

IOS 8 WIDGETS

Notification Center Today Extensions

KIM AHLBERG

We have already had some Widgets since iOS5. Calendar, Reminders, Weather... all made by Apple.

EXTENSIONS INTRODUCED AT WWDC 2014

**Share, Action, Photo Editing, Finder,
Document Provider, Custom Keyboard,
Today**

At WWDC Apple opened up several extension points to 3rd party developers in OS X Yosemite and iOS 8.

Extensions are delivered in extension bundles contained in app store apps. On iOS, the containing app must provide some functionality in addition to the extension.

TODAY EXTENSIONS

(A.K.A. WIDGETS)

Widgets appear in Notification Center. They should give quick access to information and optionally let the user perform simple tasks.

Widgets can be accessible the from lock screen. Users control which widgets are shown using the "Edit" button.



CONTENTS OF A WIDGET

A widget exposes a `UIViewController` which implements the `NCWidgetProviding` protocol and has its `extensionContext` property set to an `NSExtensionContext` instance.

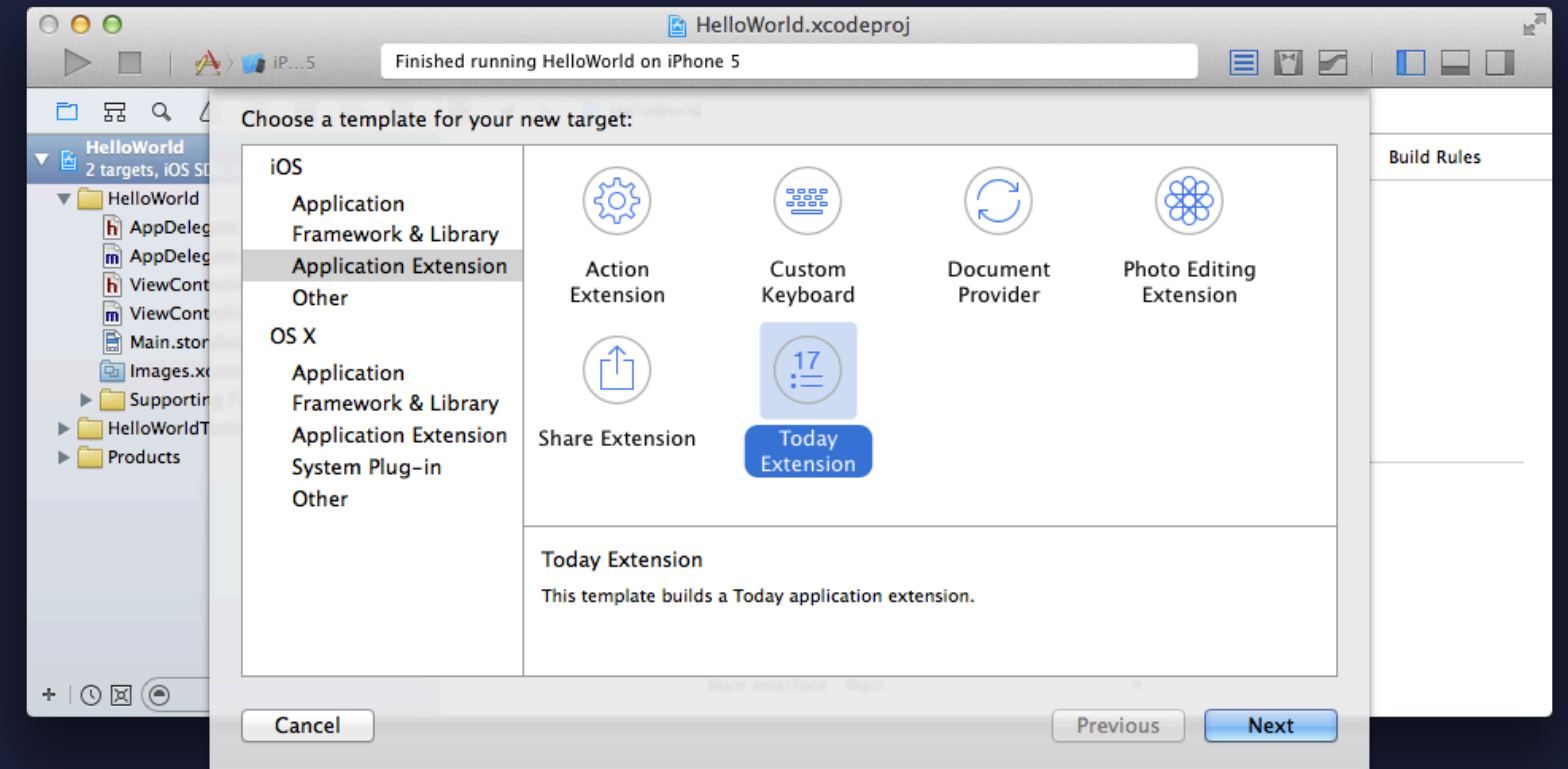
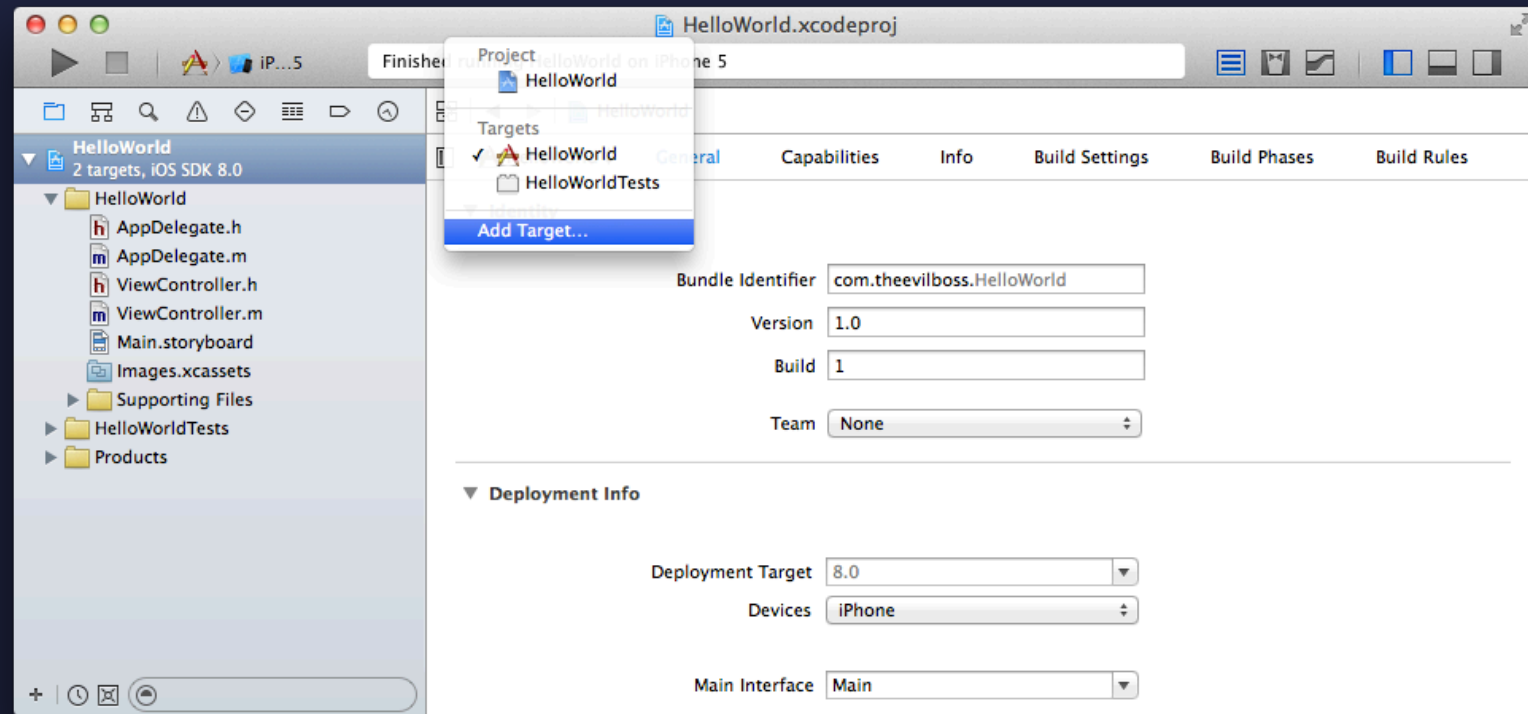
The `NCWidgetProviding` protocol lets you customize the appearance and behavior of a widget.

