



# Доступность iOS приложений

В книге я расскажу как адаптировать графические интерфейсы для незрячих: они слушают интерфейс, а команды отдают необычными жестами. Покажу как управлять голосом не имея возможности коснуться экрана. Или как взаимодействовать с миром, когда можешь только моргнуть, а больше ничего не можешь.

## Мотиваций

# Что такое доступность?

Под доступностью приложений люди имеют ввиду адаптацию приложения для разных групп людей. Если человек хуже видит (или совсем потерял зрение), не слышит, не может двигаться, но при этом может полноценно воспользоваться приложением, то оно доступно.

Есть популярная доступность которой мы пользуемся каждый день. Не знаете английский, но хочется посмотреть новый сериал на Netflix? Всегда 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.

Каждый из этих примеров можно усилить и тогда он поможет людям с другими проблемами. Аудиодескрипция вместо субтитров расскажет незрячему что происходит на экране. Вернуться на экран назад можно с помощью специального жеста в VoiceOver, а темная тема поможет людям со светобоязнью.

Отклонение бывают разные. Многие с возрастом хуже видят, но могут увеличить шрифт на телефоне и пользоваться им без очков или включить экранный зум и увеличивать очень сильно. У некоторых людей может быть нарушение восприятия цвета, для них стоит доносить информацию не только цветом. Кто-то не слышит, но узнает о новом звонке вибрацией на часах.

Отклонения бывают сильные. Кто-то совсем потерял зрение, но может включить экранный диктор, слушать как он читает что находится на экране, свайпами по экрану отдавать команды и так пользоваться телефоном. Кто-то совсем неподвижен, но движением щеки может давать телефоны команды «выбери» и «следующий элемент» и так написать книгу об устройстве вселенной.

Я расскажу о самых сложных нарушениях, в которых даже не сразу понятно как вообще можно пользоваться телефоном без кнопок. Может показаться, что незрячему удобней пользоваться телефоном с кнопками, но на самом деле мир изменился после выхода iPhone 3GS.

История доступности: [A Timeline of iOS Accessibility: It Started with 36 Seconds.](#)

## Мотивация

# 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? Иногда жет не важно, 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 небольшими, иногда серьезными, которые меняют образ жизни. Раньше это могло 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 сделать очень много. Айфон уже содержит все нужные инструменты, вам нужно им только немного помочь.

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

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, а в итоге выигрывают все пользователи.

## **Важность для компании**

Для начала: если вашем продуктом пользуются миллионы, то это сотни и тысячи людей, для которых сервис может быть жизненно важным, но недоступным.

Понимаете? Он влияет на всю жизнь человека и то что он в ней может сделать.

Не все измеряется цифрами и прибылью. Любая крупная компания несет ответственность за свое влияние на людей: она может помогать, мешать, создавать праздник, банить политиков. Все IT 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 соц.сети, ходят на работу. Не адаптируя приложения мы забираем кусочек их жизни и функционала.

При этом доступность это не фича, ее нельзя измерить сроком, сделать один раз и никогда больше не возвращаться к ней. Это майндсет, набор правил которые разработчики и дизайнеру держат в голове когда создает новый интерфейс. По сути, доступность это навык и ему можно научиться.

Доступность приложения выступает очень хорошим измерителем культуры компании. Все крупные иностранные приложения Эпла, Гугла, Фейсбука адаптированы и работают очень хорошо. Почти все крупные российские приложения не адаптированы совсем или содержат критичные ошибки.

Создавайте приложения для людей. Для всех.

## Мотивация

# 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 вторую версию приложения для особого случая, но тему обсудим.

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

Технически нужно адаптировать контролы с помощью UIAccessibility. Функционал нужен в том же объеме что и у обычной версии, при этом не ясно как должен работать графический интерфейс. Получается, что есть две версии: одна не ясно как читается VoiceOver, вторая не ясно как должна выглядеть. При этом, и там и там мы все равно работаем с UIKit.

Все технологии доступности очень много данных берут от графического интерфейса, но иногда у них не все получается и им нужно 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

Не нужно писать какой-то специальный текст для незрячих, надо лишь правильно оформить существующие контролы, добавлять текст приходится крайне редко.

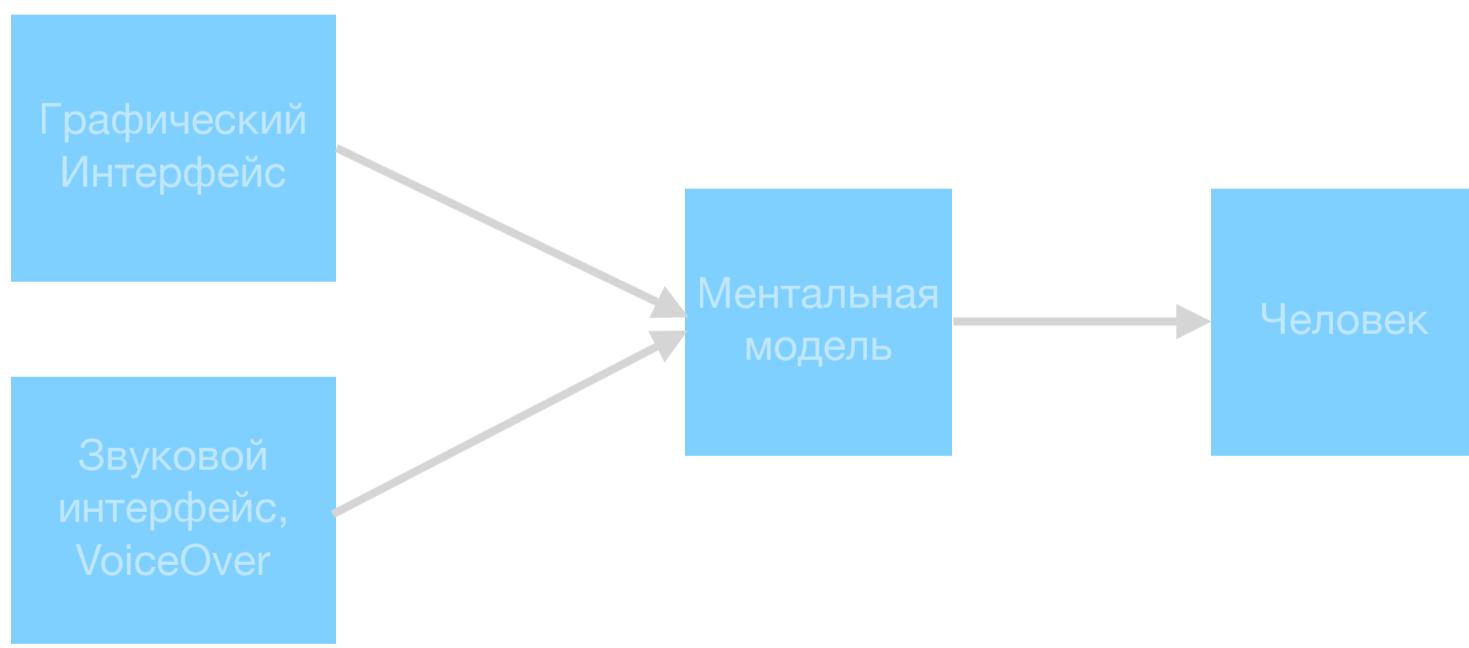
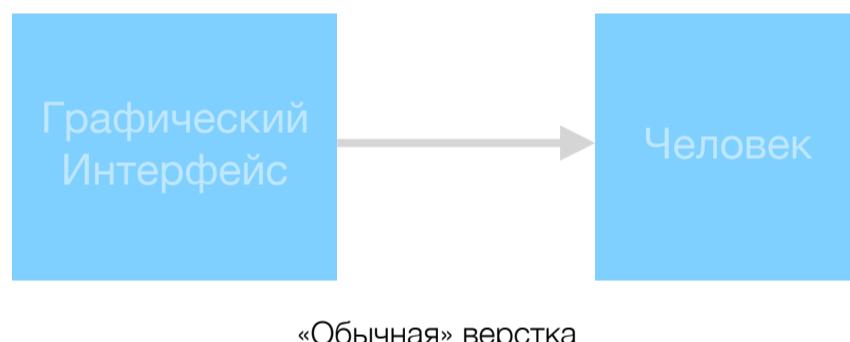
## Мотивация

