нетология

Особенности разработки приложений под разные платформы и ОС

Алексей Жемловский

Алексей Жембловский

iOS Engineer, компания EPAM

План занятия

1. App Extensions
2. iPad OS
3. tvOS
4. watchOS
5. Итоги
6. Домашнее задание

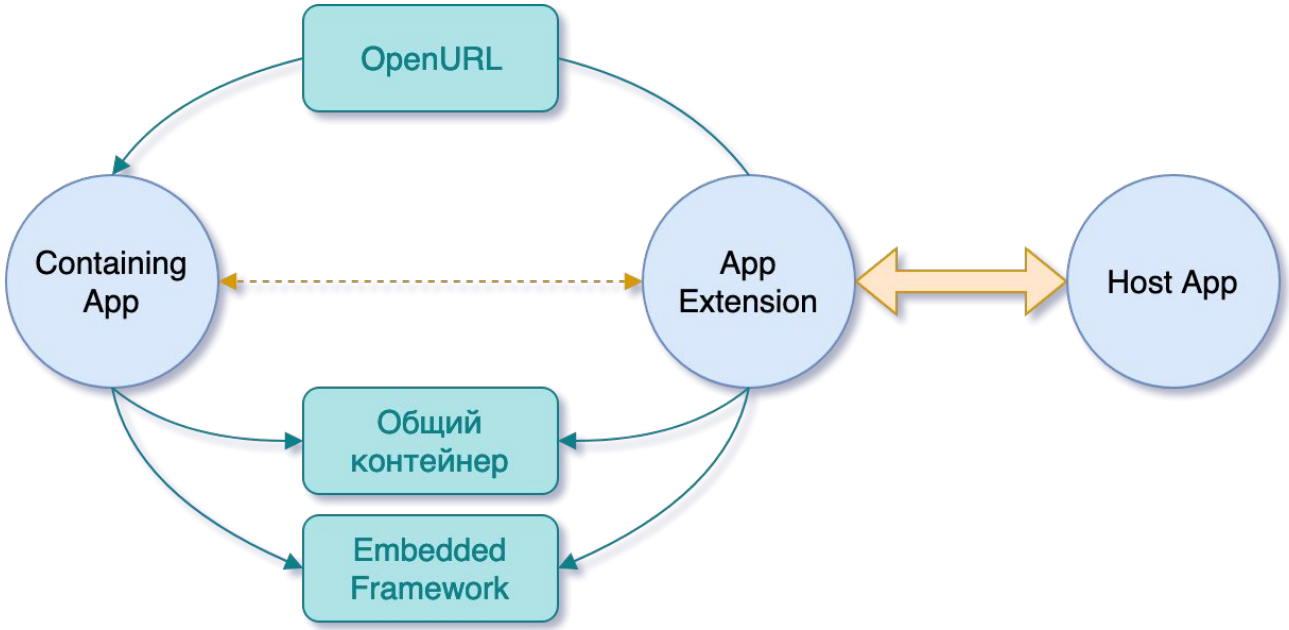
App Extensions

App Extensions расширяют функциональность и контент приложения за его границами, дают возможность взаимодействовать с ним из других приложений или из операционной системы.

Возможности App Extensions

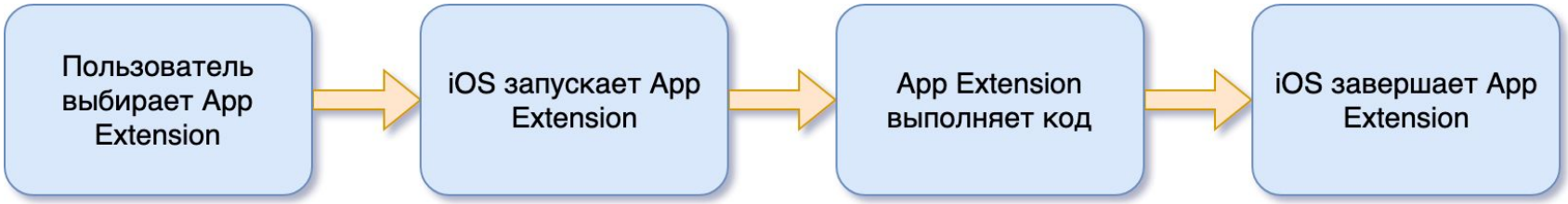
- поделиться контентом из приложения
- добавить виджет на рабочий стол
- предоставить набор дополнительных действий в action sheet
- добавить фильтры внутри системного приложения Фото
- анализатор трафика (+ vpn)
- добавить возможность логина через apple id
- идентифицировать входящие звонки
- улучшить работу Siri с вашим приложением
- хранить ваши пароли
- блокировать нежелательный контент
- создавать кастомные клавиатуры и стикеры для iMessage
- проводить индексацию вашего контента в Spotlight
- включать музыку в фоне и многое другое

Взаимодействие с App Extensions



- Containing App – Host App. Не взаимодействуют друг с другом.
- App Extension – Host App. Взаимодействуют с использованием IPC.
- App Extension – Containing App. Непрямое взаимодействие. Для обмена данными используются App Groups, а для общего кода – Embedded Frameworks. Запустить Containing App из App Extensions можно с помощью URL Schemes.

Жизненный цикл App Extension



1. Пользователь выбирает App Extension через Host App.
2. Host App отправляет запрос App Extension.
3. iOS запускает App Extension в контексте Host App и устанавливает между ними канал связи.
4. Пользователь выполняет действие в App Extension.
5. App Extension завершает запрос от Host App, выполняя задачу, или запускает фоновый процесс для ее выполнения; по завершении задачи результат может быть возвращен Host App.
6. Как только App Extension выполнит свой код, система завершает этот App Extension

Обмен данными: App groups

- Containing App и App Extension имеют собственные ограниченные участки файловой системы, и только они имеют к ним доступ.
- Чтобы Containing App и App Extension имели общий контейнер с доступом на чтение и запись, нужно создать для них App Group.
- App Group создается в Apple Developer Portal. В настройках Containing App переходим на вкладку Capabilities, активируем App Groups и выбираем созданную группу, аналогично для App Extension.

