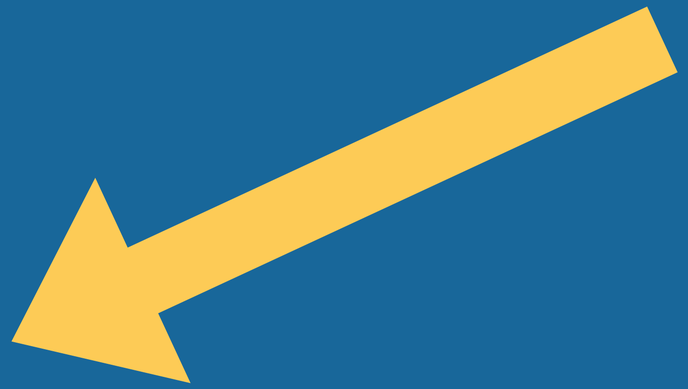
THE UNIVERSITY OF CHICAGO

# ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 4

# EXTENSIONS

# EXTENSIONS

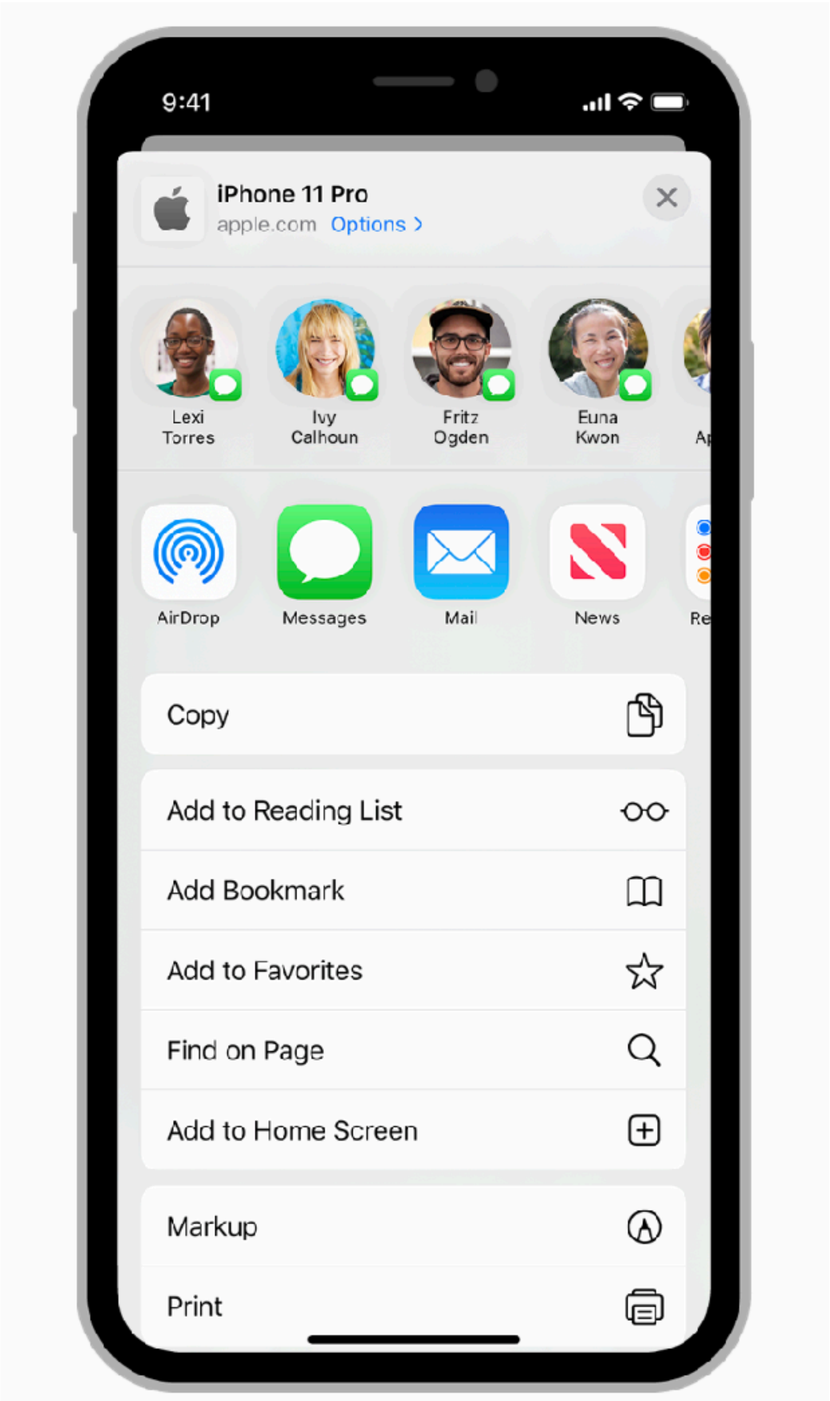


- Way to "extend" the system to provide a way to interface with your application
- More “Surface Area” for you app

# EXTENSIONS

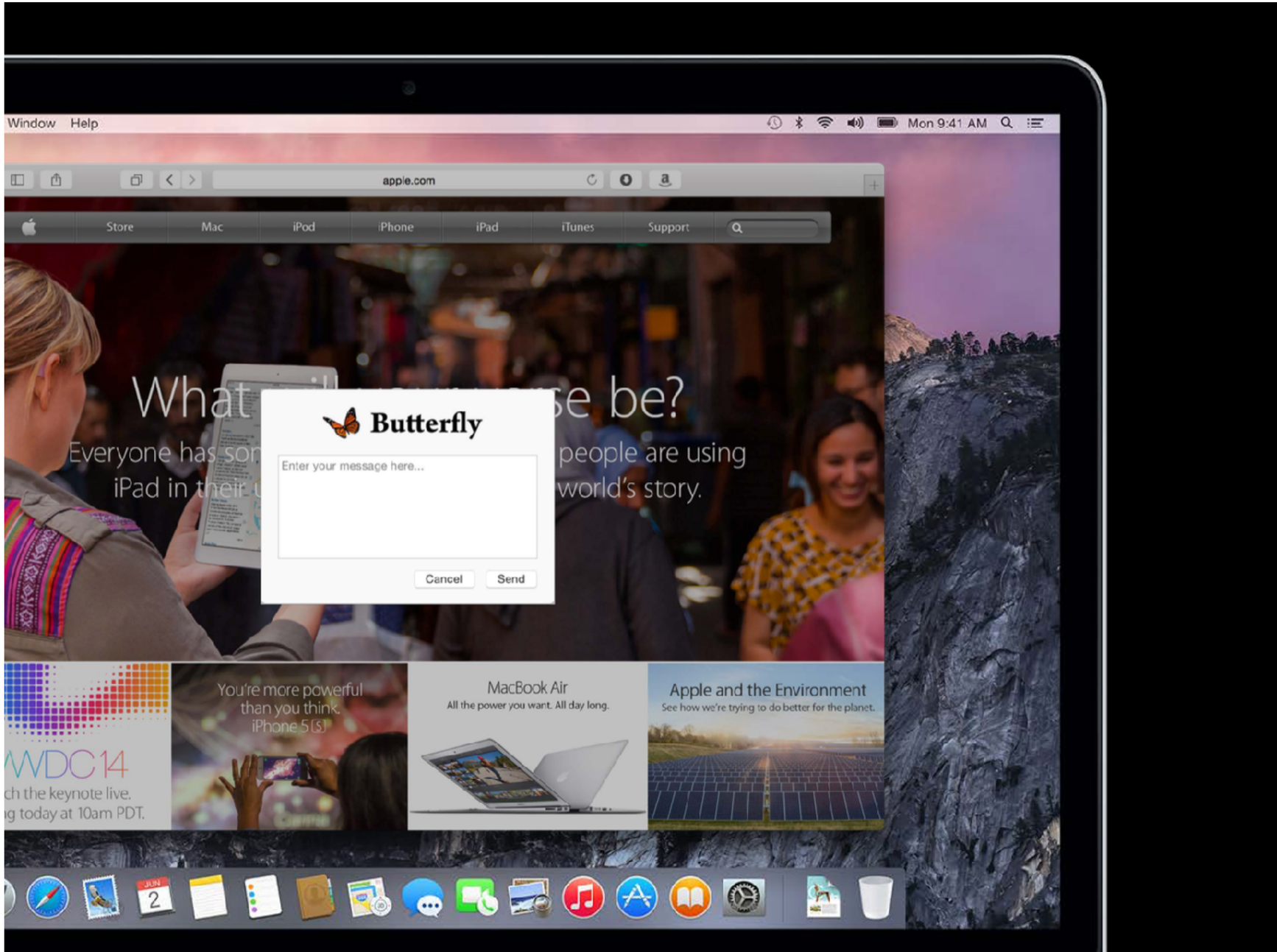
SUBTITLE

- Share extension extend the functionality of app
- Allows your app to (sort of) share data with other applications