# Ментальная модель

Не адаптируя доступность мы работаем в режиме «анимированного макета с данными», копируя только внешние атрибуты и 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. Именно поэтому не нужно делать отдельных версий для людей с нарушениями, нужно делать нормально для всех.



# Как работает VoiceOver

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

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 у вас не работает интернет, вам срочно надо понять что будет на макете, а картинку вы посмотреть не можете. Вы звоните другу с интернетом и он вам описывает картинку по телефону. Что он расскажет?

Скорее всего, перечислит элементы по порядку, даст им название и расскажет тип: заголовок «Пицца», надпись с размером теста, кнопка «купить». Из рассказа вам будет понятно в каком порядке элементы, что они обозначают, в каком состоянии находятся, что с ними можно сделать.

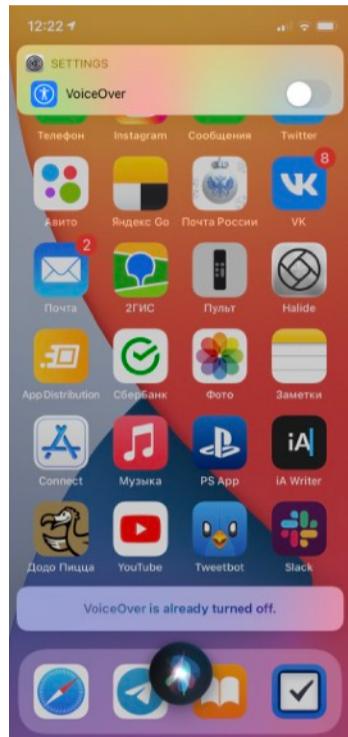
VoiceOver выступает в роли такого друга. Он берет один элемент, читает его описание, рассказывает о состоянии (выбрано, отключено), говорит какой это тип и что с ним можно делать. В ответ вы можете попросить перейти к следующему элементу, нажать на кнопку или закрыть текущий экран. Команды можно разными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

VoiceOver всеми силами старается уменьшить количество дополнительной работы. Не нужно делать специальных версий для незрячих, ваша задача лишь помочь VoiceOver назвать все правильно и обработать ввод. Крутость технологии в том, что работы нужно сделать немного: синтез речи уже встроен в iOS, нужно только подправлять читаемый текст, тип элементов и обработать несколько дополнительных функций.

Перед адаптацией приложения надо разобраться в деталях работы VoiceOver, научится ей управлять, посмотреть как она работает на примере стандартных приложений и уже с пониманием звукового интерфейса приступить к адаптации своего приложения.

## Как работает VoiceOver

# Включение



Чтобы понять как адаптировать интерфейсы нужно научиться работать с ними в таком же режиме как это делают незрячими. Это просто: включите на телефоне VoiceOver и попользуйтесь, я покажу как.

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 нее.

Включить VoiceOver можно несколькими другими способами:

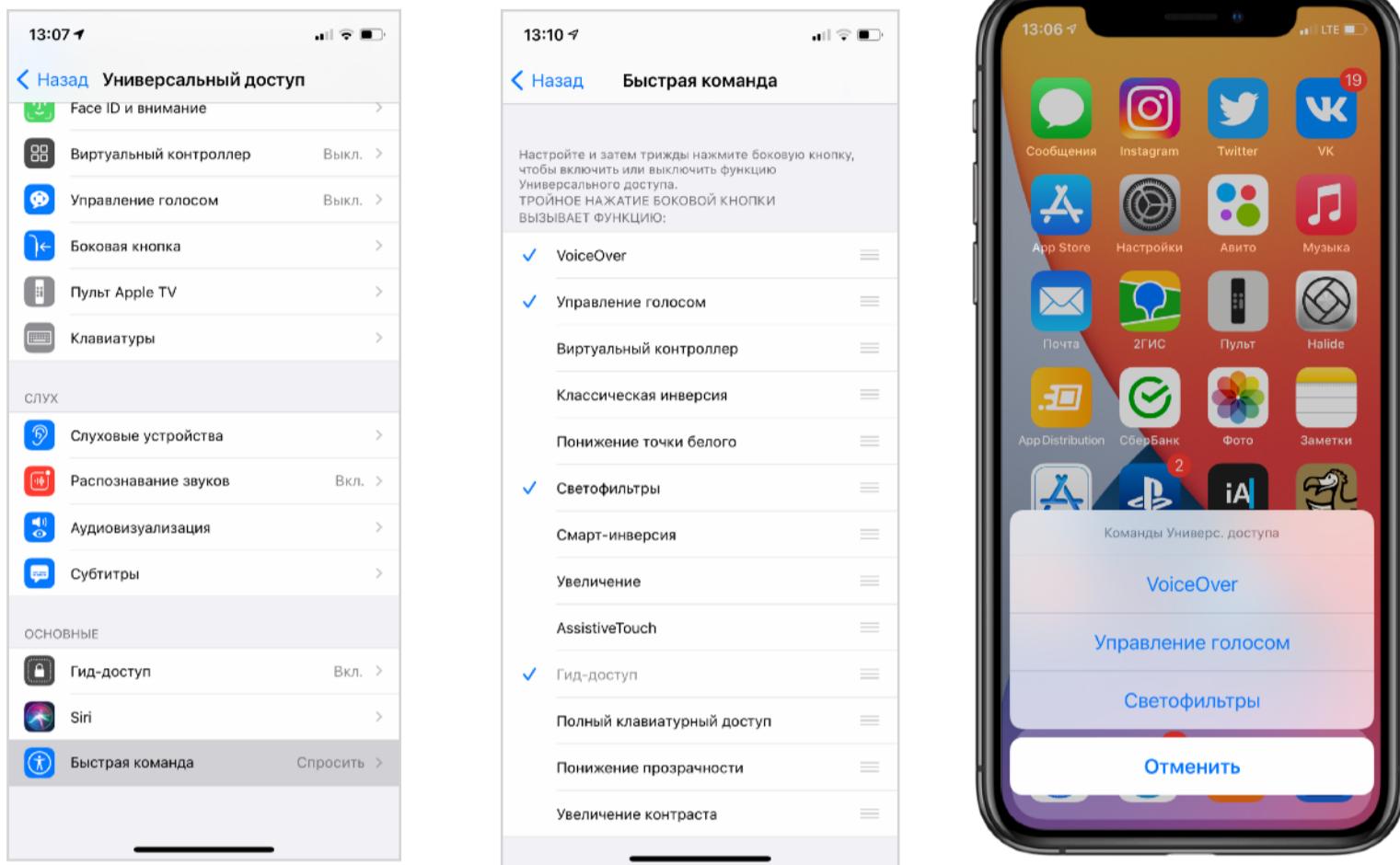
- Настройки
- Шорткат для доступности
- Бэк-тап по спинке телефона