Обмен данными: App groups

Через UserDefaults:

```
UserDefaults(suiteName: "group.com.maxial.onemoreapp")
```

Через NSFileCoordinator и NSFilePresenter:

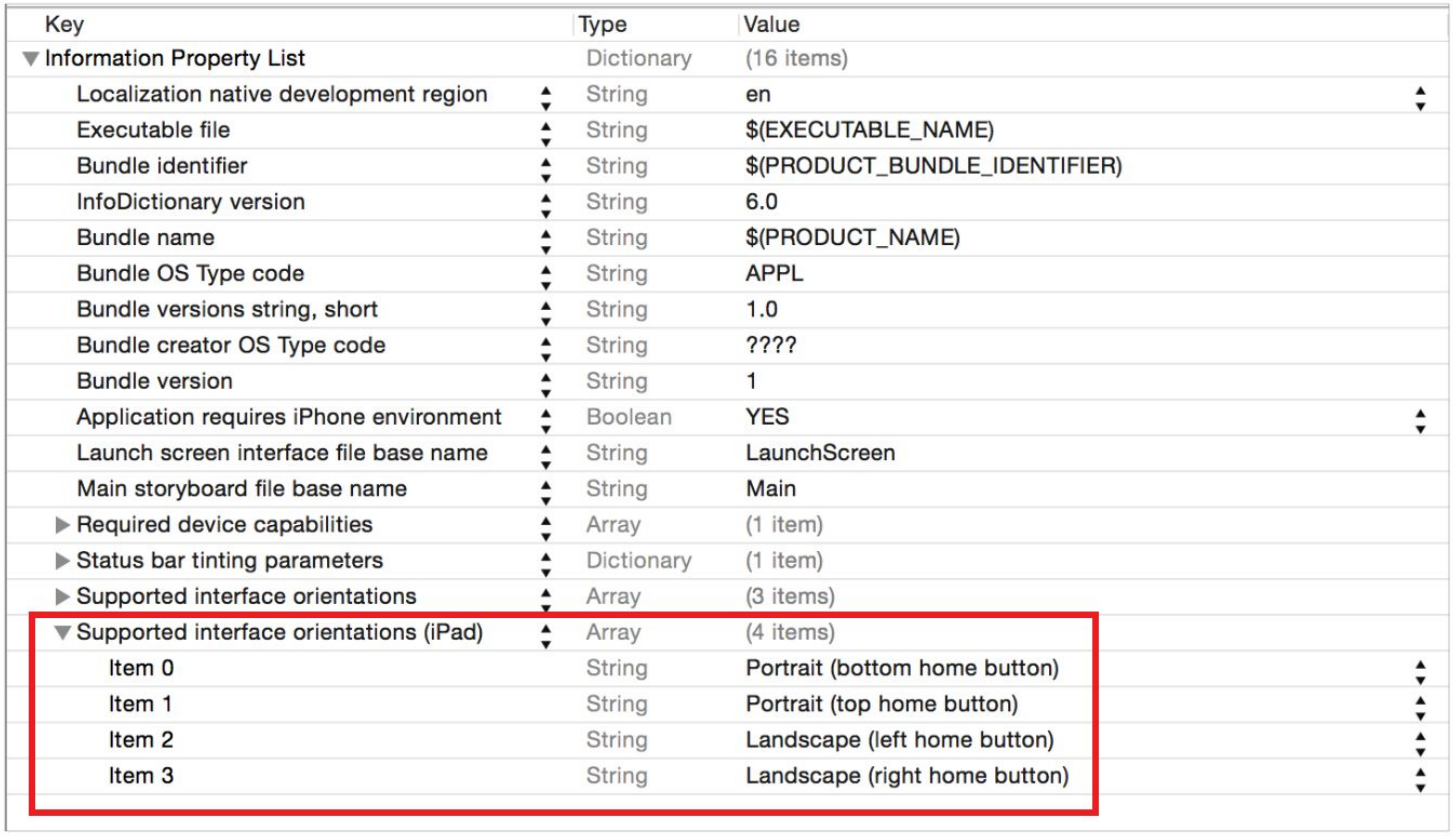
```
let sharedUrl =
FileManager.default.containerURL(forSecurityApplicationGroupIdent
ifier: "group.com.maxial.onemoreapp")
```

Через CoreData:

```
class SharedPersistentContainer: NSPersistentContainer {
    override open class func defaultDirectoryURL() -> URL {
        var storeURL =
FileManager.default.containerURL(forSecurityApplicationGroupIden
tifier: "group.com.maxial.onemoreapp")
        storeURL =
storeURL?.appendingPathComponent("OneMoreApp.sqlite")
        return storeURL!
    }
}
```

iPad split screen

Для включения ориентаций используем в split следующие данные:



Key	Type	Value
▼ Information Property List	Dictionary	(16 items)
Localization native development region	String	en
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1
Application requires iPhone environment	Boolean	YES
Launch screen interface file base name	String	LaunchScreen
Main storyboard file base name	String	Main
▶ Required device capabilities	Array	(1 item)
▶ Status bar tinting parameters	Dictionary	(1 item)
▶ Supported interface orientations	Array	(3 items)
▼ Supported interface orientations (iPad)	Array	(4 items)
Item 0	String	Portrait (bottom home button)
Item 1	String	Portrait (top home button)
Item 2	String	Landscape (left home button)
Item 3	String	Landscape (right home button)

Можно отключить все данные, используя ключ `UIRequiresFullScreen true`. Также пользователь может это отключить. Для этого нужно открыть настройки приложения в

iPad split screen

Можно отключить все данные, используя ключ `UIRequiresFullScreen true`.
Пользователь может это отключить сам. Для этого нужно открыть настройки приложения в системных настройках устройства

Возможности iPad split screen

Когда оба приложения находятся на экране, оба находятся в foreground состоянии с равными правами, но при этом главное приложение:

- Управляет статус баром
- Может работать со вторым физическим экраном
- Может иметь режим картинки в картинке
- Может занимать ⅔ физического экрана

В режиме разделения окна пользователь может менять размеры окна вашего приложения, вращая девайс или меняя размер ползунком.