# EXTENSIONS

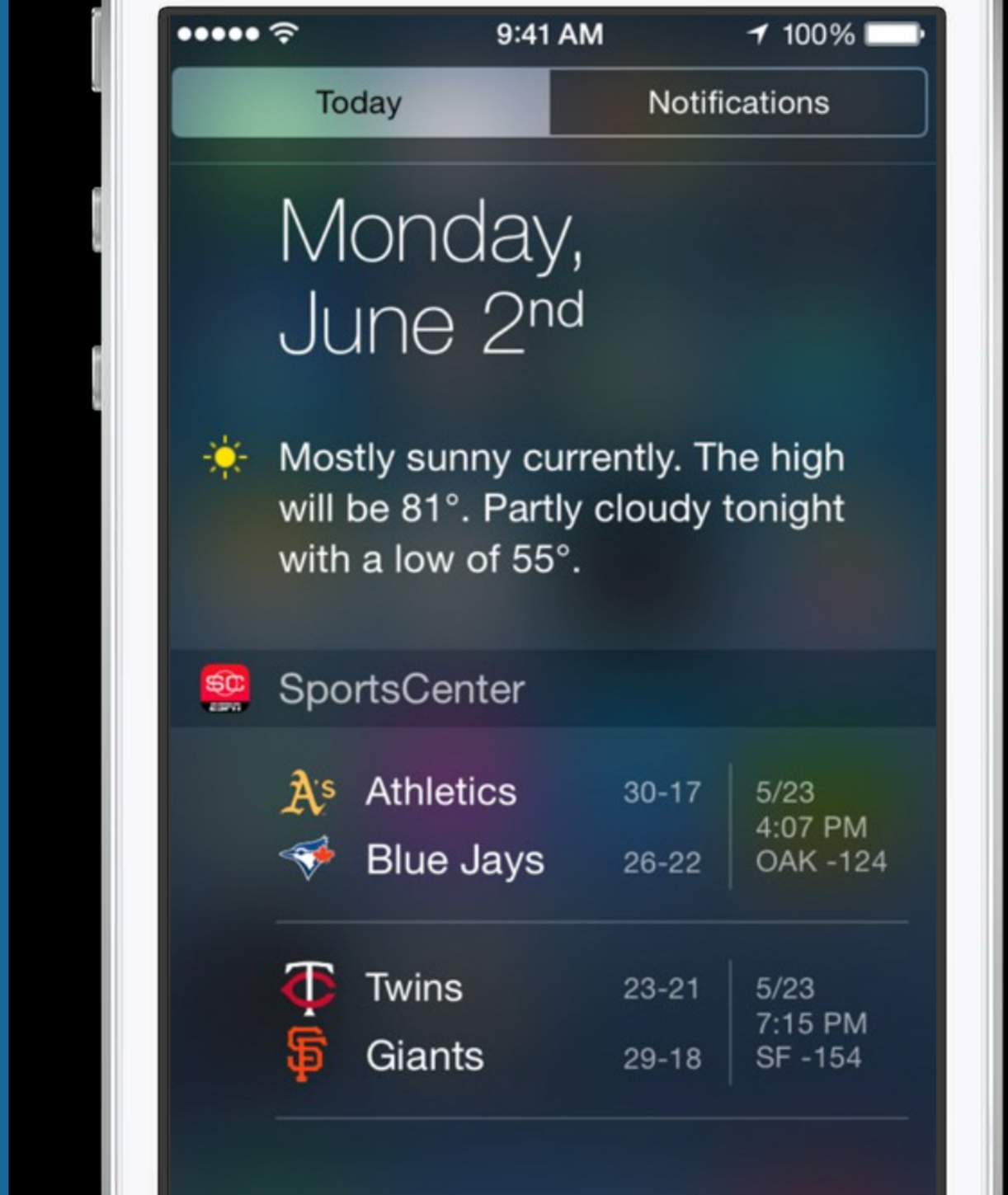
- Share extension on macOS



# EXTENSIONS

SUBTITLE

- Today extension in Notification Center



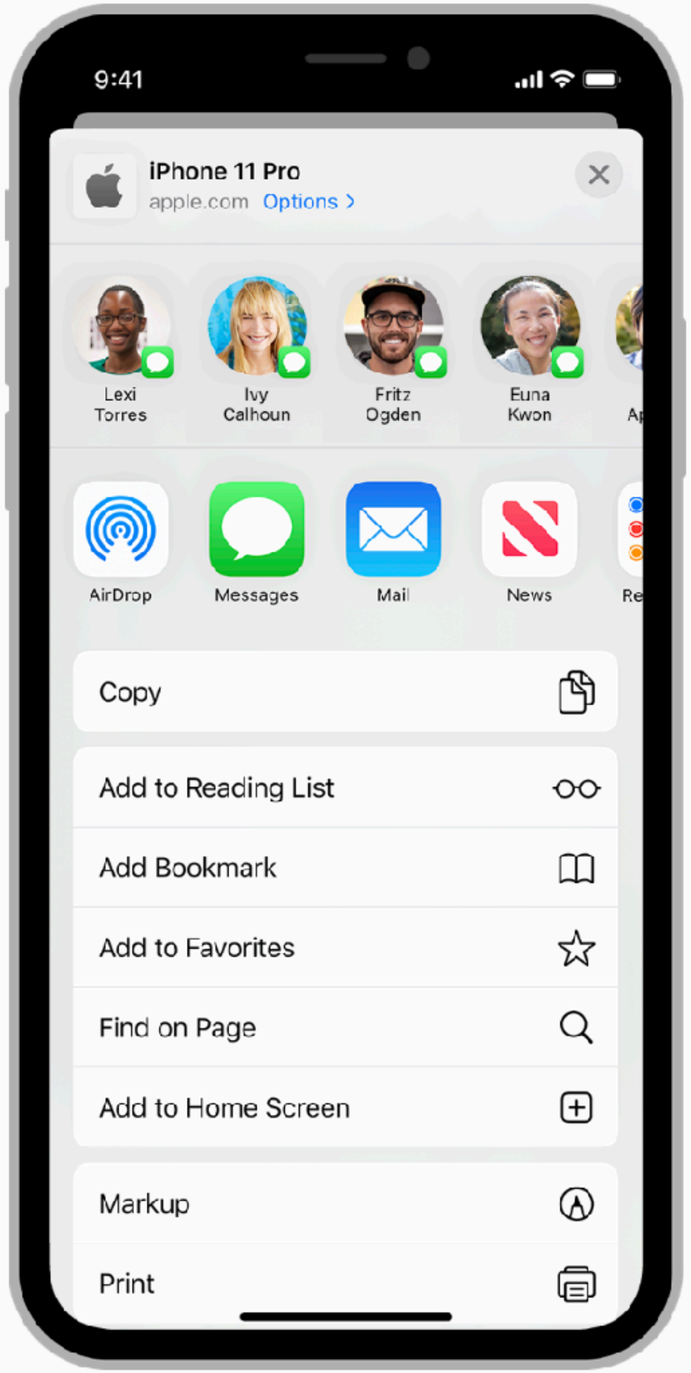
# EXTENSIONS

- Today extension in Notification Center on OSX

# EXTENSIONS

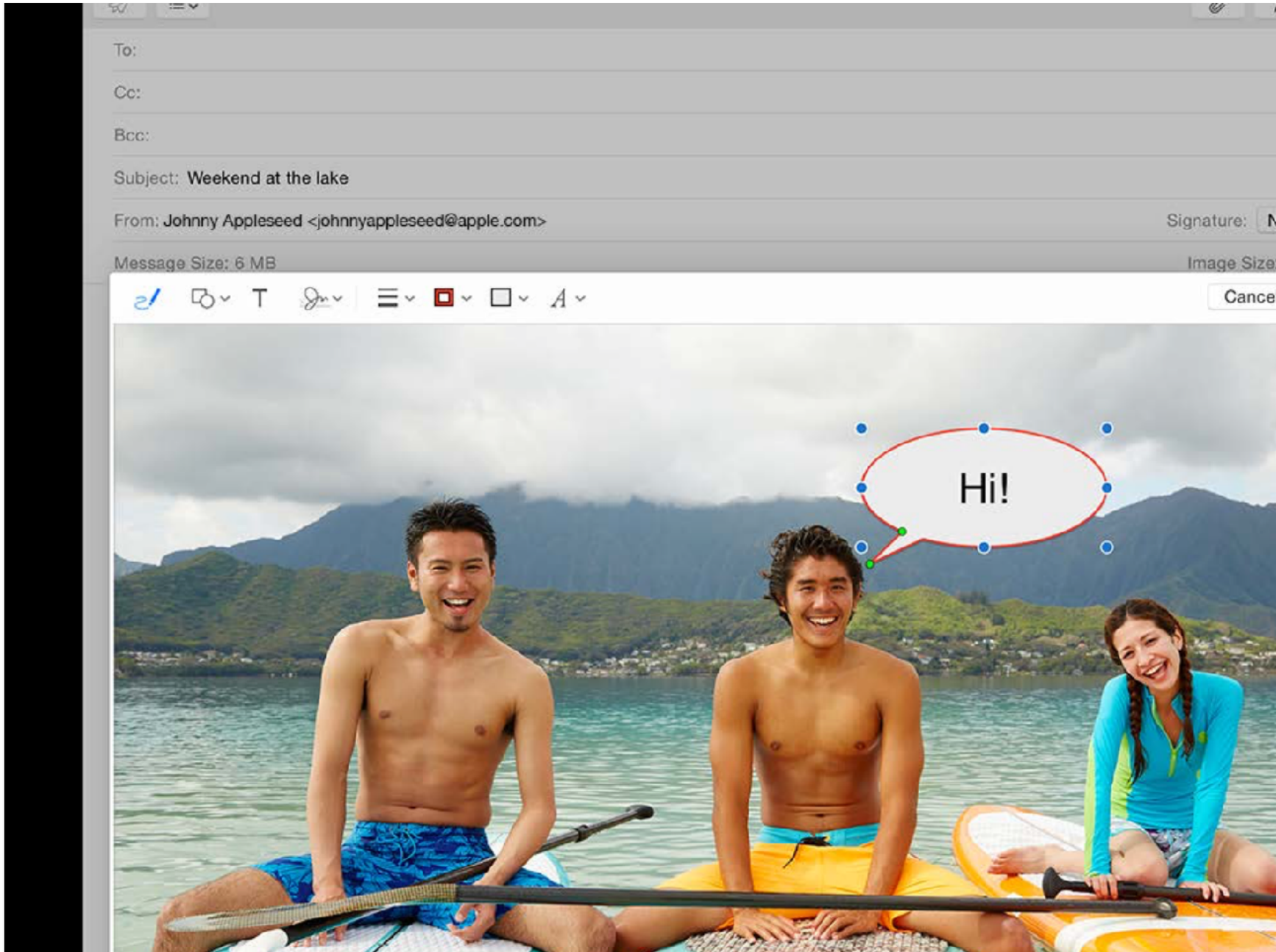
SUBTITLE

- Custom action extensions
- Siri Suggestions



# EXTENSIONS

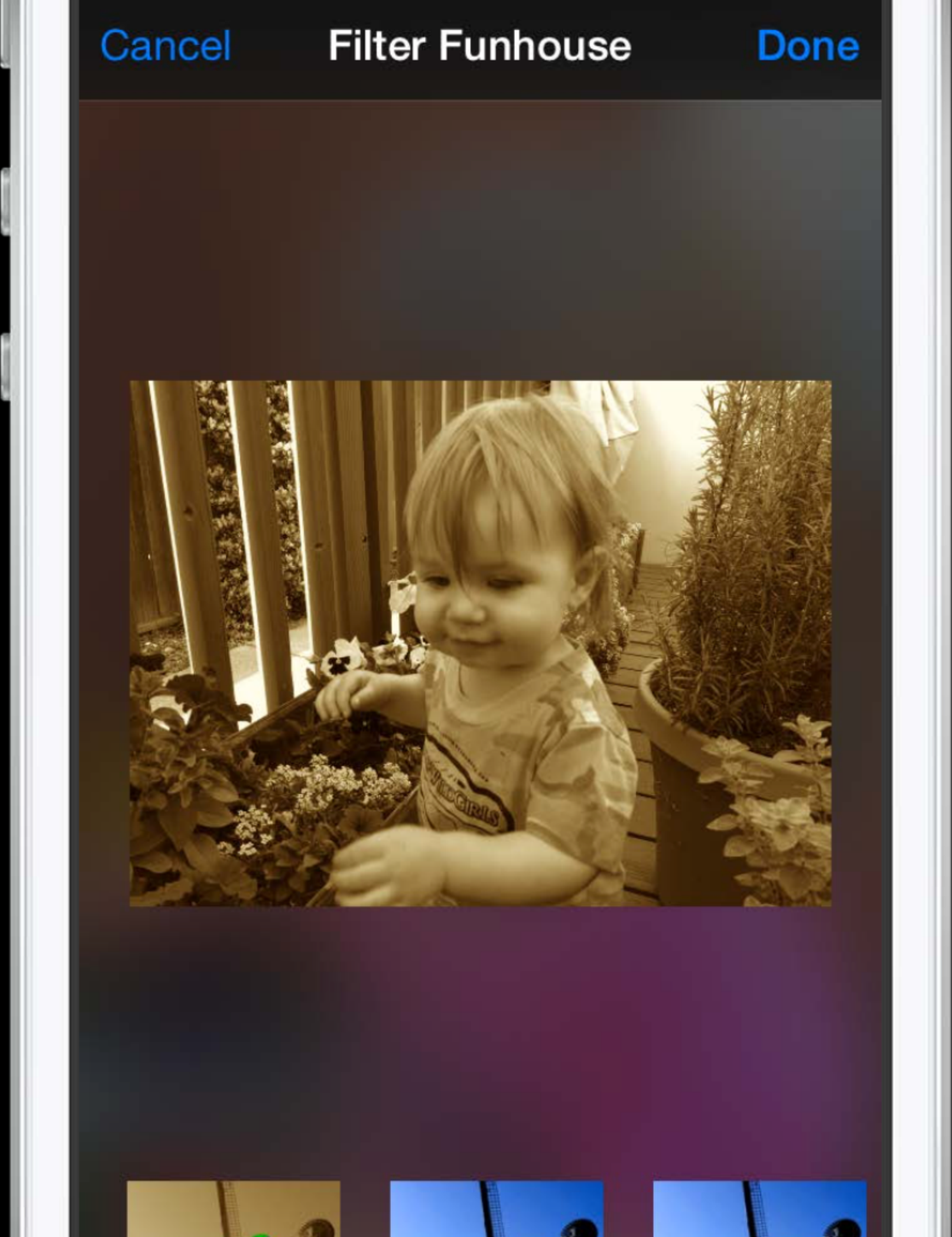
- Action Extensions on OSX



# EXTENSIONS

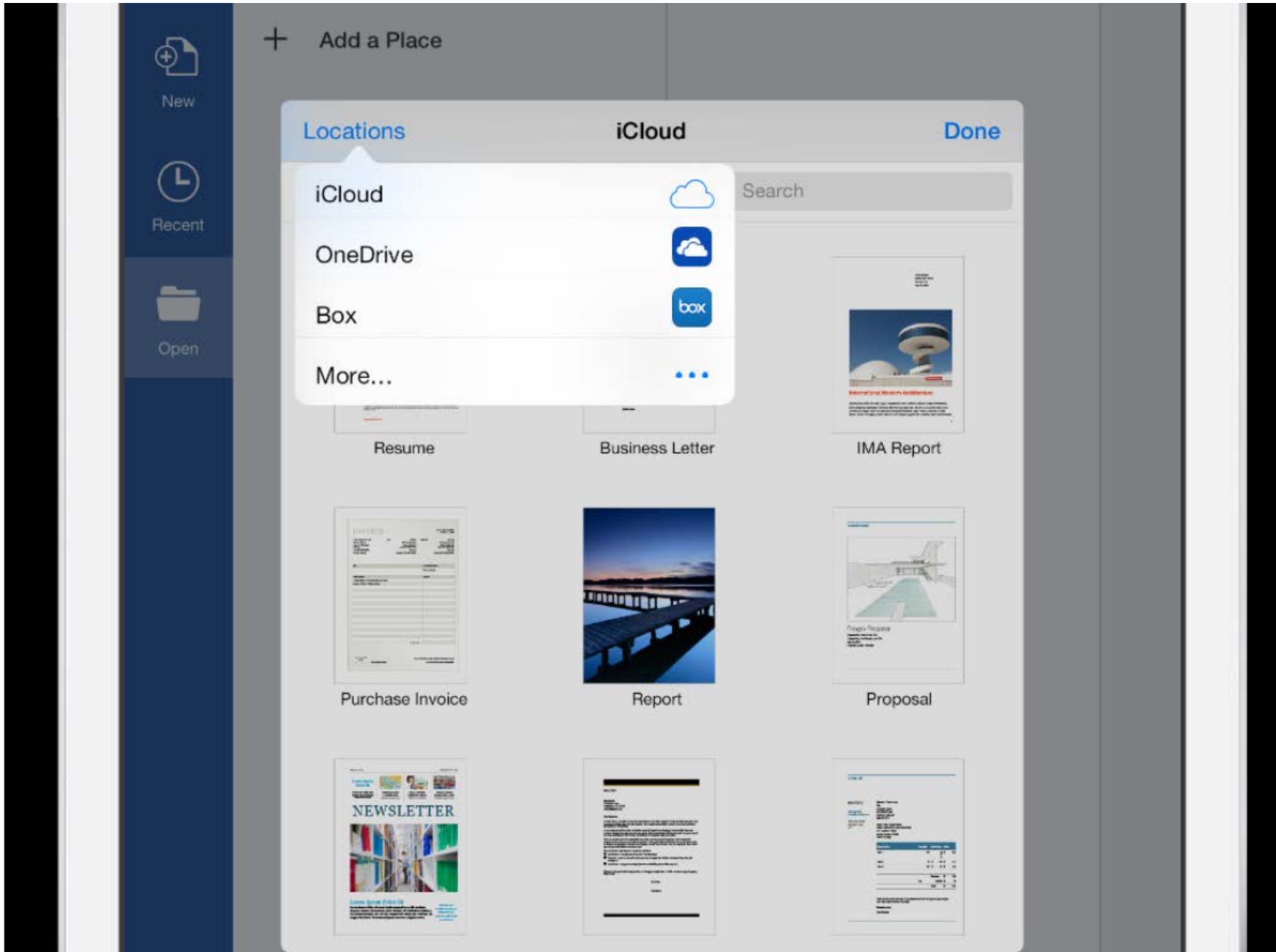
SUBTITLE

- Photo extensions



# EXTENSIONS

- Document providers



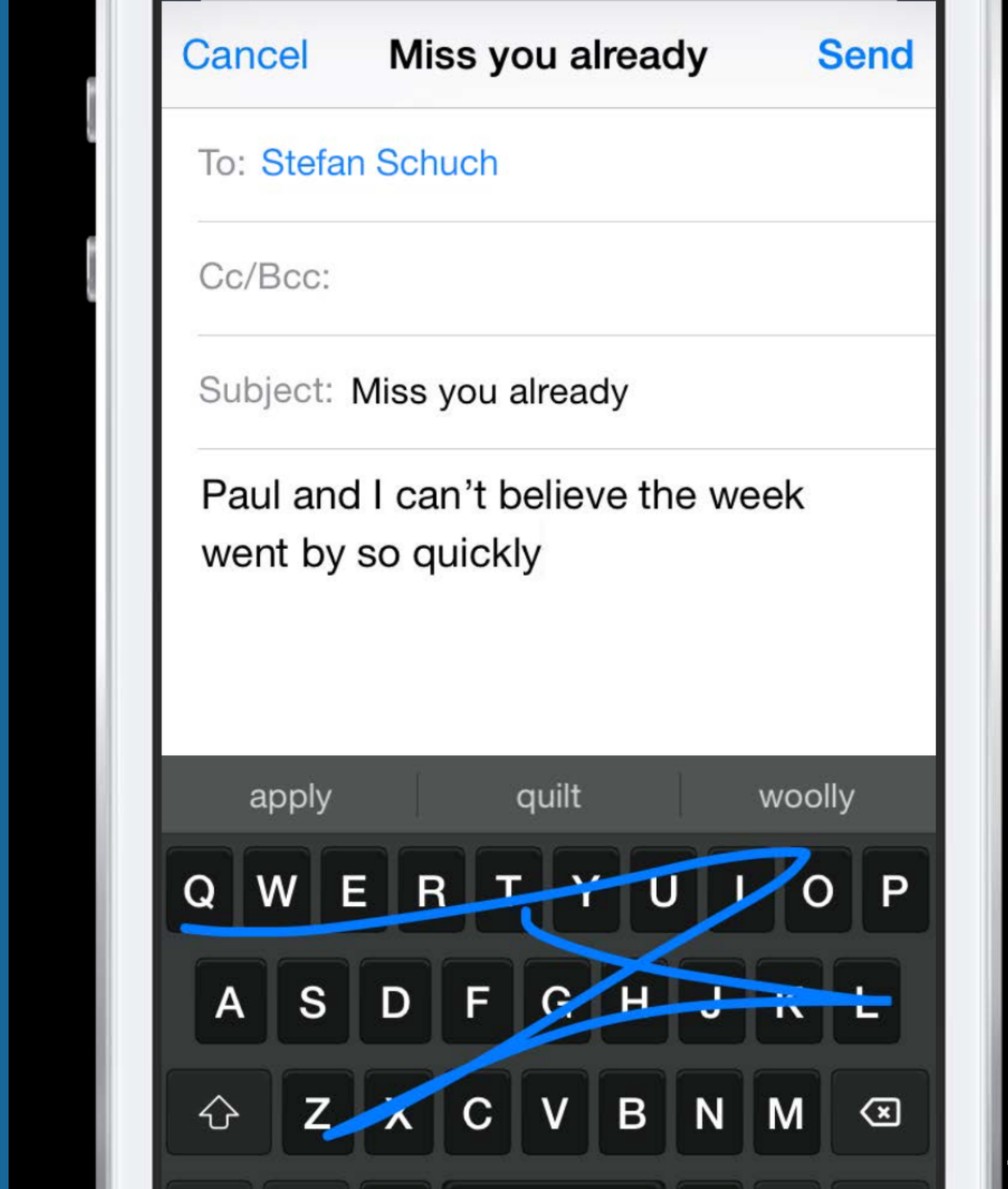
# EXTENSIONS

- Finder extensions

# EXTENSIONS

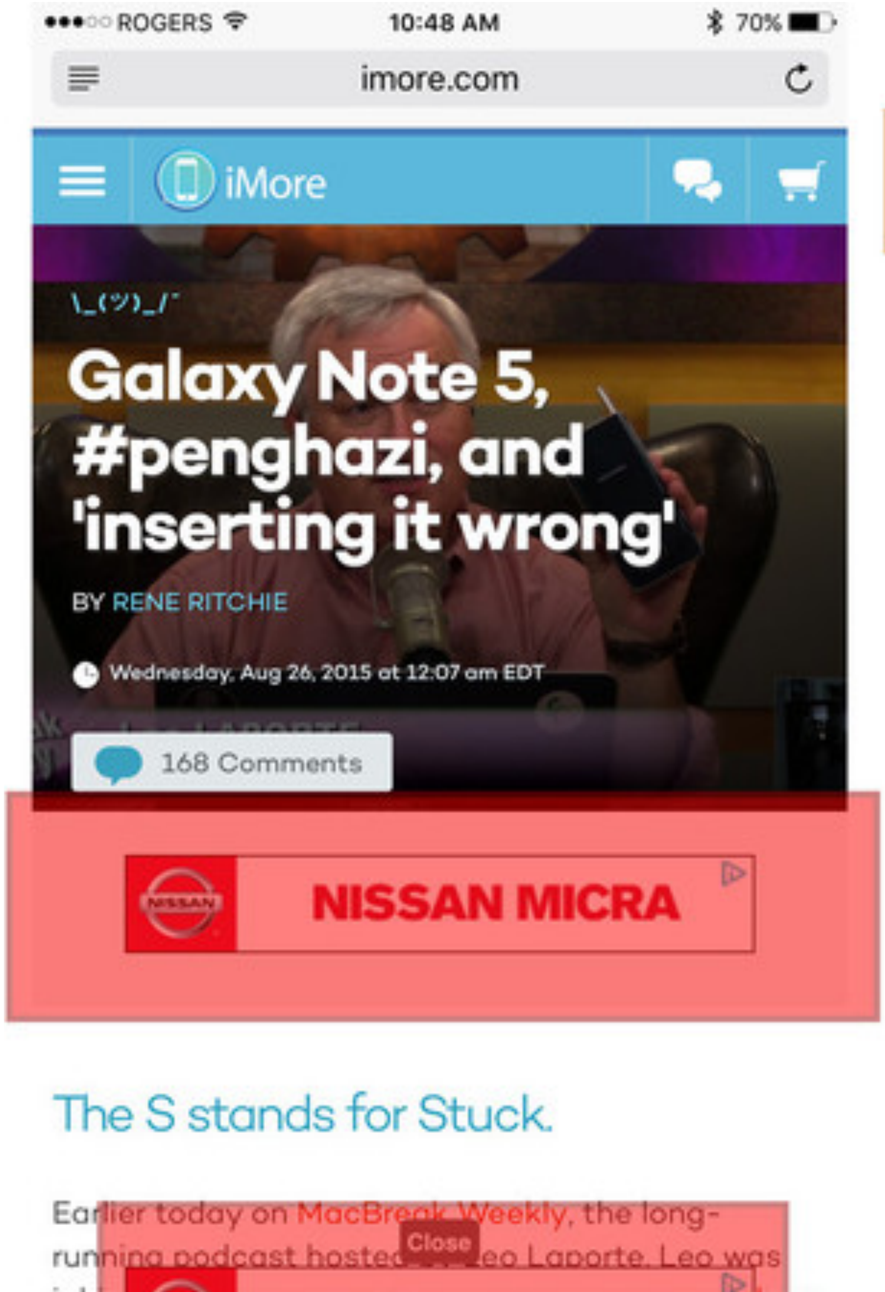
SUBTITLE

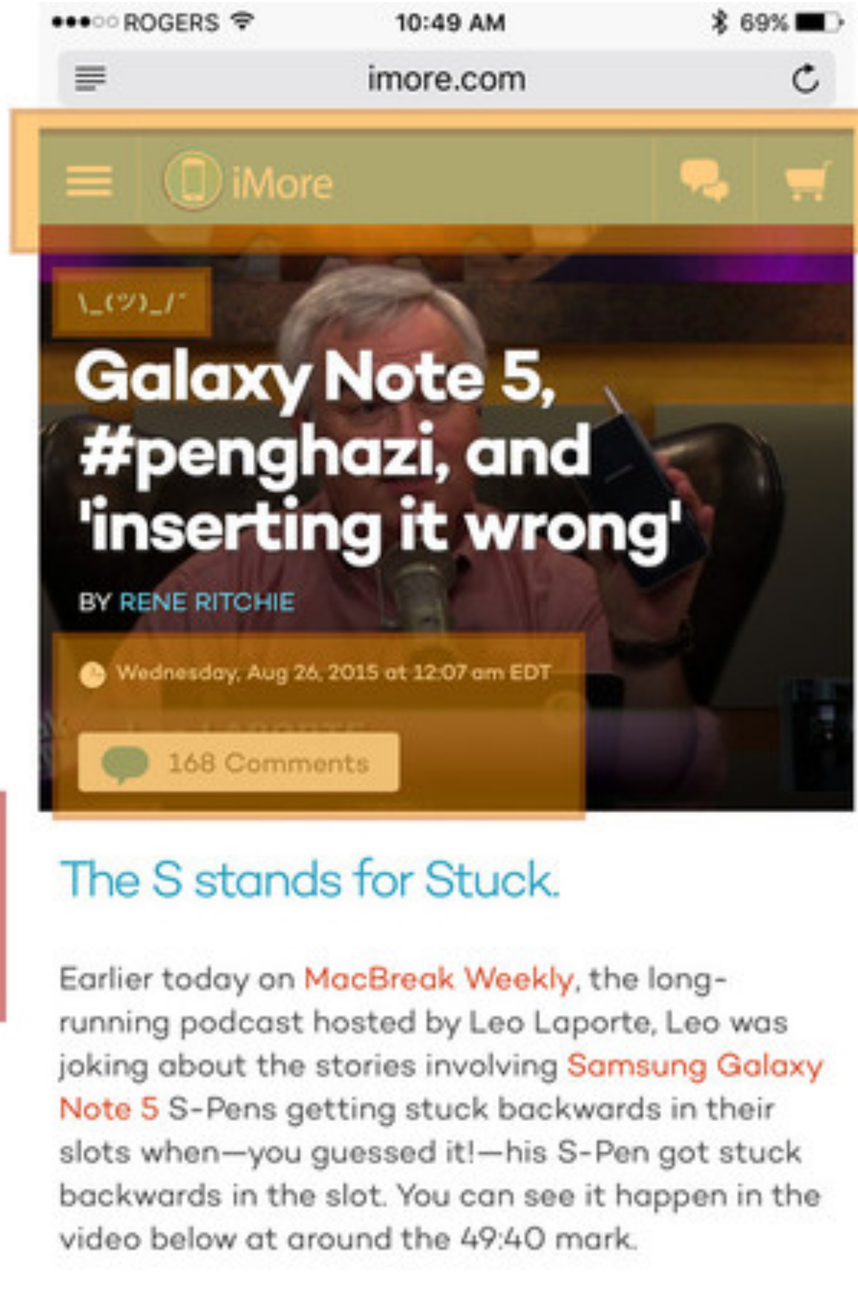
- Third party keyboards



# EXTENSIONS

- Content blockers



# EXTENSIONS

SUBTITLE

- List of all extensions
- Some are exclusive macOS, tvOS, iOS

| Extension point | Typical app extension functionality |
| --- | --- |
| Action (iOS and macOS; UI and non-UI variants) | Manipulate or view content originating in a host app. |
| Audio Unit (iOS and macOS; UI and non-UI variants) | Generates an audio stream to send to a host app, or modifies an audio stream from a host app and sends it back to the host app. |
| Broadcast UI (iOS and tvOS) | |
| Broadcast Upload (iOS and tvOS) | |
| Call Directory (iOS) | Identify and block incoming callers by their phone number. To learn more, see *CallKit Framework Reference*. |
| Content Blocker (iOS and macOS) | Indicate to WebKit that your content-blocking app has updated its rules. (This app extension has no user interface.) |
| Custom Keyboard (iOS) | Replace the iOS system keyboard with a custom keyboard for use in all apps. |
| Document Provider (iOS; UI and non-UI variants) | Provide access to and manage a repository of files. |
| | Present information about file sync (This app extension has no user interface.) |

