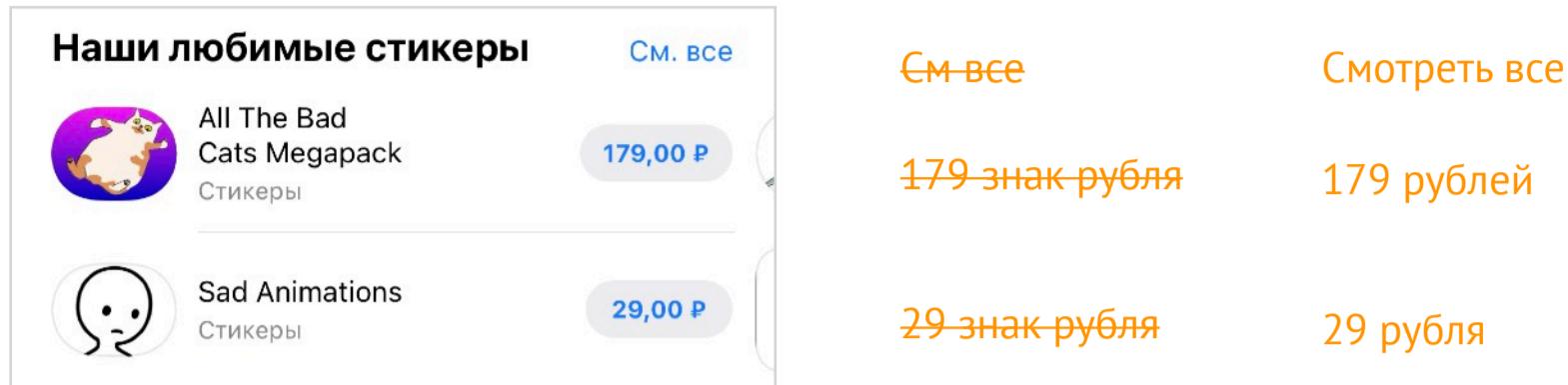
```
accessibilityTraits = .button
```

Действие от нажатия обрабатывается в функции `accessibilityActivate()`. Кнопка вызывает обычное нажатие на себя и возвращает `true`, если действие обработалось. Если вернется `false`, то VoiceOver попытается вызывать метод у следующего объекта в иерархии `UIView`.

```
override func accessibilityActivate() -> Bool {  
    // sendEvent(.touchUpInside)  
    // return true  
}
```

`AccessibilityActivate()` можно вызывать у любого доступного объекта, лишь бы у него был выставлен `isAccessibleElement = true`.

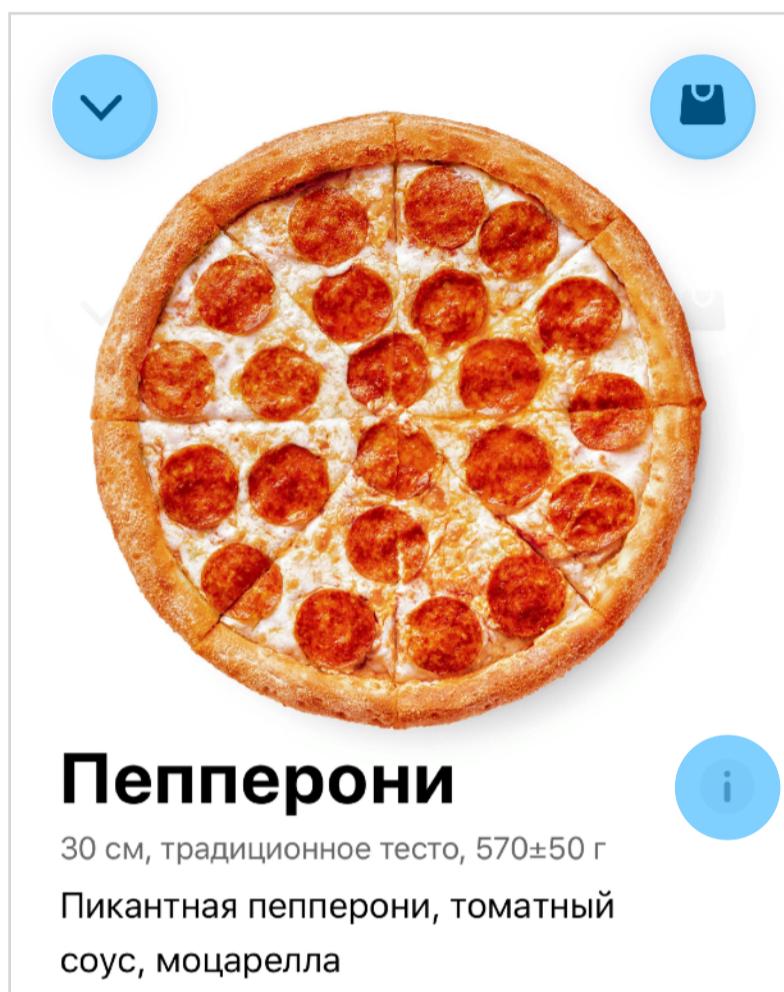
Текст кнопки надо проверять так же, как и обычные надписи. Различные сокращения и знаки могут читаться не так, как вы ожидаете.