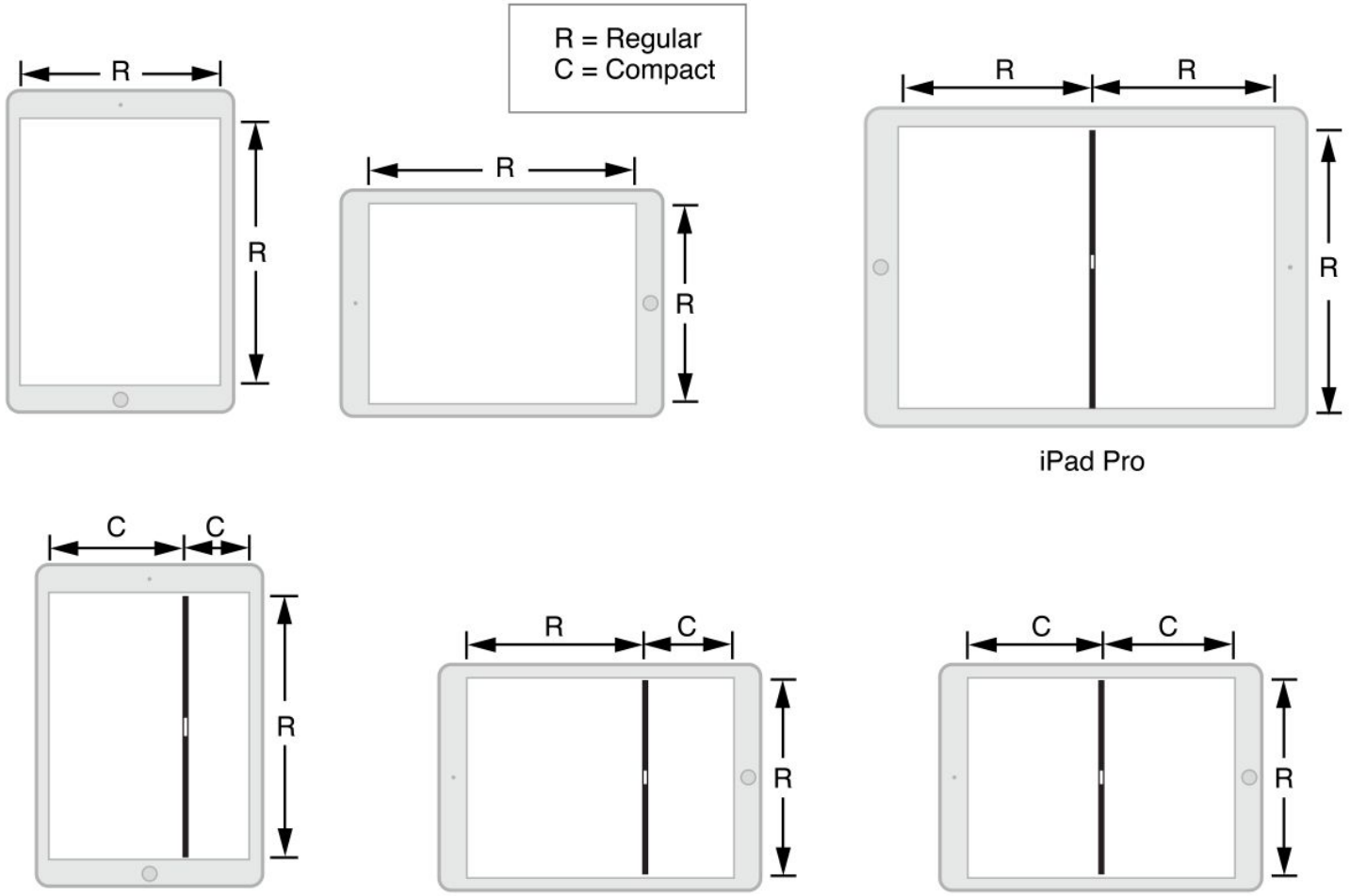
Адаптация iPad split screen

Для правильной адаптации приложения к такому режиму использования, ваше приложение должно быть адаптивным:

1. Использовать AutoLayout вместе с size classes
2. Реагировать на изменение `traitCollection`, sizeChanges UITraitEnvironment and UIContentContainer protocols.
3. Реагировать на изменение состояния приложения. Когда пользователь меняет ползунок, вызывается `willResignActive`, когда приложение смахивается вызывается `didEnterBackground`.

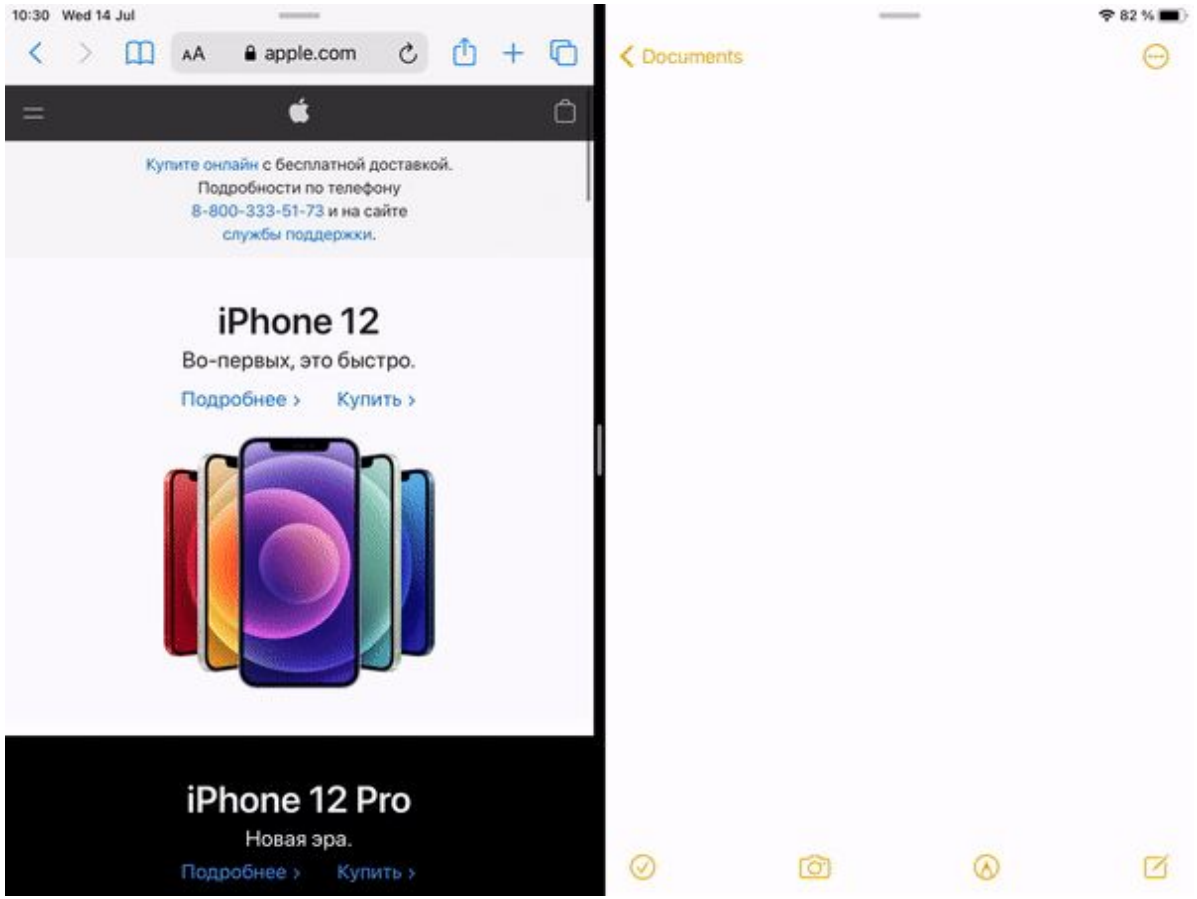
Важно: при изменениях должны сохраняться навигационный стек, данные приложения и текущий скролл