HTTPS://DEVELOPER.APPLE.COM/LIBRARY/ARCHIVE/DOCUMENTATION/GENERAL/CONCEPTUAL/EXTENSIBILITYPG/INDEX.HTML

# EXTENSIONS

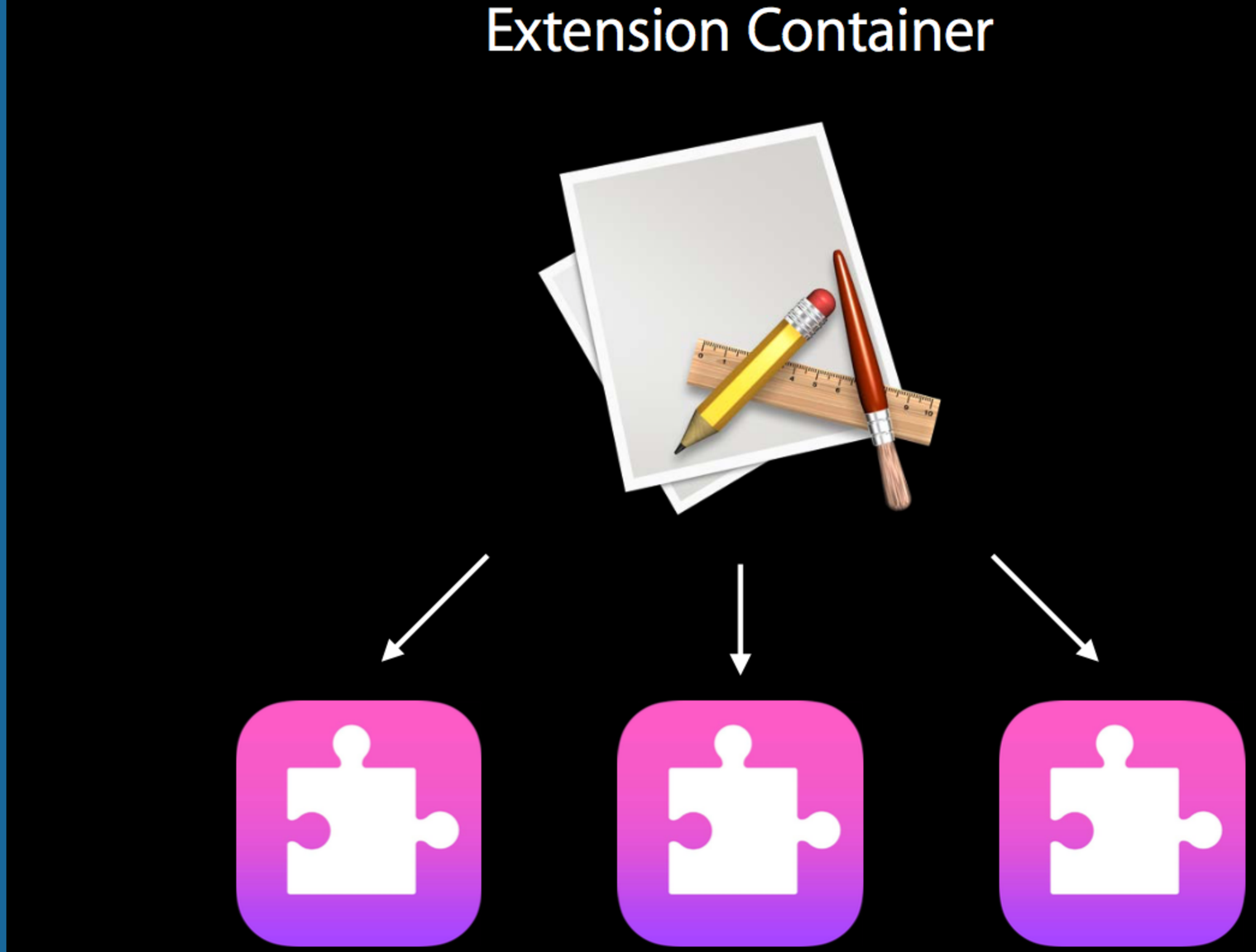
- Most common for iOS applications

| Extension point | Typical app extension functionality |
| --- | --- |
| Action (iOS and OS X; UI and non-UI variants) | Manipulate or view content originating in a host app |
| Audio Unit (iOS and OS X; UI and non-UI variants) | Generates an audio stream to send to a host app, or modifies an audio stream from a host app and sends it back to the host app |
| Content Blocker (iOS and OS X) | Indicate to WebKit that your ad blocking app has updated its rules<br>(This app extension has no user interface) |
| Custom Keyboard (iOS) | Replace the iOS system keyboard with a custom keyboard for use in all apps |
| Document Provider (iOS; UI and non-UI variants) | Provide access to and manage a repository of files. |
| Finder Sync (OS X) | Present information about file sync state directly in Finder<br>(This app extension has no user interface) |
| Photo Editing (iOS) | Edit a photo or video within the Photos app |
| Share (iOS and OS X) | Post to a sharing website or share content with others |
| | Get a quick update or perform a quick task in the |

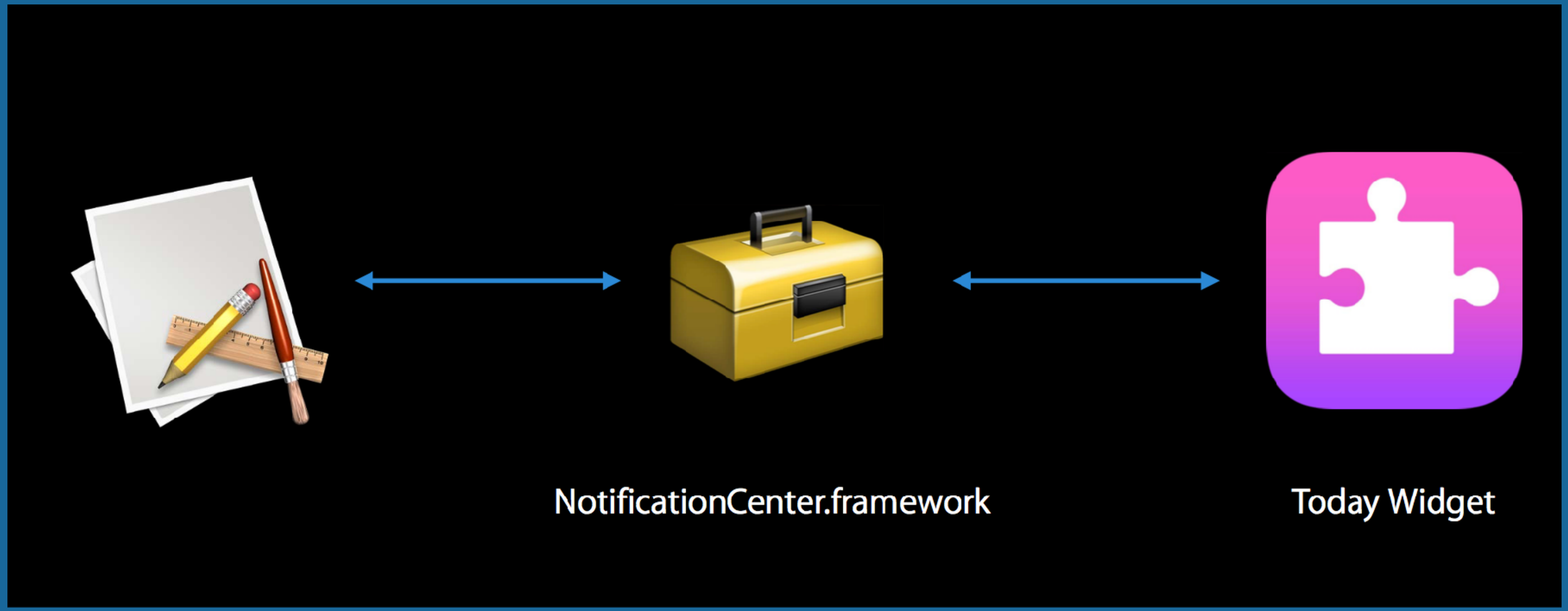
# DELIVERING EXTENSIONS

# DELIVERING EXTENSIONS

- Extension container is the main application
  - Multiple extensions per container allowed



# DELIVERING EXTENSIONS

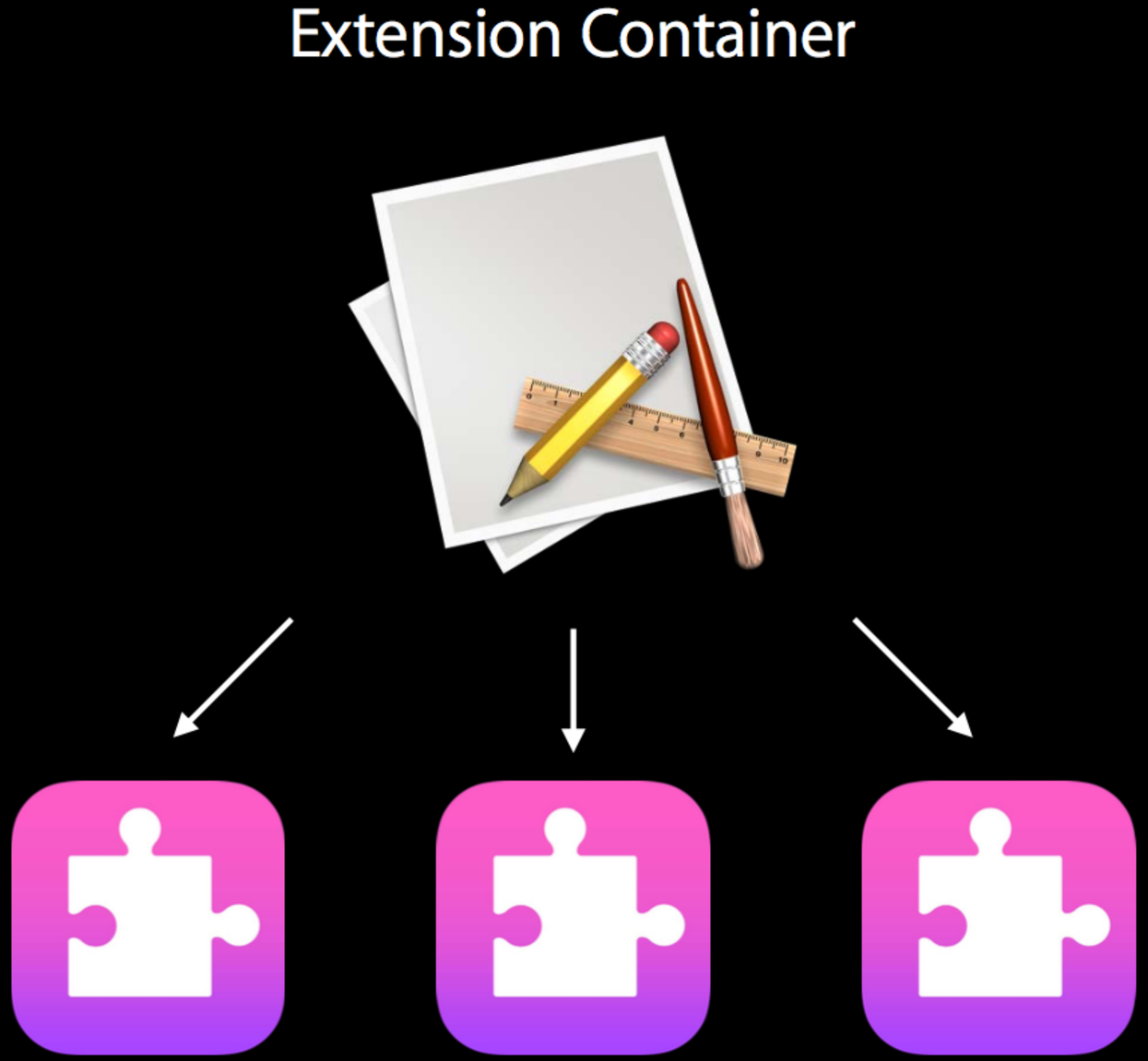



- Extension points are what your extensions binds to in the system

# DELIVERING EXTENSIONS

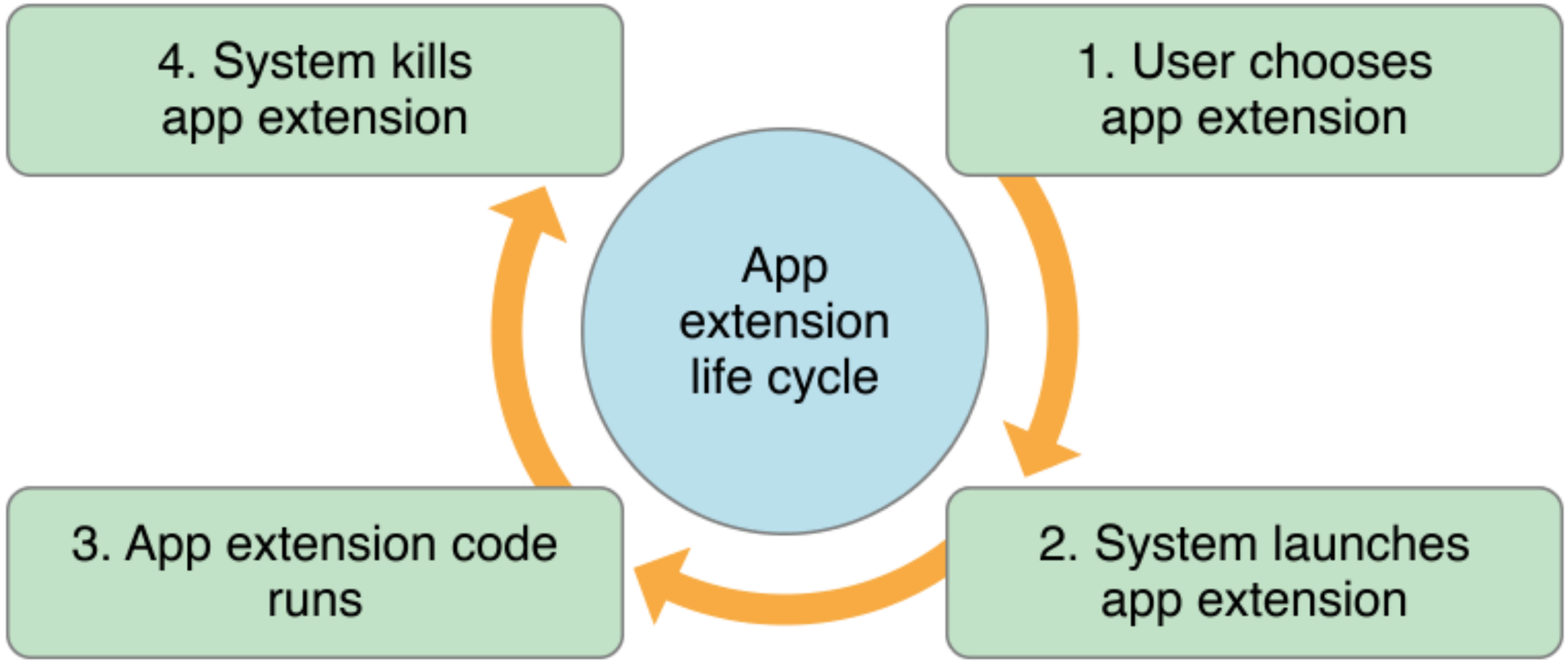
SUBTITLE

- Purpose built binaries
- Not independent applications
- Accessed via Apple frameworks code
- Not app to app interprocess communication
  - Not the same as UIApplication(openURL:)



