THE UNIVERSITY OF CHICAGO

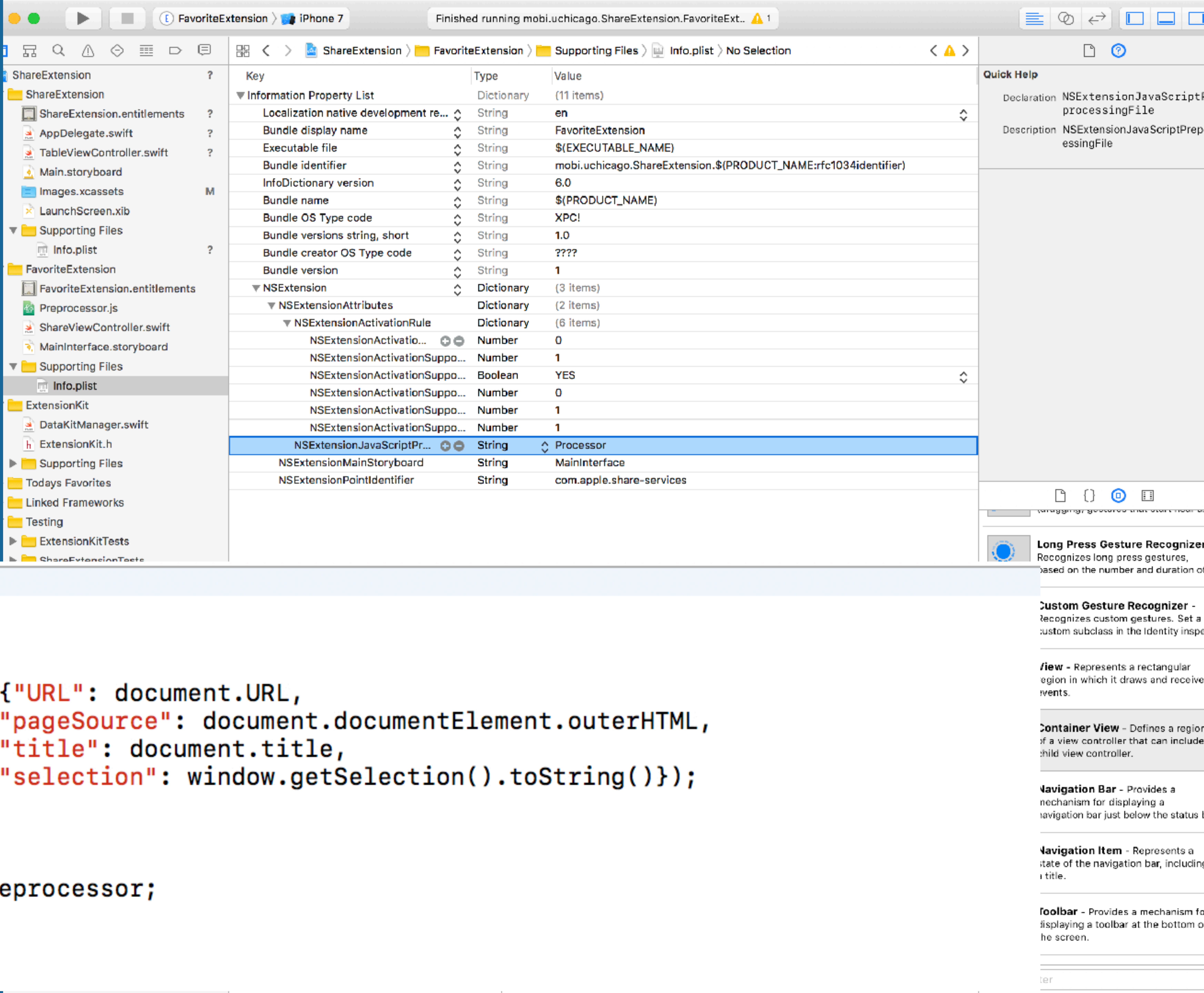
ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 4

SHARE EXTENSION FROM A WEBPAGE

SHARE EXTENSIONS

- Extension parses a web page with javascript



FavoriteExtension 〉 iPhone 7 — Finished running mobi.uchicago.ShareExtension.FavoriteExt..