iPad split screen



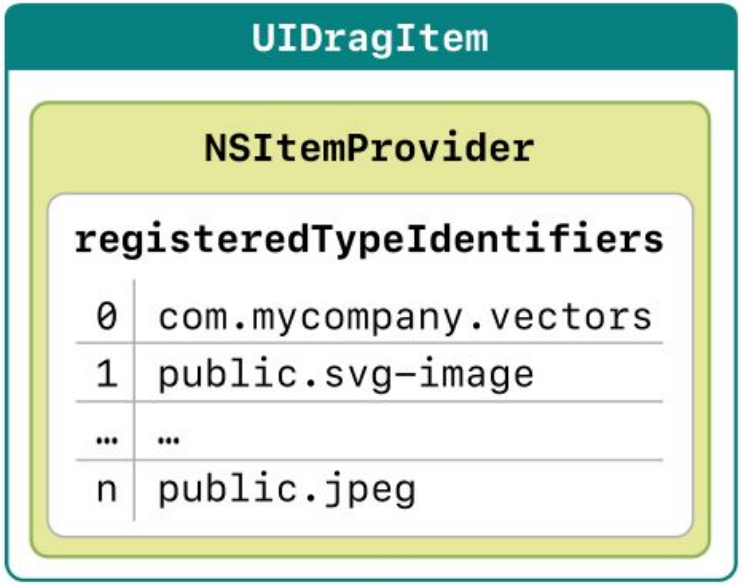
iPad drag and drop

С помощью drag and drop можно перемещать контент из одного места в приложении в другое или из одного приложения в другое, используя жест перемещения.