Кнопками могут быть не только элементы `UIButton`, но и, например, ячейки в списках. Если на ячейку можно нажать, то мы должны рассказать об этом. Самый простой способ – поставить трейт `.button` для ячейки. Про списки и ячейки мы еще подробно поговорим в следующей главе.

Подписать

Кнопки без текста



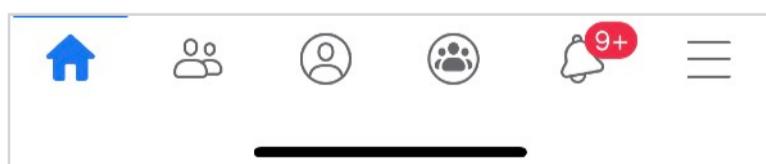
В угоду краткости, из многих кнопок убирают текст. Причин для этого много: неясно какой текст писать, кнопка неважная и не должна быть большой, ее действие очевидно и т.д. Доступность при этом сильно теряется, так как текста больше нет и VoiceOver нечего читать.

VoiceOver пытается хоть как-то помочь понять что на кнопке, поэтому читает название файла картинки!

Примеры с экрана: «айсиклоуз» (файл назывался `iClose`, сокращение для `icon close`), «айсикарт» и «айсииинфо». Что это означает догадаться можно, но не всегда.

Восстановить доступность легко, достаточно дать кнопкам название через `accessibilityLabel`.

```
closeButton.accessibilityLabel = "Закрыть"  
cartButton.accessibilityLabel = "В корзину"  
infoButton.accessibilityLabel = "Пищевая ценность"
```



Элементы в кастомном `UITabBarItem` подписываются точно так же.

Таббар в приложении Фейсбука закастомизирован, но подписан

Подписать

Изображения



14:02 4G LTE

▼ NEW

Цезарь

Средняя 30 см, традиционное тесто, 640.0 г
Свежие листья салата айсберг в конверте, цыплёнок, томаты черри, сыры чеддер и пармезан, моцарелла, сливочный соус, соус цезарь

[Убрать ингредиенты](#)

Маленькая Средняя Большая

Традиционное Тонкое

Добавить в пиццу

		
Острый халапеньо	Цыпленок	Ветчина
39 ₽	59 ₽	59 ₽

С картинками двоякая ситуация. С одной стороны, для незрячих особой ценности они не несут, iOS их даже скрывает от VoiceOver по умолчанию.

С другой стороны, может быть важно знать, что на экране есть изображение: его можно показать знакомым, сделать скриншот и много чего еще. iOS всеми силами старается рассказать о содержимом картинки, даже может распознавать изображение, рассказать что на нем, узнавать ваших знакомых. Например, у этой картинки описание «пицца на белом фоне, new». Распознавание можно включить через ротор.

Все маленькие иконки и декоративные элементы точно нужно скрывать, а вот с большими картинками посложнее. Если вы хотите явно сообщить о наличии изображения, то поставьте ему трейт `.image`, VoiceOver добавит к описанию «изображение».

Если на картинке есть ценная информация, которая больше нигде не дублируется, то это нужно где-то описать. На примере есть бейджик **NEW**, это важная информация про новинку, VoiceOver должен про это рассказать. Не обязательно сообщать об этом на картинке, можно добавить текст «новинка» к названию пиццы.

Подписи

Как называть элементы?