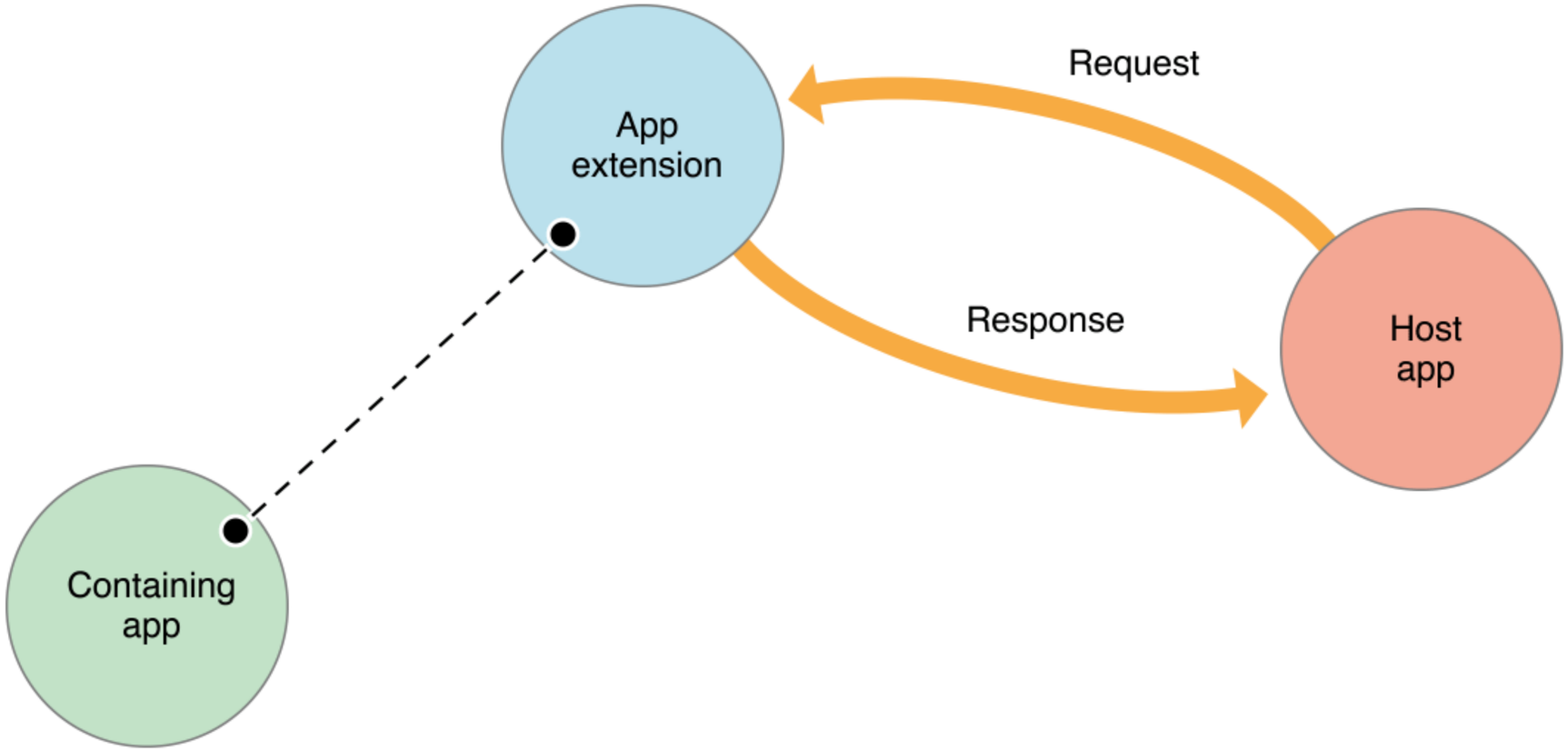
# DELIVERING EXTENSIONS

- Life cycle of an extension



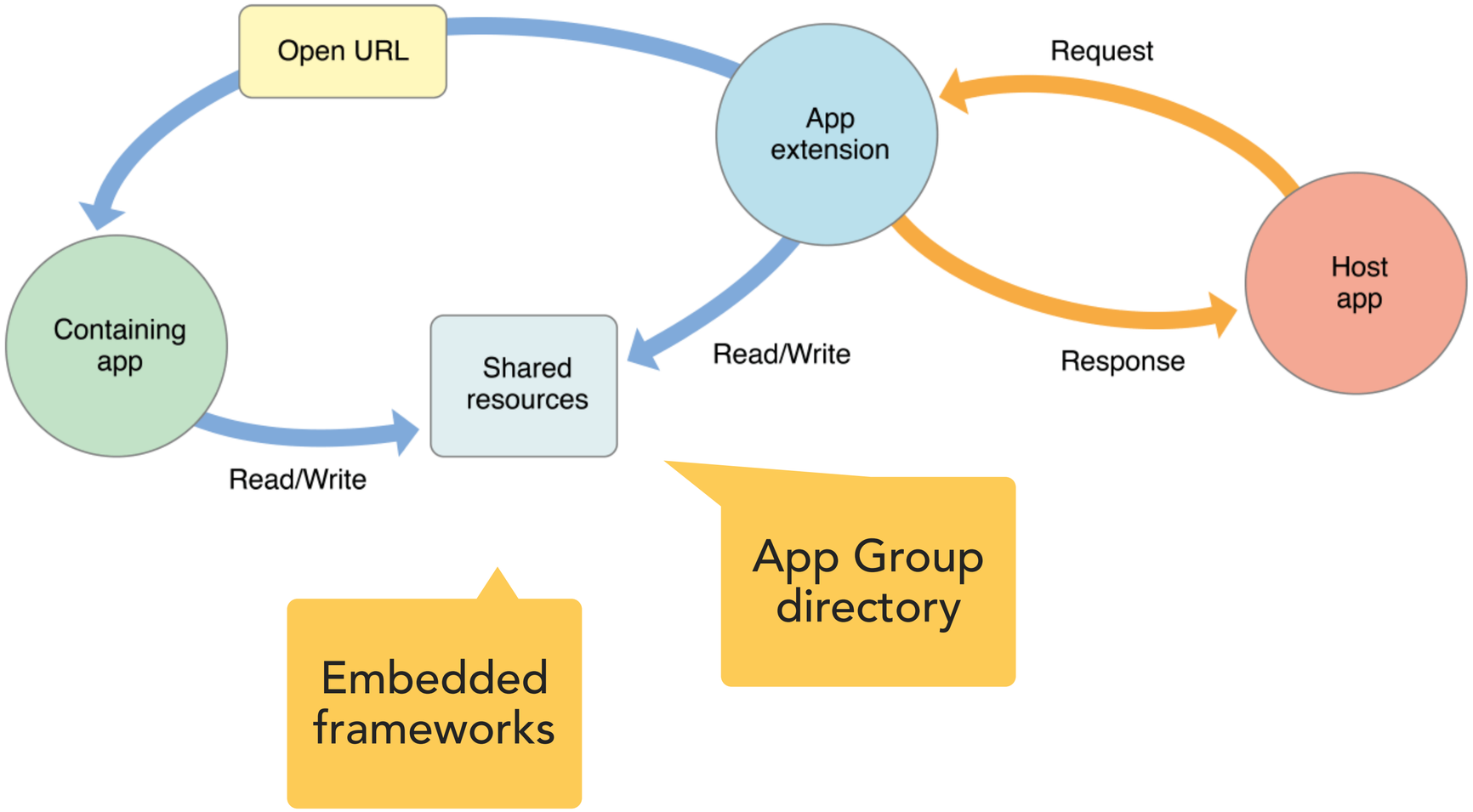
# DELIVERING EXTENSIONS

- Extensions communicate with the host (app that launched it)



# DELIVERING EXTENSIONS

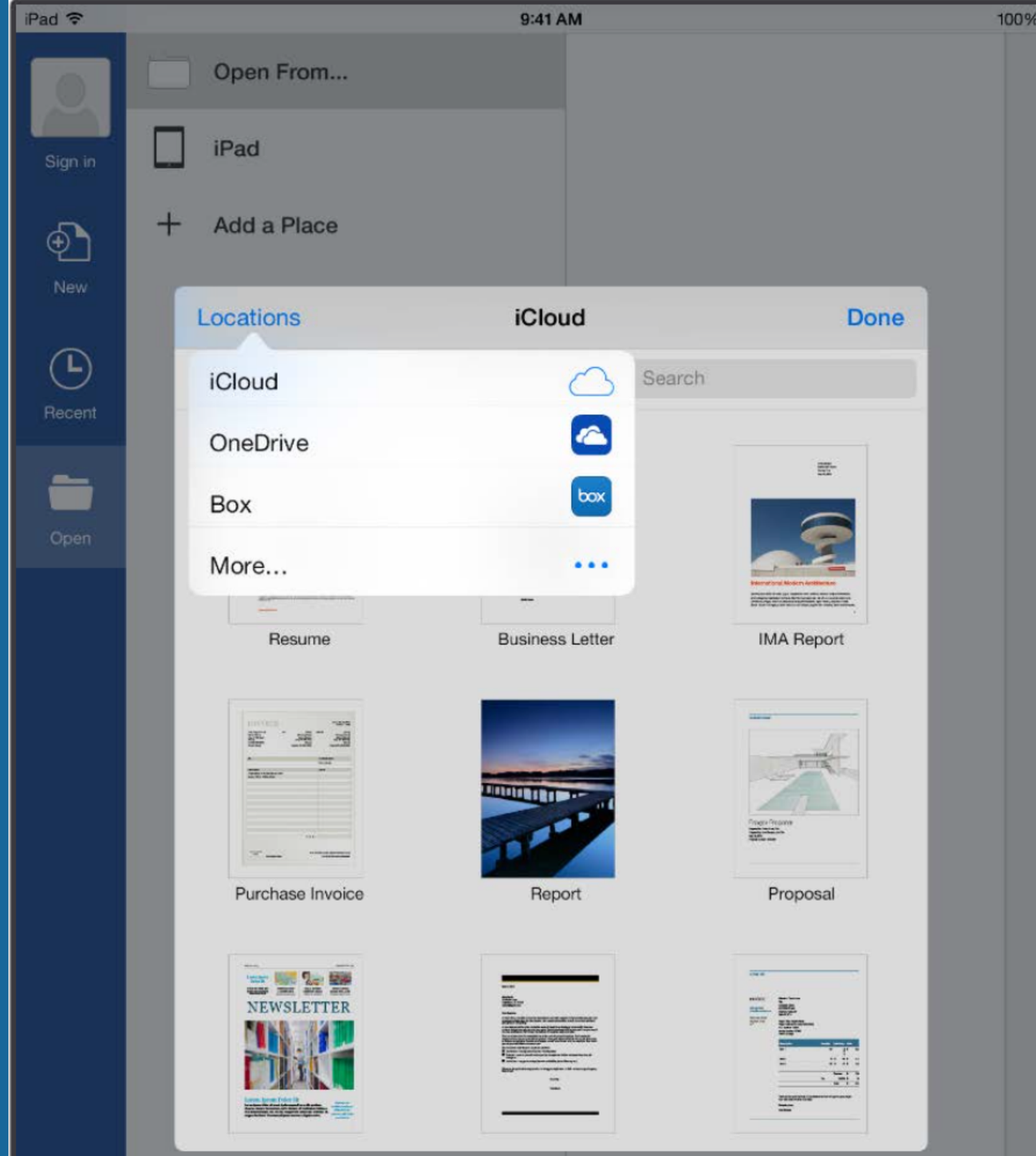
- Extension communicates with containing app via `openURL:` share resources



# DELIVERING EXTENSIONS

SUBTITLE

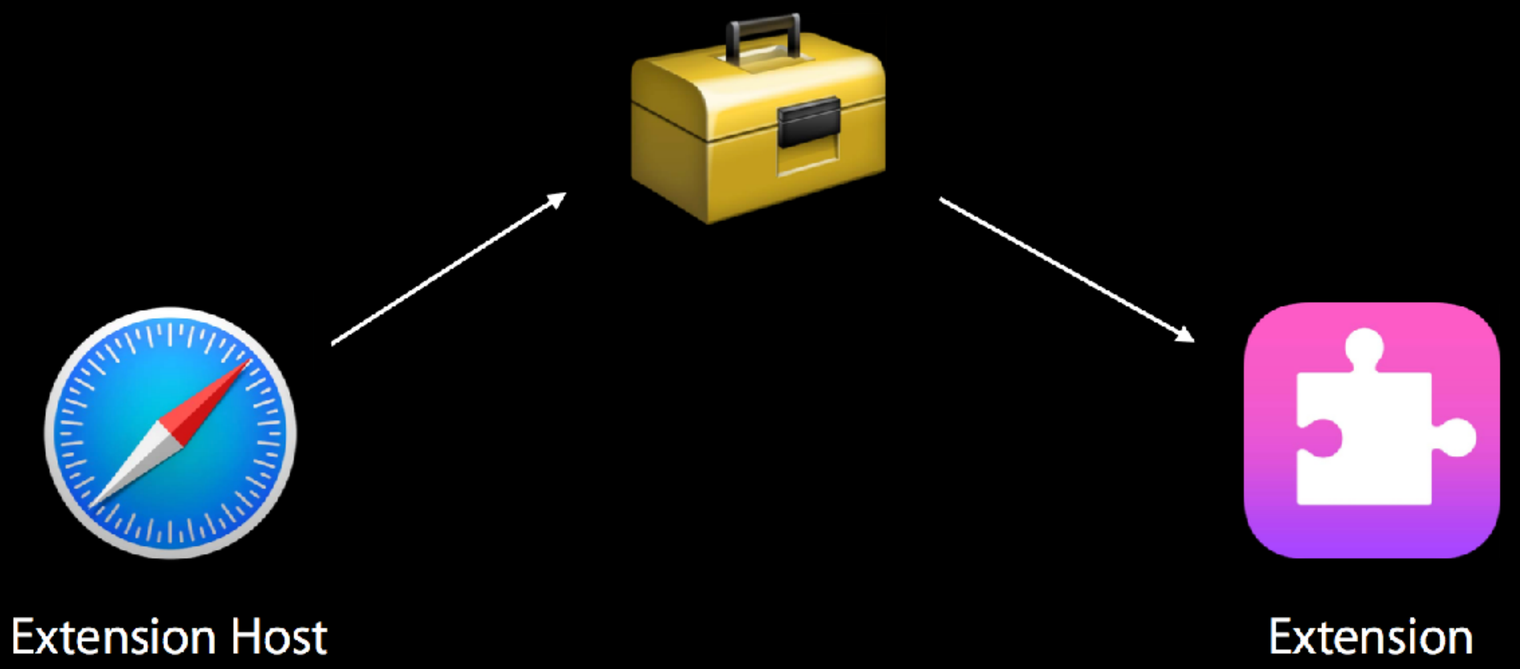
- Presentation depends on the type of extension
  - Presented (you see something on the screen)
  - Non-presented (you don’t see something)
    - Translator example



# DELIVERING EXTENSIONS

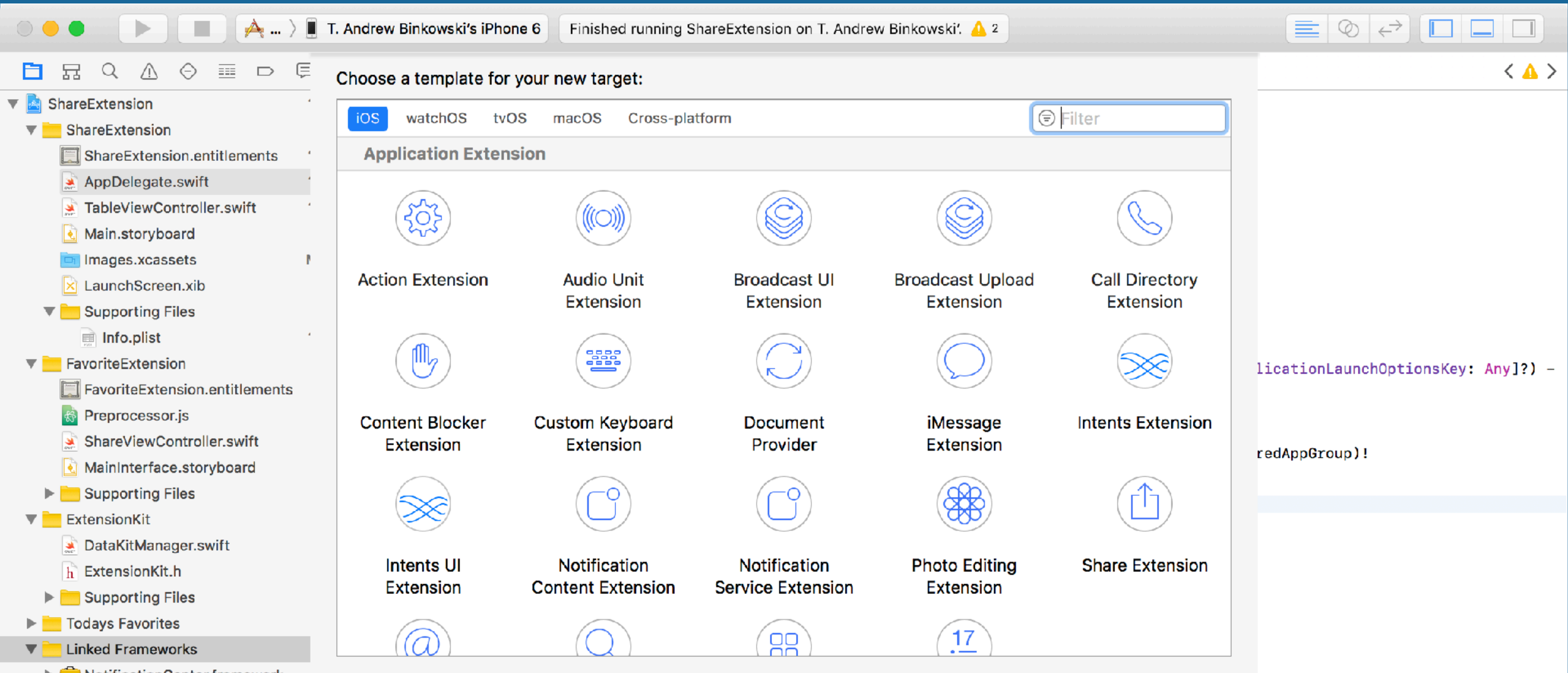
- Extensions are bounced through frameworks not launched directly