An `NSExtensionContext` object represents the context in which an app extension is invoked, we'll use its `openURL:` method later.



Demo





Adding the Hello World widget to our app was a simple matter of adding a Today Extension target to the project.

USER EXPERIENCE CONSIDERATIONS

- ▶ Immediate refresh
- ▶ Focused interactivity
- ▶ Look like it belongs

iOS shows a snapshot of your widget's previous state until the content has been refreshed.

Only expose simple interactions. Defer advanced interactions to the app.

Use the default background color.

LIMITATIONS TO KEEP IN MIND

- ▶ **Keyboard input not supported**
- ▶ **Avoid scrollable content**



Scrollable content can interfere with navigation in Notification Center.

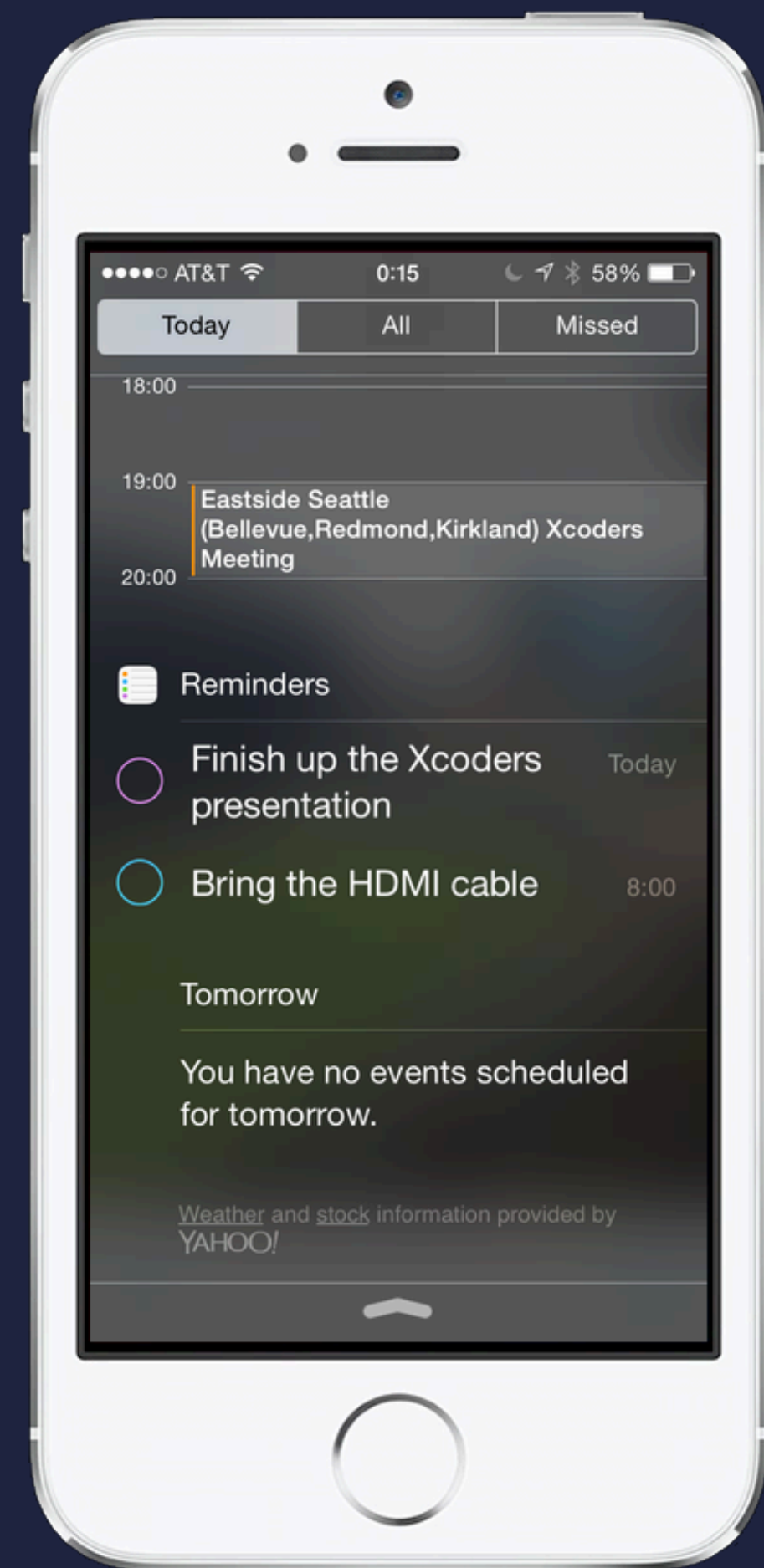
LAYOUT

- ▶ **Frame provided by Notification Center**
 - ▶ **Fixed width**
 - ▶ **Configurable margin insets**
 - ▶ **Adjustable height**

CONFIGURABLE MARGIN INSETS

THE **NCWIDGETPROVIDING** PROTOCOL
LETS YOU CUSTOMIZE YOUR WIDGET'S
MARGIN INSETS

The Reminders widget
displays checkbox images in
the left margin.



CONFIGURABLE MARGIN INSETS

```