ShareExtension 〉 FavoriteExtension 〉 Supporting Files 〉 Info.plist 〉 No Selection

- ShareExtension
  - ShareExtension
    - ShareExtension.entitlements
    - AppDelegate.swift
    - TableViewController.swift
    - Main.storyboard
    - Images.xcassets
    - LaunchScreen.xib
    - Supporting Files
      - Info.plist
  - FavoriteExtension
    - FavoriteExtension.entitlements
    - Preprocessor.js
    - ShareViewController.swift
    - MainInterface.storyboard
    - Supporting Files
      - Info.plist
  - ExtensionKit
    - DataKitManager.swift
    - ExtensionKit.h
    - Supporting Files
  - Todays Favorites
  - Linked Frameworks
  - Testing
  - ExtensionKitTests
  - ShareExtensionTests

Key	Type	Value
Information Property List	Dictionary	(11 items)
Localization native development re...	String	en
Bundle display name	String	FavoriteExtension
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	mobi.uchicago.ShareExtension.$(PRODUCT_NAME:rfc1034identifier)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	XPC!
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1
NSExtension	Dictionary	(3 items)
NSExtensionAttributes	Dictionary	(2 items)
NSExtensionActivationRule	Dictionary	(6 items)
NSExtensionActivatio...	Number	0
NSExtensionActivationSuppo...	Number	1
NSExtensionActivationSuppo...	Boolean	YES
NSExtensionActivationSuppo...	Number	0
NSExtensionActivationSuppo...	Number	1
NSExtensionActivationSuppo...	Number	1
NSExtensionJavaScriptPr...	String	Processor
NSExtensionMainStoryboard	String	MainInterface
NSExtensionPointIdentifier	String	com.apple.share-services

Quick Help

Declaration NSExtensionJavaScriptP processingFile

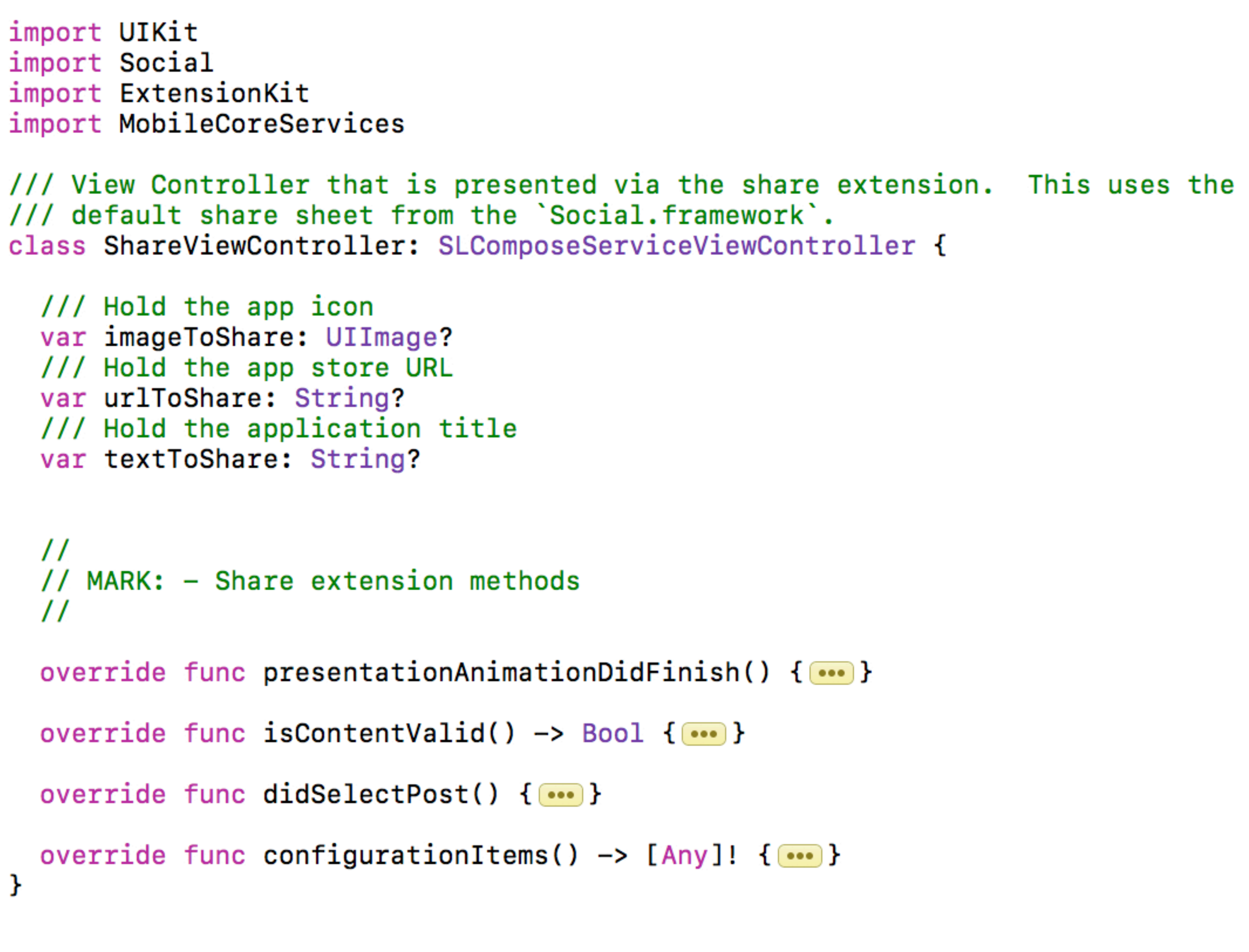
Description NSExtensionJavaScriptPrep essingFile

