

# Creating Extensions for iOS and OS X, Part Two

Architecture

Session 217

Damien Sorresso

Overloaded Operator

# What Are Apps?

Most important experience

Own entire screen

Installed/removed by user



# What Are Apps?

Most important experience

Own entire screen

Installed/removed by user

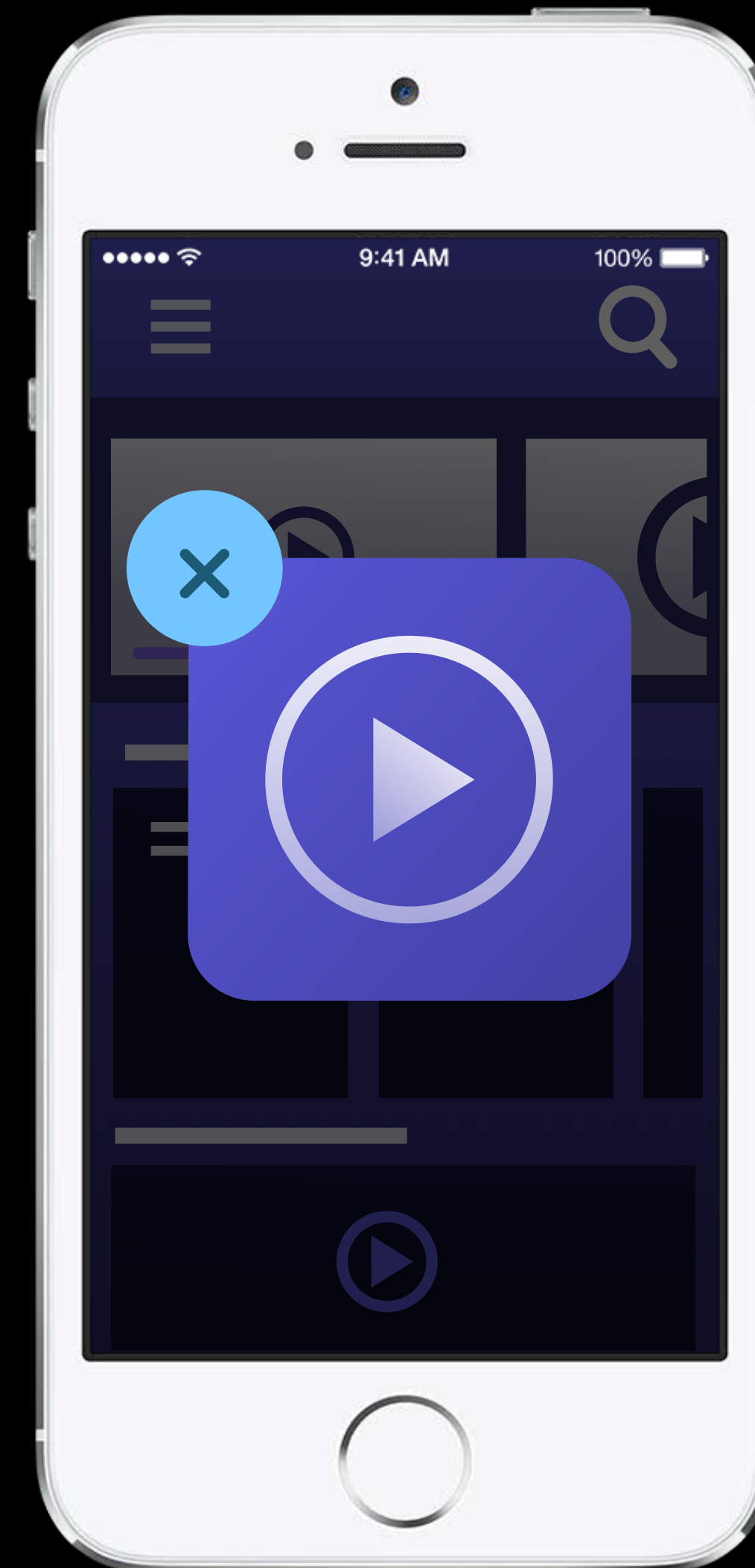


# What Are Apps?

Most important experience

Own entire screen

Installed/removed by user



# Extensions

Important to the user

Not more important than the app

Augment current experience

Come and go with apps



# Extensions

Important to the user

Not more important than the app

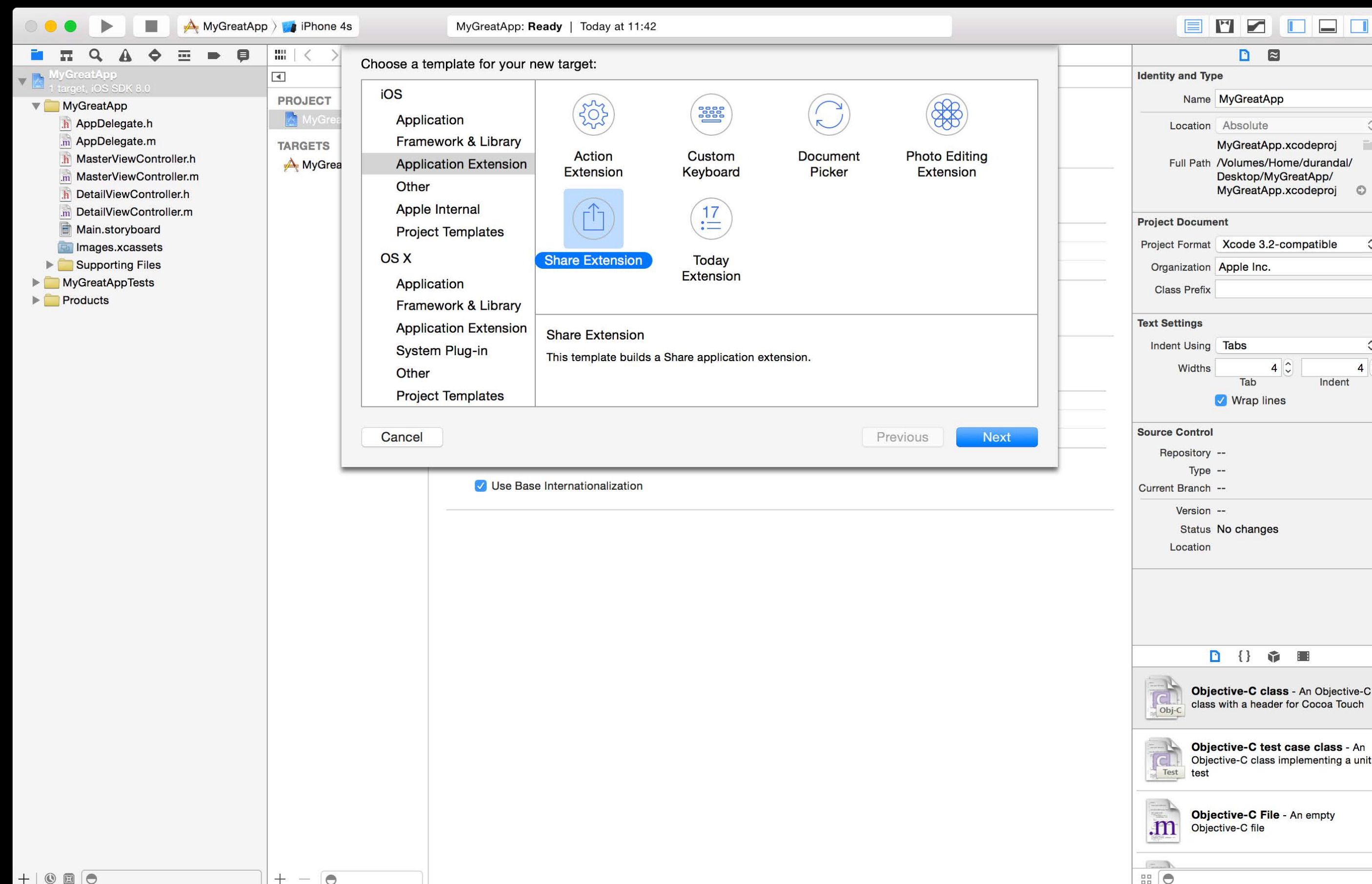
Augment current experience

Come and go with apps



Built Separately

# Built Separately





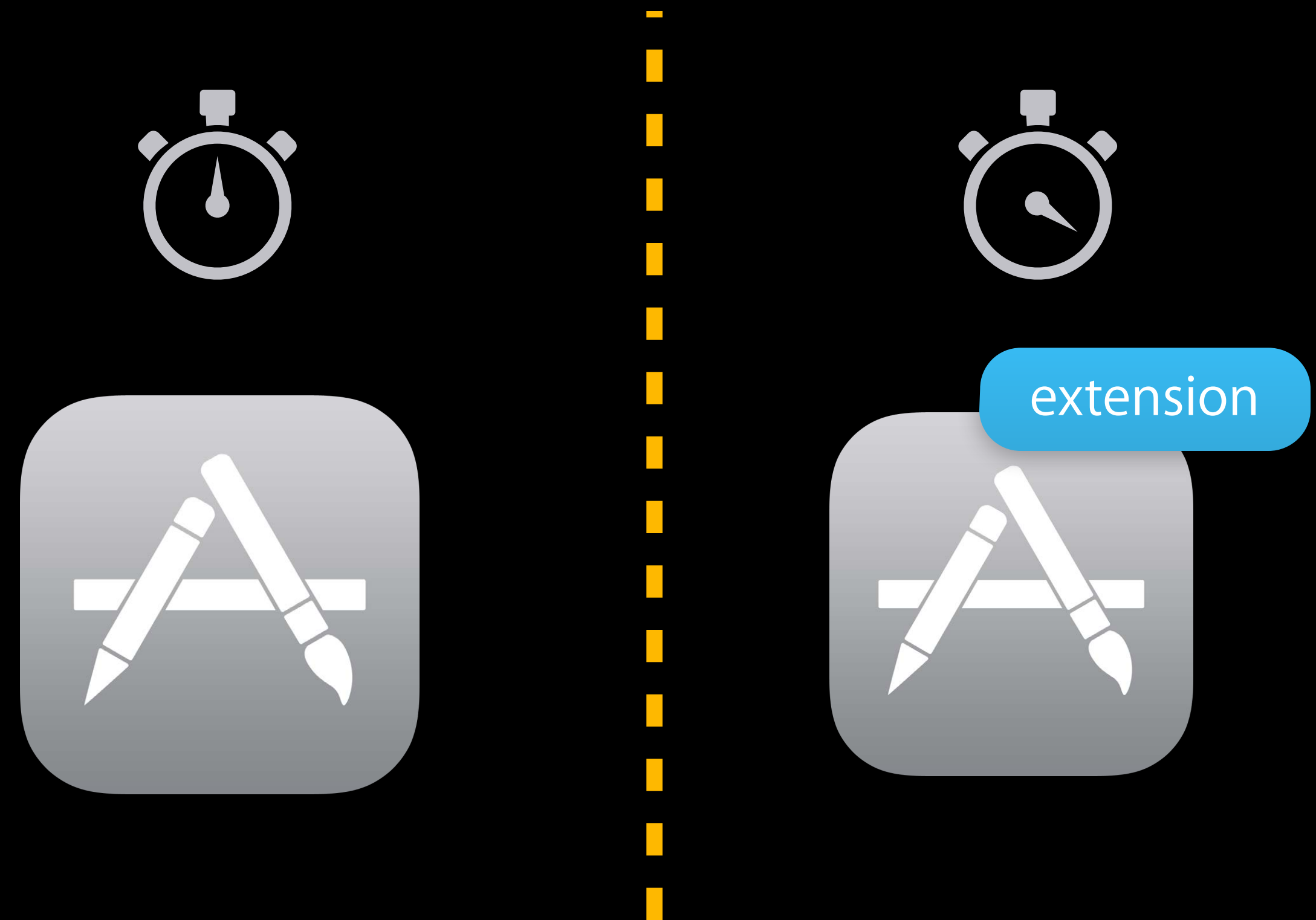
# Built Separately



# Run Separately

## Different process

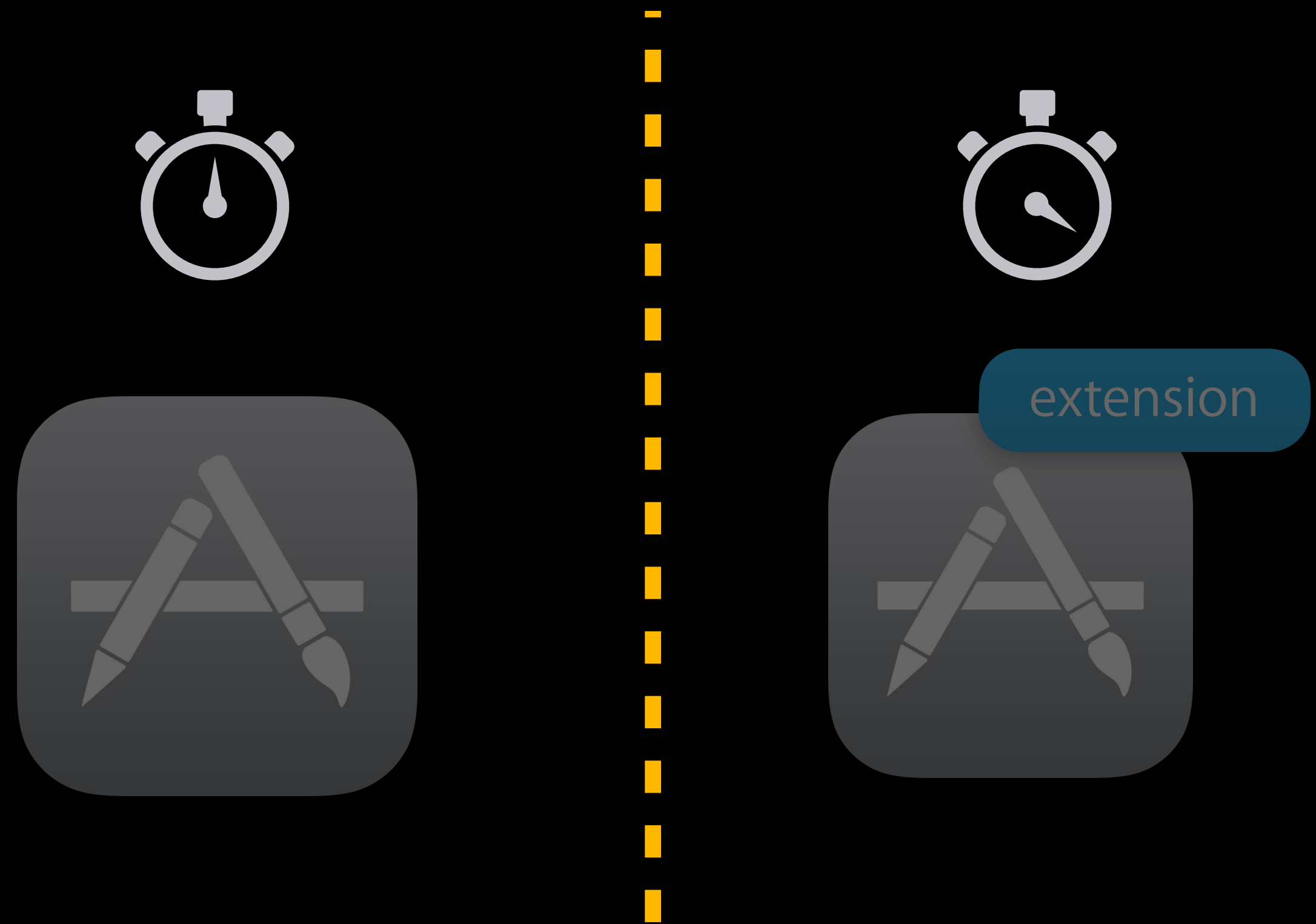
- Isolated address space
- Executes independently
- System optimizes separately



# Run Separately

## Different process

- Isolated address space