- (UIEdgeInsets)widgetMarginInsetsForProposedMarginInsets:  
    (UIEdgeInsets)defaultMarginInsets  
{  
    UIEdgeInsets newMarginInsets = defaultMarginInsets;  
    newMarginInsets.left = 0.0;  
  
    return newMarginInsets;  
}
```

It's a good idea to retain all default margins except the one you want to change.

Default insets; top: 0, left: 47, right: 0, bottom: 39 points.

ADJUSTABLE HEIGHT

WE CAN USE `UIVIEWCONTROLLER'S` `SETPREFERREDCONTENTSIZE:` METHOD TO SET THE WIDGET HEIGHT.



Alternatively we can modify our
Auto Layout constraints to
change the view's content height.

ADJUSTABLE HEIGHT

```
- (IBAction)heightToggleButtonTapped:(id)sender  
{  
    CGSize updatedSize = self.preferredContentSize;  
    updatedSize.height = 100;  
  
    [self setPreferredSize:updatedSize];  
}
```

Don't make your content too tall.

It's a good idea to only change the height dimension of the widget's content size.

ANIMATE HEIGHT ADJUSTMENTS

Animate your content to coincide with the resize animation by implementing `viewWillTransitionToSize:withTransitionCoordinator:`. Use `animateAlongsideTransition:completion:` to add your animations to the coordinator parameter.