This can make debugging a bit painful



# SETTING UP EXTENSIONS

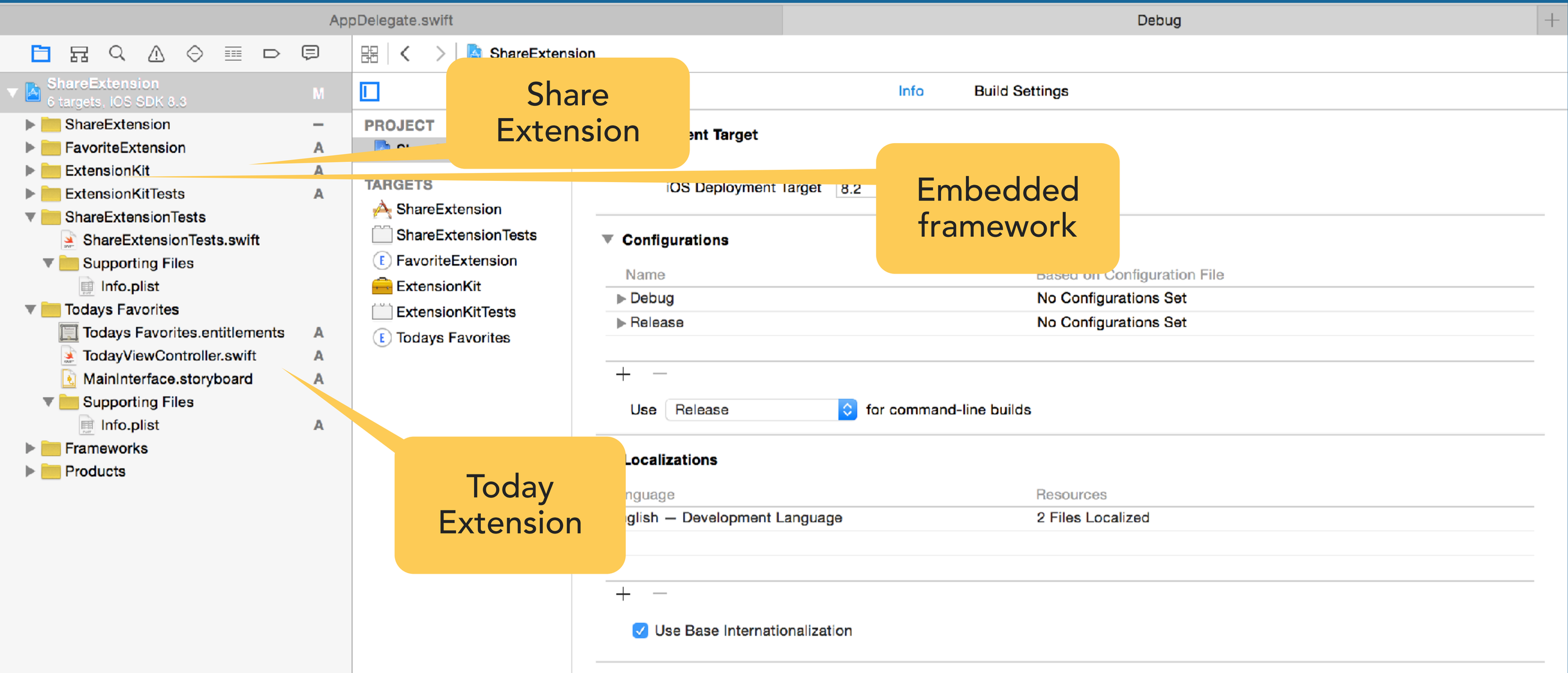
# SETTING UP EXTENSIONS



# SETTING UP EXTENSIONS

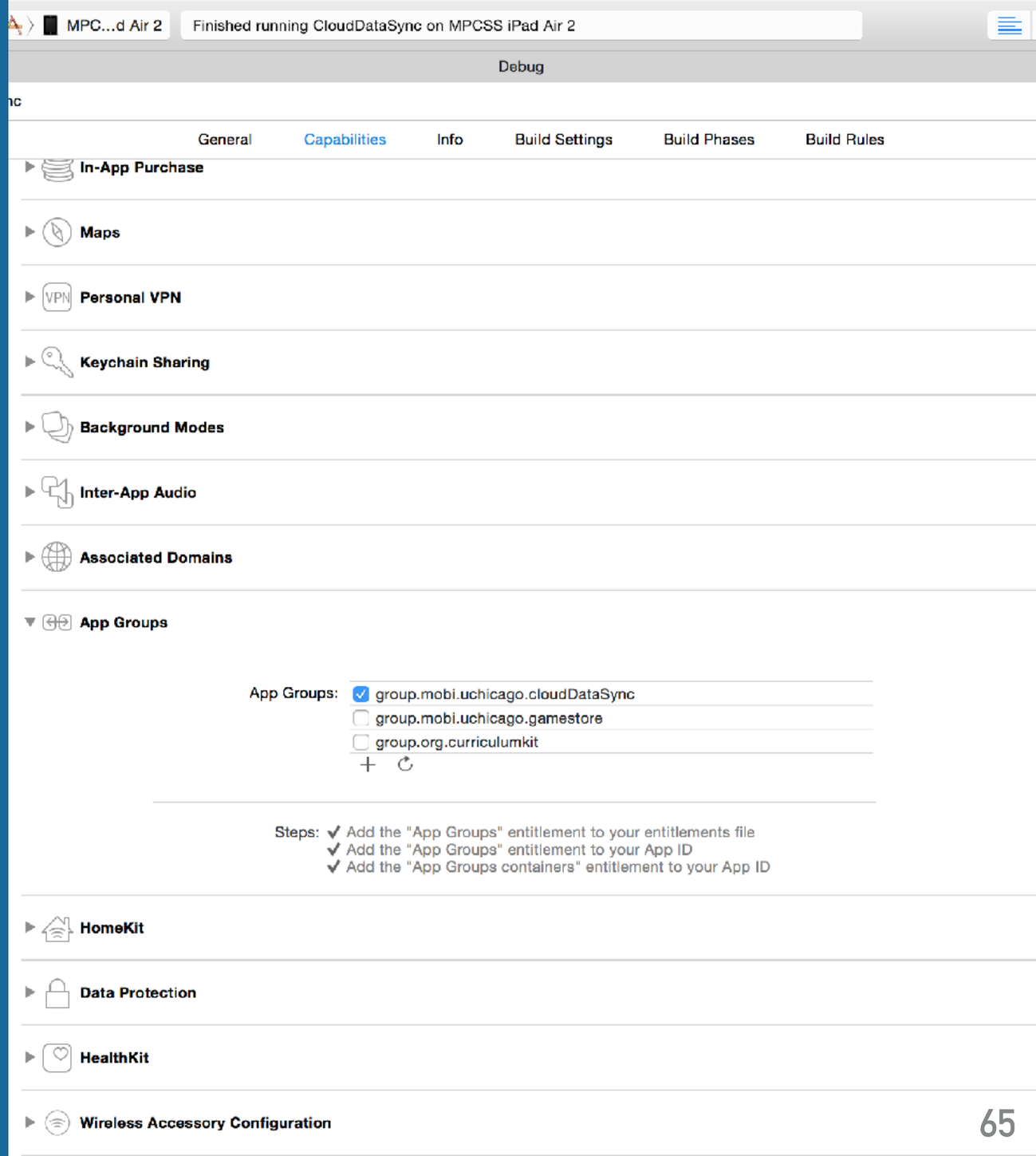
- Each share sheet has customized templates that provide basic setup of each extension
- Some have default interfaces that you can override

# SETTING UP EXTENSIONS



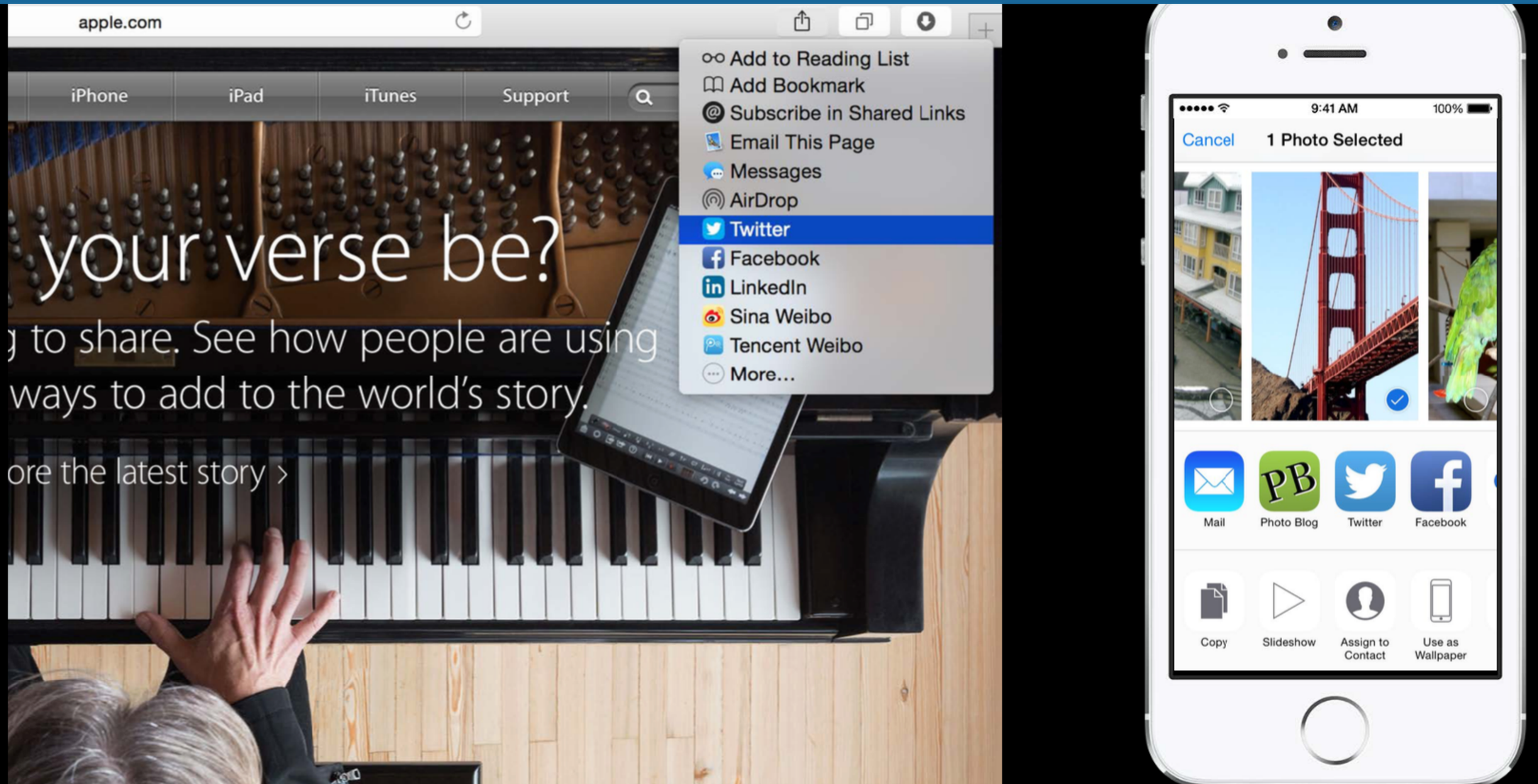
# SETTING UP EXTENSIONS

- Add app groups to share data with container extension
  - Possible that you may **not** need to share data between your extension and application



# SHARE EXTENSIONS

# SHARE EXTENSIONS

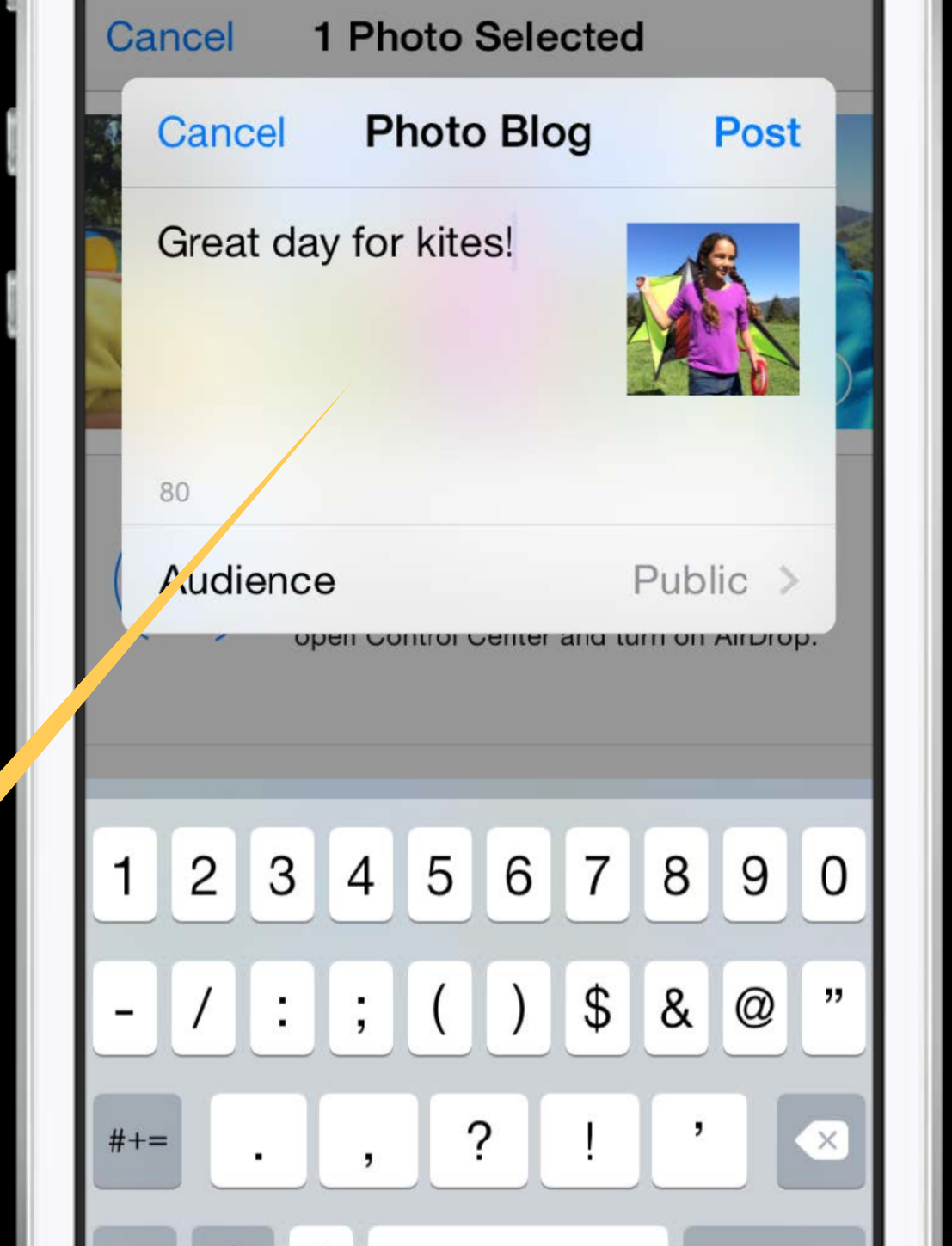


# SHARE EXTENSIONS

SUBTITLE

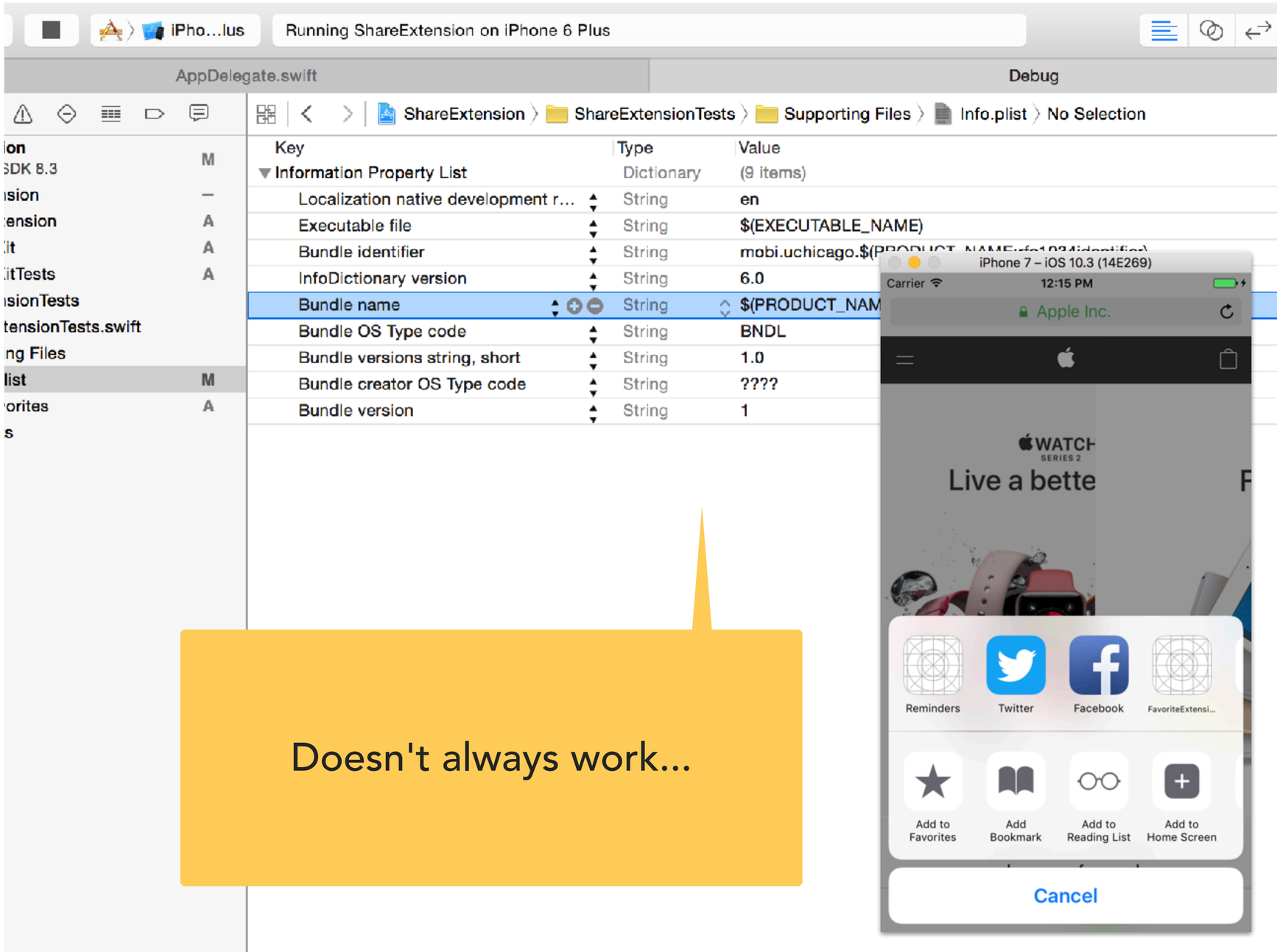
- Display name
- Activation rules
- Implement a ViewController
  - Do something with the data you have access to
- Signal completion/ cancellation

Default UI piggybacked from Social Framework



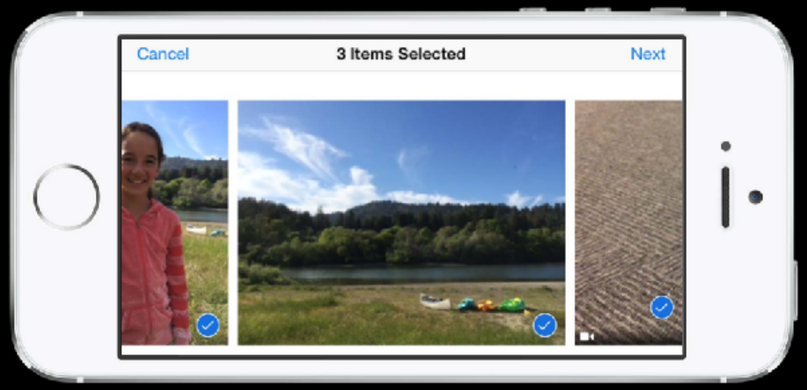
# SHARE EXTENSIONS

- Customize the name for your share extension
  - Default is product name (may need to change for visibility)



# SHARE EXTENSIONS

- Activation Rules
  - When system show my share extension?
  - What content is available





???