- Executes independently
- System optimizes separately



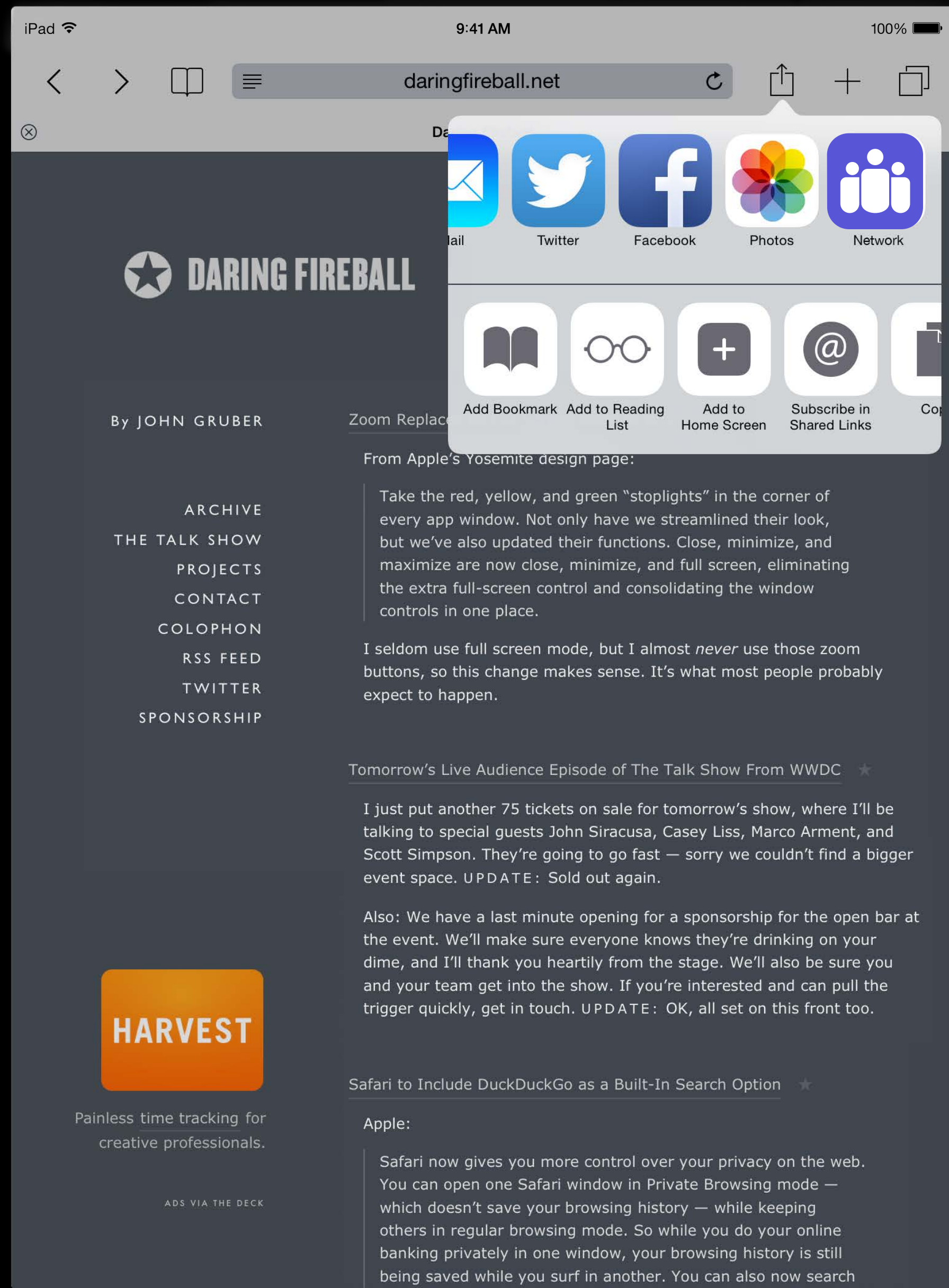
# Run Separately

## Different process

- Isolated address space
- Executes independently
- System optimizes separately








iPad 9:41 AM 100%

daringfireball.net

Daring Fireball

 **DARING FIREBALL**

Cancel Network Post


Message:

Look at this great website I found!

Location: San Francisco, CA

...I be  
talking to special guests John Siracusa, Casey Liss, Marco Arment, and  
Scott Simpson. They're going to go fast — sorry we couldn't find a bigger  
event space. UPDATE: Sold out again.

Also: We have a last minute opening for a sponsorship for the open bar at  
the event. We'll make sure everyone knows they're drinking on your  
dime, and I'll thank you heartily from the stage. We'll also be sure you  
and your team get into the show. If you're interested and can pull the  
trigger quickly, get in touch. UPDATE: OK, all set on this front too.

 **emma**  
FOR  
**AGENCIES**

Emma For Agencies: The  
email marketing platform  
built for agencies.