Long Press Gesture Recognizer
Recognizes long press gestures,
based on the number and duration o

```
var Preprocessor = function() {};

Preprocessor.prototype = {
    run: function(arguments) {
        arguments.completionFunction({"URL": document.URL,
                                      "pageSource": document.documentElement.outerHTML,
                                      "title": document.title,
                                      "selection": window.getSelection().toString()});
    }
};

var ExtensionPreprocessingJS = new Preprocessor;
```

SHARE EXTENSIONS

- SLComposeServiceViewController subclass
- Custom UI will implement typical view controller methods



```swift
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 {

  /// Hold the app icon
  var imageToShare: UIImage?
  /// Hold the app store URL
  var urlToShare: String?
  /// Hold the application title
  var textToShare: String?


  //
  // MARK: - Share extension methods
  //

  override func presentationAnimationDidFinish() {•••}

  override func isContentValid() -> Bool {•••}

  override func didSelectPost() {•••}

  override func configurationItems() -> [Any]! {•••}
}
```

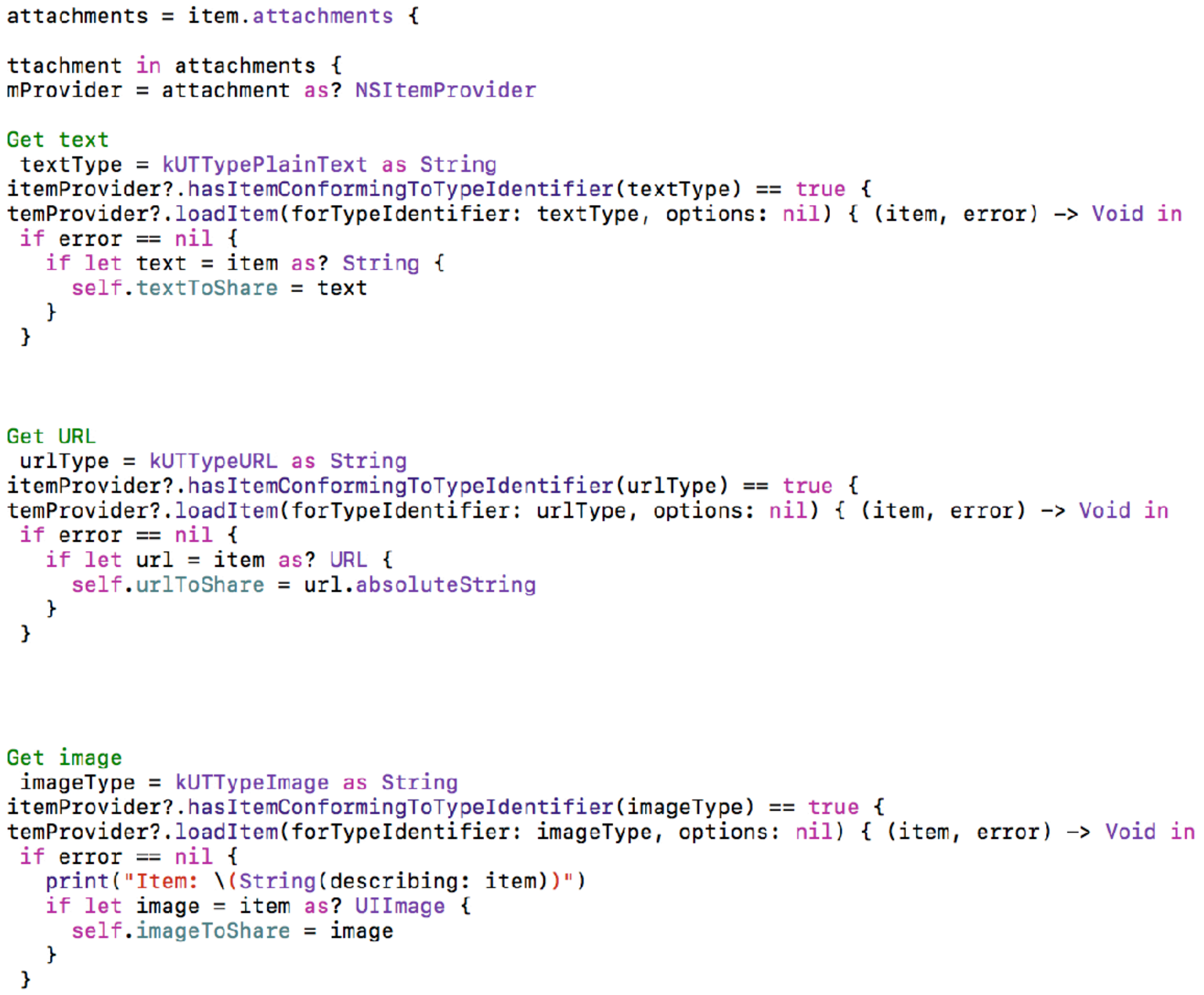
SHARE EXTENSIONS

- Iterate through and get all the attachments

```
override func presentationAnimationDidFinish() {
  let items = extensionContext?.inputItems
  var itemProvider: NSItemProvider?

  // Exit early if there are no items
  if items == nil && items!.isEmpty != false { return }

  // Get item
  let item = items![0] as! NSExtensionItem
  print("Attachment items: \(String(describing: items))")

  // Loop through all the attachment items and store their values in properties
  //
  // Note: The App Store makes three attachements available: image, plain-text, and url
  //

  if let attachments = item.attachments {

    for attachment in attachments {
      itemProvider = attachment as? NSItemProvider

      // Get text
      let textType = kUTTypePlainText as String
      if itemProvider?.hasItemConformingToTypeIdentifier(textType) == true {
        itemProvider?.loadItem(forTypeIdentifier: textType, options: nil) {•••}
      }

      // Get URL
      let urlType = kUTTypeURL as String
      if itemProvider?.hasItemConformingToTypeIdentifier(urlType) == true {•••}

      // Get image
      let imageType = kUTTypeImage as String
      if itemProvider?.hasItemConformingToTypeIdentifier(imageType) == true {
        itemProvider?.loadItem(forTypeIdentifier: imageType, options: nil) {•••}
      }

    }
  }
```

SHARE EXTENSIONS

- Process the callback from javascript to extract the information
- Get info and save to property in view controller to access later