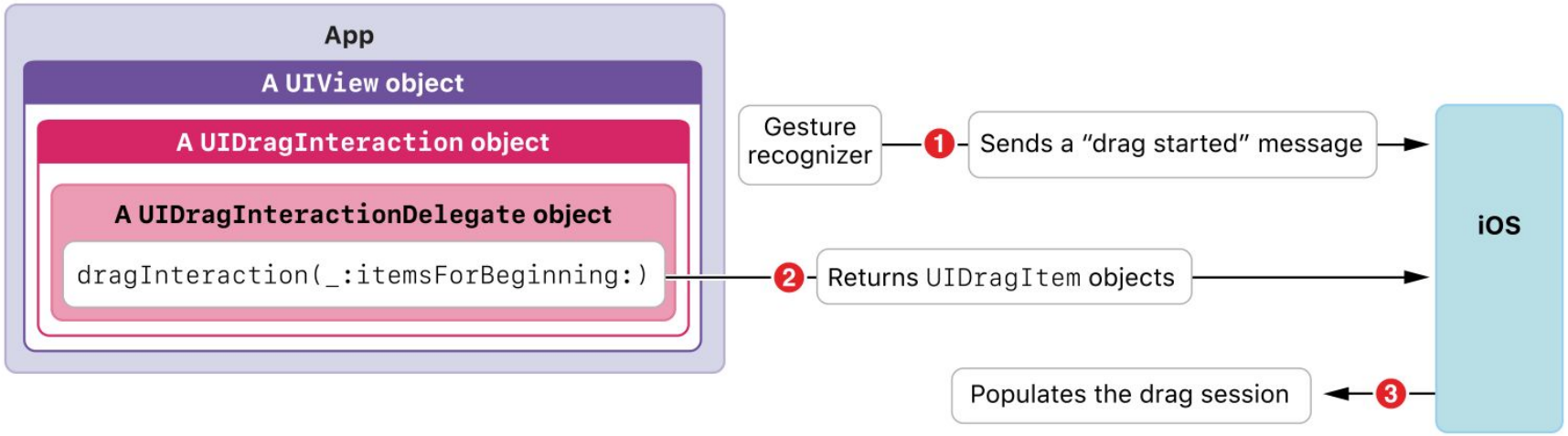


iPad drag and drop

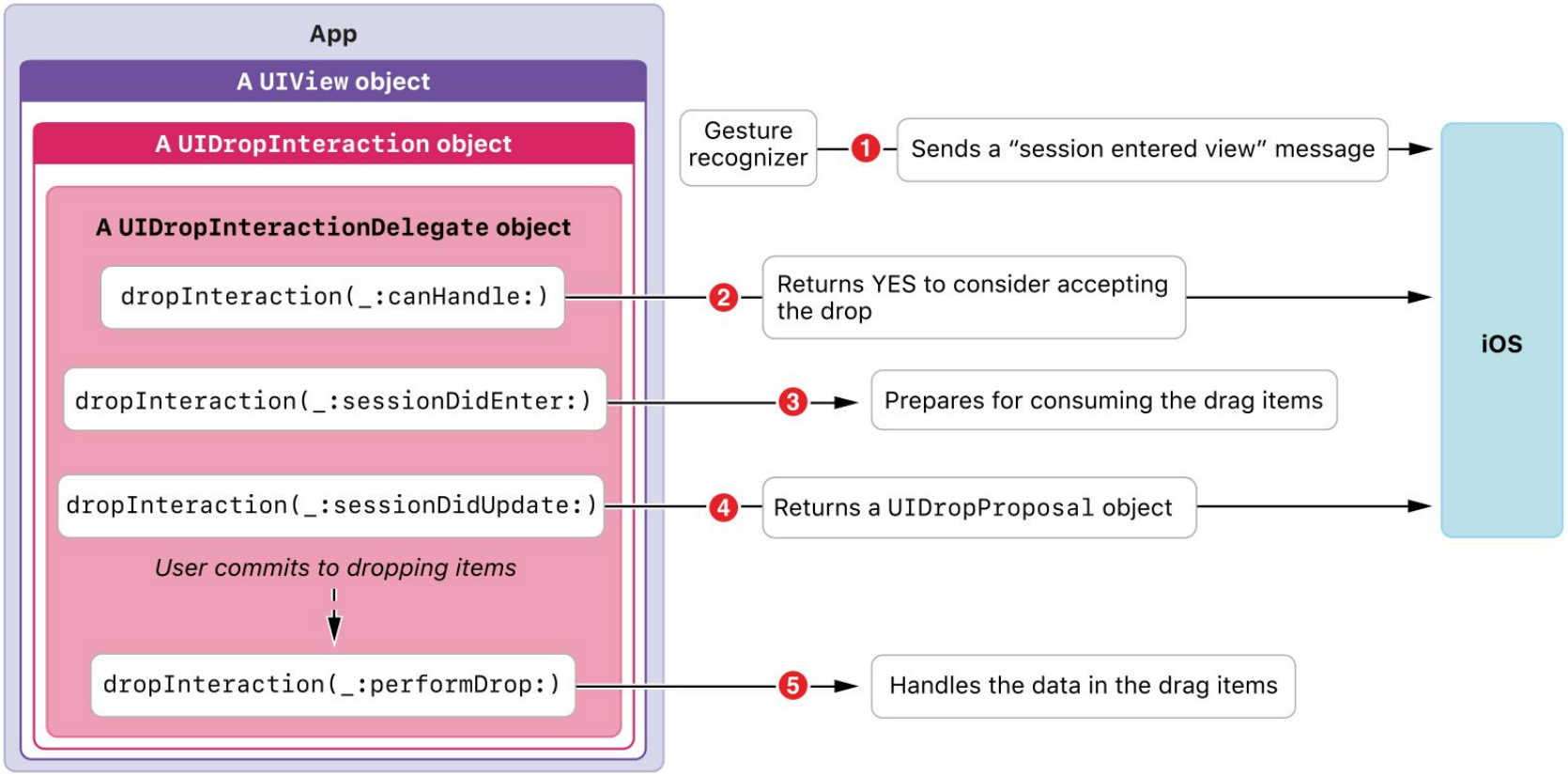
- Текстовые поля поддерживают функцию drag and drop автоматически, таблицы предоставляют свое API, можно адаптировать и для своих view.
- Для перемещения системой iOS создается `UIDragSession`, которая включает в себя `UIDragItem`.



Drag-взаимодействие с системой



Drop-взаимодействие с системой



tvOS

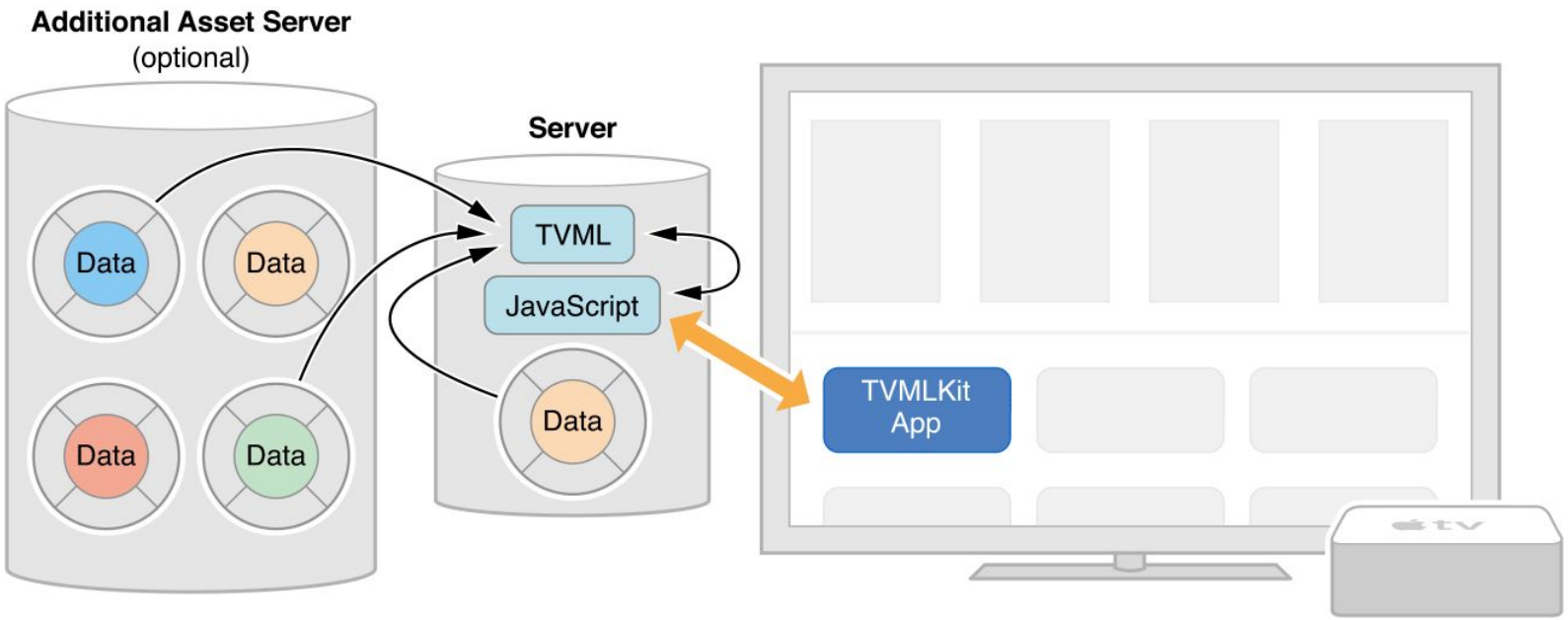
- Для Apple TV можно писать приложения, точно также как и для iOS, только со своими тонкостями и способом ввода (пульт).
- Можно создавать игры, обычные приложения, медиаприложения, используя те же техники и фреймворки, как в iOS.
- Основная задача приложений для tvOS – это стриминг контента, используя веб-технологии (HTTPS, XMLHTTPRequest, DOM and JavaScript).
- Apple дает доступ к своему языку разметки TVML для создания интерфейсов и поведения через JavaScript.