У хорошего названия элемента несколько свойств:

- Он кратко описывает элемент.
 - Одно слово:
Добавить, Играть, Удалить, Искать.
 - Короткая фраза:
Играть музыку, Указать имя, Добавить к событию.
- Не содержит тип, его помечаем через трейт.
- Начинается с заглавной буквы, не заканчивается точкой. Название от значения VoiceOver отделит сам короткой паузой.
- На языке пользователя: переведенное и простое по смыслу.
- Если надпись составная, то части нужно разделить запятой.
Пепперони, 25 см, тонкое тесто

Больше про синтаксис можно прочитать [в документации](#).

Практика

- Попользуйся приложением, найди элементы, которые не подписаны.
- На какие элементы можно нажать, но они не обозначены как кнопки? Чаще всего непонятно, что ячейки в списках кликабельны.
- В каких случаях текст читается неправильно из-за сокращений? На странные ударения пока внимания не обращай.
- Попадались ли неподписанные картинки, которые стоит скрыть? На каких наоборот, есть важная информация, о которой нужно сообщить вслух?

Упростить

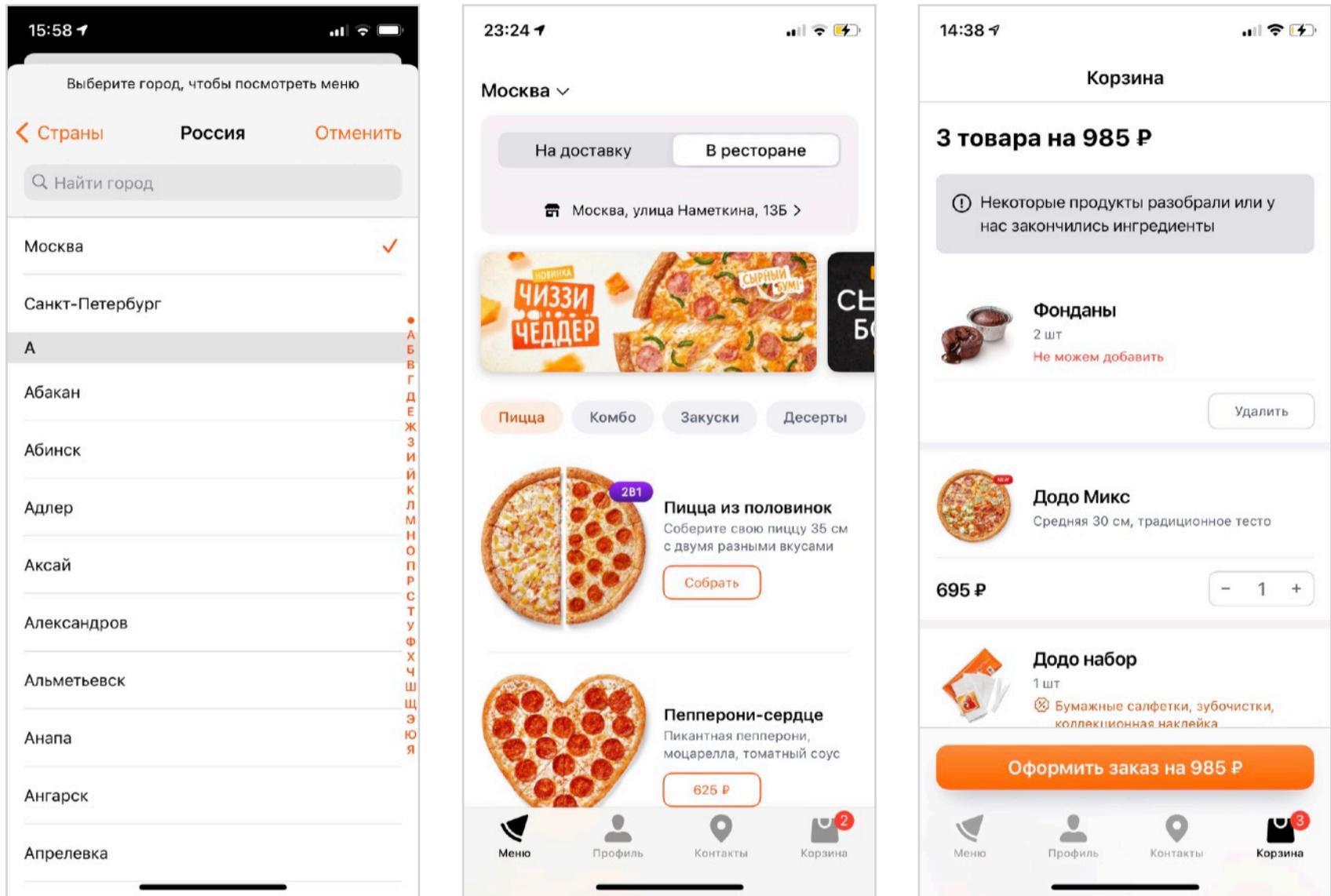
Подписав все элементы и указав их тип, мы исправили самый простой, но массовый слой проблем. Для большинства приложений этого будет достаточно.