Настройки VoiceOver можно найти в разделе Универсальный доступ настроек телефона. Для начала достаточно задать комфортную скорость чтения. Незрячие ставят высокую скорость, они к ней привыкли

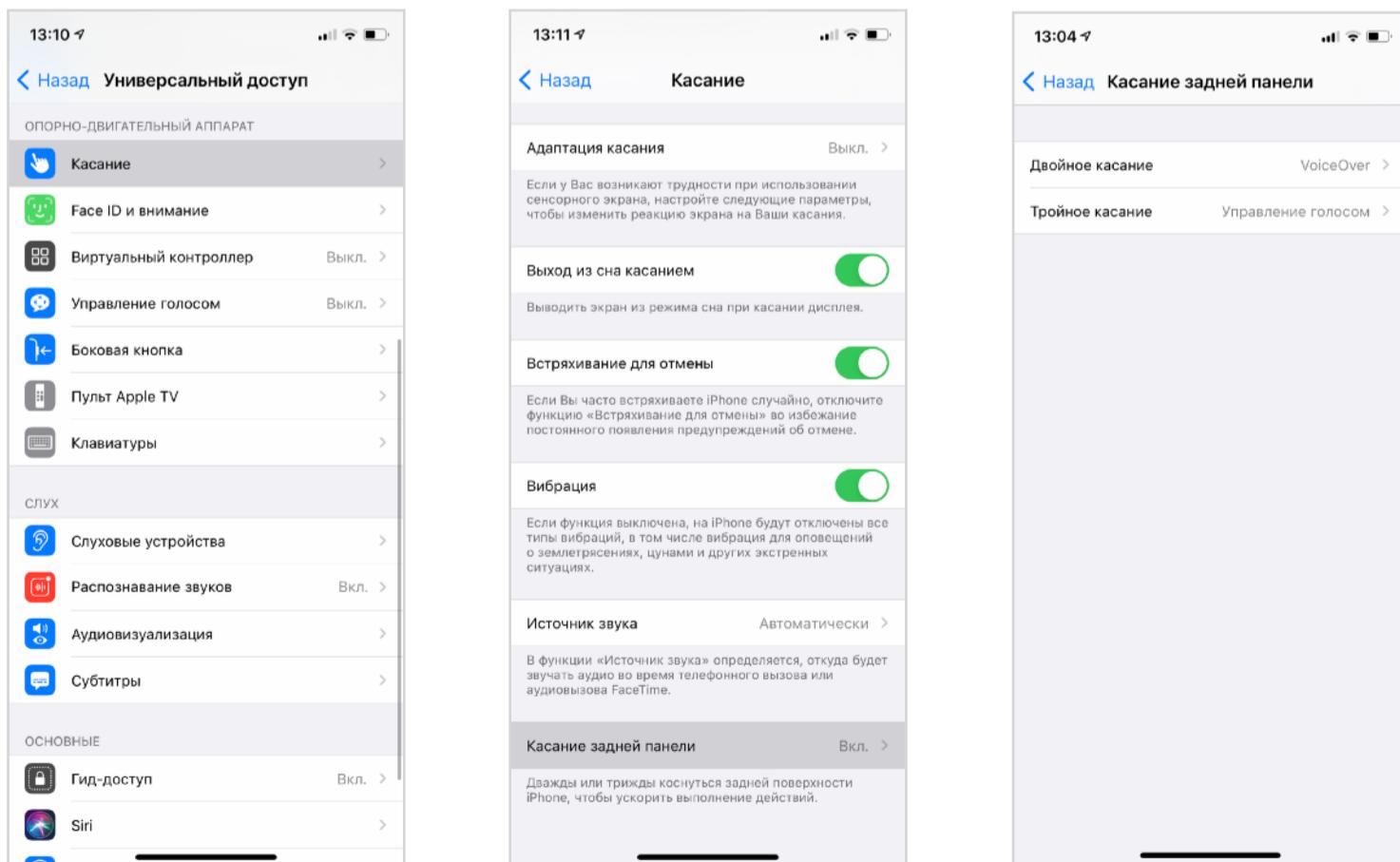
The screenshots show the following navigation steps:

- First Screenshot:** Shows the main **Настройки** (Settings) screen with various categories like Основные (General), Пункт управления (Control Center), Экран и яркость (Screen & Brightness), etc.
- Second Screenshot:** Shows the **Универсальный доступ** (Universal Access) section under the **ЗРЕНИЕ** (Vision) heading. It lists settings for VoiceOver, Увеличение (Zoom), Лупа (Loupe), Дисплей и размер текста (Display & Text Size), Движение (Motion), Устный контент (Speakable Results), and Аудиодескрипция (Audio Description).
- Third Screenshot:** Shows the **VoiceOver** settings screen. At the top, there is a large green switch labeled "VoiceOver". Below it, the text says "VoiceOver произносит названия объектов на экране:" followed by two bullet points: "• Коснитесь для выбора объекта." and "• Коснитесь дважды для активации выбранного объекта.". There is also a "Подробнее..." (More info...) link. Further down, there are sections for **СКОРОСТЬ РЕЧИ** (Speech Rate) with a slider, **Речь** (Speech), **Брайль** (Braille), **Распознавание VoiceOver** (VoiceOver Recognition), **Детализация** (Detail), **Аудио** (Audio), **Команды** (Commands), and **Пейджинг** (Paging). The bottom of the screen shows a navigation bar with "Настройки", "VoiceOver", and "Кнопка-переключатель, вкл., Коснитесь д....