Social Network Extension

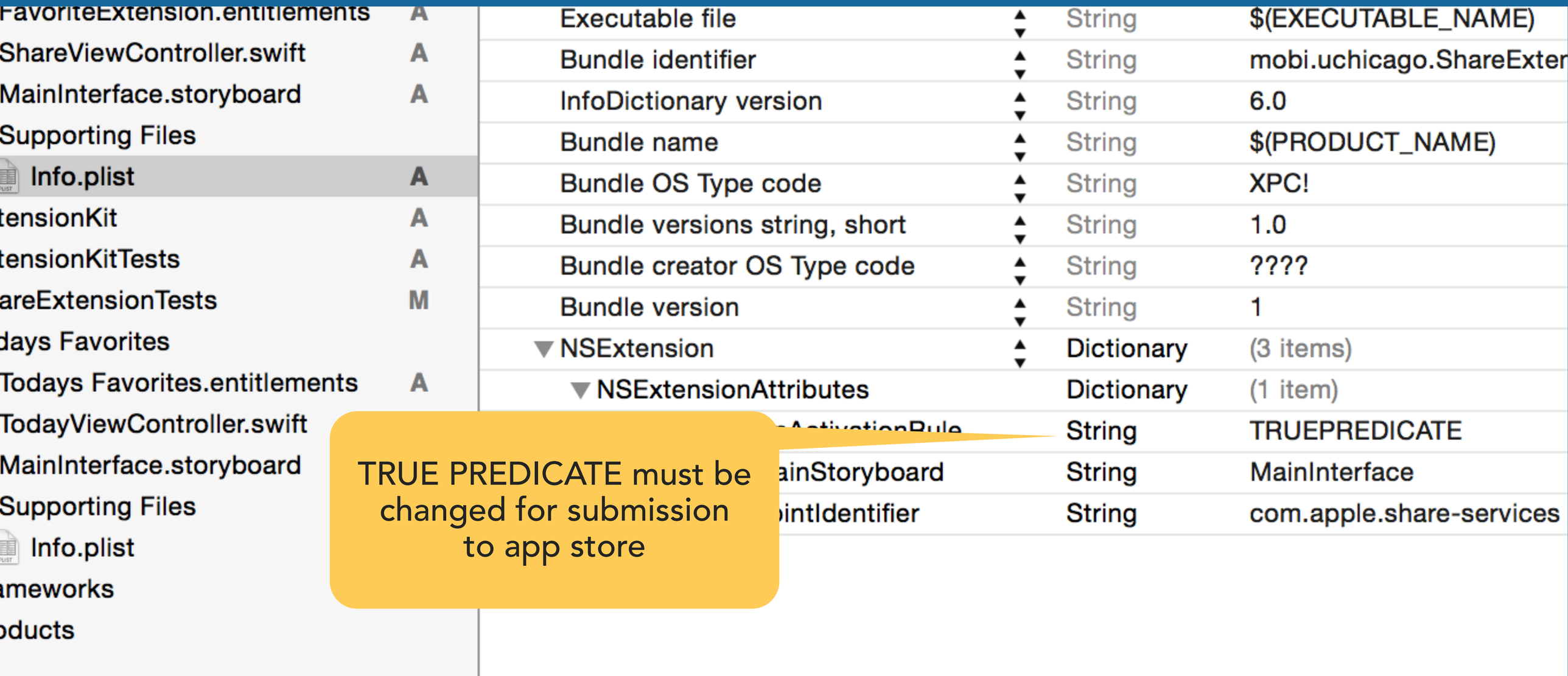

Photo Sharing Extension

Microblog Extension

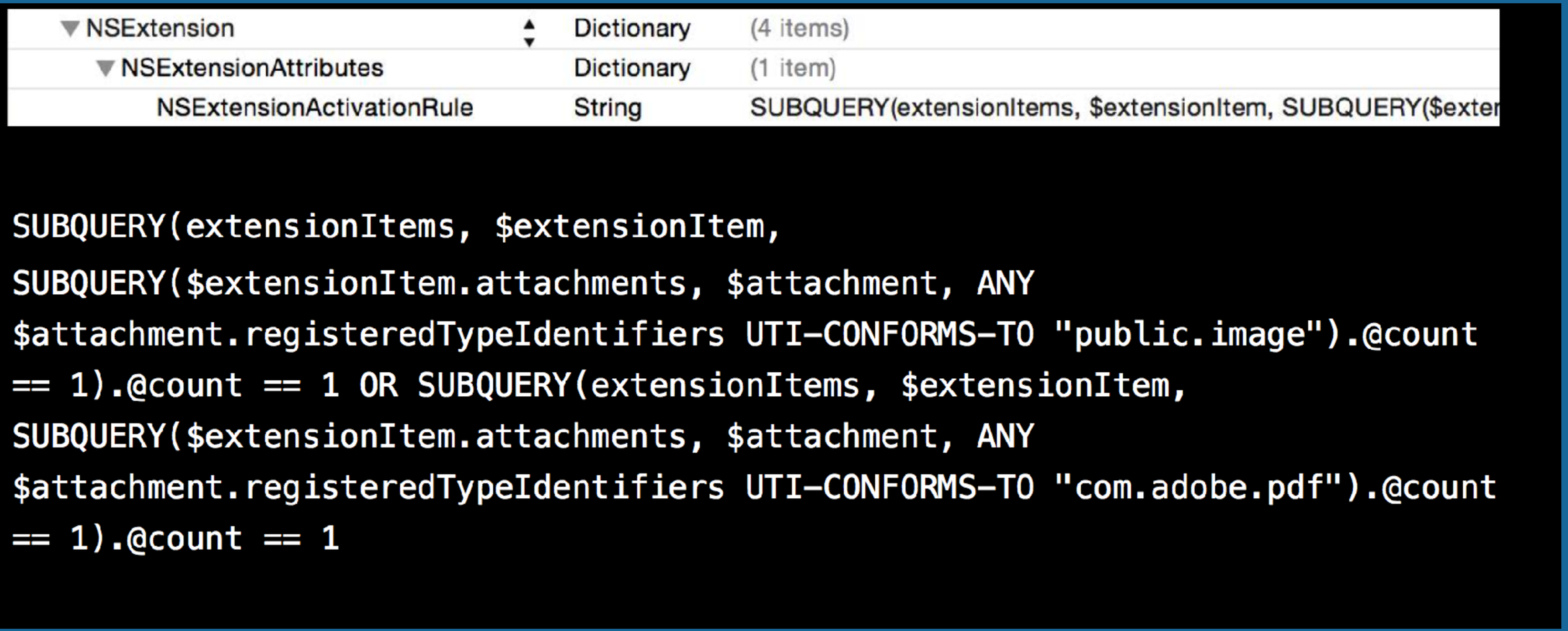
Video Sharing Extension

URLs, text, Images…

# SHARE EXTENSIONS

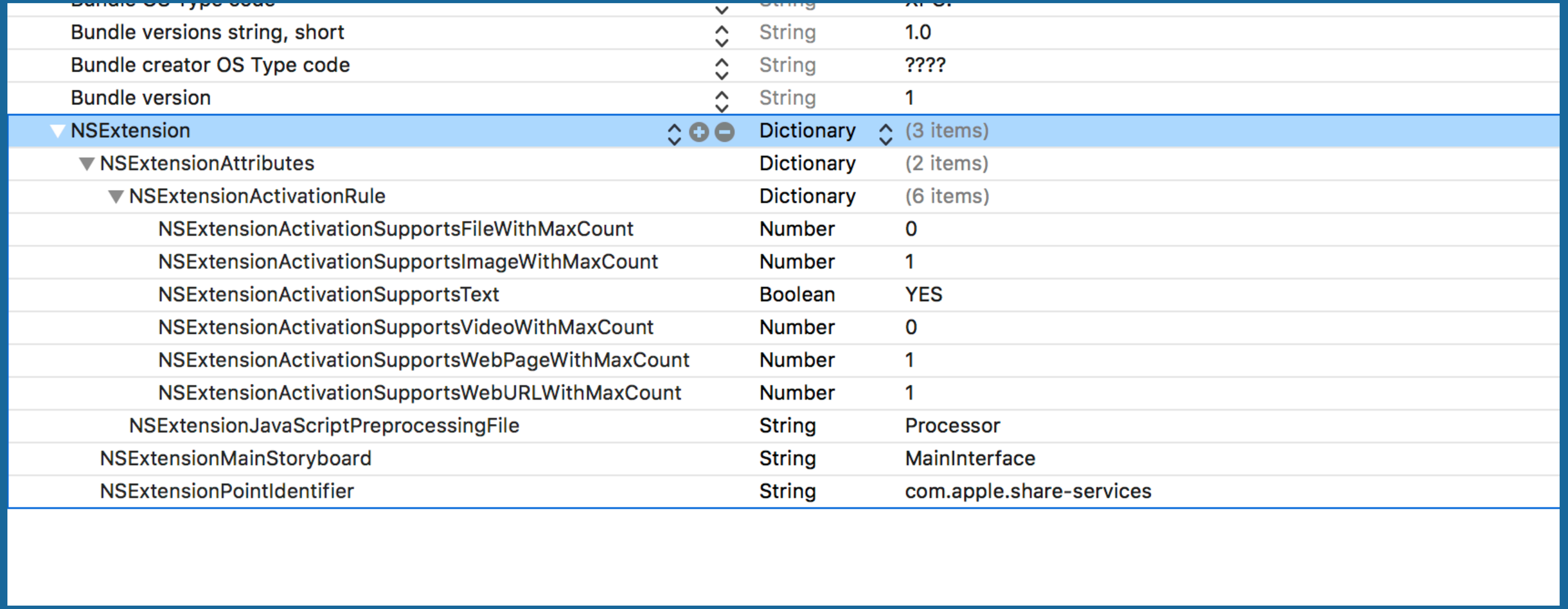


# SHARE EXTENSIONS




- Set activation rules in Info.plist of the extension

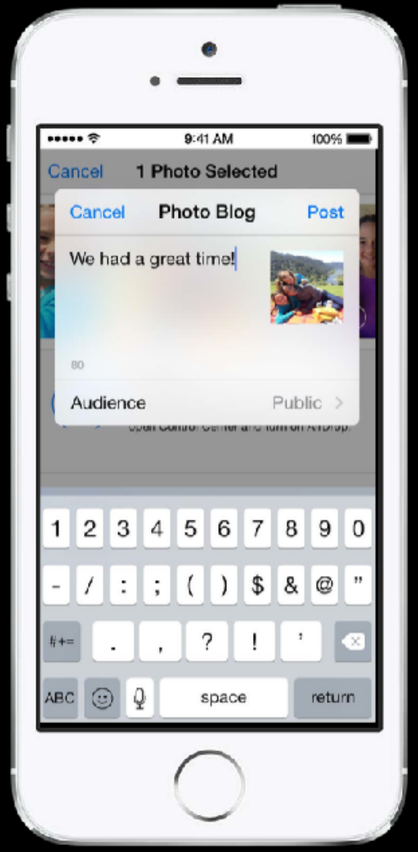
# SHARE EXTENSIONS

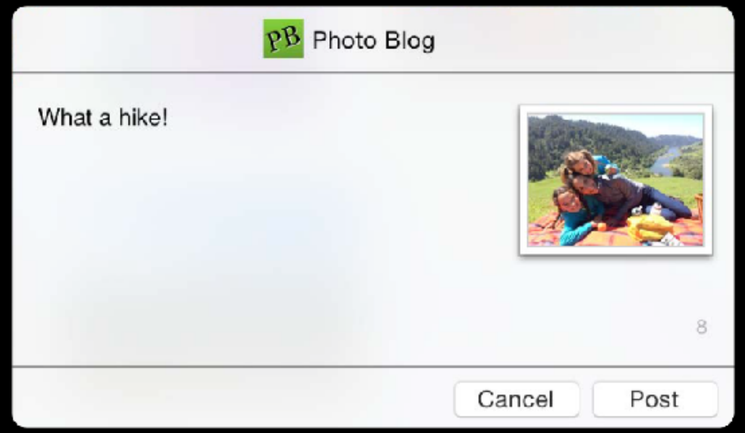



- Condensed rules (instead of SQL query); compile to a predicate

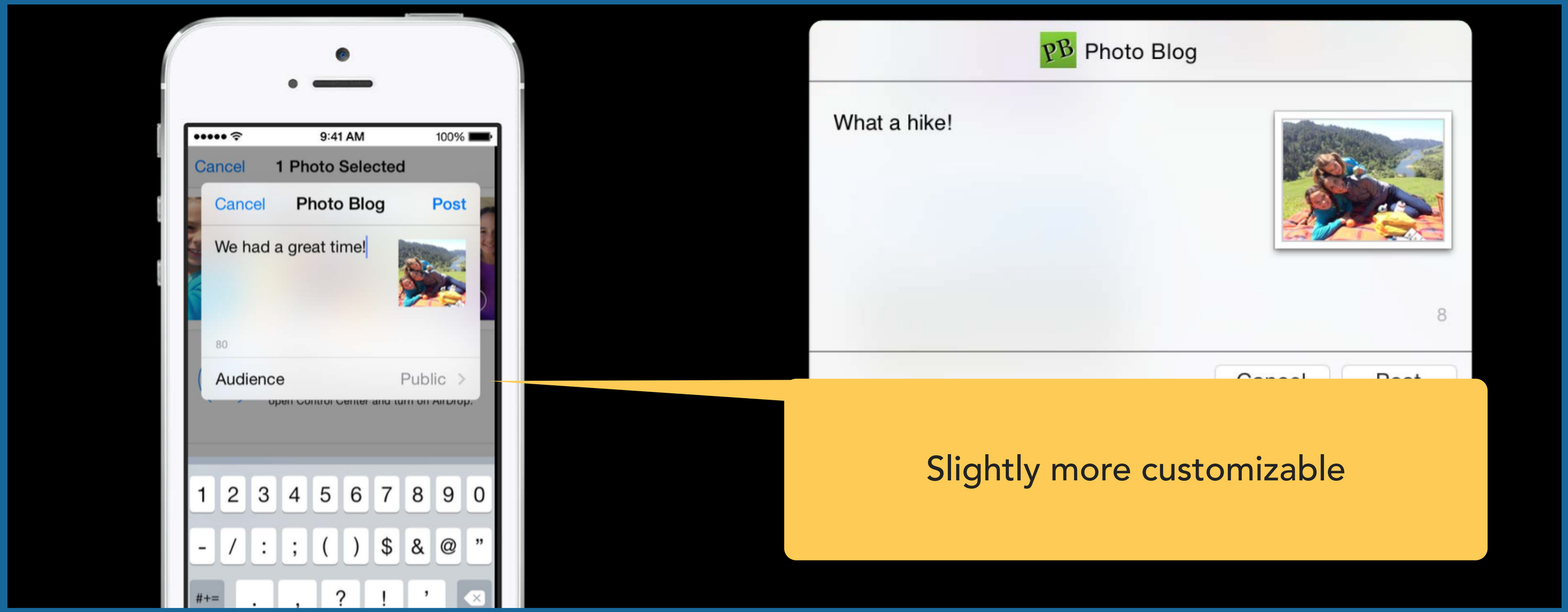
# SHARE EXTENSIONS

- Default UI is `SLComposeServiceViewController` subclass that has same interface as system sharing



# SHARE EXTENSIONS




- You can additional information about the sharedItem from “extensionContext” property

# SHARE EXTENSION FROM PHOTOS

# SHARE EXTENSIONS

SUBTITLE

- Share extension with Photos App
  - Get the image
  - Get the text

```
import UIKit
import Social
import ExtensionKit
import MobileCoreServices

/// View Controller that is presented via the share extension.  This uses the
/// default share sheet from the `Social.framework`.
class ShareViewController: SLComposeServiceViewController {

  override func didSelectPost() {
    printLog("📷")

    // This is called after the user selects Post. Do the upload of
    // contentText and/or NSExtensionContext attachments.

    // While it is possible to save an UIImage to a dictionary as NSData as
    // we are doing here, it is generally not a good idea.  You could store
    // image on server or in App Group folder and keep the URL of the location
    // to store

    // Also, note that you get the app store URL, so you could use it to get
    // the image that way.

    if let item = self.extensionContext?.inputItems[0] as? NSExtensionItem {
      for ele in item.attachments!{
        let itemProvider = ele as! NSItemProvider

        if itemProvider.hasItemConformingToTypeIdentifier("public.jpeg"){
          NSLog("itemprovider: %@", itemProvider)
          itemProvider.loadItem(forTypeIdentifier: "public.jpeg", options: nil,
                                completionHandler: { (item, error) in

            var imgData: Data!
            if let url = item as? URL {
              imgData = try! Data(contentsOf: url)
            }

            if let img = item as? UIImage {
              imgData = UIImagePNGRepresentation(img)
            }

            let sharedItem: NSDictionary = [ "date": Date(),
                                             "contentText": self.contentText,
                                             "imageData": imgData]

            LocalDefaultsManager.sharedInstance.add(object: sharedItem)


          })
```

# SHARE EXTENSIONS

```
override func didSelectPost() {

  if let item = self.extensionContext?.inputItems[0] as? NSExtensionItem {
    for ele in item.attachments!{
      let itemProvider = ele as! NSItemProvider

      if itemProvider.hasItemConformingToTypeIdentifier("public.jpeg"){
        NSLog("itemprovider: %@", itemProvider)
        itemProvider.loadItem(forTypeIdentifier: "public.jpeg", options: nil,
                              completionHandler: { (item, error) in

          var imgData: Data!
          if let url = item as? URL {
            imgData = try! Data(contentsOf: url)
          }

          if let img = item as? UIImage {
            imgData = UIImagePNGRepresentation(img)
          }

          let sharedItem: NSDictionary = [ "date": Date(),
                                           "contentText": self.contentText,
                                           "imageData": imgData]

          LocalDefaultsManager.sharedInstance.add(object: sharedItem)


        })
      }
    }
  }
```