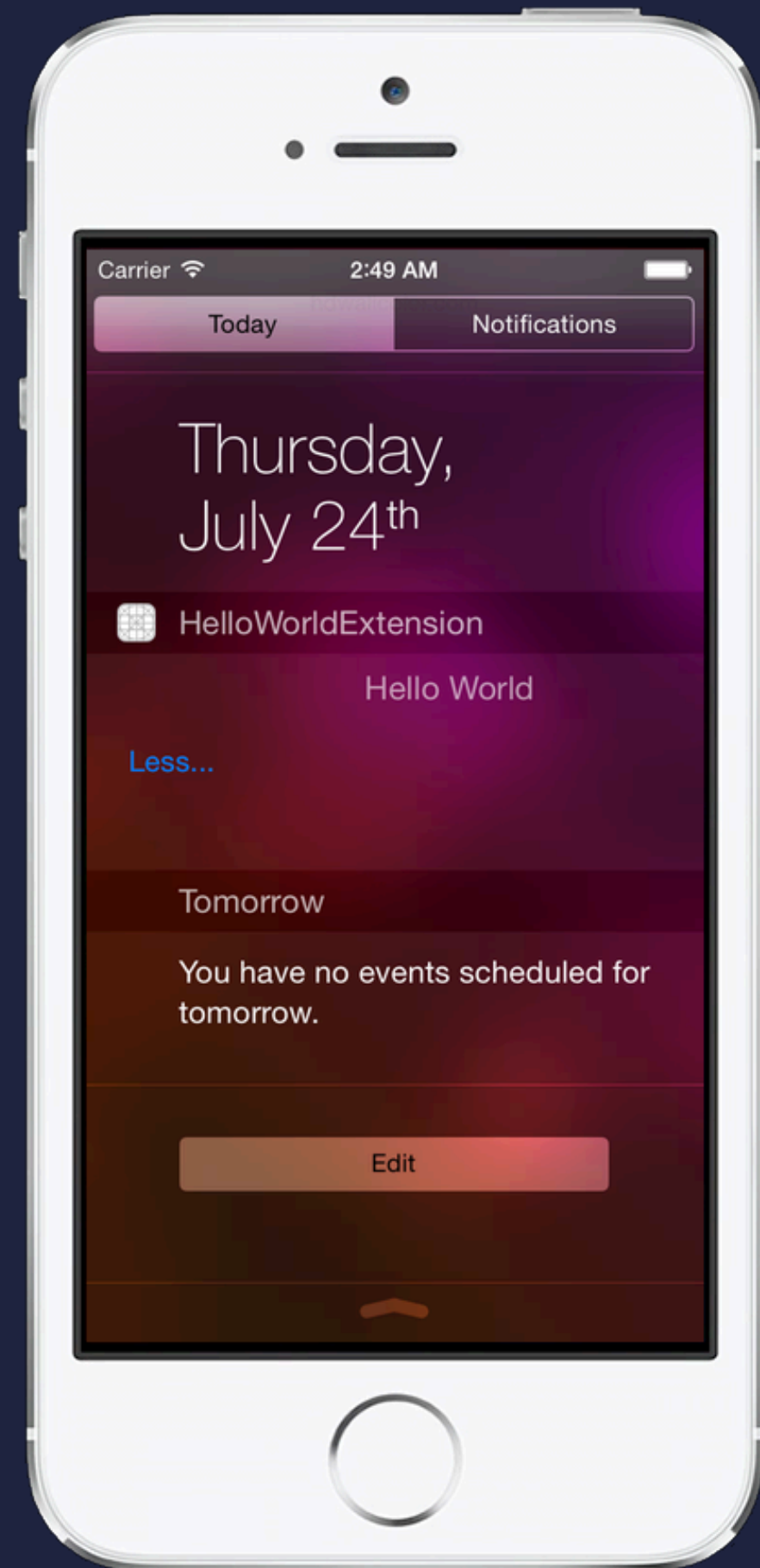
ANIMATE HEIGHT ADJUSTMENTS

```
- (void)viewWillTransitionToSize:(CGSize)size
    withTransitionCoordinator:
        (id<UIViewControllerTransitionCoordinator>)coordinator
{
    UIView *myView = ... // The view to animate
    CGFloat *newAlpha = ... // The new alpha value for the view

    [coordinator animateAlongsideTransition:
        ^(id<UIViewControllerTransitionCoordinatorContext> context) {
            myView.alpha = newAlpha;
        } completion:nil];
}
```

MARGINS AND HEIGHT

Demo



CONTROLLING VISIBILITY

Using `NCWidgetController`'s `method` `setHasContent:forWidgetWithBundleIdentifier:` you can indicate whether the widget has content to display or should be hidden.


Both a widget and its containing app can specify whether the widget should appear in Notification Center.

HIDING THE WIDGET

```
#import <NotificationCenter/NotificationCenter.h>
```

...

```
[[NCWidgetController widgetController] setHasContent:NO  
forWidgetWithBundleIdentifier:@"com.example.MyWidgetBundleIdentifier"];
```



If a widget indicates it doesn't have any content to display Notification Center won't launch it again until the containing app specifies that the widget should be displayed.