tvOS

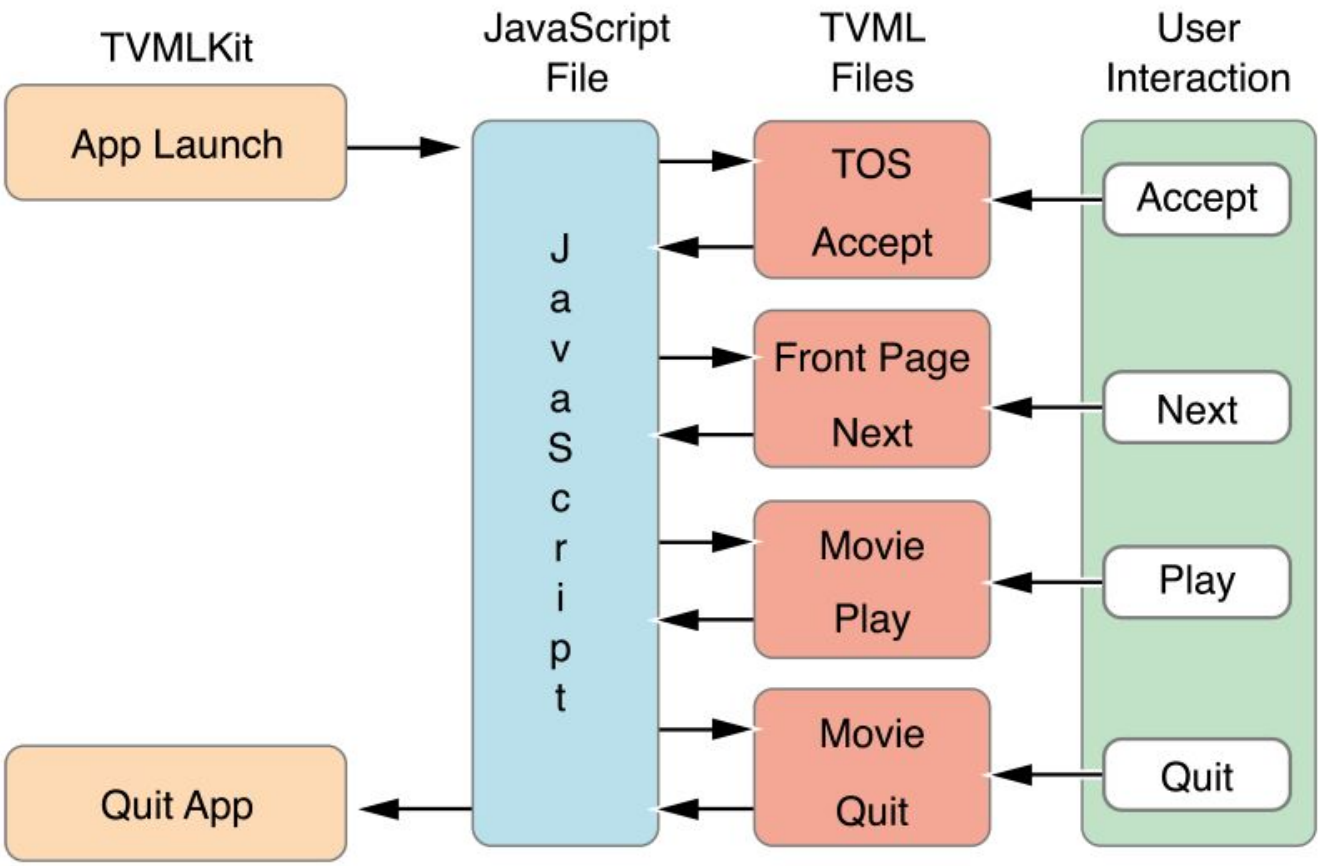
Есть 3 фреймворка для tvOS:

- TVMLJS – API JavaScript для загрузки TVML-страниц
- TVMLKit – предоставляет доступ к взаимодействию между JavaScript и TVML-элементами
- TVServices – для создания Top Shelf

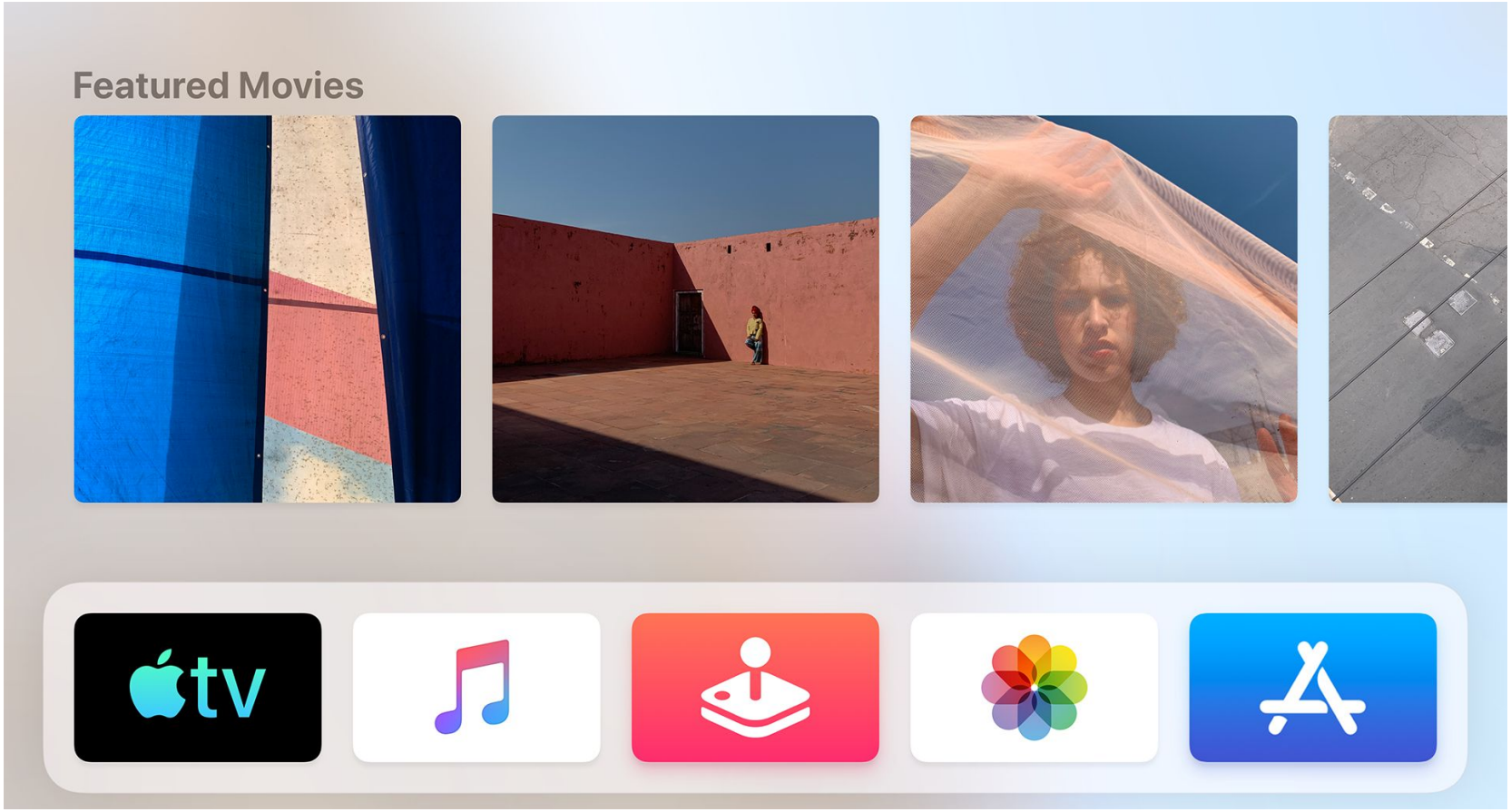
Создание TVML клиент-серверного приложения