Get image

Get text

Save somewhere that you app can read it

# SHARE EXTENSIONS

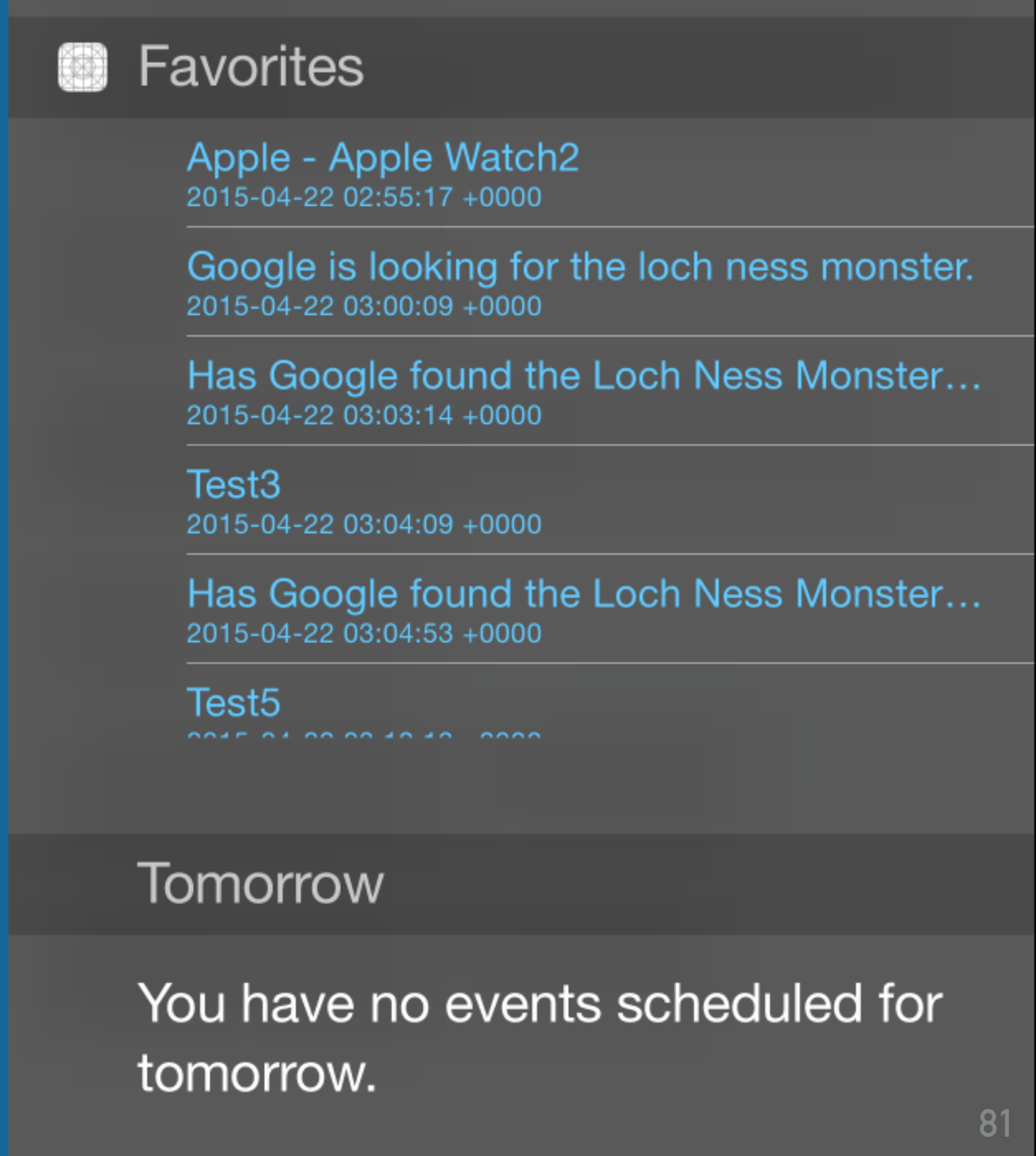
```
  override func isContentValid() -> Bool {
    // Do validation of contentText and/or NSExtensionConte
    return true
  }

  override func configurationItems() -> [Any]! {
    // To add configuration options via table cells at the
    // return an array of SLComposeSheetConfigurationItem h
    return []
  }
}
```

# NOTIFICATION CENTER WIDGETS

# Notification Center Widgets

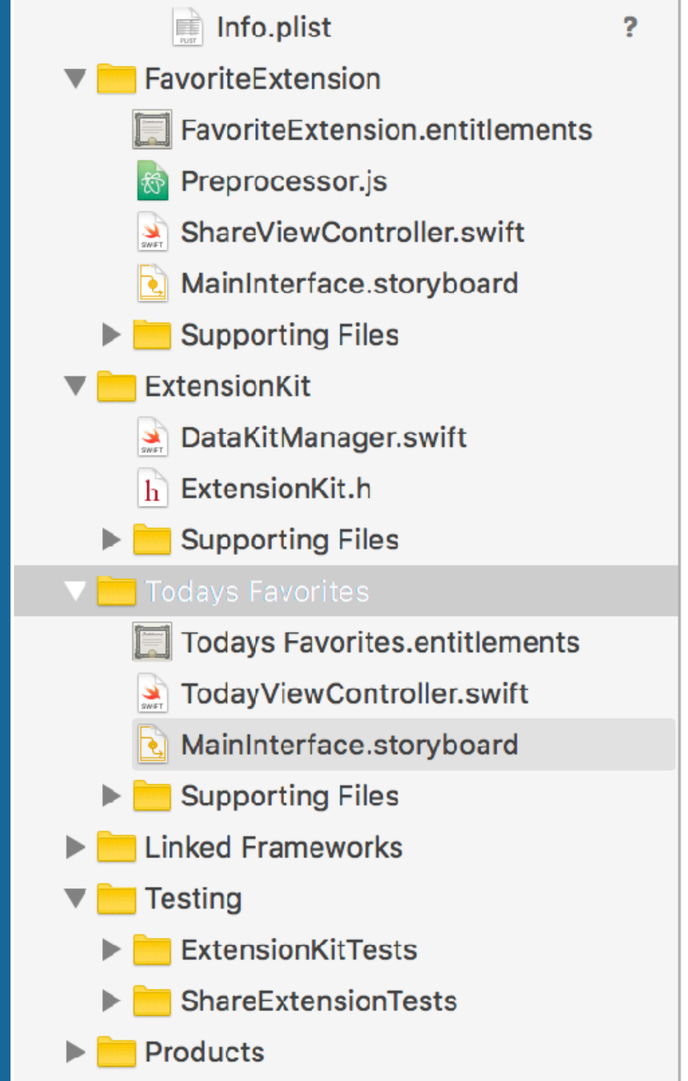
SUBTITLE

- Widgets are view controllers
  - Use the same methods
- Notification Center handles layout
  - Express preferred height with constraints or `preferredContentSize`
- Your responsibility
  - Animate content along with resize animation
  - Handle update requests

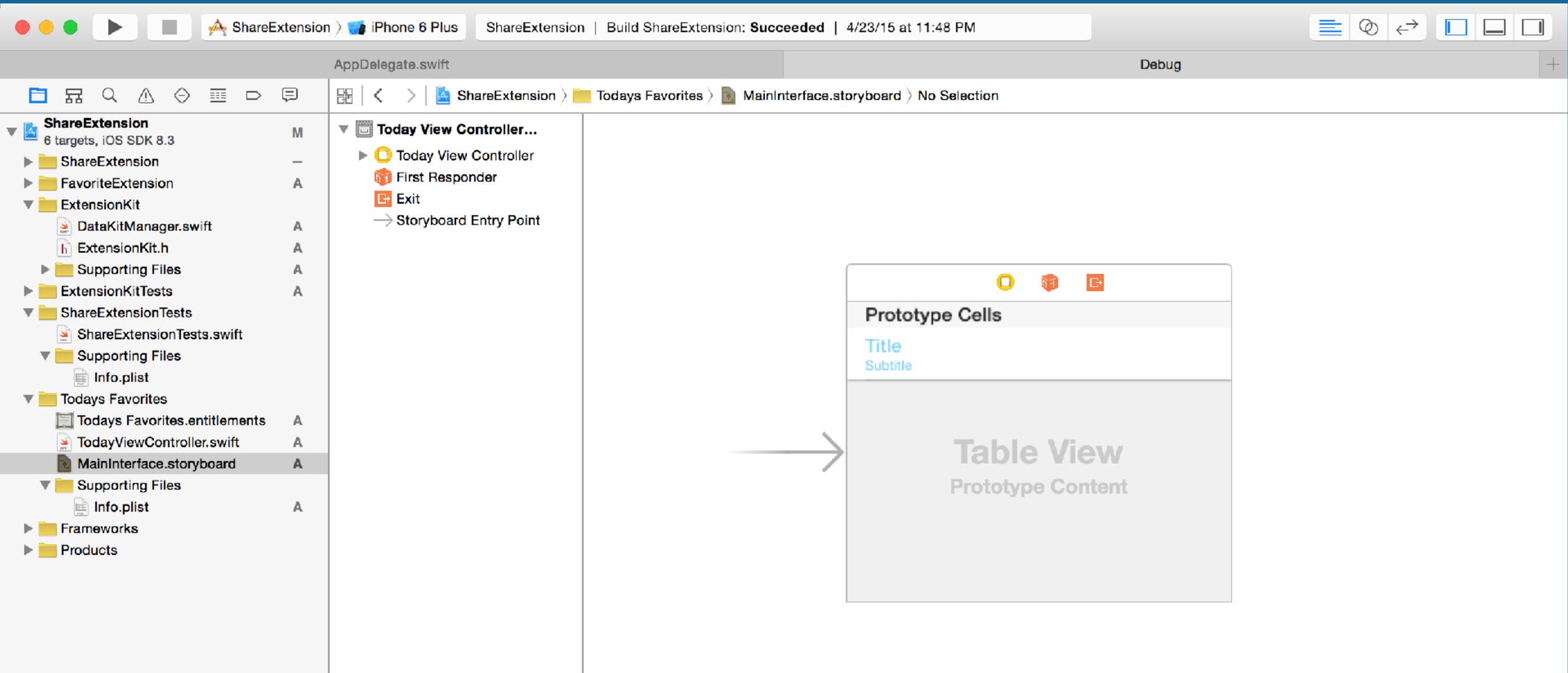


# NOTIFICATION CENTER WIDGETS

- Entitlements
- View controller
- Storyboard (optional)




# NOTIFICATION CENTER WIDGETS



# NOTIFICATION CENTER WIDGETS

- `NCWidgetProviding` protocol allows you to customize appearance of a widget
  - Size
  - Updating
  - Insets

NotificationCenter › NCWidgetProviding

Protocol

## NCWidgetProviding

The `NCWidgetProviding` protocol lets you customize some of the appearance and behavior of a widget (a widget is an app extension).

### Symbols

Customizing the Display

```
func widgetMarginInsets(forProposedMarginInsets: U
```

Called to let a widget accept the default margin inset values o instead.

```
func widgetActiveDisplayModeDidChange(NCWidgetDispl
CGSize)
```

# NOTIFICATION CENTER WIDGETS

- Data source needs to be accessible from widget
  - App Groups

```
import UIKit
import NotificationCenter
import ExtensionKit

class TodayViewController: UIViewController {

  /// Array to hold all the shared items loaded from NSUserDefaults
  var sharedItems: NSMutableArray = []

  @IBOutlet weak var tableView: UITableView!


  //
  // MARK: - Lifecycle
  //

  override func viewDidLoad() {
    super.viewDidLoad()

    // Size the widget for initial view
    self.extensionContext?.widgetLargestAvailableDisplayMode = NCWidgetDisplayMode.expanded

    // Retrieve the list of apps that is stored in AppGroups UserDefaults
    sharedItems = LocalDefaultsManager.sharedInstance.currentList()
  }

  //
  // MARK: - Table view data source
  //
  func numberOfSectionsInTableView(_ tableView: UITableView) -> Int {
    return 1
  }

  func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return sharedItems.count
  }

  func tableView(_ tableView: UITableView, cellForRowAtIndexPath indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)
    let sharedItemDictionary: NSDictionary = sharedItems[indexPath.row] as! NSDictionary

    cell.textLabel!.text = sharedItemDictionary["contentText"] as? String
    cell.detailTextLabel!.text = (sharedItemDictionary["date"] as? Date)!.description

    // The images are stored here as NSData inside of NSUserDefaults, we
    // need to convert to an UIImage before we add it to the table
    if let imageData = sharedItemDictionary["imageData"] as? Data {
      cell.imageView!.image = UIImage(data: imageData)
    }

    return cell
  }
```

# NOTIFICATION CENTER WIDGETS

```
// MARK: - Table view data delegate methods
//


func tableView(_ tableView: UITableView, didSelectRowAtIndexPath indexPath: IndexPath) {
  tableView.deselectRow(at: indexPath, animated: true)
  let row = indexPath.row

  // We are using a custom URL to pass the index of the tapped cell
  // You can use this to customize the launch of the container application
  let launchString = "mobiuchicagoshare://indexPathRow=\(row)"
  self.extensionContext?.open(URL(string: launchString)!, completionHandler:nil)
}
```

- Launch the application with a custom URL to  pass the cell tapped

# NOTIFICATION CENTER WIDGETS

OPEN FROM TERMINATED

```
class AppDelegate: UIResponder, UIApplicationDelegate {

  var window: UIWindow?


  func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

    // Print out the app group path for debugging
    let sharedAppGroup: String = "group.extensiontest2"
    let directory: URL = FileManager.default.containerURL(forSecurityApplicationGroupIdentifier: sharedAppGroup)!
    print("App Group URL: \(directory)")

    // If the app was not in the background, it will launch with the options
    // key `UIAPplicationLaunchOptionsURLKey`
    if let options = launchOptions {
      let alertView = UIAlertView(title: "Launch Options", message: options.description,
                                  delegate: nil,
                                  cancelButtonTitle: "Cancel",
                                  otherButtonTitles: "OK")
      alertView.alertViewStyle = .default
      alertView.show()
    }
    return true
  }
```

OPEN FROM SUSPENDED

```
  func application(_ application: UIApplication, open url: URL, sourceApplication: String?, annotation: Any) -> Bool {
    print("The application was launched from the url: \(url)")
    return true
  }
```

# NOTIFICATION CENTER WIDGETS

```
class AppDelegate: UIResponder, UIApplicationDelegate {

  var window: UIWindow?


  func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

    // Print out the app group path for debugging
    let sharedAppGroup: String = "group.extensiontest2"
    let directory: URL = FileManager.default.containerURL(forSecurityApplicationGroupIdentifier: sharedAppGroup)!
    print("App Group URL: \(directory)")

    // If the app was not in the background, it will launch with the options
    // key `UIAPplicationLaunchOptionsURLKey`
    if let options = launchOptions {
      let alertView = UIAlertView(title:
                                  delegat
                                  cancelB
                                  otherBu
      alertView.alertViewStyle = .default
      alertView.show()
    }
    return true
  }

  func application(_ application: UIApplication, open url: URL, sourceApplication: String?, annotation: Any) -> Bool {
    print("The application was launched from the url: \(url)")
    return true
  }
```

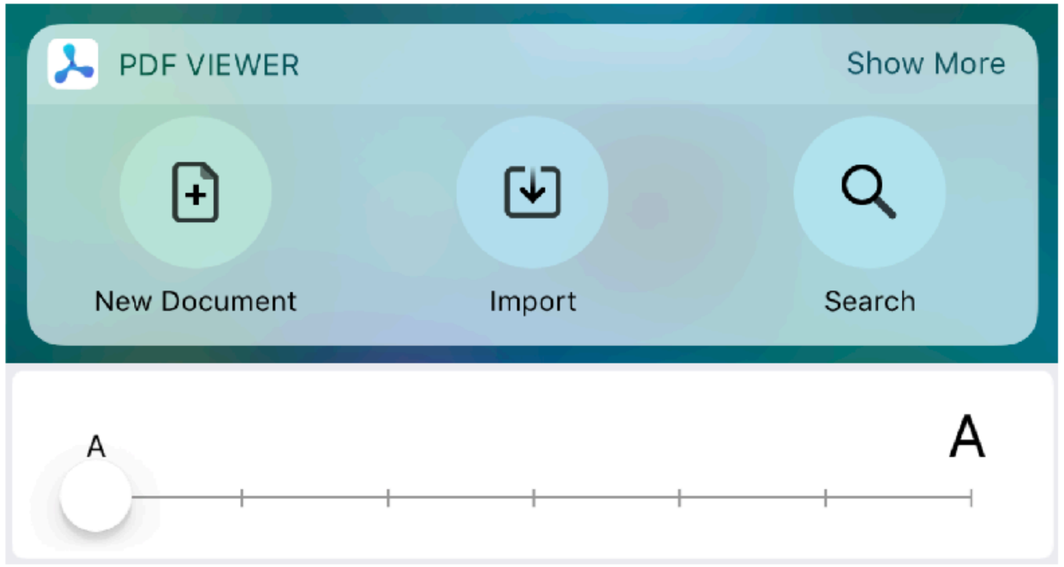
SINCE WE ARE USING THE SAME DATA SOURCE WE CAN CUSTOMIZE THE LAUNCH

# NOTIFICATION CENTER WIDGETS

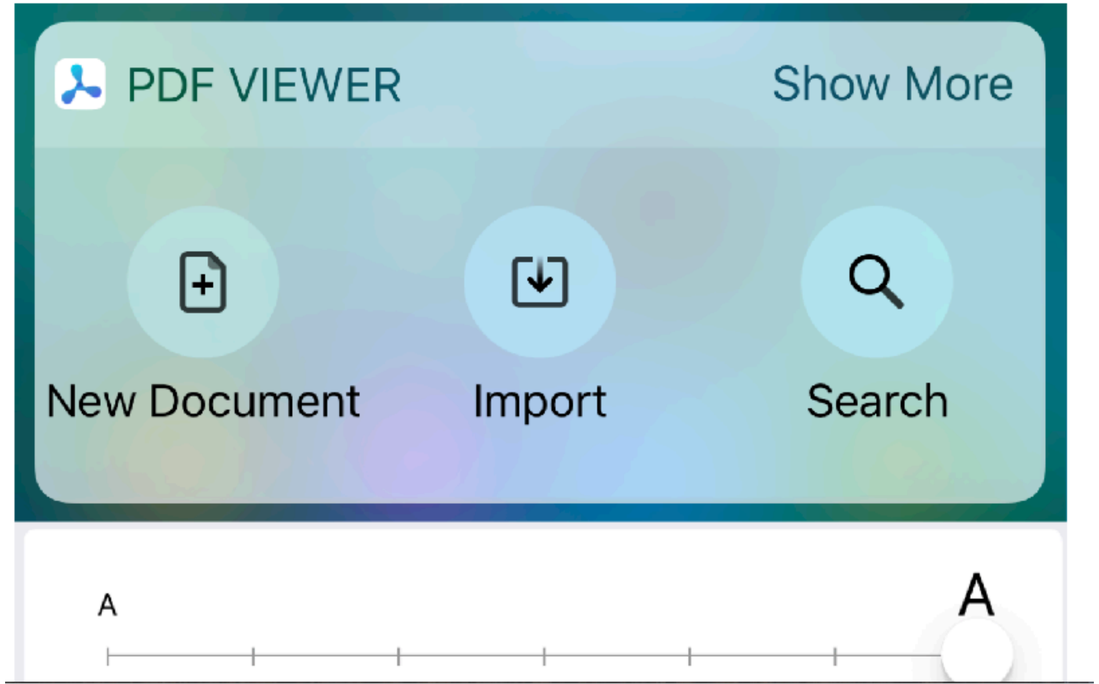
- Dynamic sizing
  - Widget is on the left or right column on iPad
  - Widget is shown in Notification Center or the Today view
  - Accessed from the Home or Lock screen
  - Portrait or landscape orientation
  - Users text size setting