CONTROLLING VISIBILITY

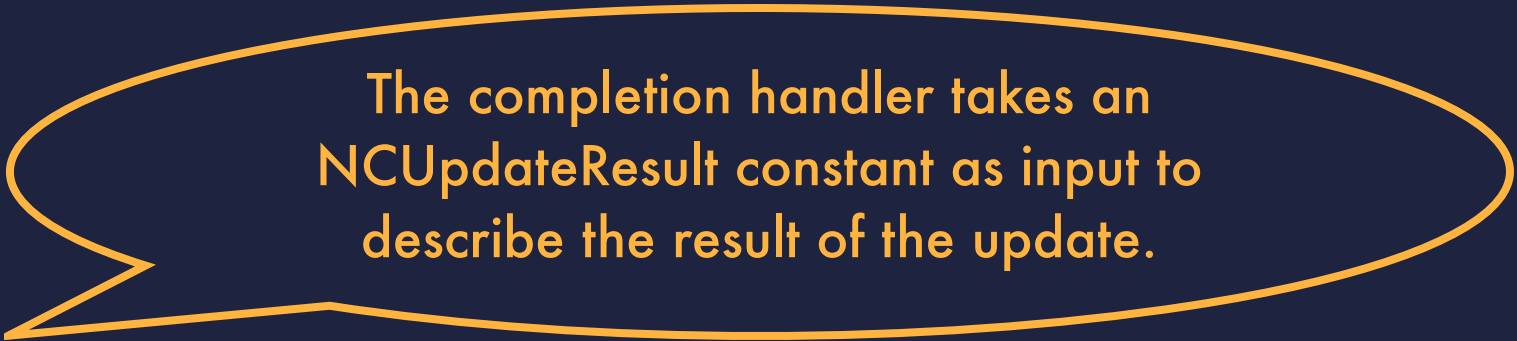
Demo

UPDATING CONTENTS

To help your widget look up to date, the system occasionally captures snapshots of your widget's view. When the widget becomes visible again, the most recent snapshot is displayed until replaced with a live version of the view.

UPDATING CONTENTS

To update a widget's state before a snapshot is taken, conform to the `NCWidgetProviding` protocol. When your widget receives the `widgetPerformUpdateWithCompletionHandler:` call, update its view with the most recent content and then call the completion handler.



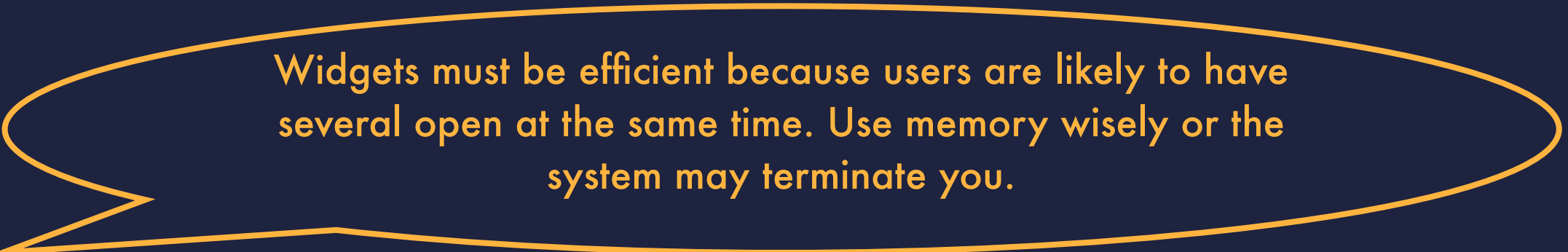
The completion handler takes an `NCUpdateResult` constant as input to describe the result of the update.