ADS VIA THE DECK

Safari to Include DuckDuckGo as a Built-In Search Option

Apple:

Safari now gives you more control over your privacy on the web.  
You can open one Safari window in Private Browsing mode —  
which doesn't save your browsing history — while keeping  
others in regular browsing mode. So while you do your online  
banking privately in one window, your browsing history is still  
being saved while you surf in another. You can also now search

Extending the Experience



# A Piece of Your App

Focused experience

Seamlessly merged with current app

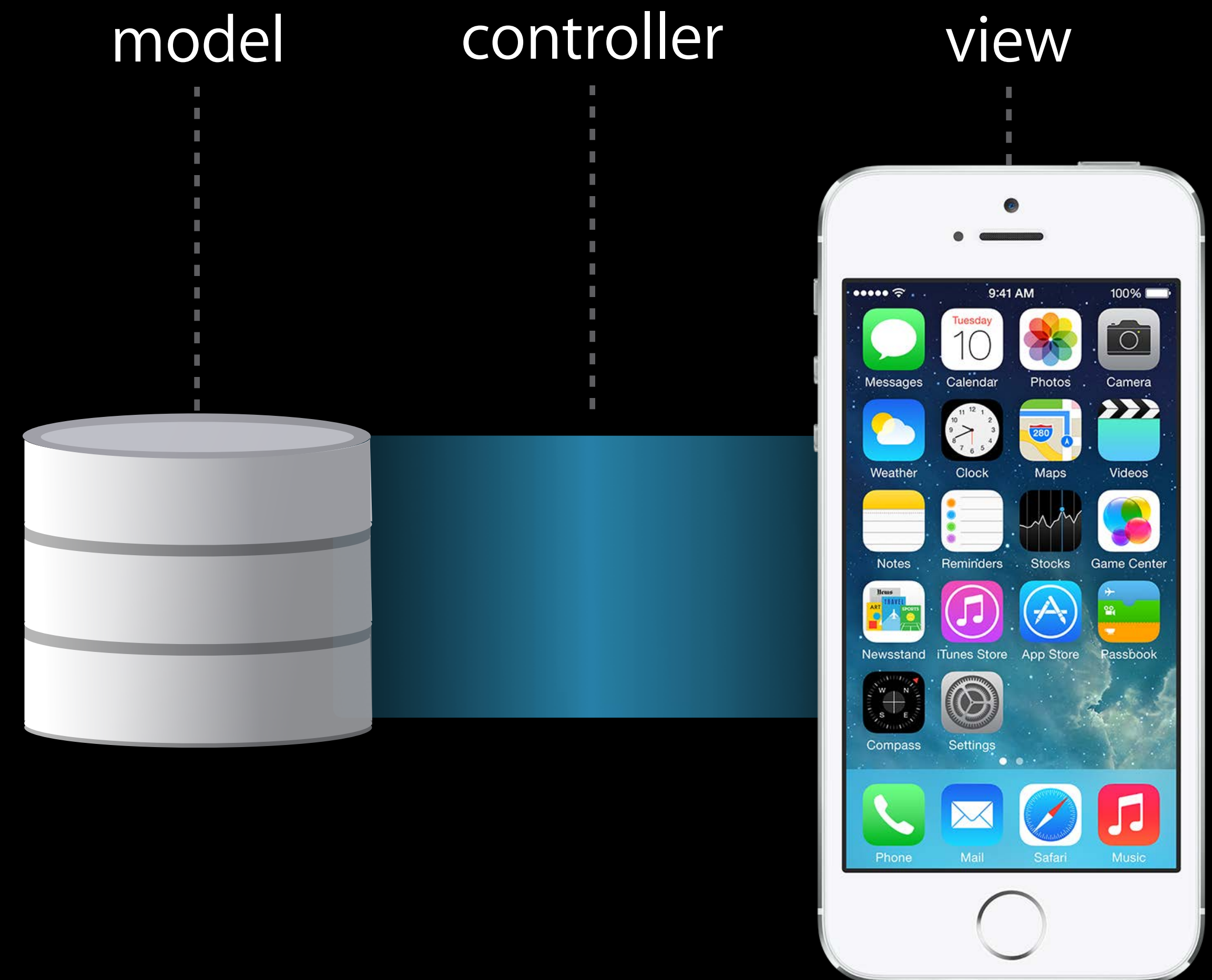
Reuse code from your app

# Sharing Code

Hope you used MVC!

Same data model

Same view controllers

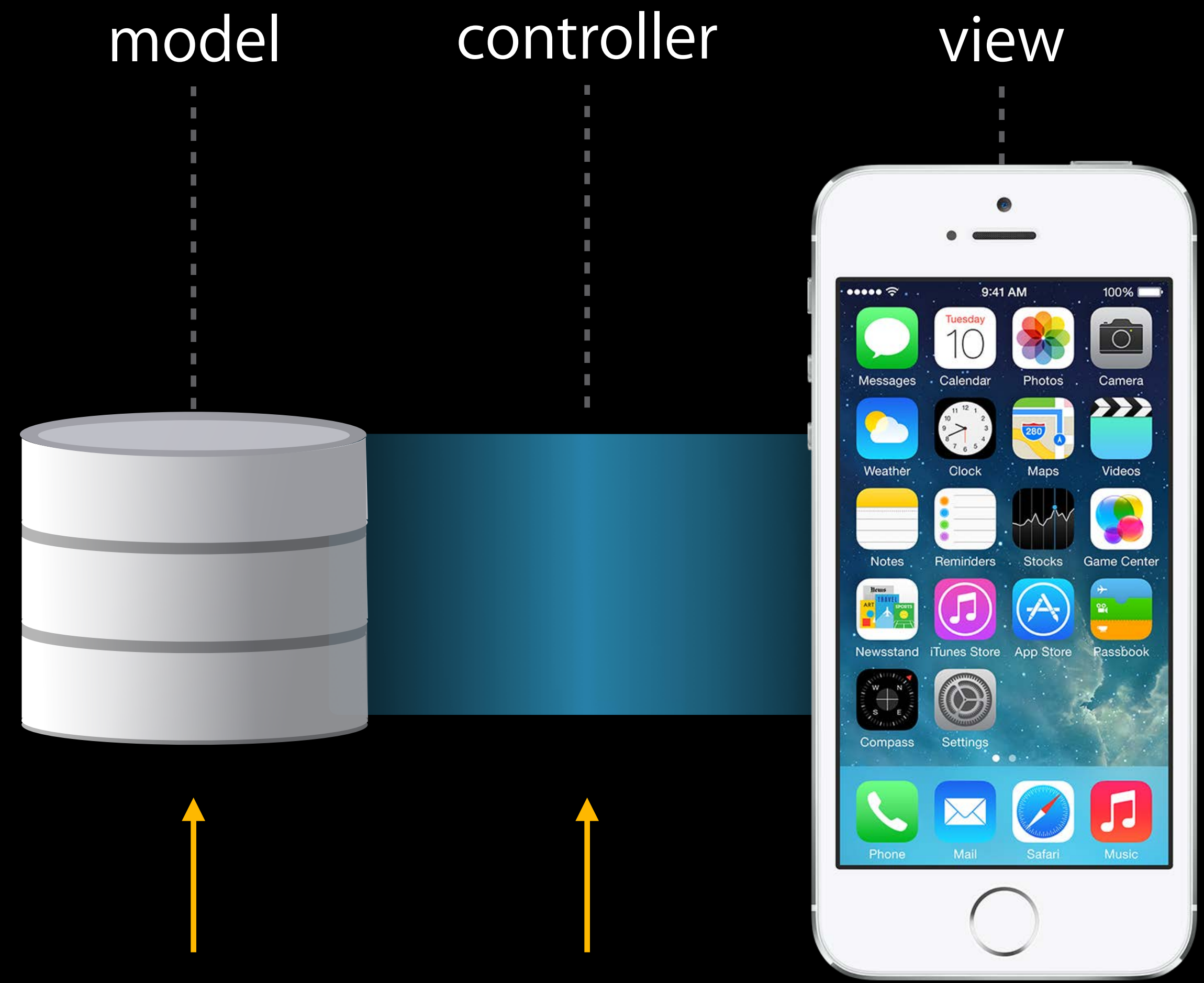


# Sharing Code

Hope you used MVC!

Same data model

Same view controllers



What's the best way  
to share code?



Frameworks!

# Embedded Frameworks



Can ship frameworks in iOS apps

App and extensions can link

Encrypted

Already possible in OS X

# Minimum Deployment Target



Minimum deployment target may change

App linkage requires iOS 8

Not affected by extension linkage

# Minimum Deployment Target



Minimum deployment target may change

App linkage requires iOS 8

Not affected by extension linkage





# Related Sessions

- 
- Building Modern Frameworks Presidio Thursday 3:15PM
-

# API Availability

Vast majority of API is available

Some exceptions

Marked specifically with "unavailability" macro

# Unavailability

```
+ (UIApplication *)sharedApplication
    NS_EXTENSION_UNAVAILABLE_IOS("Use view controller based solutions where
appropriate instead.");
```

# Unavailability

```
__OS_EXTENSION_UNAVAILABLE("Not available for extensions")  
extern void  
super_1337_api_with_underscores(const char *astring);
```