https://pspdfkit.com/blog/2017/today-widget/#

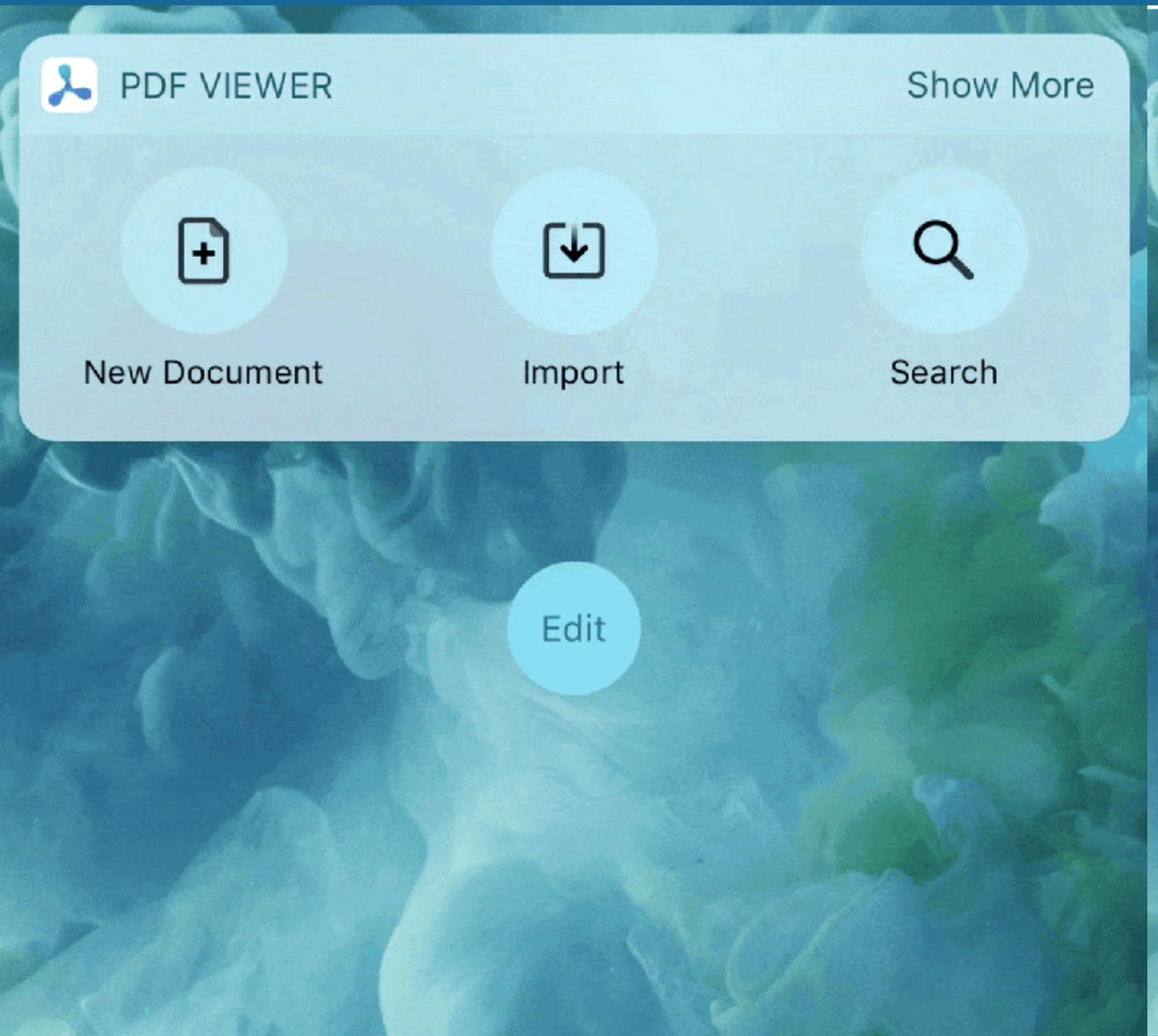
Small Text Size

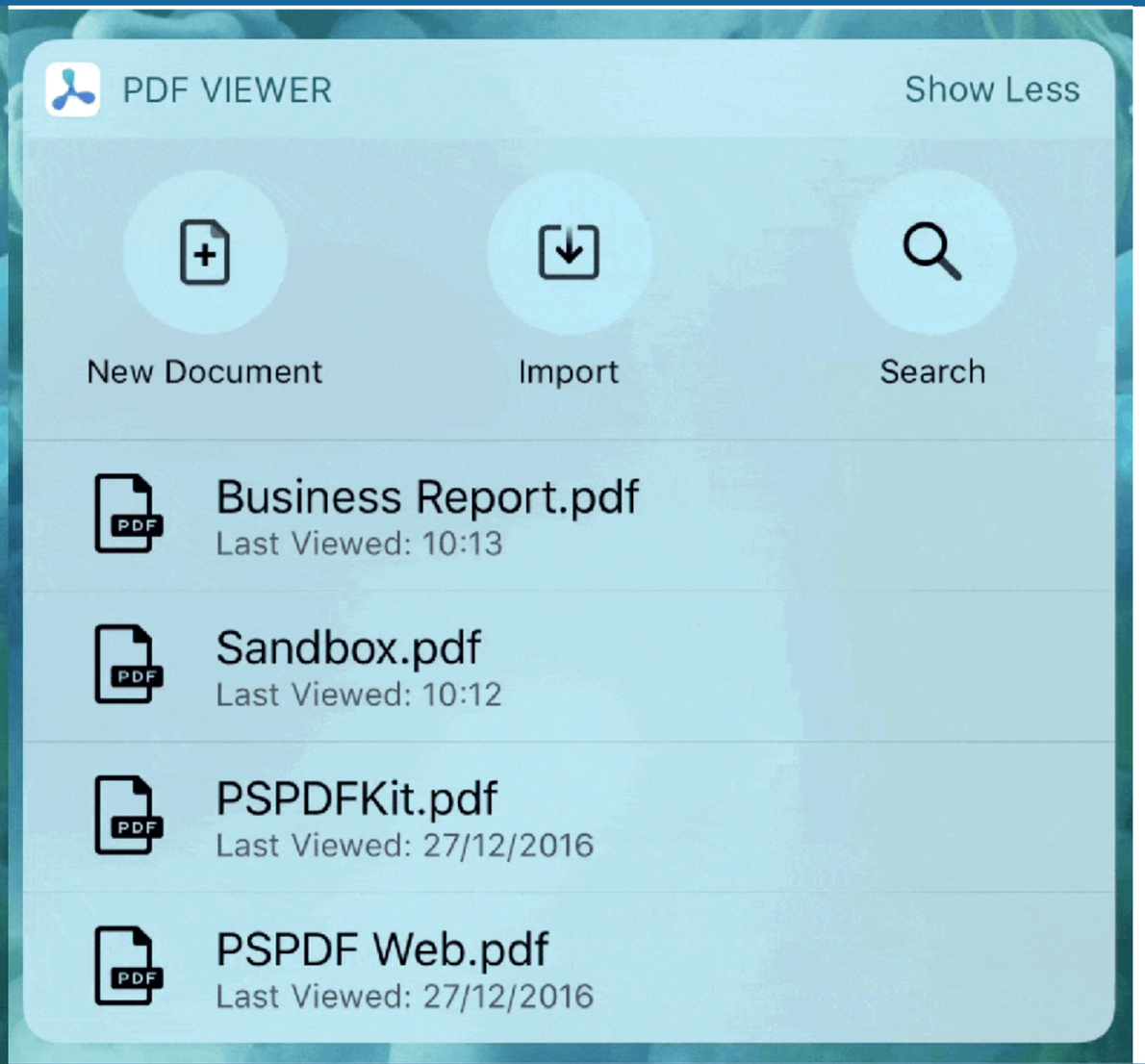



Large Text Size



# NOTIFICATION CENTER WIDGETS



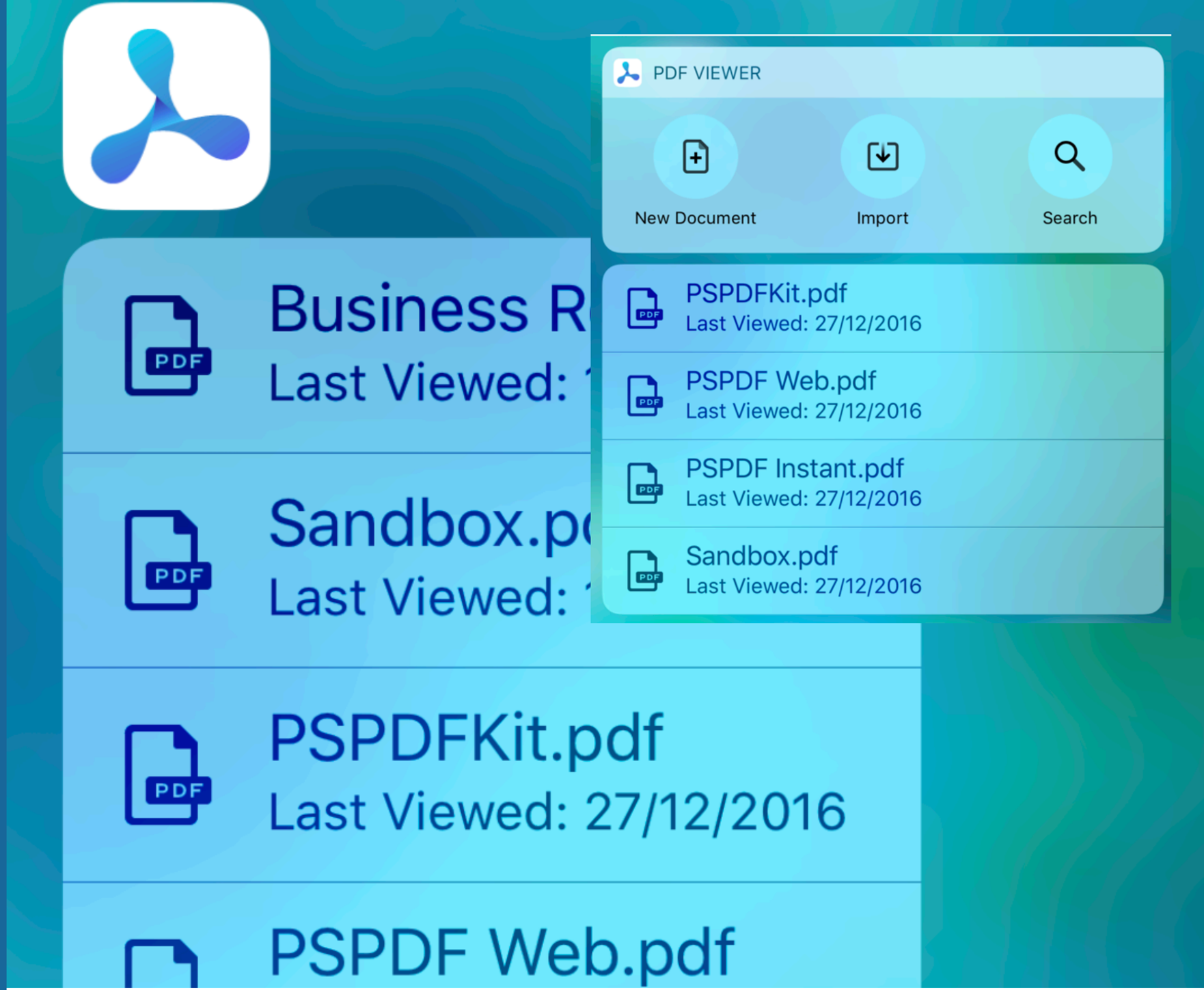
# NOTIFICATION CENTER WIDGETS

```
// MARK: - NCWidgetProviding Methods
//


extension TodayViewController : NCWidgetProviding {

  func widgetActiveDisplayModeDidChange(_ activeDisplayMode: NCWidgetDisplayMode, withMaximumSize maxSize: CGSize){
    if (activeDisplayMode == NCWidgetDisplayMode.compact) {
      self.preferredContentSize = maxSize;
    } else {
      self.preferredContentSize = CGSize(width: 0, height: 400);
    }
  }


  func widgetPerformUpdate(completionHandler: (@escaping (NCUpdateResult) -> Void))
    // Perform any setup necessary in order to update the view.
    // If an error is encountered, use NCUpdateResult.Failed
    // If there's no update required, use NCUpdateResult.NoData
    // If there's an update, use NCUpdateResult.NewData

    // Refresh table
    self.tableView.reloadData()
    completionHandler(NCUpdateResult.newData)
  }

}
```

SIZING IS NOT GUARANTEED, USE STACK VIEWS FOR MORE FLEXIBILITY

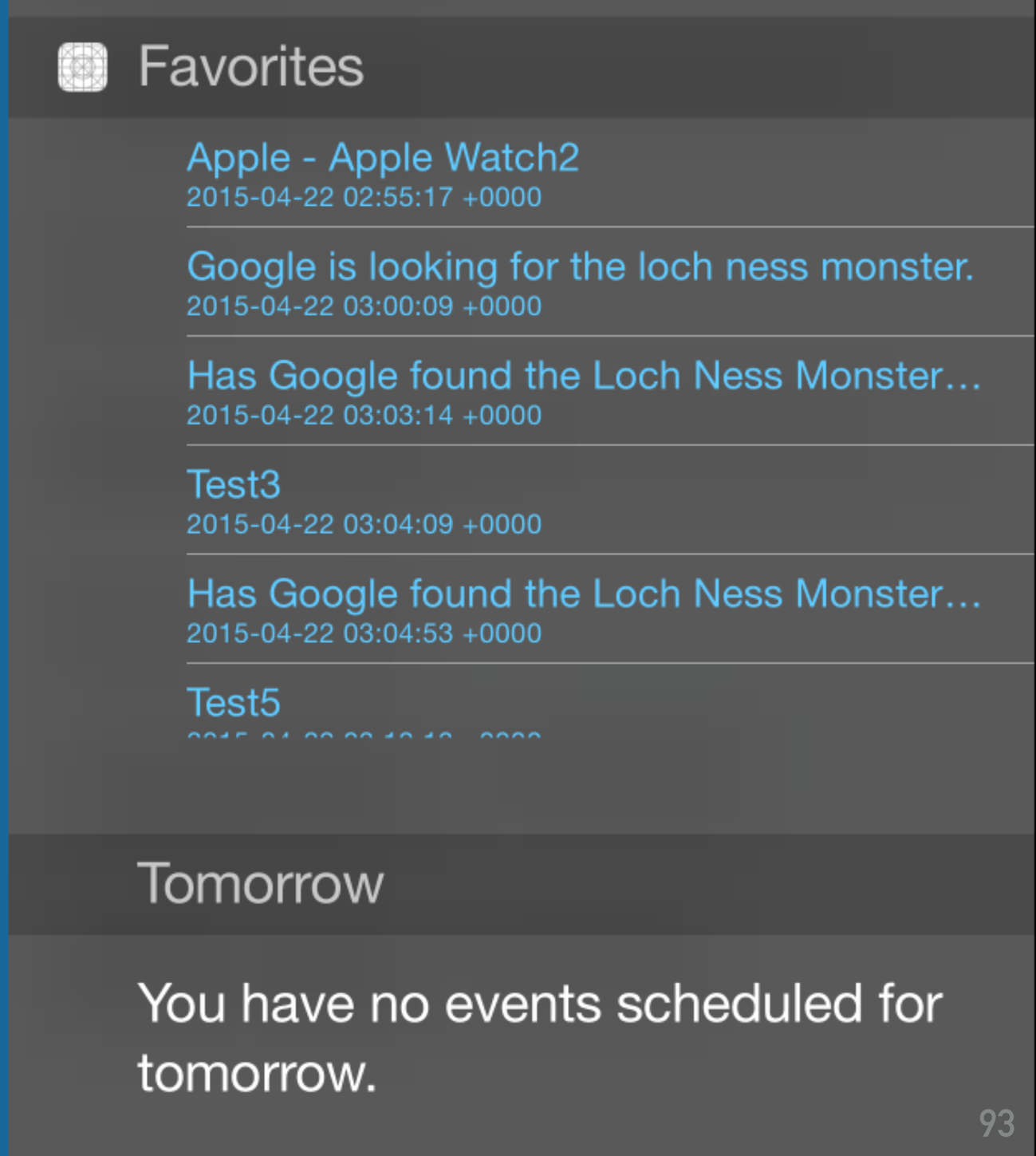
# NOTIFICATION CENTER WIDGETS

- Quick actions also show up on home screen force touch
- Different appearance based on quick actions



# NOTIFICATION CENTER WIDGETS

- Debugging notes
  - Run the container application first
  - Run the extension from its scheme to get debugging information
  - User Interface
    - Optimize for size and visibility
    - Think about how users use today widget



THE UNIVERSITY OF CHICAGO

# ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 4