Теперь надо разобраться со следующей проблемой: на экране очень много элементов, связи между ними не всегда понятны, нужно постоянно между ними переключаться. Количество элементов можно уменьшить, связи добавить, а удобство использования повысить.

Большинство экранов мобильных приложений – это списки. Мы воспринимаем ячейку в списке как одно целое, поэтому и VoiceOver должен воспринимать ее как единый элемент доступности. Посмотрим на разные ячейки и будем постепенно усложнять их, наблюдая за тем, как меняется доступность этих ячеек.

Упростить

Списки и ячейки



Большая часть интерфейсов приложений – это списки, а для доступности это:

- много ячеек,
- много контроллов внутри каждой ячейки.

В идеальном списке каждая ячейка – это один доступный элемент, а описание ячейки рассказывает о всем содержимом, что находится внутри нее.

Интересно, что поведение таблиц и коллекций отличается: в таблицах есть системные стили, они влияют и на правильное описание для VoiceOver. Коллекции примитивней, для них всю доступность нужно поддерживать самостоятельно. Рассмотрим на примерах, постепенно усложняя.

Начнем с ячейки из одной подписи и воссоздадим ее доступность с ноля.

Ячейка является контейнером для `UILabel`, поэтому VoiceOver ее пропустит и поставит фокус именно на надпись. Фокус получится маленький, его размер будет отличаться для каждой ячейки. Самое критичное, что на ячейку можно нажать, но VoiceOver об этом не сообщает, ведь он смотрит на простой текст.



Ячейка, в которой элементом доступности является подпись.

Адаптированная ячейка: «Москва, кнопка». Фрейм занимает весь размер ячейки

Для адаптации ячейки нужно:

- сделать всю ячейку доступным элементом,
- дать ей `label`, который опишет содержимое,
- если на ячейку можно нажать, то указать трейт `.button`.

Доступность самой подписи можно не отключать, VoiceOver уже не увидит ее внутри ячейки. Код для адаптации простой: сообщаем ячейке, что она стала доступным элементом, ставим ей трейт `.button`, а текст заголовка ставим не только в подпись, но и в `.accessibilityLabel`.

```
class LocaleCell: UITableViewCell {
    @IBOutlet private weak var nameLabel: UILabel!

    override func awakeFromNib() {
        super.awakeFromNib()

        isAccessibilityElement = true
        accessibilityTraits = .button
    }

    var title: String? {
        didSet {
            nameLabel.text = title
            accessibilityLabel = title
        }
    }
}
```

Москва, улица Миклухо-Маклая, 36А
С 09:00 до 23:00

Усложним ячейку. Теперь у нее есть вторая строчка со временем работы. В ячейке две смысловые части: адрес и время работы, причем время нам интересно, только если это нужный нам адрес.

Мы можем объединить два текста в один через запятую и записать в `accessibilityLabel`. Можно интересней: время работы записать в `accessibilityValue`, тогда VoiceOver добавит небольшую паузу и прочитает время с другой интонацией, так появится живость речи и будет проще воспринимать на слух.

`label: Москва, улица Миклухо-Маклая, 36А`
`value: С 9 до 23`
`traits: кнопка`

```
class PizzeriasTableViewCell: UITableViewCell {
    @IBOutlet weak var titleLabel: UILabel!
    @IBOutlet weak var scheduleLabel: UILabel!

    override func awakeFromNib() {
        super.awakeFromNib()

        isAccessibilityElement = true
        accessibilityTraits = .button
    }

    var title: String? {
        didSet {
            titleLabel.text = title
            accessibilityLabel = title
        }
    }

    var schedule: String? {
        didSet {
            scheduleLabel.text = schedule
            accessibilityValue = schedule
        }
    }
}
```

Москва, улица Миклухо-Маклая, 36А
С 09:00 до 23:00



label: Выбрано, Москва, улица Миклухо-Маклая, 36А
value: с 9 до 23
traits: кнопка

Продолжим усложнять ячейку.