# App and Extension Consistency

# Consistency

# Consistency

Common resources

Reflect changes

Share data

# Sharing Data

Separate containers

No access to app data

Opt into data sharing



# Shared Container

App Group

Shared storage area

General data sharing

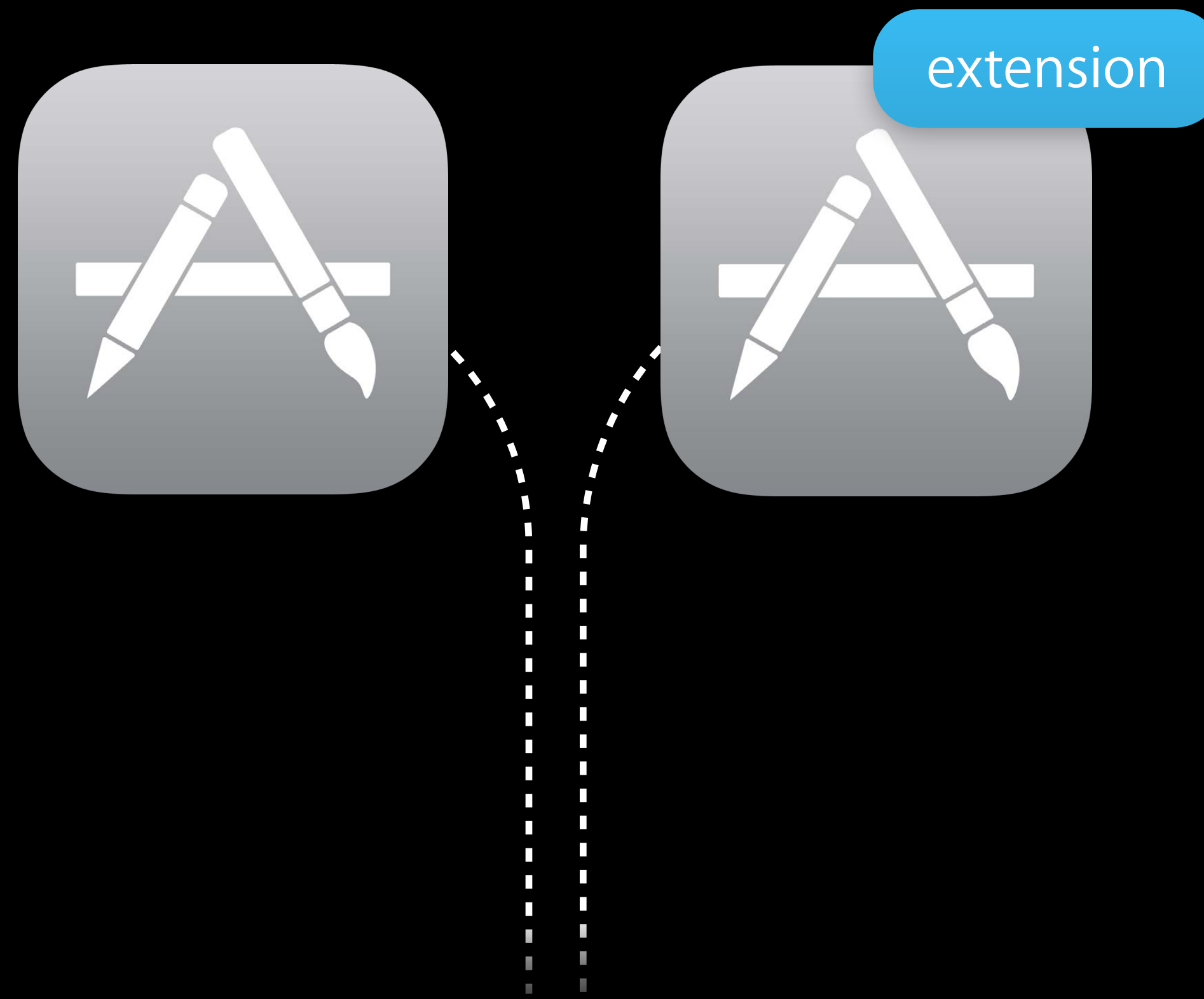
Coordinate file access



# Safely Sharing



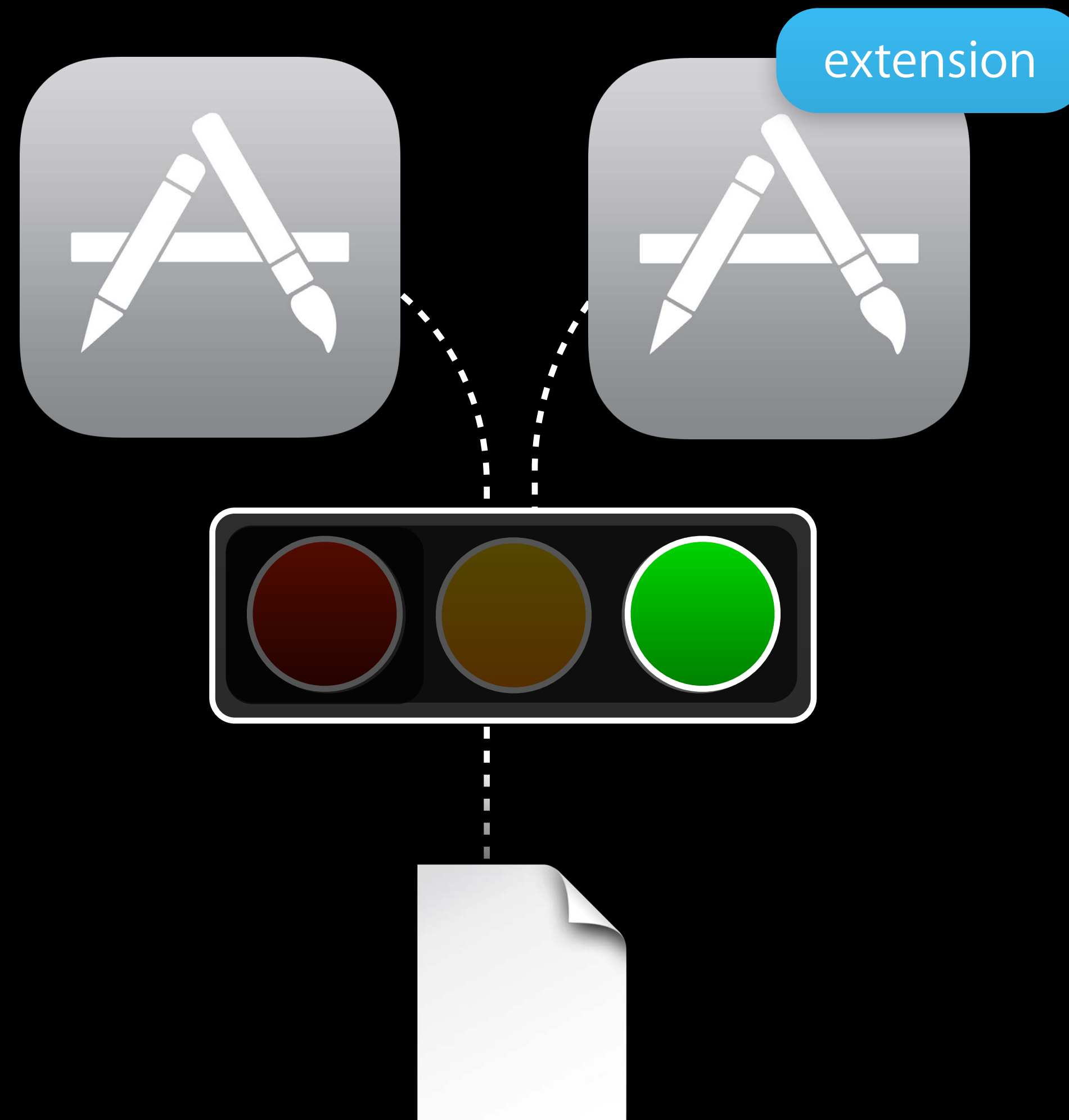
# Safely Sharing



# Safely Sharing



# Safely Sharing



# Synchronization Technologies

NSFileCoordination

CoreData

sqlite

# NSUserDefaults

Distinct user defaults from app

Can set up shared domain

```
NSUserDefaults *sharedDefaults = [[NSUserDefaults alloc]
    initWithSuiteName:@"com.mycompany.MyGreatApp.shared"];
```

API coordinates access

# Shared Keychains

App Group

Distinct keychain from app

Can set up keychain access group

Like sharing keychains between apps



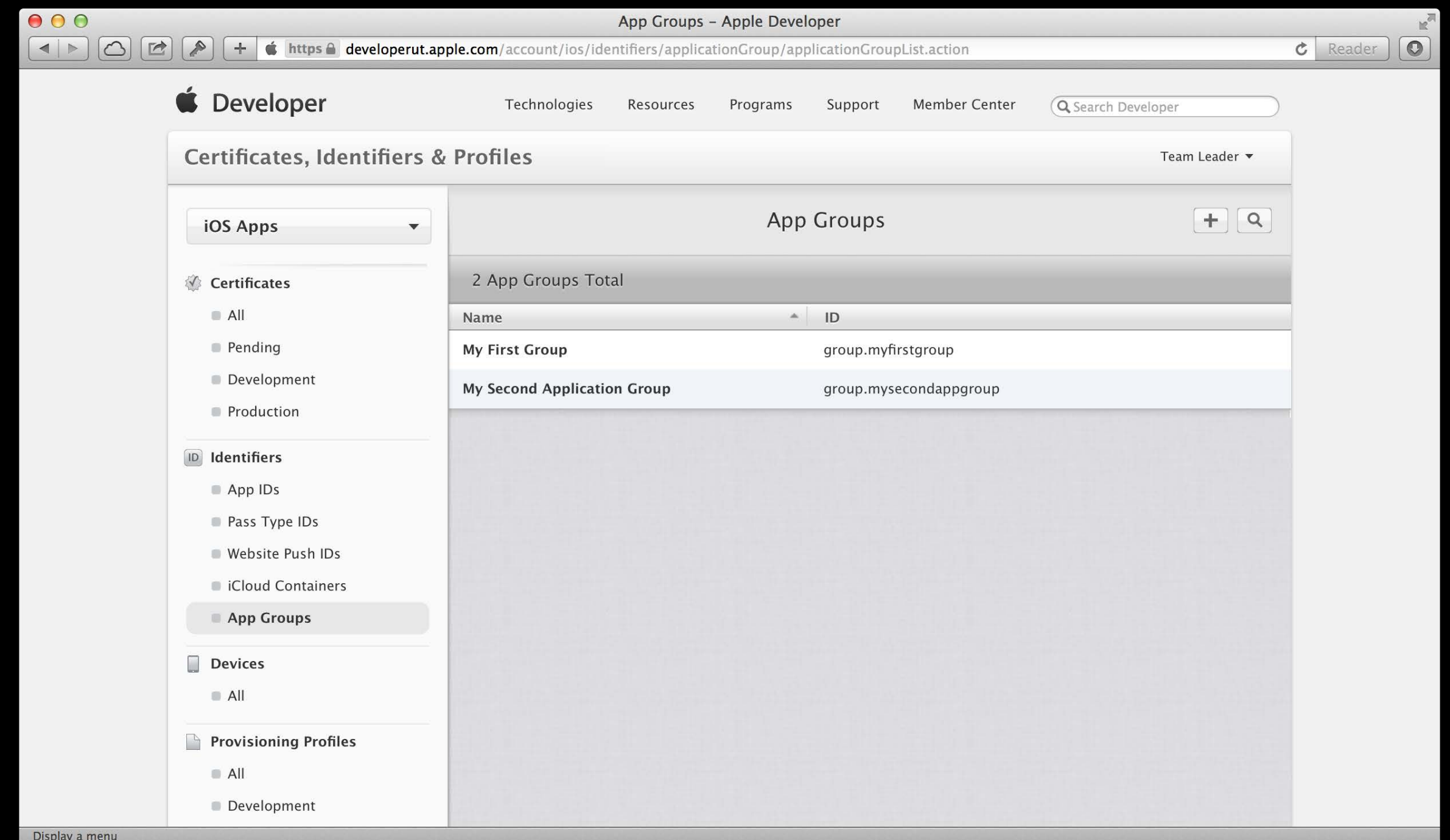


# Privacy

Approval encompasses app and extensions

Some exceptions

User approves entirety of app



# Best Practices

# Be Lean

Frontmost app is still most important

Don't have full use of system resources

Get in, get out

# Be Stateless

May be killed aggressively

No general multitasking

Support for short task-completion

Flush state to disk

# Be Awesome

Make it seamless

Make it useful

Surprise and delight!