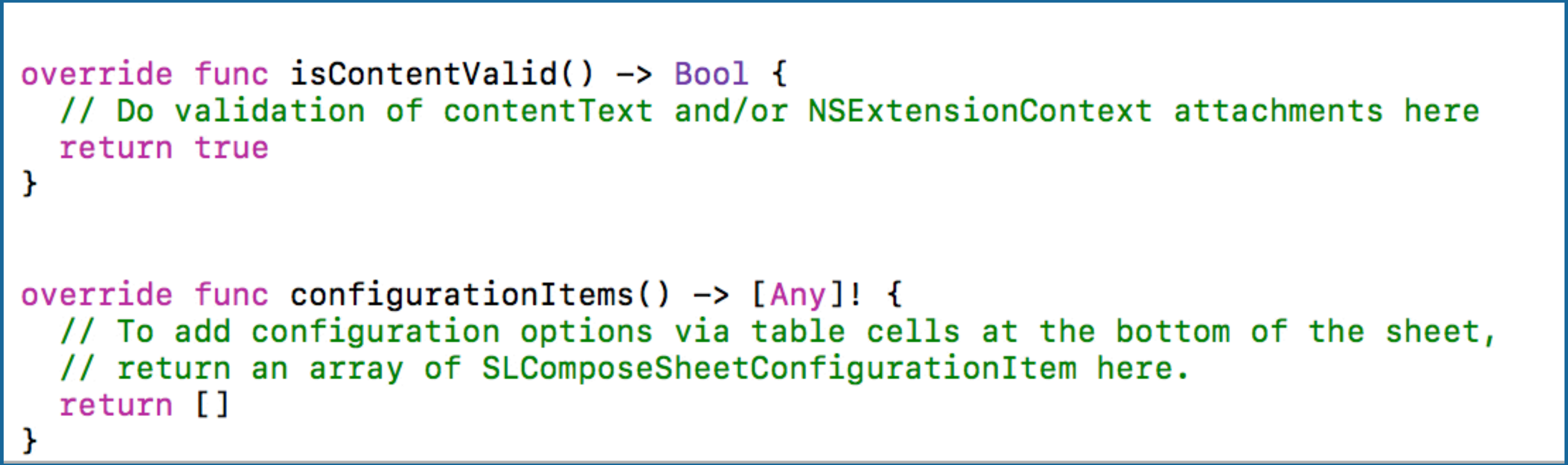
```
attachments = item.attachments {

ttachment in attachments {
mProvider = attachment as? NSItemProvider

Get text
 textType = kUTTypePlainText as String
itemProvider?.hasItemConformingToTypeIdentifier(textType) == true {
temProvider?.loadItem(forTypeIdentifier: textType, options: nil) { (item, error) -> Void in
 if error == nil {
   if let text = item as? String {
     self.textToShare = text
   }
 }



Get URL
 urlType = kUTTypeURL as String
itemProvider?.hasItemConformingToTypeIdentifier(urlType) == true {
temProvider?.loadItem(forTypeIdentifier: urlType, options: nil) { (item, error) -> Void in
 if error == nil {
   if let url = item as? URL {
     self.urlToShare = url.absoluteString
   }
 }




Get image
 imageType = kUTTypeImage as String
itemProvider?.hasItemConformingToTypeIdentifier(imageType) == true {
temProvider?.loadItem(forTypeIdentifier: imageType, options: nil) { (item, error) -> Void in
 if error == nil {
   print("Item: \(String(describing: item))")
   if let image = item as? UIImage {
     self.imageToShare = image
   }
 }
```

SHARE EXTENSIONS

```swift
override func didSelectPost() {
  // 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.
  let imageData: Data? = UIImagePNGRepresentation(imageToShare!)
  let sharedItem: NSDictionary = [ "date": Date(),
                                   "contentText": textToShare!,
                                   "imageData": imageData!,
                                   "url": urlToShare!
  ]

  LocalDefaultsManager.sharedInstance.add(object: sharedItem)


  // Inform the host that we're done, so it un-blocks its UI.
  // Note: Alternatively you could call super's -didSelectPost, which will similarly complete the extension context.
  self.extensionContext!.completeRequest(returningItems: [], completionHandler: nil)
}
```