У ячейки может стоять галочка, так мы отмечаем текущую пиццерию. У VoiceOver есть стандартный подход к обозначению выбранного элемента – трейт .selected.

Элемент можно отметить одновременно и типом (кнопка), и состоянием (выбрано), поэтому трейты надо указывать через функции работы с OptionSet.

```
override public var isSelected: Bool {  
    didSet {  
        if isSelected {  
            accessibilityTraits.formUnion(.selected)  
        } else {  
            accessibilityTraits.subtract(.selected)  
        }  
    }  
}
```

Интересно, что даже в таком формате VoiceOver читает время как «с девять до двадцать три». Слова не склоняет, но и лишние нули не читает.

Москва, улица Миклухо-Маклая, 36А

С 09:00 до 23:00, сейчас закрыто

label: Москва, улица Миклухо-Маклая, 36А

value: С 9 до 23, сейчас закрыто

traits: недоступно, кнопка

Если вам нужно показать, что ячейку сейчас выбрать нельзя, то используйте стандартный трейт `.notEnabled`, он добавит к описанию «недоступно».

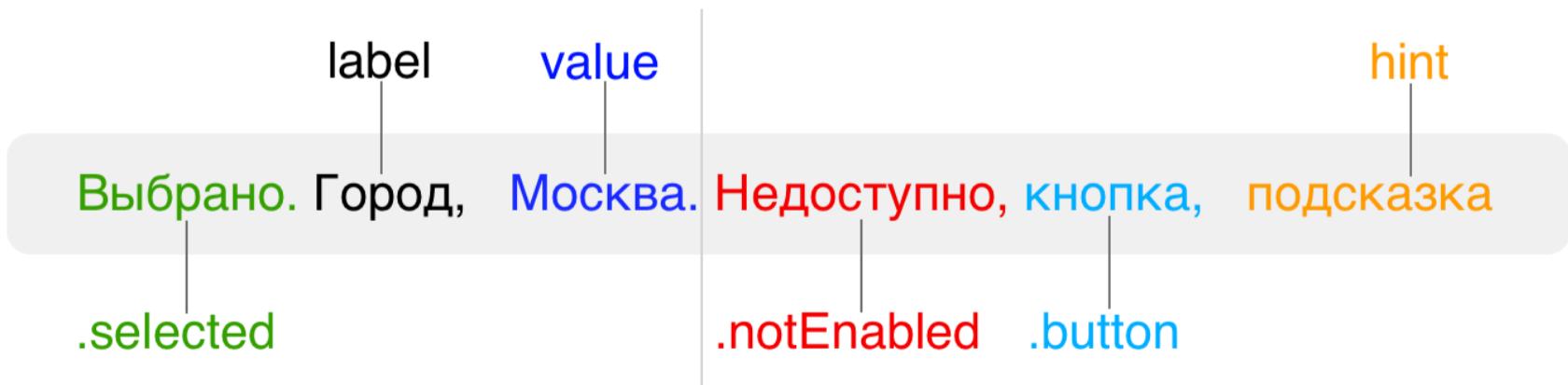
```
accessibilityTraits.formUnion(.notEnabled)
```

Использовать трейт `.notEnabled` нужно только для тех элементов, с которыми нельзя взаимодействовать. Если я могу выбрать адрес, то трейт `.notEnabled` указывать не нужно и о закрытой пиццерии надо рассказать через `value`.

За последние 3 страницы мы столкнулись с несколькими трейтами:

- `.selected`,
- `.notEnabled`,
- `.button`.

Каждый из них описывает одно из стандартных поведений, при этом подписи появляются в разных местах. С другой стороны, у нас есть `accessibilityLabel`, `accessibilityValue` и `accessibilityHint`. Пора разобраться в порядке чтения.



Если вы листаете большой список из элементов, то вам важнее всего узнать, какой элемент **выбран**. Затем, что это за элемент, его **label** и **value**. Когда вы поняли, что за элемент в фокусе, вы можете решить, что нужно с ним делать, поэтому **доступность** элемента, его **тип** и **подсказка**, как с ним взаимодействовать, находятся в самом конце описания. Перед `value` и `hint` есть небольшие паузы.