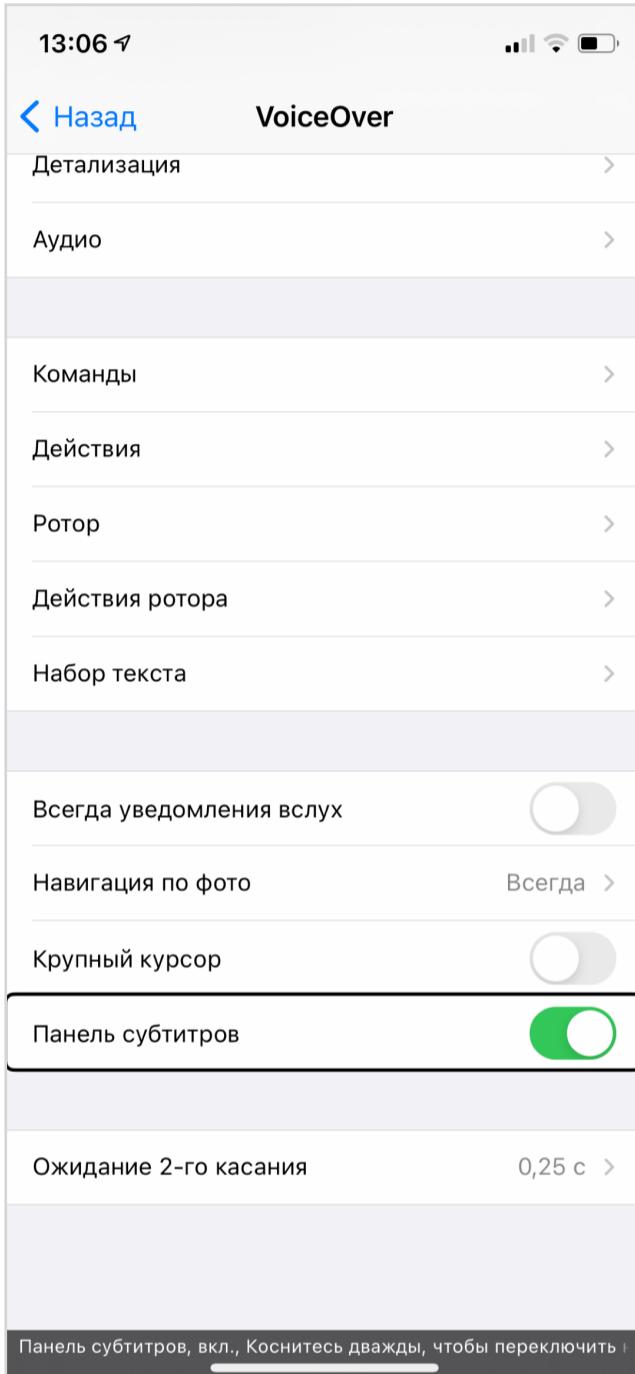
Разные варианты доступности можно активировать по тройному нажатию на кнопку включения. Ищите «Быстрые команды» в конце экрана «Универсальный доступ».



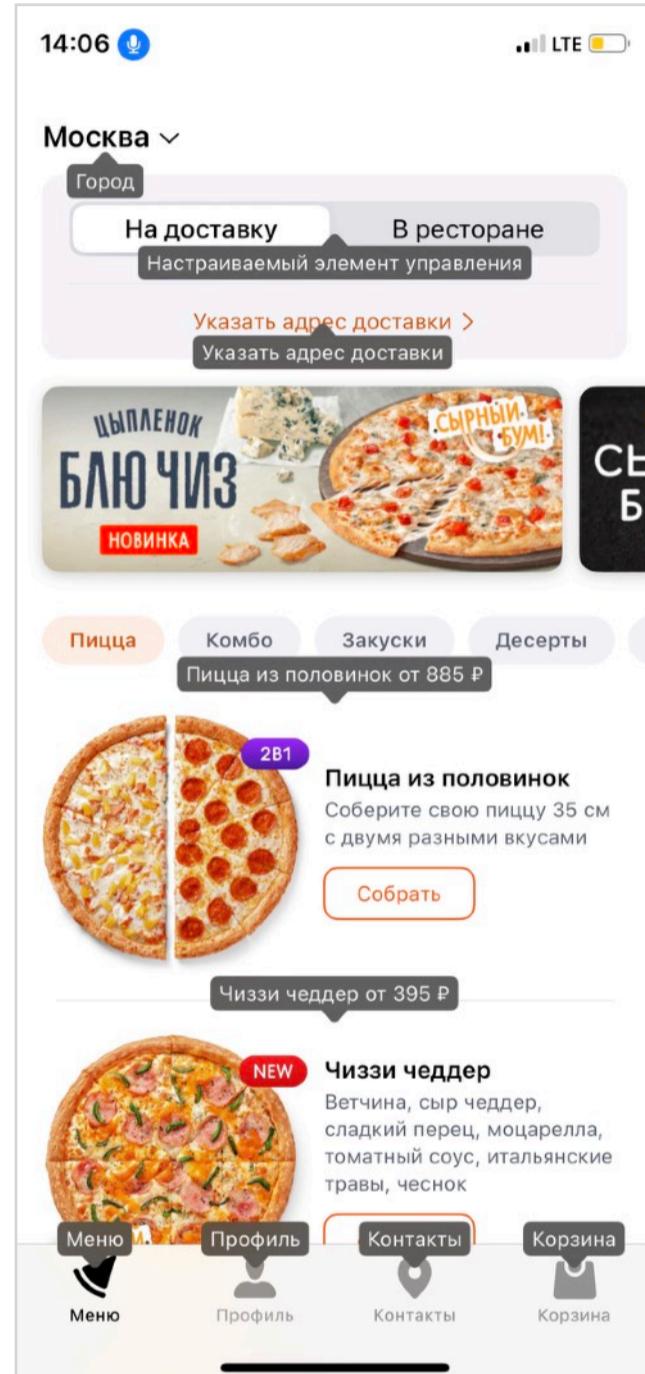
Удобно настроить включение VoiceOver на двойной (или тройной) тап по спинке телефона. Я пользуюсь этим способом чаще всего.



Для отладки удобно не слушать VoiceOver, а смотреть текст который он читает. Включив субтитры вы увидите описание текущего элемента. Voice Control 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 VoiceOver. Фокус перемещается по элементам от свайпов по экрану. Когда он встает на новый элемент он читает его описание.

У такого интерфейса сразу несколько следствий:

- Работать можно только с одним элементов за раз.
- Все действия с экрана передаются элементу в фокусе.
- Не нужно прицеливаться на элемент. Свайп в любом месте экрана будет влиять на элемент в фокусе.
- Если хотите прочитать описание еще раз, то нужно тапнуть по экрану.

Чтобы перейти к следующему контролу нужно свайпнуть вправо. Это логично, 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. Все 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 влезит 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, VoiceOver будет читать элемент который находится под вашим пальцем. Удобно при отладке программ, но незрячие пользуются этим редко, чаще всего, когда что-то в программе сломалось.

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

## Действия

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

**Нажать кнопку** можно тапнув дважды. Тапать можно в любом месте экрана, действие передастся элементу в фокусе.

Попробуйте двойным тапом открыть программу. Если у вас айфон «с челкой», то закрыть ее можно привычным свайпом снизу, только чуть медленнее: вы должны почувствовать легкую вибрацию, после этого палец можно отпустить и программа свернется.

**Проскролить экран** можно свайпнув тремя пальцами в нужном направлении. Полистайте главный экран айфона со значками.

Если вы хотите погрузиться во взаимодействие незрячих людей с интерфейсами, то можете **отключить экран** тапнув трижды тремя пальцами. Включается также.

Можно еще в некоторых случаях свайпать вертикально, но про это расскажу позже в главах «Элемент регулировки», «Контекстные действия» и «Ротор».

Мы познакомились с несколькими базовыми жестами, но у VoiceOver очень их намного больше. Я не буду рассказывать сейчас про все, какие-то упомяну в тексте позже, а полный список можно посмотреть [в гайде](#).