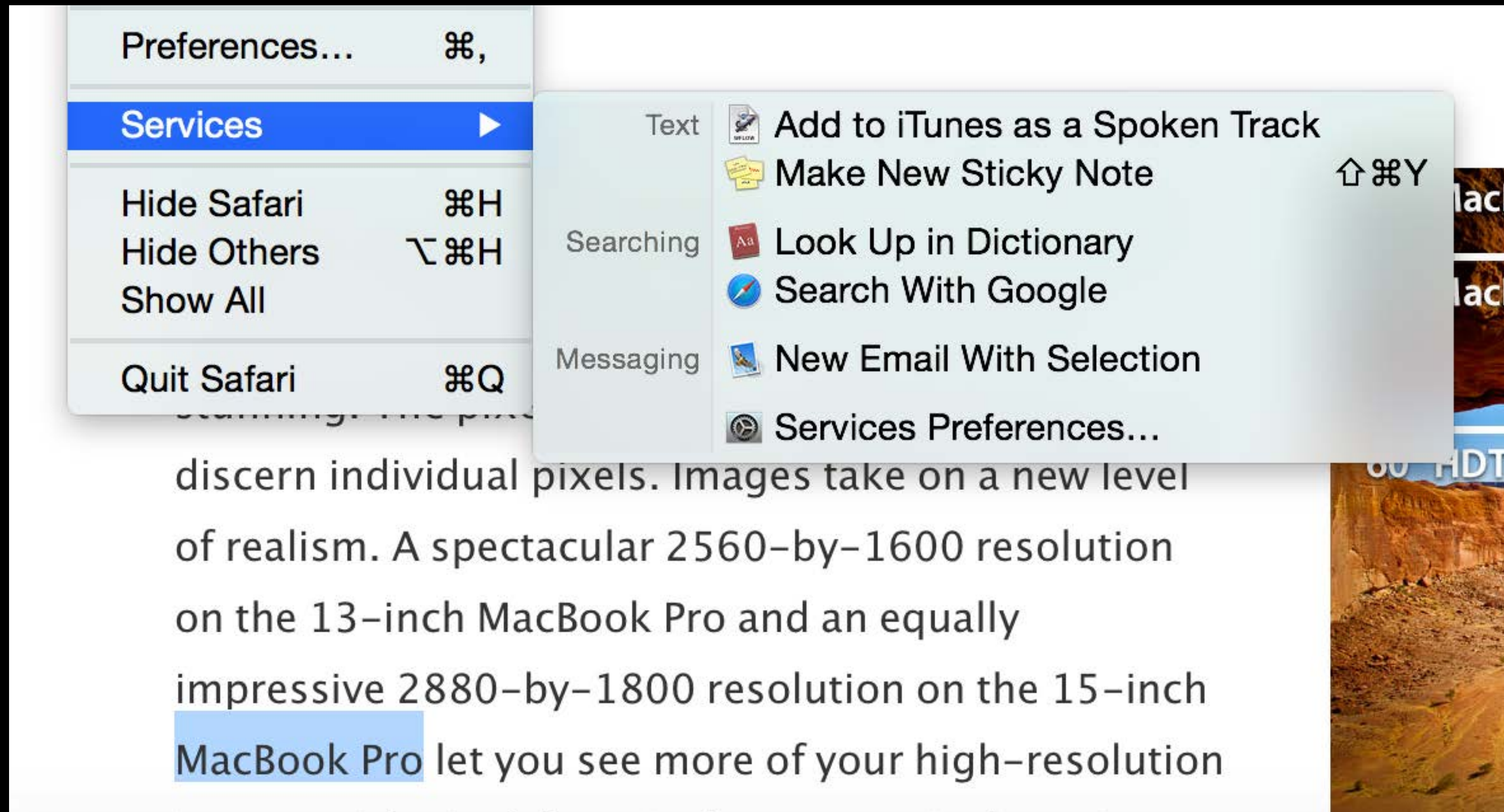
# Action Extensions

Aki Inoue

Cocoa Senior Engineer

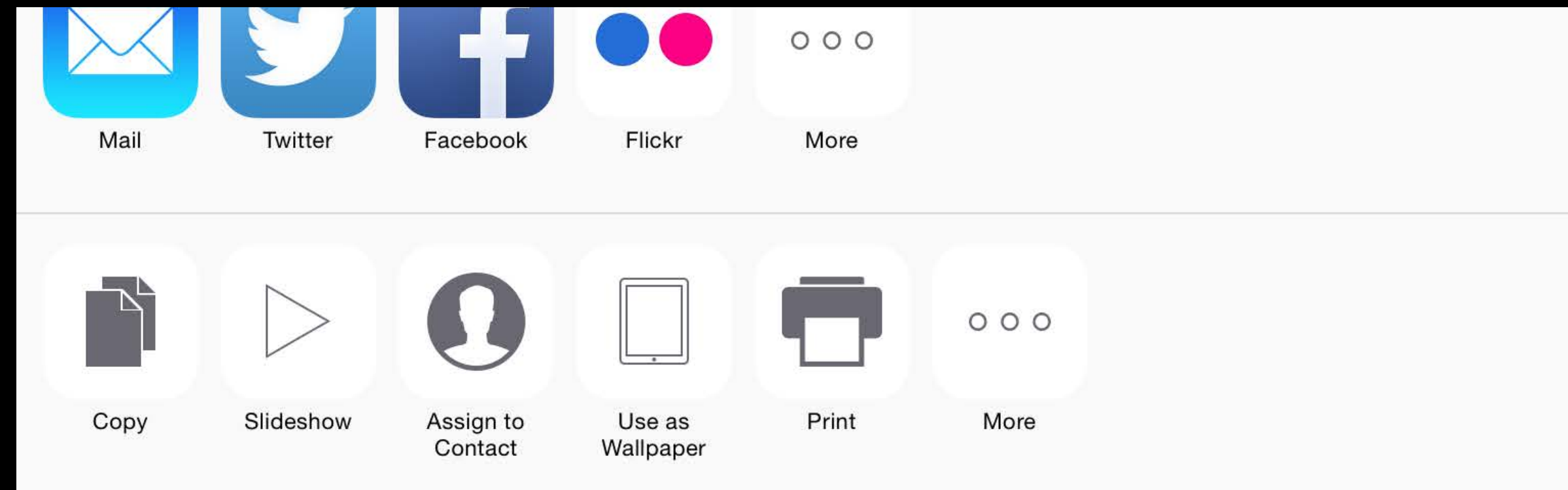
# Action Extensions

# Action Extensions

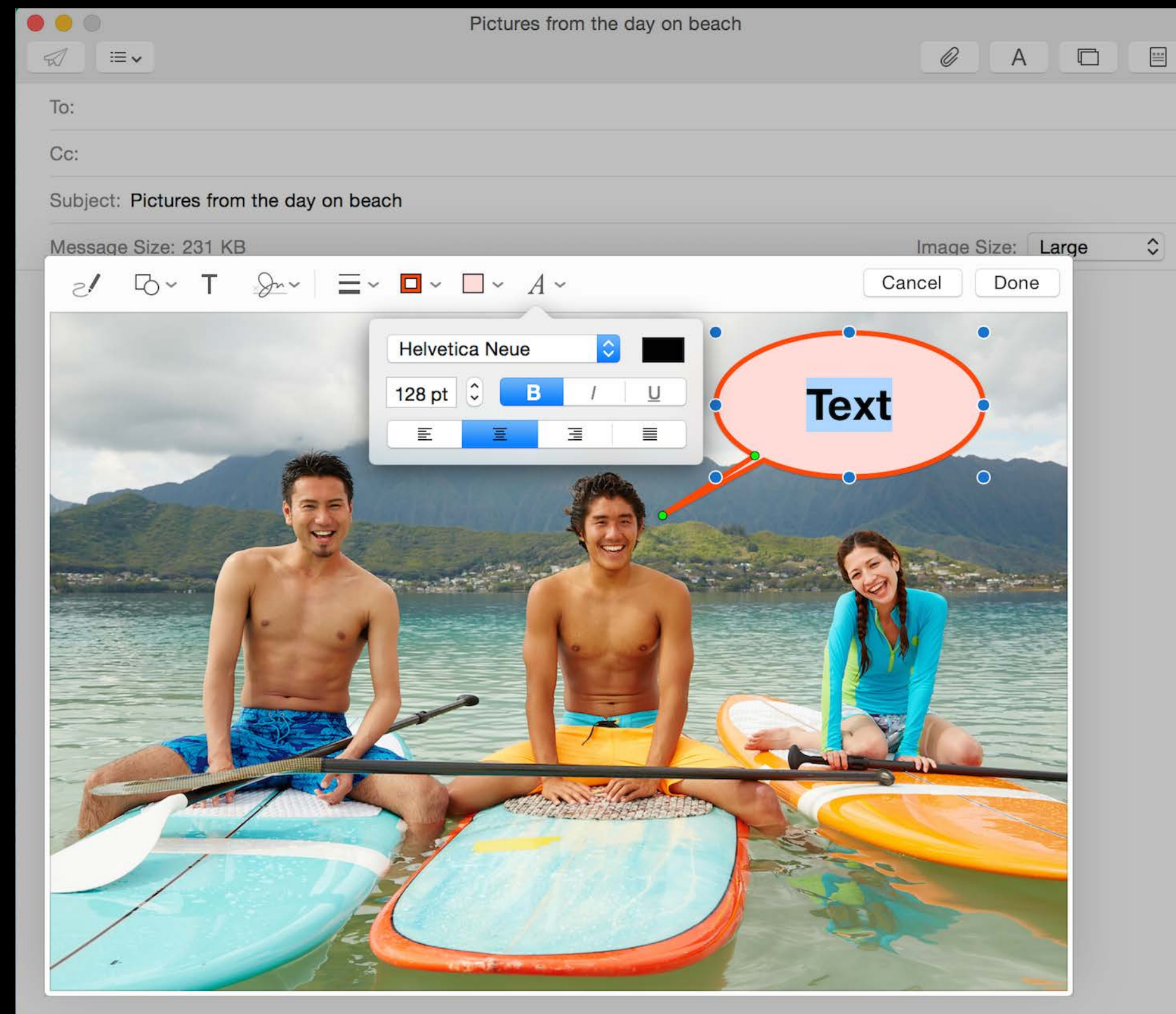




# Action Extensions



# Action Extensions



# Describing Action Extensions

▼ NSExtension	↕	Dictionary	(3 items)
NSExtensionPointIdentifier	+ -	String	com.apple.ui-services
NSExtensionPrincipalClass		String	ActionViewController
▼ NSExtensionAttributes		Dictionary	(2 items)
NSExtensionServiceRoleType		String	NSExtensionServiceRoleTypeEditor
▼ NSExtensionActivationRules		Dictionary	(2 items)
NSExtensionActivationSupportsText		Boolean	YES
NSExtensionActivationSupportsImageWithMaxCount		Number	1