Упростить

Сложная ячейка



Крэйзи пепперони
Пикантная пепперони, цыпленок, моцарелла, томатный соус, кисло-сладкий соус.
от 395 рублей

Перейдем к сложным ячейкам, когда внутри несколько контроллов. Без адаптации фокус будет читать каждую надпись отдельно и надо будет свайпнуть минимум три раза, чтобы перейти к следующему продукту.

Для комфортной работы ячейка должна быть одним элементом, тогда текст будет читаться один за другим. При этом, его можно прервать, выполнив одно из действий: нажать на ячейку, перейти к следующей, или просто остановить чтение.

Составим модель ячейки

Картинка незрячему не нужна, знать, что она там есть тоже бесполезно. Остается 3 надписи. Как их считывает зрячий человек? Название пиццы первое и выделено жирным. Скорее всего, следующим он прочитает цену, ведь на ней цветовой акцент, а состав в последнюю очередь, потому что он длинный и блеклый. Итак, порядок восприятия такой:

- название,
- цена,
- состав.

Название и цену можно объединить через запятую и поставить в `label`. Состав отделить интонацией и сохранить в `value`.

`label: Крэйзи пепперони, от 395 рублей`
`value: Пикантная пепперони, цыпленок, моцарелла, томатный соус, кисло-сладкий соус.`
`trait: кнопка`



Крэйзи пепперони
Пикантная пепперони, цыпленок, моцарелла, томатный соус, кисло-сладкий соус.
от 395 рублей

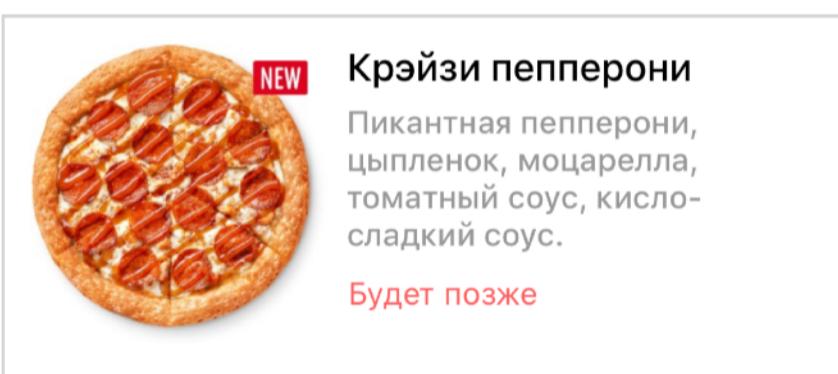
Интересно, что в графическом дизайне порядок восприятия не всегда правильный, опрятность верстки иногда важнее смысла.

Код для доступности достаточно прост: делаем ячейку доступной, даем ей описание и поведение кнопки, а затем лишь собираем две строчки с описанием.

```
override func awakeFromNib() {
    super.awakeFromNib()

    isAccessibilityElement = true
    accessibilityTraits = .button
}

func updateAccessibility(title: String,
                        price: String,
                        ingredients: String?) {
    accessibilityLabel = [title, price].joined(separator: ", ")
    accessibilityValue = ingredients
}
```



Если продукт недоступен, то кнопка с ценой заменится на надпись «будет позже». Для VoiceOver графическая смена контроллов неважна, поэтому просто меняем текст цены:

```
label: Крэйзи пепперони, будет позже
value: Пикантная пепперони, цыпленок, моцарелла, томатный соус, кисло-сладкий соус.
trait: кнопка
```

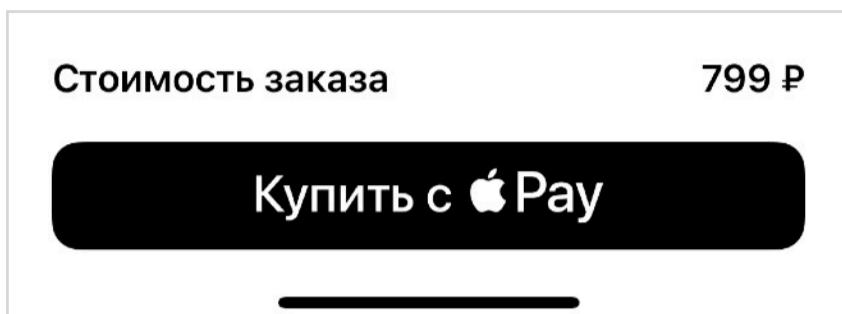
С учетом этого, код становится лишь чуть сложнее:

```
func updateAccessibility(title: String,
                        price: String,
                        ingredients: String?,
                        isProductAvailable: Bool) {
    let price = isProductAvailable ? price : "Будет позже"

    accessibilityLabel = [title, price].joined(separator: ", ")
    accessibilityValue = ingredients
}
```