## Как работает VoiceOver

# До и после

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

Я запускаю приложения и оказываюсь на каком-то экране. Чтобы понять что на нем я свайпаю двумя пальцами вверх, чтобы VoiceOver прочитал все элементы на экране. Я могу пройтись по элементам вручную, но это будет долго, ведь на экране больше 22 элементов.

Прочитайте. Можете понять на каком экране мы находимся? 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 все 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.

Элементов немного, у всех указан тип, понятно как взаимодействовать с элементами, 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.

Сравните, насколько экраны похожи по восприятию. При этом, мы не сделали ничего сложного: только дали всем элементам тип, немного подправили описание и сгруппировали элементы. При этом 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 выглядит экран меню для зрячего. Много элементов и видов контроллов, но легко понять из каких блоков он состоит.

23:24 1

Москва ▾

На доставку    В ресторане

📍 Москва, улица Наметкина, 13Б >

Пицца    Комбо    Закуски    Десерты

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

Собрать

**Пепперони-сердце**  
Пикантная пепперони,  
моцарелла, томатный соус

625 ₽

Меню    Профиль    Контакты    Корзина

2

**Сколько в адаптированной  
версии занимают 22 элемента?**

**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 должны обратиться к корневому виду и спросить у него: есть ли у него **доступные элементы**? Чтобы отвечать на вопрос `UIView` реализует протокол `UIAccessibilityContainer` и может пройтись по всем своим дочерним элементам чтобы спросить у них, являются ли они доступным элементом. Маркер простой: `isAccessibilityElement` должен быть `true`.

Если элемент не доступен, то может быть он является **контейнером** для других доступных элементов? Тогда надо у всех по очереди спросить есть ли у них элементы. Вложенность может быть большой, но в конце только два варианта: либо сама виду является доступным элементом, либо дочерних элементов больше нет совсем.

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

Элемент мы нашли и обвели. Теперь нужно рассказать о нем. **Описание элемента** хранится в `accessibilityLabel`. Если у элемента есть какое-то **значение**, то оно хранится в `accessibilityValue`, читается после короткой паузы и немного другим голосом. Это удобно и создает динамику в речи VoiceOver.

У контрола может быть одно из стандартных свойств: его **тип** (надпись или кнопка), **состояние** (обычное, выбранное, отключенное) или **особое свойство** (часто обновляется, начинает проигрывать медиа и т.д.). Обычно, свойство добавится к описанию элемента и как-то меняют поведение VoiceOver с этим элементом.

Вы поняли что это за элемент, что это кнопка и вы хотите на нее **нажать**. Нажать кнопку можно двойным тапом, он вызовет специальный метод `accessibilityActivate()`. Для обычной кнопки он эмулирует нажатие, при этом нажимает **на точку активации** которая указана в `accessibilityActivationPoint`. Обычно это центр кнопки.

Если после открытия кнопки 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 почти со всеми интерфейсами. 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 будет тоже 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 ее 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 интерфейса дополнительной информацией.

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 нужно уметь адаптировать их.

Для адаптации надо сделать 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.

43

44

@interface NSObject : UIAccessibility

45

Добавить в пиццу



Сырный бортик

169 ₽



Шампиньоны

29 ₽



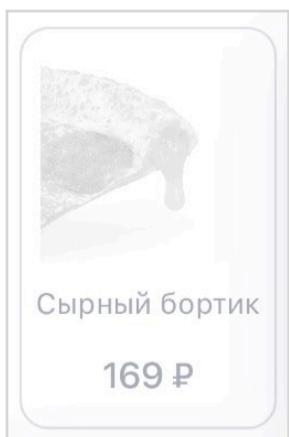
Моцарелла

49 ₽

Чаще всего VoiceOver запрашивает доступность элементов у responder chain, а уже из этих данных создает фокус и накладывает его поверх интерфейса.

Разберем устройств VoiceOver на примере добавок в пиццу. Посмотрим как мы воспринимаем интерфейс, чем он является технически и где нужно помочь VoiceOver.

Добавить в пиццу



Сырный бортик

169 ₽



Шампиньоны

29 ₽



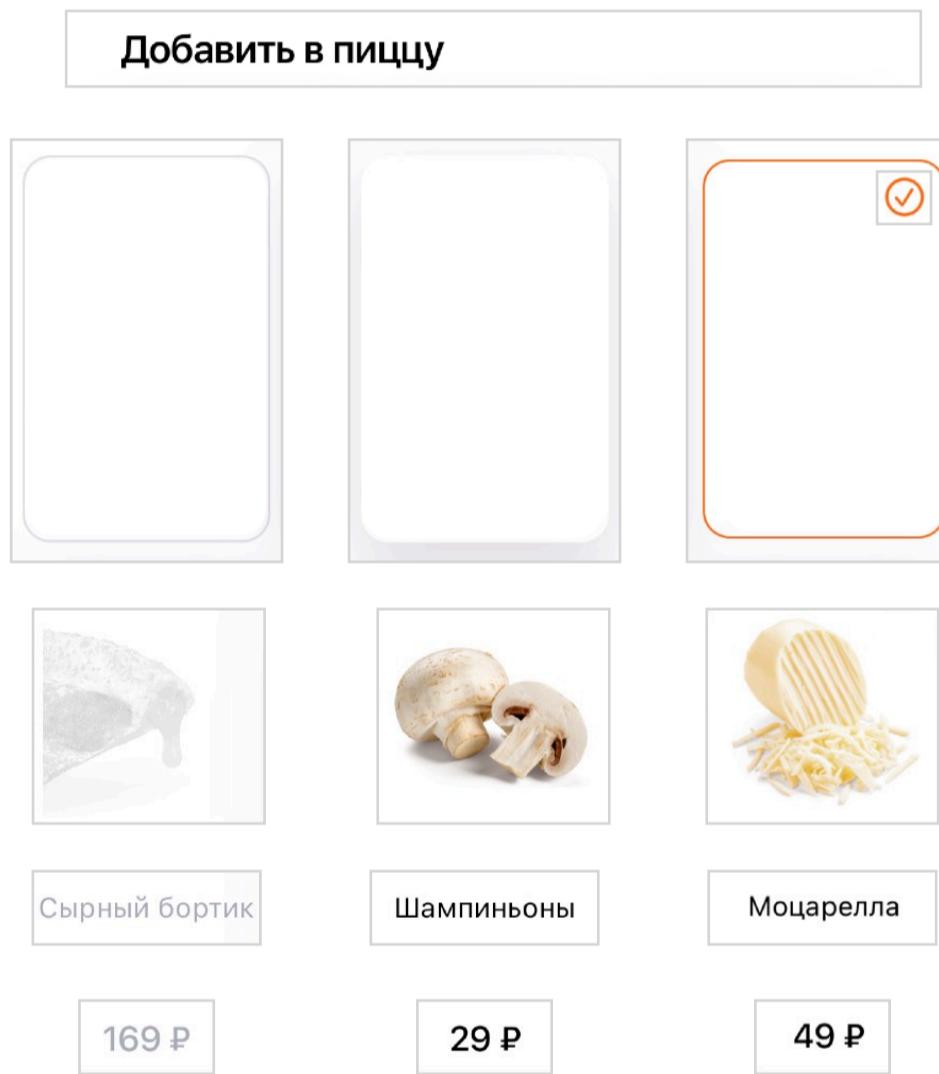
Моцарелла

49 ₽

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,
- 3 картинки,
- 3 названия,
- 3 цены.