# Describing Action Extensions

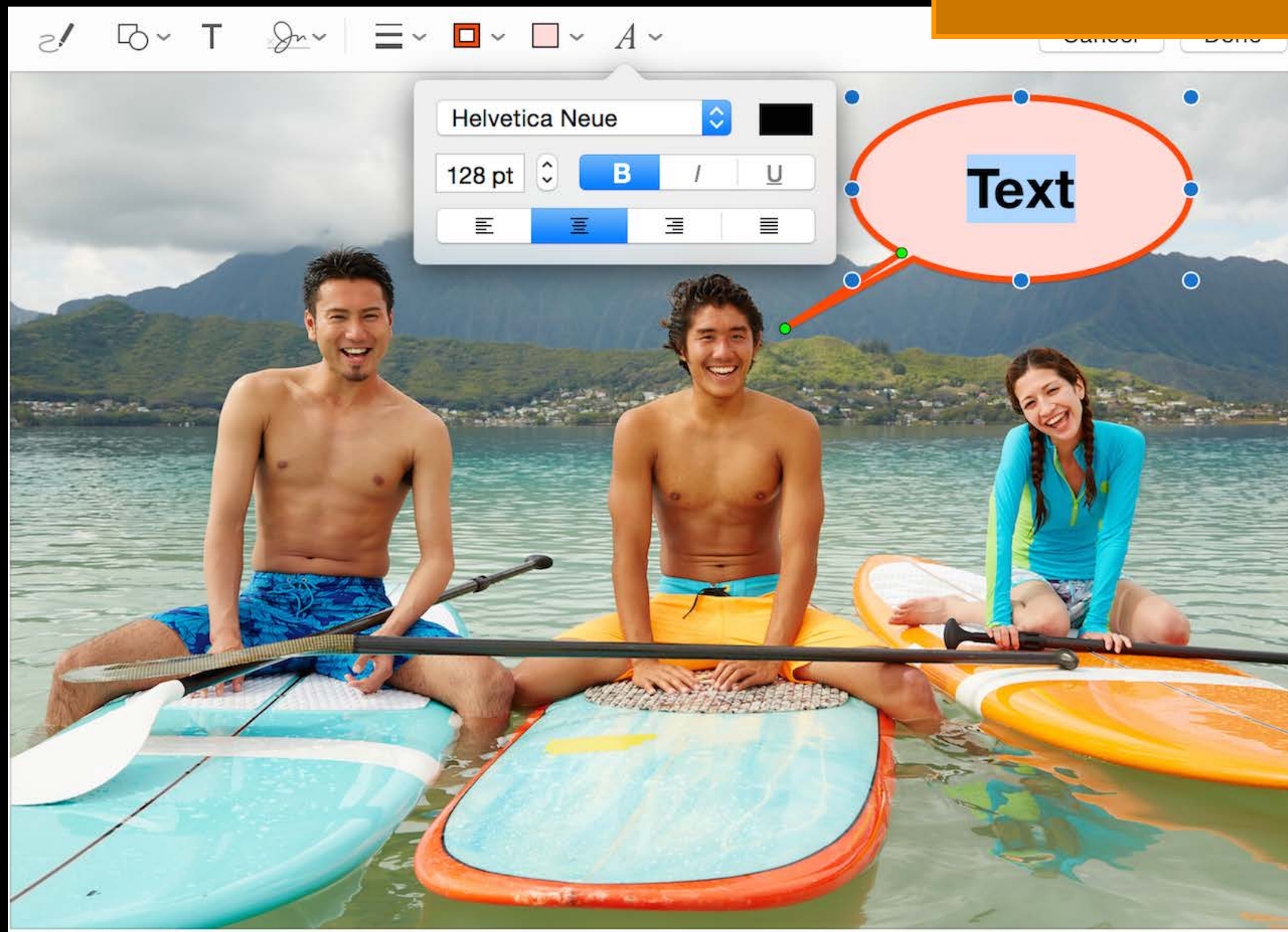
▼ NSExtension	Dictionary	(3 items)
NSExtensionPointIdentifier	String	com.apple.ui-services
NSExtensionPrincipalClass	String	ActionViewController
▼ NSExtensionAttributes	Dictionary	(2 items)
NSExtensionServiceRoleType	String	NSExtensionServiceRoleTypeEditor
▼ NSExtensionActivationRules	Dictionary	(2 items)
NSExtensionActivationSupportsText	Boolean	YES
NSExtensionActivationSupportsImageWithMaxCount	Number	1

# Describing Action Extensions

▼ NSExtension	↕	Dictionary	(3 items)
NSExtensionPointIdentifier	+ -	String	com.apple.ui-services
NSExtensionPrincipalClass		String	ActionViewController
▼ NSExtensionAttributes		Dictionary	(2 items)
NSExtensionServiceRoleType		String	NSExtensionServiceRoleTypeEditor
▼ NSExtensionActivationRules		Dictionary	(2 items)
NSExtensionActivationSupportsText		Boolean	YES
NSExtensionActivationSupportsImageWithMaxCount		Number	1