Пример навигации в TVML-приложении



Top Shelf



watchOS

2 extensions:

- WatchKit App – интерфейс
- WatchKit Extension – остальное

Особенности разработки:

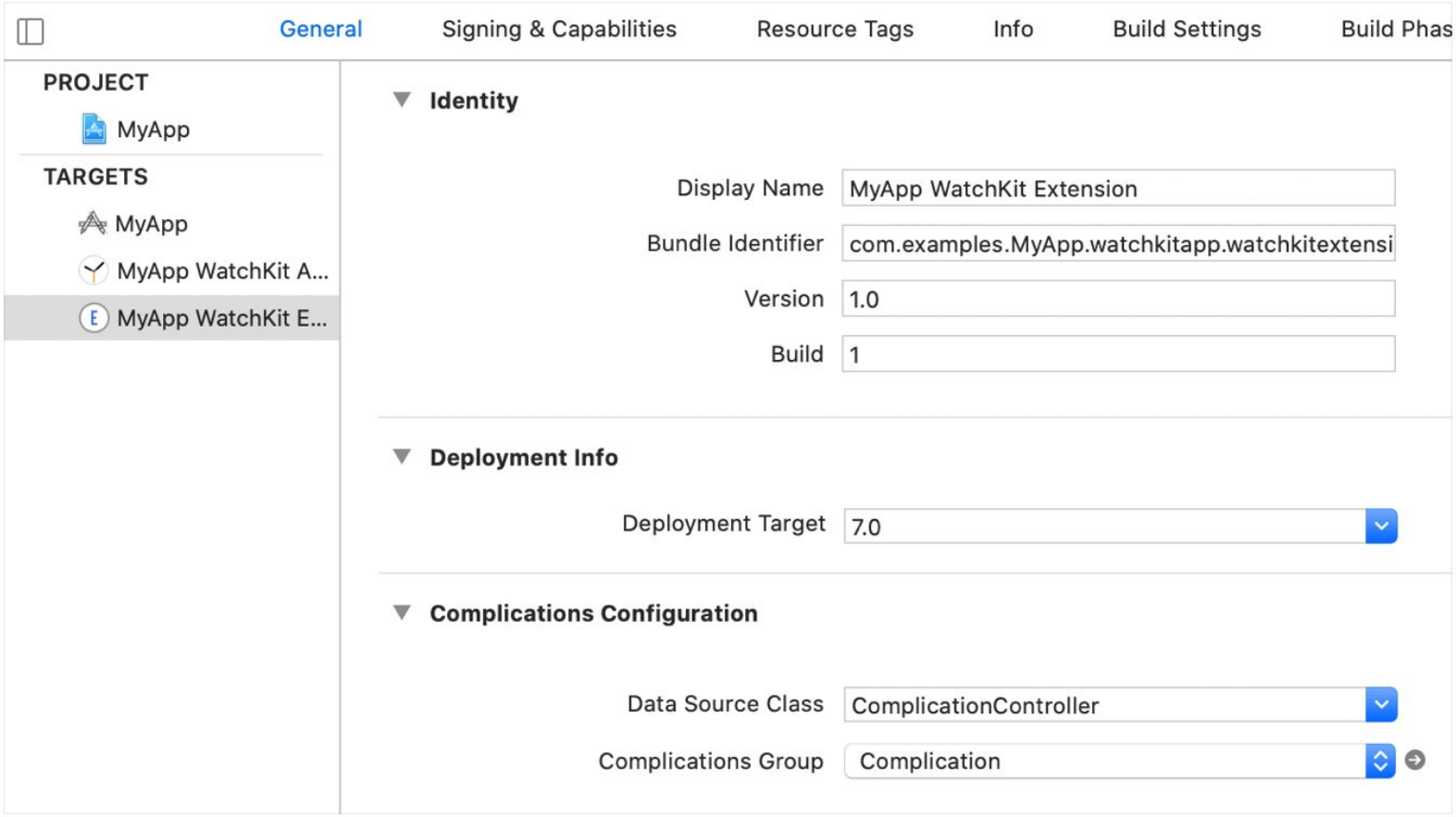
- Для интерфейса используется WatchKit
- Нет клавиатуры (только голос или рукописный ввод)
- Другой подход при работе с таблицами
- Нельзя позиционировать элементы относительно друг друга
- Маленький экран и батарея

watchOS Complications

Для поддержки циферблатов нам нужно:

- Создать класс, поддерживающий `CLKComplicationDataSource`, и реализовать его методы.
- Создать ассет-группу для заглушек (набор картинок), которые будут показываться при настройке циферблата.