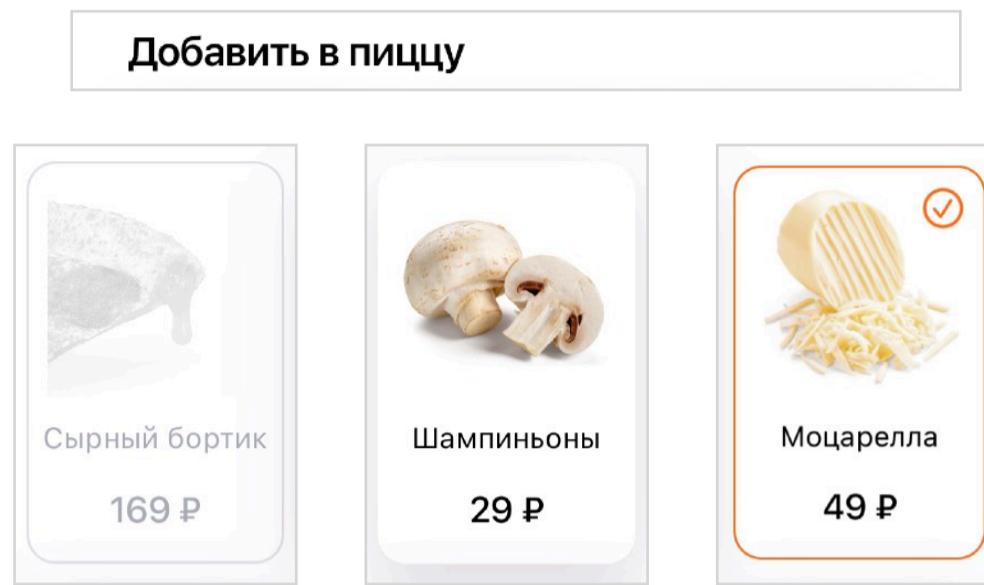


VoiceOver уже имеет ряд встроенных правил по которым она пытается правильно прочитать интерфейс: например, она не фокусируется на контейнерах других элементов, поэтому на ячейке она останавливаться не будет, картинки по умолчанию недоступны и VoiceOver их тоже не видит. А вот надписи и цены VoiceOver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

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

Задача разработчика — подсказать VoiceOver как правильно читать элементы, какие у них свойства, как группировать элементы и что с ними можно делать. По сути, вернуться к первой ментальной модели.

В процессе мы будем разбирать как применить все правила так, чтобы VoiceOver не был многословен, но рассказывал о всем 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 еще и скажет о новой цене всей пиццы.

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

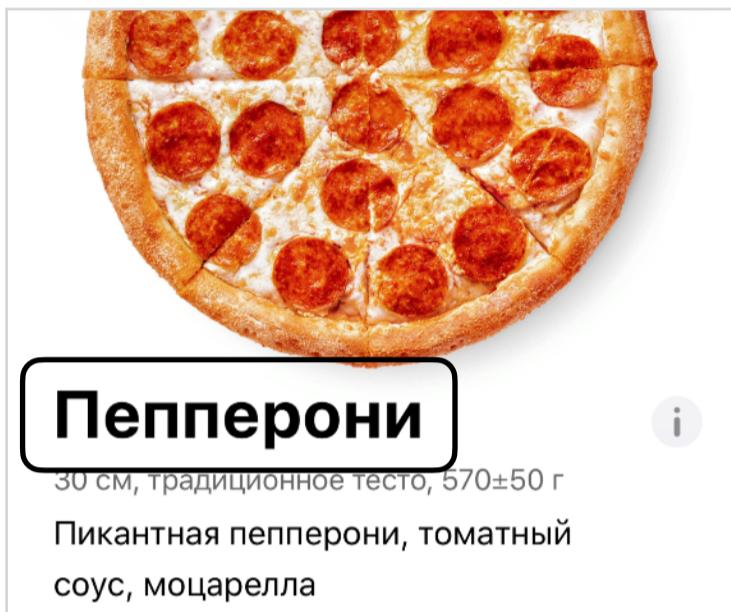
# Подписать

Первая проблема доступности, с которой сталкиваются незрячие: элементы подписаны неправильно или не подписаны совсем. Из-за этого не понятно что с ними можно сделать 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 – базовый элемент доступности.

Обычные подписи уже адаптированы для VoiceOver: они доступный элемент, у них есть название и размеры для фокуса.

Если вы рисуете текст через CATextLayer, то информация о доступности теряется. Восстановить ее можно самостоятельно указав параметры. Разберем такой случай и посмотрим что UILabel делает за нас.

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

```
isAccessibilityElement = true
```

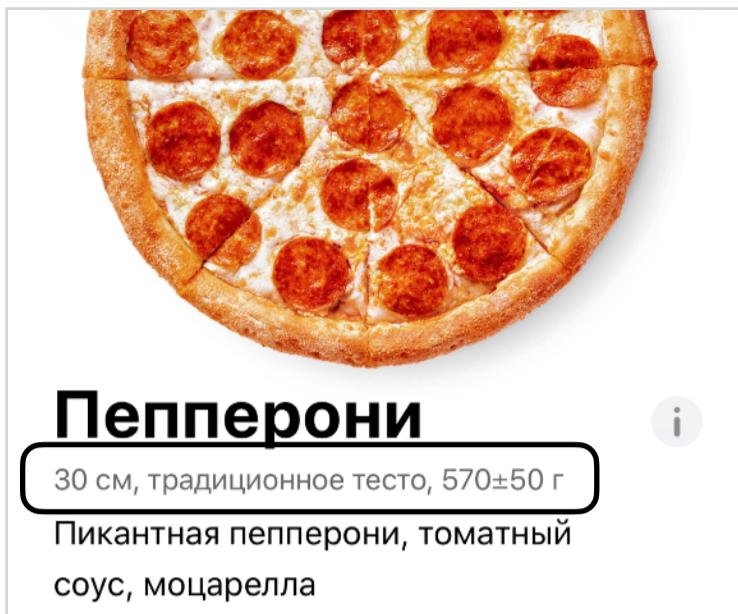
Затем нужно подписать элемент, для описания возьмем текст в элементе. Этот текст VoiceOver прочитает, когда фокус попадет на контрол.

```
accessibilityLabel = text
```

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

```
accessibilityFrameInContainerSpace = frame
```

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 ужасное «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 текст для VoiceOver.