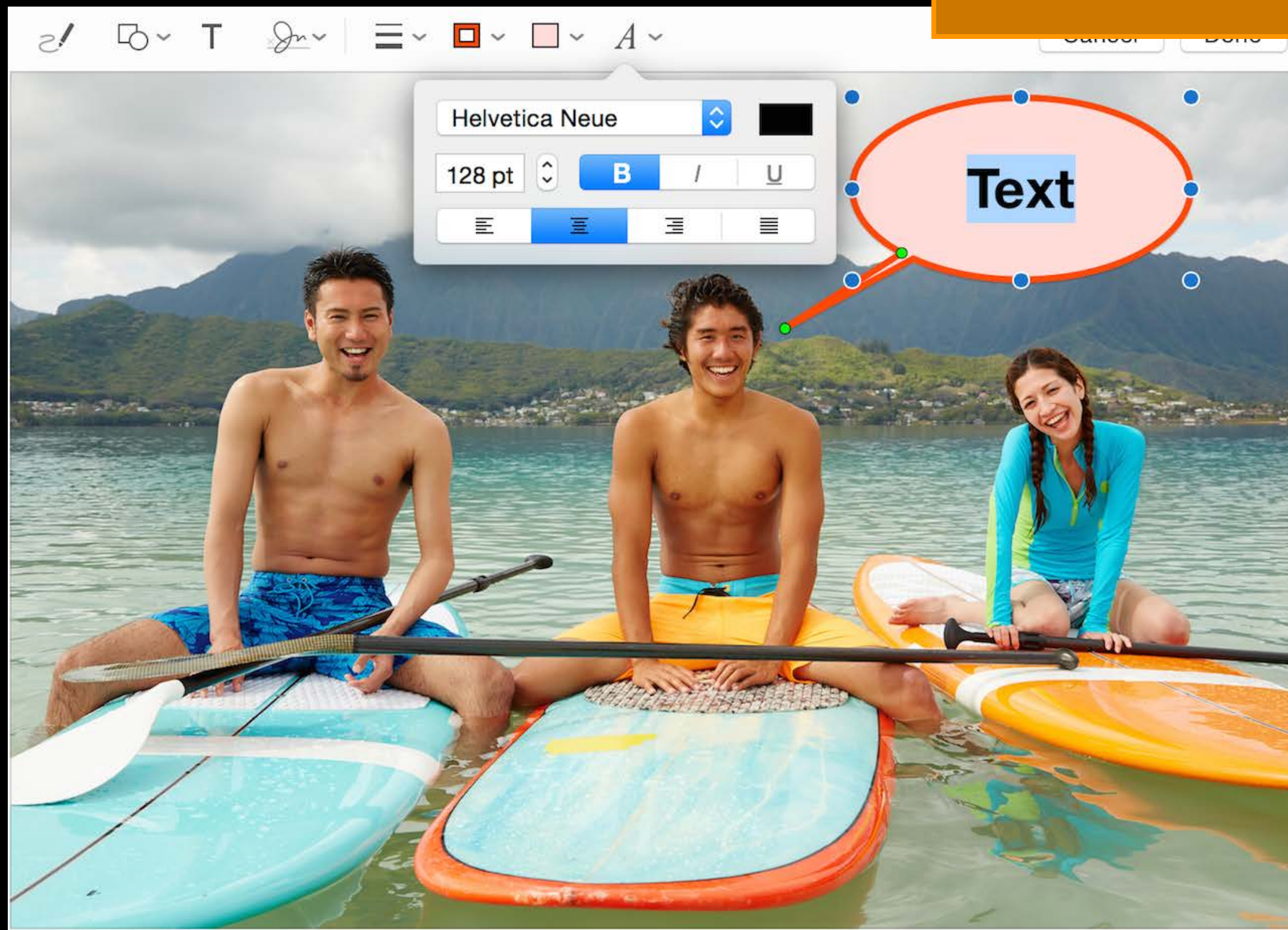
# Principal View Controller

View Controller



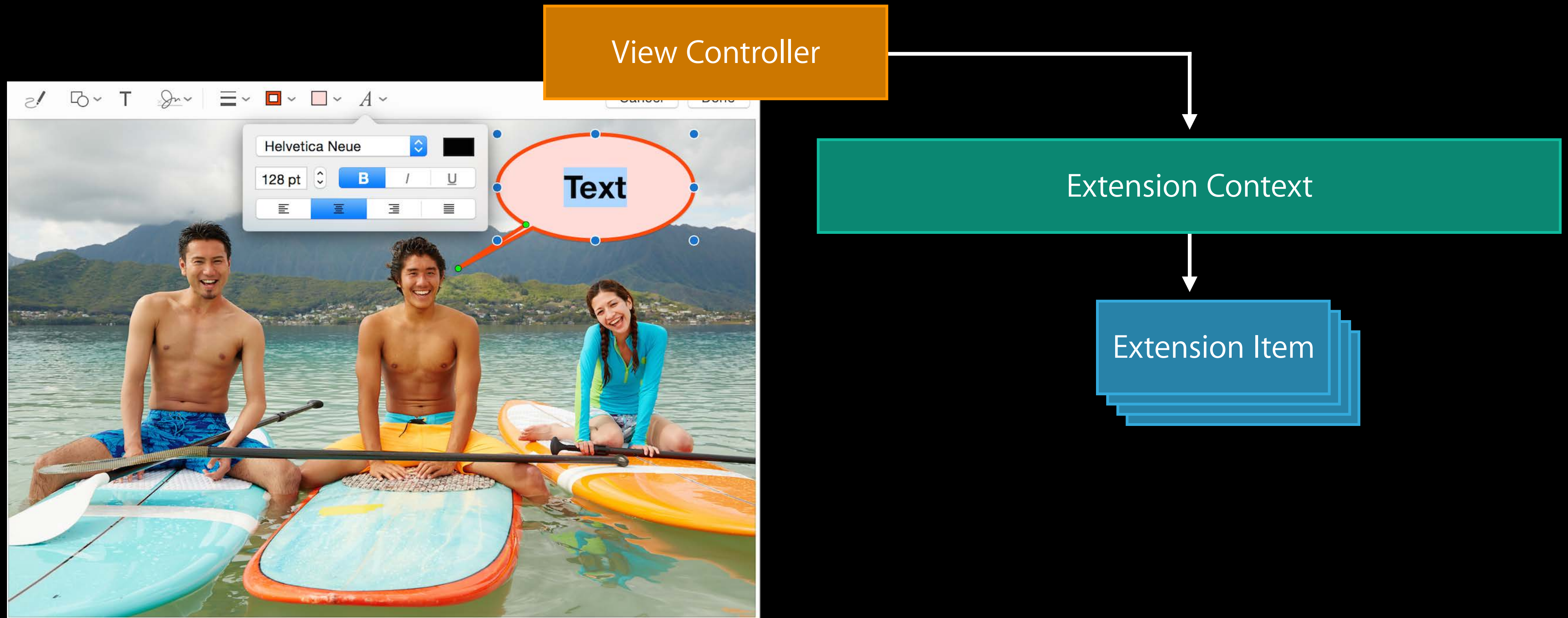
# Principal View Controller

View Controller



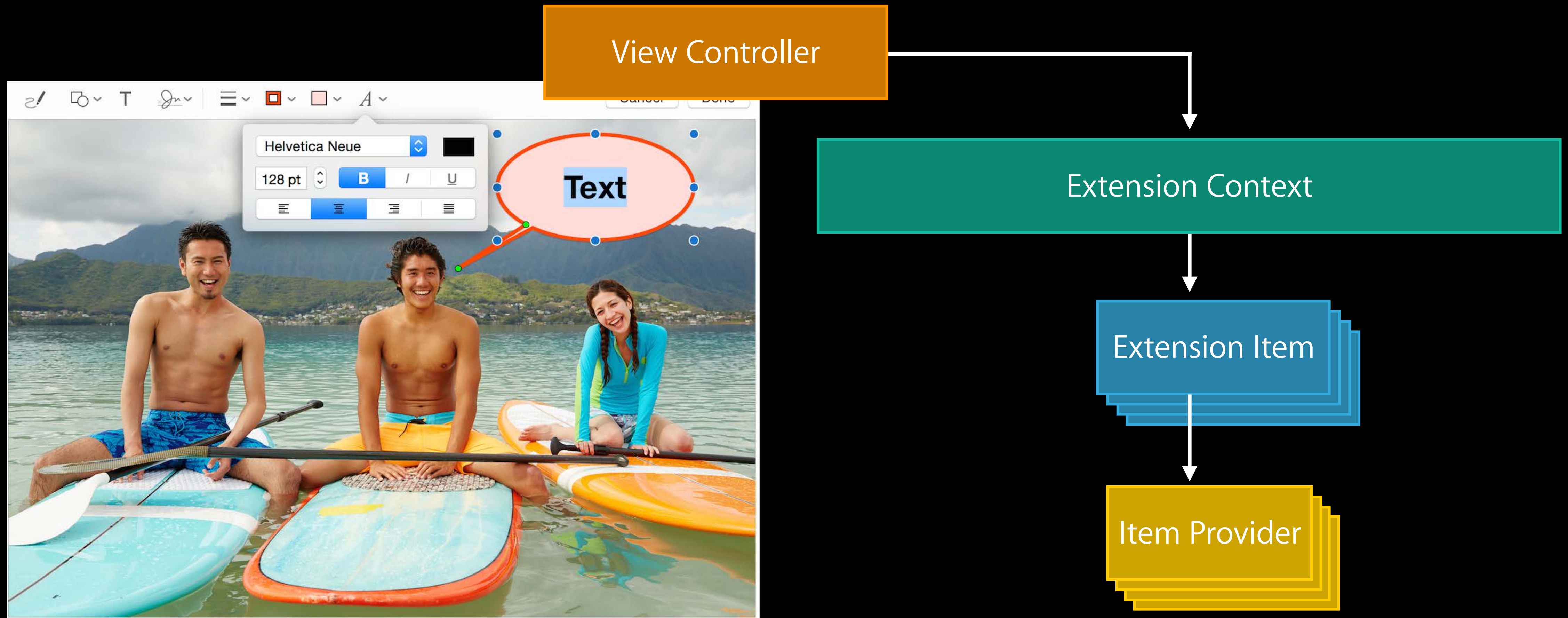
Extension Context

# Principal View Controller





# Principal View Controller



# Describing Action Extensions

▼ NSExtension	↕	Dictionary	(3 items)
NSExtensionPointIdentifier	+ -	String	com.apple.ui-services
NSExtensionPrincipalClass		String	ActionViewController
▼ NSExtensionAttributes		Dictionary	(2 items)
NSExtensionServiceRoleType		String	NSExtensionServiceRoleTypeEditor
▼ NSExtensionActivationRules		Dictionary	(2 items)
NSExtensionActivationSupportsText		Boolean	YES
NSExtensionActivationSupportsImageWithMaxCount		Number	1

# Describing Action Extensions

▼ NSExtension	↕	Dictionary	(3 items)
NSExtensionPointIdentifier	+ -	String	com.apple.ui-services
NSExtensionPrincipalClass		String	ActionViewController
▼ NSExtensionAttributes		Dictionary	(2 items)
NSExtensionServiceRoleType		String	NSExtensionServiceRoleTypeEditor
▼ NSExtensionActivationRules		Dictionary	(2 items)
NSExtensionActivationSupportsText		Boolean	YES
NSExtensionActivationSupportsImageWithMaxCount		Number	1

# UI Action Setup

```
- (void)viewDidLoad {  
    [super viewDidLoad];  
  
    // Get NSExtensionContext  
    NSExtensionContext *extensionContext = self.extensionContext;  
  
    // Get NSExtensionItem  
    NSExtensionItem *inputItem = extensionContext.inputItems.firstObject;  
  
    // Get NSItemProvider  
    NSItemProvider *itemProvider = inputItem.attachments.firstObject;
```

# UI Action Setup

```
- (void)viewDidLoad {  
    [super viewDidLoad];
```

```
    // Get NSExtensionContext
```

```
    NSExtensionContext *extensionContext = self.extensionContext;
```

```
    // Get NSExtensionItem
```

```
    NSExtensionItem *inputItem = extensionContext.inputItems.firstObject;
```

```
    // Get NSItemProvider
```

```
    NSItemProvider *itemProvider = inputItem.attachments.firstObject;
```

# UI Action Setup

```
- (void)viewDidLoad {  
    [super viewDidLoad];  
  
    // Get NSExtensionContext  
    NSExtensionContext *extensionContext = self.extensionContext;  
  
    // Get NSExtensionItem  
    NSExtensionItem *inputItem = extensionContext.inputItems.firstObject;  
  
    // Get NSItemProvider  
    NSItemProvider *itemProvider = inputItem.attachments.firstObject;
```

# UI Action Setup

```
- (void)viewDidLoad {  
    [super viewDidLoad];  
  
    // Get NSExtensionContext  
    NSExtensionContext *extensionContext = self.extensionContext;  
  
    // Get NSExtensionItem  
    NSExtensionItem *inputItem = extensionContext.inputItems.firstObject;  
  
    // Get NSItemProvider  
    NSItemProvider *itemProvider = inputItem.attachments.firstObject;
```

# UI Action Setup (continued)

```
// Request document data
[itemProvider loadItemForTypeIdentifier:MyDocumentUTI options:nil
  completionHandler:^(NSData *data, NSError *error) {
  if (data) self.contents = data;
}];
}
```



# UI Action Setup (continued)

```
// Request document data
[itemProvider loadItemForTypeIdentifier:MyDocumentUTI options:nil
 completionHandler:^(NSData *data, NSError *error) {
    if (data) self.contents = data;
}];
}
```

# UI Action Setup (continued)

```
// Request document data
[itemProvider loadItemForTypeIdentifier:MyDocumentUTI options:nil
  completionHandler:^(NSData *data, NSError *error) {
  if (data) self.contents = data;
}];
}
```

# UI Action Setup (continued)

```
// Request document data
[itemProvider loadItemForTypeIdentifier:MyDocumentUTI options:nil
 completionHandler:^(NSData *data, NSError *error) {
    if (data) self.contents = data;
}];
}
```

# Editor UI Action Conclusion

```
- (IBAction)done:(id)sender {
    NSData *data = self.contents; // Document contents

    // Create NSItemProvider
    NSItemProvider *itemProvider = [[NSItemProvider alloc] initWithItem:data
                                                                    typeIdentifier:MyDocumentUTI];

    // Create NSExtensionItem
    NSExtensionItem *item = [[NSExtensionItem alloc] init];
    item.attachments = @[itemProvider];

    // Send the data back
    [self.extensionContext completeRequestReturningItems:@[item]
                          completionHandler:nil];
}
```

# Editor UI Action Conclusion

```
- (IBAction)done:(id)sender {
    NSData *data = self.contents; // Document contents

    // Create NSItemProvider
    NSItemProvider *itemProvider = [[NSItemProvider alloc] initWithItem:data
                                                                    typeIdentifier:MyDocumentUTI];

    // Create NSExtensionItem
    NSExtensionItem *item = [[NSExtensionItem alloc] init];
    item.attachments = @[itemProvider];

    // Send the data back
    [self.extensionContext completeRequestReturningItems:@[item]
                          completionHandler:nil];
}
```

# Editor UI Action Conclusion

```
- (IBAction)done:(id)sender {
    NSData *data = self.contents; // Document contents

    // Create NSItemProvider
    NSItemProvider *itemProvider = [[NSItemProvider alloc] initWithItem:data
                                                                    typeIdentifier:MyDocumentUTI];

    // Create NSExtensionItem
    NSExtensionItem *item = [[NSExtensionItem alloc] init];
    item.attachments = @[itemProvider];

    // Send the data back
    [self.extensionContext completeRequestReturningItems:@[item]
                          completionHandler:nil];
}
```

# Editor UI Action Conclusion

```
- (IBAction)done:(id)sender {
    NSData *data = self.contents; // Document contents

    // Create NSItemProvider
    NSItemProvider *itemProvider = [[NSItemProvider alloc] initWithItem:data
                                                                    typeIdentifier:MyDocumentUTI];

    // Create NSExtensionItem
    NSExtensionItem *item = [[NSExtensionItem alloc] init];
    item.attachments = @[itemProvider];

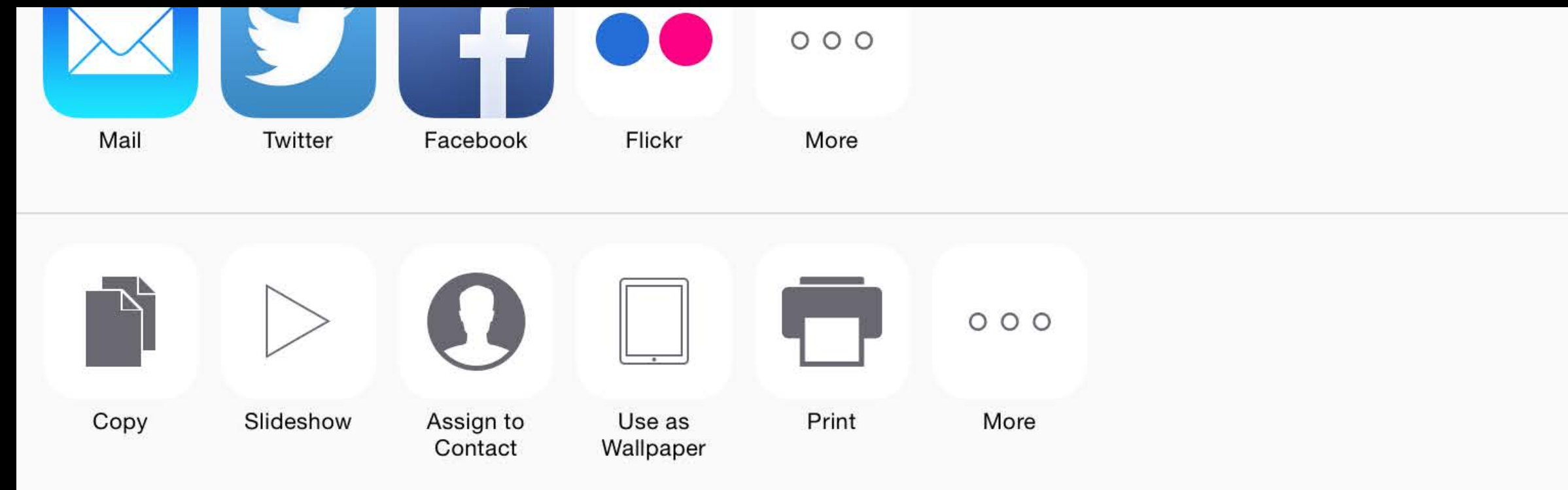
    // Send the data back
    [self.extensionContext completeRequestReturningItems:@[item]
                          completionHandler:nil];
}
```