watchOS Complications



watchOS Complications

```
import Foundation
import ClockKit

class ComplicationController: NSObject, CLKComplicationDataSource
{
       func getCurrentTimelineEntry(for complication:
CLKComplication, withHandler handler: @escaping
(CLKComplicationTimelineEntry?) -> Void) {
           // TODO: Finish implementing this required method.
       }
    }
```

watchOS Complications

ClockKit использует провайдеры для текстовых данных, картинок и других данных:

- CLKSimpleTextProvider
- CLKDateTextProvider
- CLKTimeTextProvider
- CLKRelativeDateTextProvider
- CLKTimeIntervalTextProvider
- CLKImageProvider
- CLKFullColorImageProvider
- CLKSimpleGaugeProvider
- CLKTimeIntervalGaugeProvider

watchOS Complications

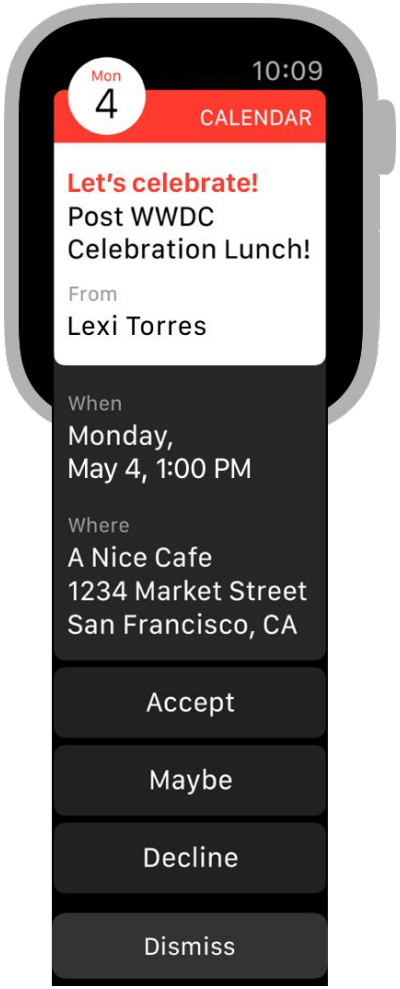
Создаем шаблон и формируем из него TimeLineEntry:

```
let template = CLKComplicationTemplateModularSmallStackText(
        line1TextProvider: temperatureProvider,
        line2TextProvider: nameProvider)
CLKComplicationTimelineEntry(date: Date(), complicationTemplate:
template)
```

watchOS Notifications

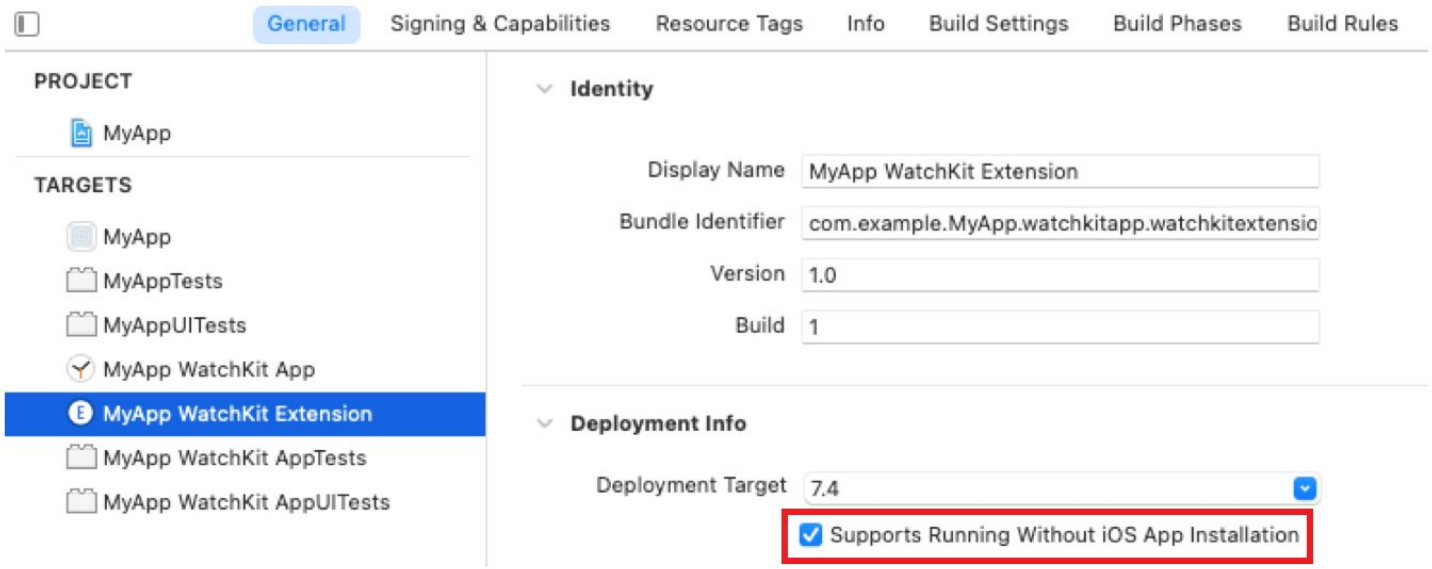
Short look

Long look



watchOS Independent apps

До watchOS 5 все приложения были зависимы от основного приложения на телефоне. Теперь можно создавать полностью независимые приложения.



Зависимые от компаньона приложения создаются только если нам нужно взаимодействовать с приложением на телефоне.

watchOS Independent apps

Главными отличиями независимого приложения от зависимого являются:

- Можно создавать аккаунты и логинить пользователя
- Запрашивать доступы от системы часов
- Загружать данные непосредственно на часы. Независимые приложения не могут надеяться на Watch Connectivity framework (фреймворк для общения часов и телефона) для передачи данных с компаньона. Поэтому, для синхронизации между девайсами используют CloudKit.
- Принимать push-уведомления

Итоги

- Благодаря расширениям мы можем создавать как и узкоспециализированные приложения, так и расширять функционал за его пределами. Обмениваться данными можно через App Groups.
- Современный iOS-разработчик должен уметь разбираться как писать код под разные платформы и экосистемы, а также расширять функциональность для расширений. Необходимость в расширениях только возрастает.

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** учебной группы.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Алексей Жембловский