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

Использовать можно напрямую, или завернуть в красивый экстеншен 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.localizedStringWithFormat(  
    NSLocalizedString(@"%@", comment: ""),  
    price)
```

Еще в таком файле можно использовать не только ключ NSStringLocalizedFormatKey, но и NSStringVariableWidthRuleType. С его помощью можно задать разный 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 = Int(UIScreen.main.bounds.width)  
  
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,  
цыплёнок, томаты черри, сыры чеддер и  
пармезан, моцарелла, сливочный соус, соус  
цезарь

[Убрать ингредиенты](#)

i

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 с элементом можно сделать. 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 кнопки обрабатывается в функции accessibilityActivate(). Кнопка вызывается обычное нажатие на себя. Возвращает true если действие обработалось. Если вернется false, то VoiceOver попытается вызывать метод у следующего объекта в responder chain.

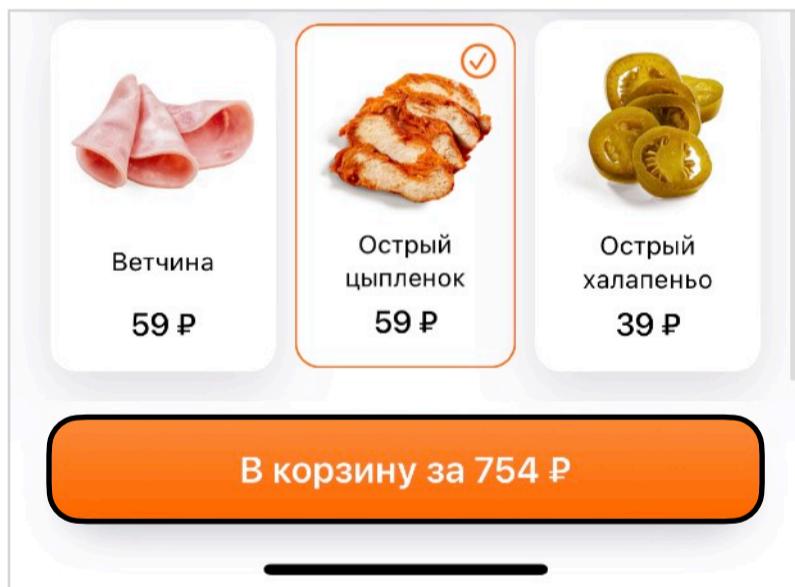
```
override func accessibilityActivate() -> Bool {  
    // sendEvent(.touchUpInside) // По умолчанию  
}
```

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

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

| Наши любимые стикеры   | См. все                  | См. все                   | Смотреть все          |
|--|--------------------------|---------------------------|-----------------------|
|  All The Bad Cats Megapack<br>Стикеры | <a href="#">179,00 ₽</a> | <del>179 знак рубля</del> | <del>179 рублей</del> |
|  Sad Animations<br>Стикеры            | <a href="#">29,00 ₽</a>  | <del>29 знак рубля</del>  | <del>29 рубля</del>   |

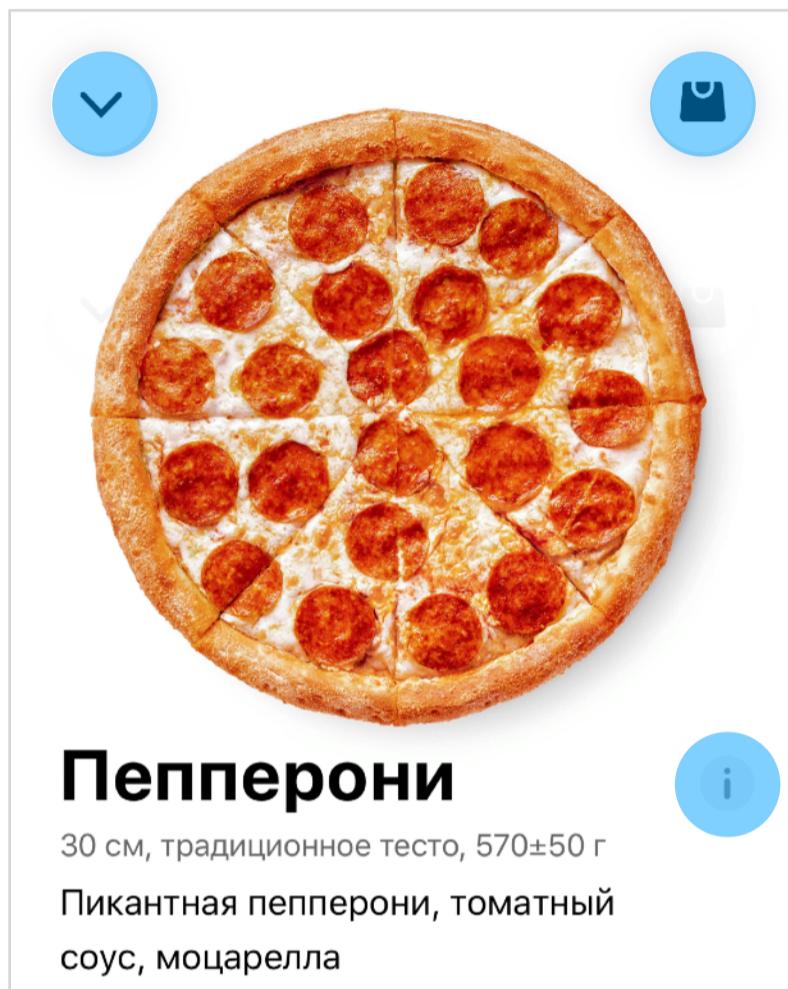
Иногда, для VoiceOver кнопке нужно дать больше информации чем в ней написано. Например, если у вас есть кнопка добавления товара в корзину, то в ней можно подытожить все что вы настроили на экране. Так можно не проходить все контролы снова чтобы убедиться, что ничего лишнего не добавили и количество с ценой такие, какие нужно.



В корзину за 695 рублей,  
*Пицца Песто, убрать «томаты черри»,  
добавить «Острый цыпленок».*  
**Кнопка**

Подписать

# Кнопки без текста



В угоду краткости из многих кнопок убирают текст. Причин для этого много: не ясно какой текст писать, кнопка не важная и не должна быть большой, ее действие очевидно и т.д. Доступность при этом сильно теряется.

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

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

Восстановить доступность легко, достаточно дать кнопкам название в `accessibilityLabel`

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



Элементы в кастомном UITabBarItem подписываются точно также.

Подписать

# Размер кнопки

Размер кнопки должен быть минимум 44 пикселя, это важно для всех людей. Для незрячих чем кнопка больше, тем лучше, так ее легче найти касанием.

Фрейм кнопки можно поменять специально для доступности, для этого есть свойство accessibilityFrameInContainerSpace. Используйте его, особенно если работаете с UIScrollView.

```
accessibilityFrameInContainerSpace  
= bounds.inset(by: .all(10))
```

Можно управлять размером и через accessibilityFrame, но ему нужен фрейм в координатах экрана. Конвертировать можно с помощью специальной функции. UIScrollView работает только с accessibilityFrameInContainerSpace.