USER TAPS POST; ADD TO USER DEFAULTS IN APP GROUP

SHARE EXTENSIONS

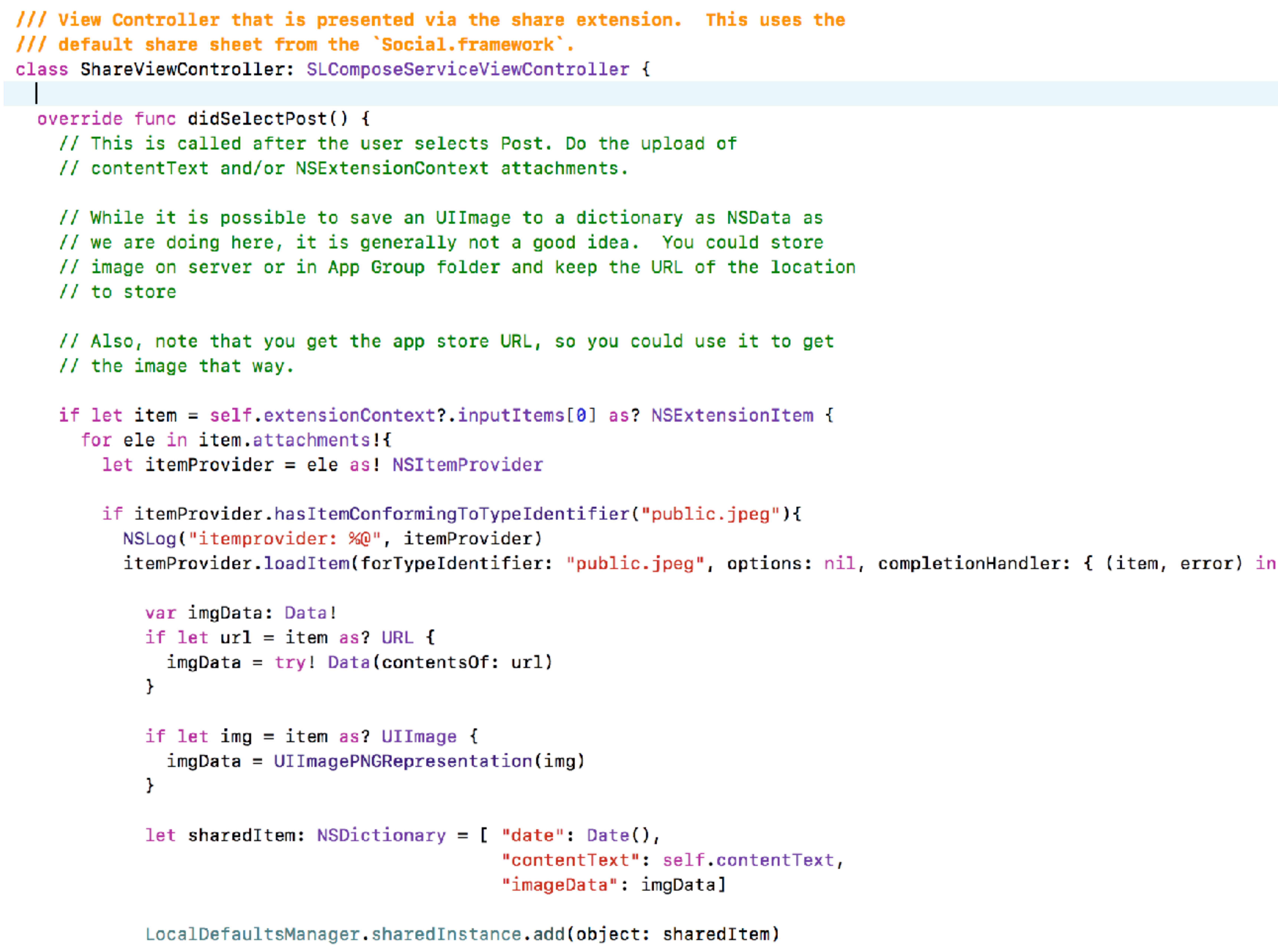


```swift
override func isContentValid() -> Bool {
  // Do validation of contentText and/or NSExtensionContext attachments here
  return true
}


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

- Do custom validation or add more configuration items

PHOTO SHARING EXTENSION

- Customize the parsing of the NSItemProviders



```swift
/// 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() {
    // 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)
```

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 4