Упростить

Название+значение



В интерфейсах часто встречается паттерн: одна надпись рассказывает о названии, другая о значении. Мы воспринимаем строчки как одно целое, но для VoiceOver это отдельные элементы. На примере хорошо, что расположение стоимости и цены совпадает с порядком чтения: название, а затем величина. Так человек получает информацию в правильной последовательности даже без адаптации.

Мы можем улучшить, сделав из всей строки один доступный элемент. Для этого описание двух лейблов нужно склеить в одну строку, а рамку фокуса растянуть на всю строку. Вариантов адаптации несколько. Можно взять текст из первой надписи, через запятую добавить из второй (VoiceOver учитывает пунктуацию в речи) и записать это все в `label`.

```
titleLabel.accessibilityLabel = "Стоимость заказа, 799 рублей"
```

Текст можно забирать прямо из контроллов:

```
titleLabel.accessibilityLabel = titleLabel.text!
                                + ", "
                                + valueLabel.text!
```

Можно сделать это чуть изящней: название оставить в `label`, а величину записать в `value`. В таком случае VoiceOver немного изменит интонацию для значения, получится красиво.

```
titleLabel.accessibilityLabel = "Стоимость заказа"
titleLabel.accessibilityValue = "799 рублей"
```

Динамически получится так:

```
titleLabel.accessibilityLabel = titleLabel.text
titleLabel.accessibilityValue = valueLabel.text
```

Теперь нам нужно скрыть label с цифрой от VoiceOver, его текст уже читается у label с названием, а дублирование нам не нужно.

```
valueLabel.isAccessibilityElement = false
```

Можно адаптировать иначе: скрыть от VoiceOver обе надписи, вместо них самому сделать «невидимый» доступный элемент и поставить ему все свойства. Мне это кажется слишком сложным, но пример с этим я покажу в главе про контейнеры.



Если дизайнер разместил значения под подписями, то порядок чтения без адаптации сбьется: сначала прочитает все названия, потом все значения, понять что к чему относится будет сложно. Исправить проблему можно таким же объединением в один элемент.

Пример из апстора показывает, что иногда надо чуть поменять текст специально для VoiceOver:

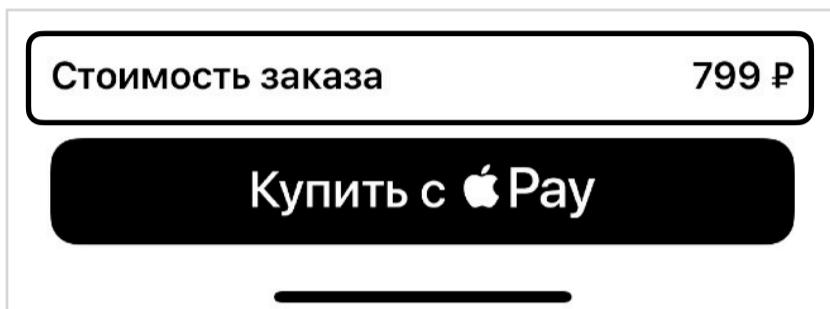
Оценка 4.8 на основе 61 тысячи оценок

Возраст 12 плюс

Рейтинг 9 место в «Еда и напитки»

Упростить

Рамка фокуса



При объединении двух элементов нужно увеличить рамку, чтобы она обхватывала обе подписи. Это поможет правильно определять элемент под пальцем при изучении касанием. С другой стороны, это поможет правильно ставить фокус другим технологиям доступности, например, Switch Control.

За рамку фокуса отвечает свойство `accessibilityFrame`. Рамка должна быть рассчитана в координатах экрана. Самый простой способ – это использовать уже рассчитанные фреймы и просто объединить их. Важно, чтобы ваш расчет был после того,

```
override func layoutSubviews() {
    super.layoutSubviews()

    titleLabel.accessibilityFrame =
        titleLabel.accessibilityFrame
            .union(valueLabel.accessibilityFrame)
}
```

как посчитает iOS, поэтому лучшее место – внутри `layoutSubviews()`.