```
accessibilityFrame = UIAccessibility  
.convertToScreenCoordinates(bounds.inset(by: .all(10)),  
in: self)
```

В идеале, кнопка должна занимать всю ширину экрана, чтобы можно было пройти по экрану сверху вниз касанием и встретить все кнопки, чтобы ничего не искать по горизонтали.

**Идеальный контрол занимает всю ширину экрана**

Не подписывать

# Изображения



14:02 4G LTE

▼ NEW

## 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](#)

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, 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 можно распознавать изображение и рассказать что на нем. Например, у этой картинки описание «пицца на белом фоне». 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 картинке, то поставьте ей трейт `.image`, VoiceOver добавит к описанию картинки «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 узнать. Не обязательно сообщать об этом на картинке, можно добавить текст «новинка» к названию пиццы.

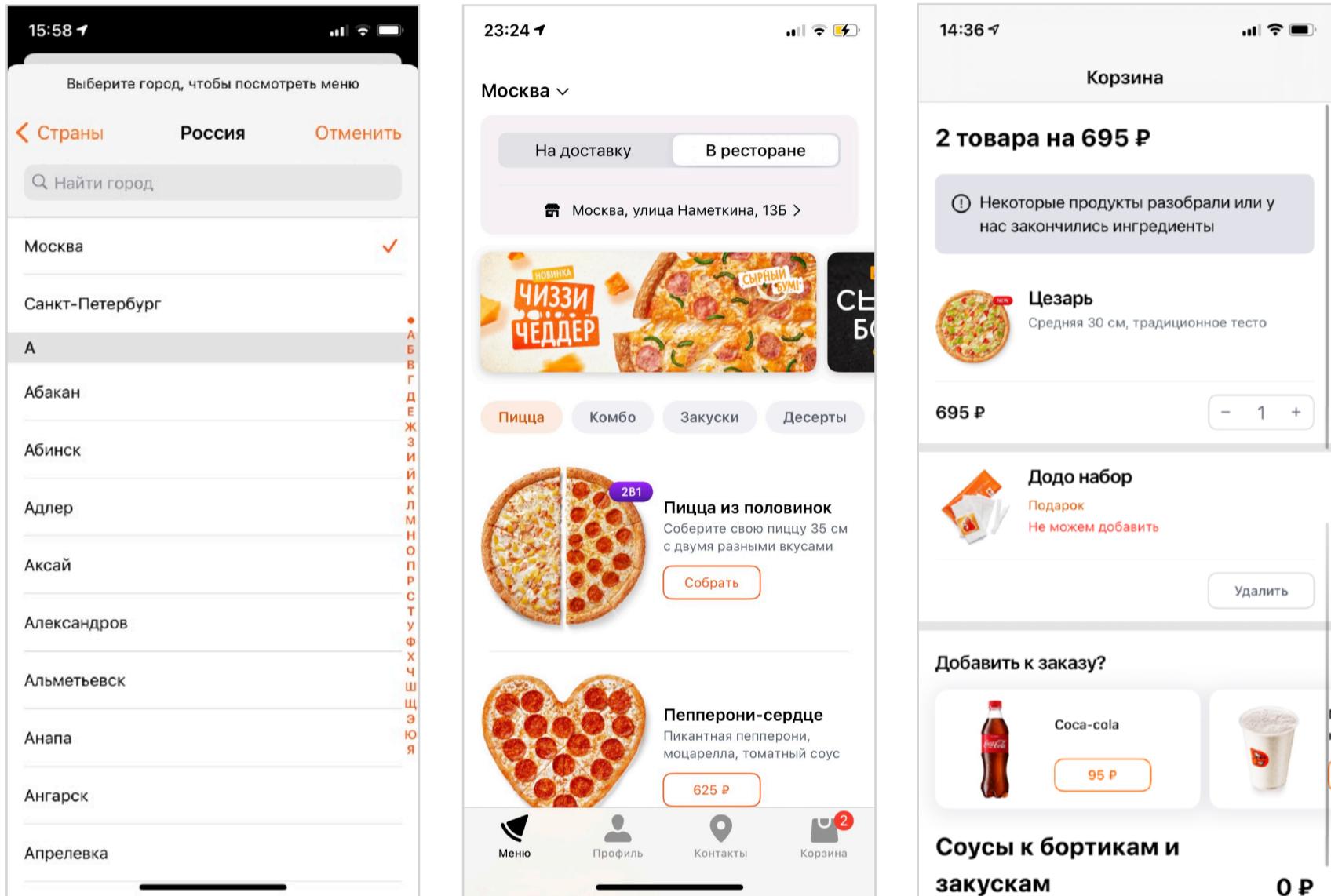
# Упростить

Подписывав все элементы и указав их тип мы исправили самый простой, но массовый слой проблем. Для большинства приложений этого было бы 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.

Упростить

# Списки и ячейки



Большая часть интерфейсов приложений это списки. Списки для доступности это:

- много ячеек в списке
- 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. Всю адаптацию 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, будет разный для каждой ячейки. Самое критичное, что на ячейку можно нажать, но VoiceOver об этом не сообщает.



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  
        }  
    }  
}
```

**Трейт .button нужен и для ячеек  
на которые можно нажать**

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? {
```

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 текущего элемента — трейт .selected.

Трейты можно объединять, ведь элемент может быть одновременно и кнопкой и выбранным.

```
override public var isSelected: Bool {  
    didSet {  
        if isSelected {  
            accessibilityTraits = [.button, .selected]  
        } else {  
            accessibilityTraits = [.button]  
        }  
    }  
}
```

Для правильного чтения времени нужно не только дать другую интонацию, но и упростить сам текст. Нужно, чтобы время читалось как «с 9 до 23», а не «с 9 ноль до 23 ноль». Самый простой способ правильно отформатировать время это задать форматтеру даты стиль .spellOut.

```
if UIAccessibility.isVoiceOverRunning {  
    formatter.numberStyle = .spellOut  
}
```

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 = [.button, .notEnabled]
```

Использовать трейт `.notEnabled` нужно только для тех элементов, с которыми нельзя взаимодействовать. Если адрес я могу выбрать, например, чтобы заказать на завтра, то трейт `.notEnabled` указывать не нужно.

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.



## 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, а чтобы перейти к следующему элементу свайпнуть надо будет минимум три раза. 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.

Картина 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, ведь 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 поставить в value.

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

label: Крэйзи пепперони, от 395 рублей

value: Пикантная пепперони, цыпленок, моцарелла, томатный соус, кисло-сладкий соус.

trait: кнопка

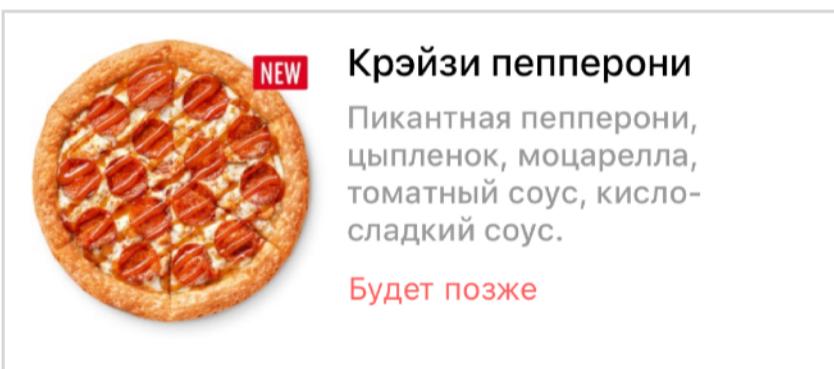
Интересно, что в графическом дизайне порядок восприятия не всегда правильный, иногда опрятность верстки важнее.

Код для доступности достаточно прост: делаем ячейку доступной, уточняем, что у нее 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 замена контроллов не важна, поэтому просто меняем текст цены, чтобы получилось:

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
}
```