UPDATING CONTENTS

```
- (void)widgetPerformUpdateWithCompletionHandler:  
    (void (^)(NCUpdateResult))completionHandler  
{  
    // Perform any setup necessary in order to update the view  
    NSString *updatedContent = ...  
    self.label.text = updatedContent;  
  
    // If an error is encountered, use NCUpdateResultFailed  
    // If there's no update required, use NCUpdateResultNoData  
    // If there's an update, use NCUpdateResultNewData  
    completionHandler(NCUpdateResultNewData);  
}
```

PERFORMANCE CONSIDERATIONS

- ▶ Memory limits
- ▶ Cache data
- ▶ Don't block the main run loop
- ▶ Perform expensive operations in background



Widgets must be efficient because users are likely to have several open at the same time. Use memory wisely or the system may terminate you.

SHARING DATA WITH YOUR CONTAINING APP

Apps supporting URL schemes can be launched and passed data using the `openURL:completionHandler:` method available on the `NSExtensionContext` object.

Apps and Widgets that are part of the same App Group can share data via the shared container.

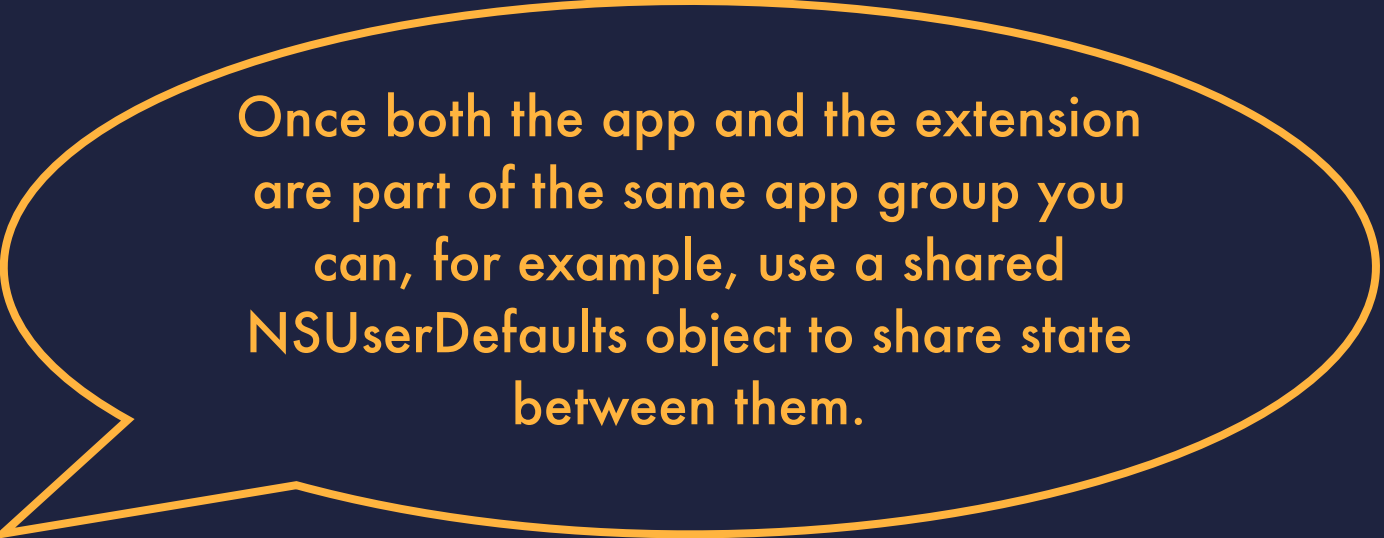
URL SCHEMES

```
NSExtensionContext *context = self.extensionContext;
NSURL *url = [NSURL URLWithString:@"myUr1Scheme://"];

[context openURL:url completionHandler:^(BOOL success) {
    if (success) {
        // Opened app
    } else {
        // Failed to open app
    }
}];
```

APP GROUPS

By default, your containing app and its extensions have no access to each other's containers. If you want them to be able to share data enable app groups for the app and its extensions.



Once both the app and the extension are part of the same app group you can, for example, use a shared `NSUserDefaults` object to share state between them.