Для конвертации фрейма в координаты экрана есть специальная функция `UIAccessibility.convertToScreenCoordinates`. Вызывать ее нужно так же

```
override func layoutSubviews() {
    super.layoutSubviews()

    let commonFrame = titleLabel.frame.union(valueLabel.frame)
    titleLabel.accessibilityFrame = UIAccessibility
        .convertToScreenCoordinates(commonFrame, in: self)
}
```

Упростить

UISwitch в ячейке



Свитчер – это кнопка с состоянием, VoiceOver его называет «кнопка-переключатель» и сообщает текущее состояние: включено или выключено. Переключить можно двойным тапом, VoiceOver прочитает новое состояние. В общем, по умолчанию все адаптировано.

Часто рядом со свитчером есть подпись. Семантически подпись и свитчер – это один контрол, поэтому и VoiceOver должен работать с ним как одним целым: читать подпись в качестве названия и тут же давать возможность нажать.

Должно получиться так:

```
label    Сообщать о бонусах, акциях и новых продуктах  
value   Включено  
trait   кнопка-переключатель, вкл.
```

Интересно, что трейт «кнопка-переключатель» незадокументирован, но доступен под номером 53:

```
let switchButtonTrait: UIAccessibilityTraits = 1 << 53
```

Как можно обработать:

- сделать всю ячейку доступным элементом,
- перенести текст надписи в лейбл ячейки ,
- перенести трейт со свитчера,
- value брать динамически из свитчера.

Делаем всю ячейку доступной, указываем трейт как у свитчера.

```
override func awakeFromNib() {
    super.awakeFromNib()
    isAccessibilityElement = true
    accessibilityTraits = switcher.accessibilityTraits
}
```

Даем правильный label из двух строк.

```
func configure(title: String, subtitle: String) {
    titleLabel?.text = title
    subtitleLabel?.text = subtitle
    accessibilityLabel = [title, subtitle]
        .joined(separator: ", ")
}
```

Значение должно считываться прямо из свитчера. Оно уже локализовано, но вы можете написать свой текст.

```
override var accessibilityValue: String? {
    get {
        switcher.accessibilityValue
    }
    set {}
}
```

Нажатие на ячейку должно активировать свитчер. Можно вызвать свитчер внутри функции accessibilityActivate().

```
override func accessibilityActivate() -> Bool {
    switcher.accessibilityActivate()
}
```

Но есть и другой способ: переместить точку активации на свитчер. Про него в следующей главе.

```
override func layoutSubviews() {
    super.layoutSubviews()
    accessibilityActivationPoint = switcher
        .accessibilityActivationPoint
}
```

Упростить

Activation Point



Во всех примерах мы старались сделать всю ячейку активным элементом. Обычно, нажатие на ячейку выполняет главное действие, но иногда нужно нажать на кнопку внутри ячейки, даже если в фокусе вся ячейка.

Мы могли бы симулировать нажатие кнопки в функции `accessibilityActivate()`, но есть решение проще – передвинуть точку активации.

За координату отвечает свойство `accessibilityActivationPoint`, по умолчанию нажимает в центр экрана. Новое значение надо задавать в координатах экрана, поэтому:

- координату нужно задавать в методе `layoutSubviews()`, когда координаты уже известны,
- проще всего брать `activationPoint` другого элемента, чтобы не конвертировать координаты самому.

```
override func layoutSubviews() {  
    super.layoutSubviews()  
    accessibilityActivationPoint =  
        selectButton.accessibilityActivationPoint  
}
```

Если нужно конвертировать, то можно воспользоваться функцией для фрейма, а уже у него взять центр.

```
UIAccessibility.convertToScreenCoordinates(  
    selectButton.bounds,  
    in: selectButton).center
```

Здесь будет еще

Продолжение книги будет выходить еженедельно. Подписывайтесь [на канал Додо Пиццы](#) или [мой твиттер](#), я сообщу о выходе.

Можно приходить [на курс про доступность](#). Все расскажем, покажем, научим. Курс для дизайнеров, исследователей, iOS, андроид и web-программистов.

Актуальную версию книги можно узнать [на сайте](#). Сейчас у вас версия 0.2, выпущена 24 мая.