Register – App Groups – Apple Developer

Apple Inc.

developer.apple.com/account/ios/identifiers/applicationGroup/applicationG

Reader

Certificates, Identifiers & Profiles

Kim Ahlberg

iOS Apps

Certificates

All

Pending

Development

Production

Identifiers

App IDs

Pass Type IDs

Website Push IDs

iCloud Containers

App Groups

Devices

All

Provisioning Profiles

All

Development

Distribution

Register App Groups

ID

Registering an App Group

Registering your App Group allows access to group containers that are shared among multiple related apps, and allows certain additional interprocess communication between the apps.

App Groups Description

Description:

MyAppGroup

You cannot use special characters such as @, &, *, ', "

Identifier

Enter a unique identifier for your App Group, starting with the string 'group'.

ID:

group.com.example.MyAppGroup

We recommend using a reverse-domain name style string (i.e., com.domainname.appname).

APP GROUPS

```
// Create and share access to an NSUserDefaults object.  
NSUserDefaults *mySharedDefaults = [[NSUserDefaults alloc]  
    initWithSuiteName:@"group.com.example.MyAppGroup"];  
  
// Store an object in the shared user defaults object.  
[mySharedDefaults setObject:myObject forKey:@"myObjectKey"];  
  
// Retrieve an object from the shared user defaults object.  
myObject = [mySharedDefaults objectForKey:@"MyStringKey"];
```

URL SCHEMES & APP GROUPS DEMO

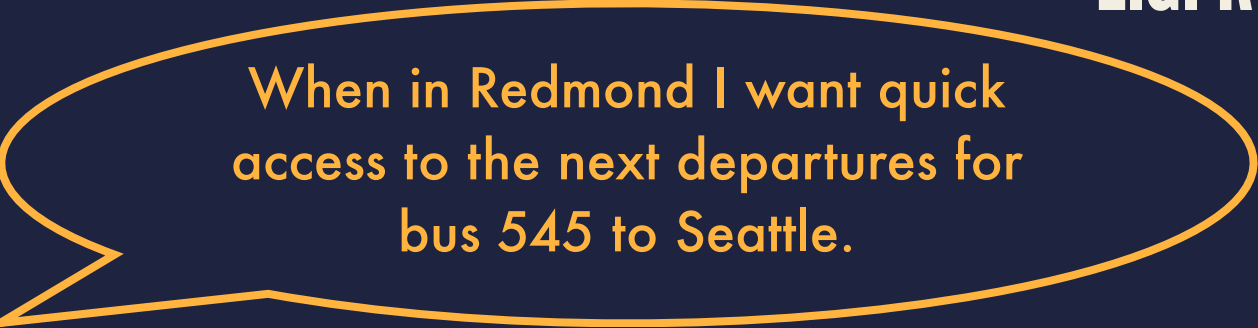
SHARING CODE

IN IOS 8.0 YOU CAN USE AN **EMBEDDED FRAMEWORK** TO SHARE CODE BETWEEN YOUR EXTENSION AND ITS CONTAINING APP. PUT THE CODE INTO A FRAMEWORK AND **EMBED IT IN BOTH TARGETS.**

**WHAT ARE YOUR COOL IDEAS
FOR WIDGETS?**

USE GEOFENCES TO PROVIDE LOCATION BASED INFO IN NOTIFICATION CENTER.

E.G. RELEVANT BUS STOP DEPARTURES.



When in Redmond I want quick
access to the next departures for
bus 545 to Seattle.

CONTACT

twitter.com/kimahlberg

alpha.app.net/kimahlberg

github.com/kimahlberg

linkedin.com/in/kimahlberg

www.TheEvilBoss.com

FOR FURTHER STUDIES

WWDC 2014 video — CREATING EXTENSIONS FOR IOS AND OS X, PART 1

developer.apple.com/videos/wwdc/2014/

Pre-release documentation — EXTENSION ESSENTIALS

[developer.apple.com/library/prerelease/ios/documentation/General/
Conceptual/ExtensibilityPG](https://developer.apple.com/library/prerelease/ios/documentation/General/Conceptual/ExtensibilityPG)