# UIActivityViewController



# Receiving Result Items

```
- (void)setupActivityViewController {
    UIActivityViewController *controller; // initialize the controller here

    // Specifies the completion handler block
    controller.completionWithItemsHandler = ^(NSString *activityType, BOOL
                                             completed, NSArray *returnedItems, NSError *error) {
        if (completed && (returnedItems.count > 0)) {
            // process the result items
        }
    };
}
```

# Receiving Result Items

```
- (void)setupActivityViewController {
    UIActivityViewController *controller; // initialize the controller here

    // Specifies the completion handler block
    controller.completionWithItemsHandler = ^(NSString *activityType, BOOL
                                             completed, NSArray *returnedItems, NSError *error) {
        if (completed && (returnedItems.count > 0)) {
            // process the result items
        }
    };
}
```

# Receiving Result Items

A white rounded square containing the word "NEW" in a colorful, outlined font.

```
- (void)setupActivityViewController {
    UIActivityViewController *controller; // initialize the controller here

    // Specifies the completion handler block
    controller.completionWithItemsHandler = ^(NSString *activityType, BOOL
        completed, NSArray *returnedItems, NSError *error) {
        if (completed && (returnedItems.count > 0)) {
            // process the result items
        }
    };
}
```

# Receiving Result Items

A white rounded square containing the word "NEW" in a colorful, outlined font.

```
- (void)setupActivityViewController {
    UIActivityViewController *controller; // initialize the controller here

    // Specifies the completion handler block
    controller.completionWithItemsHandler = ^(NSString *activityType, BOOL
                                             completed, NSArray *returnedItems, NSError *error) {
        if (completed && (returnedItems.count > 0)) {
            // process the result items
        }
    };
}
```

# NSTextView

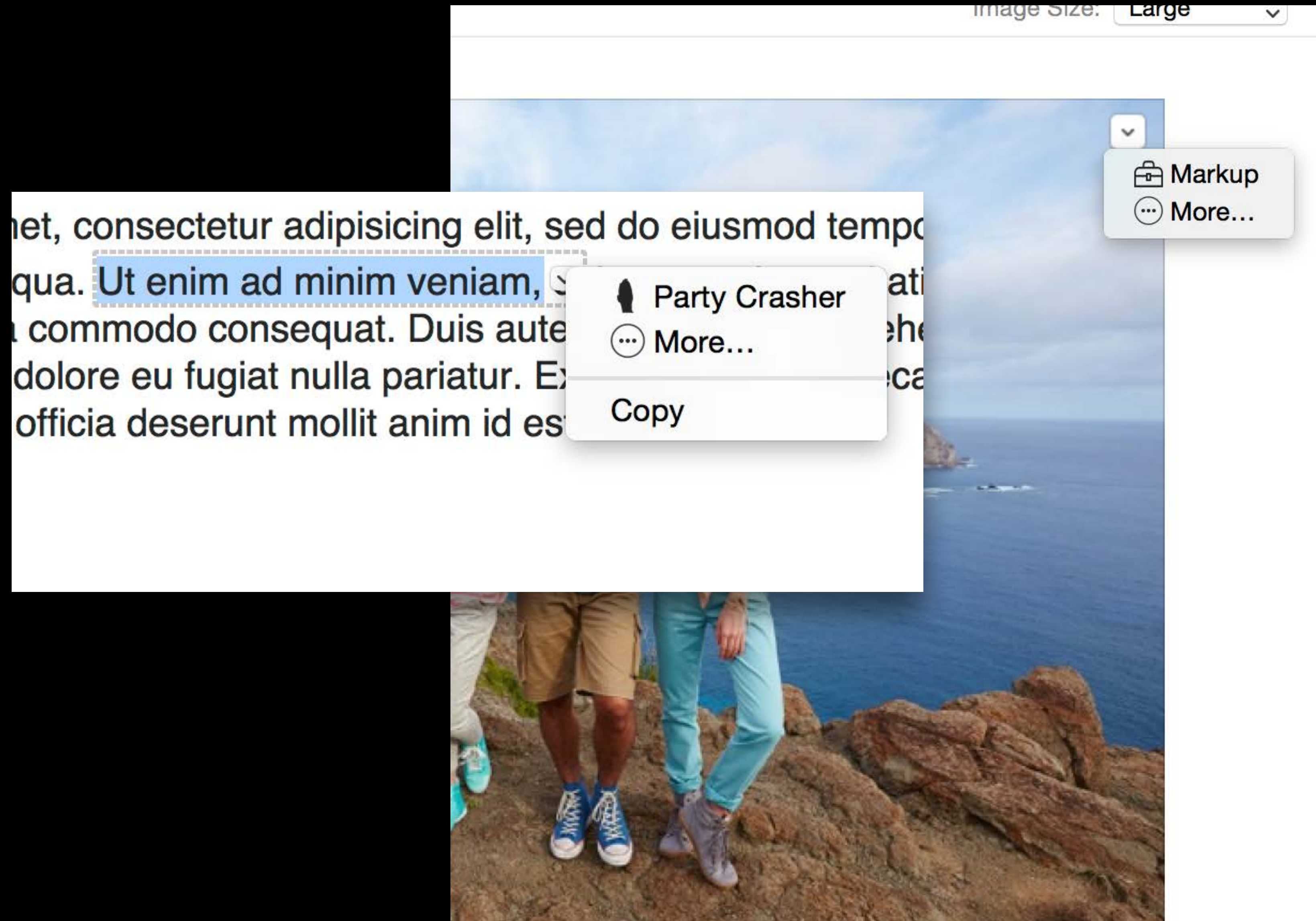


*Demo*

OS X Action Extension



# Supporting Multiple Data Types



# Reacting to Data Types

```
NSString *textType = (NSString *)kUTTypeText;

// is it text ?
if ([itemProvider hasItemConformingToTypeIdentifier:textType]) {
    [itemProvider loadItemForTypeIdentifier:textType options:nil
        completionHandler:^(NSAttributedString *string, NSError *error) {
        if (string) {
            // process text here
        }
    }];
}
```

# Reacting to Data Types

```
NSString *textType = (NSString *)kUTTypeText;
```

```
// is it text ?
if ([itemProvider hasItemConformingToTypeIdentifier:textType]) {
    [itemProvider loadItemForTypeIdentifier:textType options:nil
        completionHandler:^(NSAttributedString *string, NSError *error) {
        if (string) {
            // process text here
        }
    }];
}
```

# Reacting to Data Types

```
NSString *textType = (NSString *)kUTTypeText;
```

```
// is it text ?
```

```
if ([itemProvider hasItemConformingToTypeIdentifier:textType]) {  
    [itemProvider loadItemForTypeIdentifier:textType options:nil  
        completionHandler:^(NSAttributedString *string, NSError *error) {  
        if (string) {  
            // process text here  
        }  
    }];  
}
```

# Reacting to Data Types

```
NSString *textType = (NSString *)kUTTypeText;

// is it text ?
if ([itemProvider hasItemConformingToTypeIdentifier:textType]) {
    [itemProvider loadItemForTypeIdentifier:textType options:nil
        completionHandler:^(NSAttributedString *string, NSError *error) {
        if (string) {
            // process text here
        }
    }];
}
```

# Reacting to Data Types

```
NSString *textType = (NSString *)kUTTypeText;

// is it text ?
if ([itemProvider hasItemConformingToTypeIdentifier:textType]) {
    [itemProvider loadItemForTypeIdentifier:textType options:nil
        completionHandler:^(NSAttributedString *string, NSError *error) {
        if (string) {
            // process text here
        }
    }];
}
```

# Reacting to Data Types

```
NSString *textType = (NSString *)kUTTypeText;

// is it text ?
if ([itemProvider hasItemConformingToTypeIdentifier:textType]) {
    [itemProvider loadItemForTypeIdentifier:textType options:nil
        completionHandler:^(NSAttributedString *string, NSError *error) {
        if (string) {
            // process text here
        }
    }];
}
```

# Reacting to Data Types (continued)

```
NSString *imageType = (NSString *)kUTTypeImage;

// is it image ?
if ([itemProvider hasItemConformingToTypeIdentifier:imageType]) {
    [itemProvider loadItemForTypeIdentifier:imageType options:nil
        completionHandler:^(UIImage *image, NSError *error) {
        if (image) {
            // process image here
        }
    }];
}
```



# Reacting to Data Types (continued)

```
NSString *imageType = (NSString *)kUTTypeImage;
```

```
// is it image ?
if ([itemProvider hasItemConformingToTypeIdentifier:imageType]) {
    [itemProvider loadItemForTypeIdentifier:imageType options:nil
        completionHandler:^(UIImage *image, NSError *error) {
        if (image) {
            // process image here
        }
    }];
}
```

# Reacting to Data Types (continued)

```
NSString *imageType = (NSString *)kUTTypeImage;

// is it image ?
if ([itemProvider hasItemConformingToTypeIdentifier:imageType]) {
    [itemProvider loadItemForTypeIdentifier:imageType options:nil
        completionHandler:^(UIImage *image, NSError *error) {
        if (image) {
            // process image here
        }
    }];
}
```

# Reacting to Data Types (continued)

```
NSString *imageType = (NSString *)kUTTypeImage;

// is it image ?
if ([itemProvider hasItemConformingToTypeIdentifier:imageType]) {
    [itemProvider loadItemForTypeIdentifier:imageType options:nil
        completionHandler:^(UIImage *image, NSError *error) {
        if (image) {
            // process image here
        }
    }];
}
```

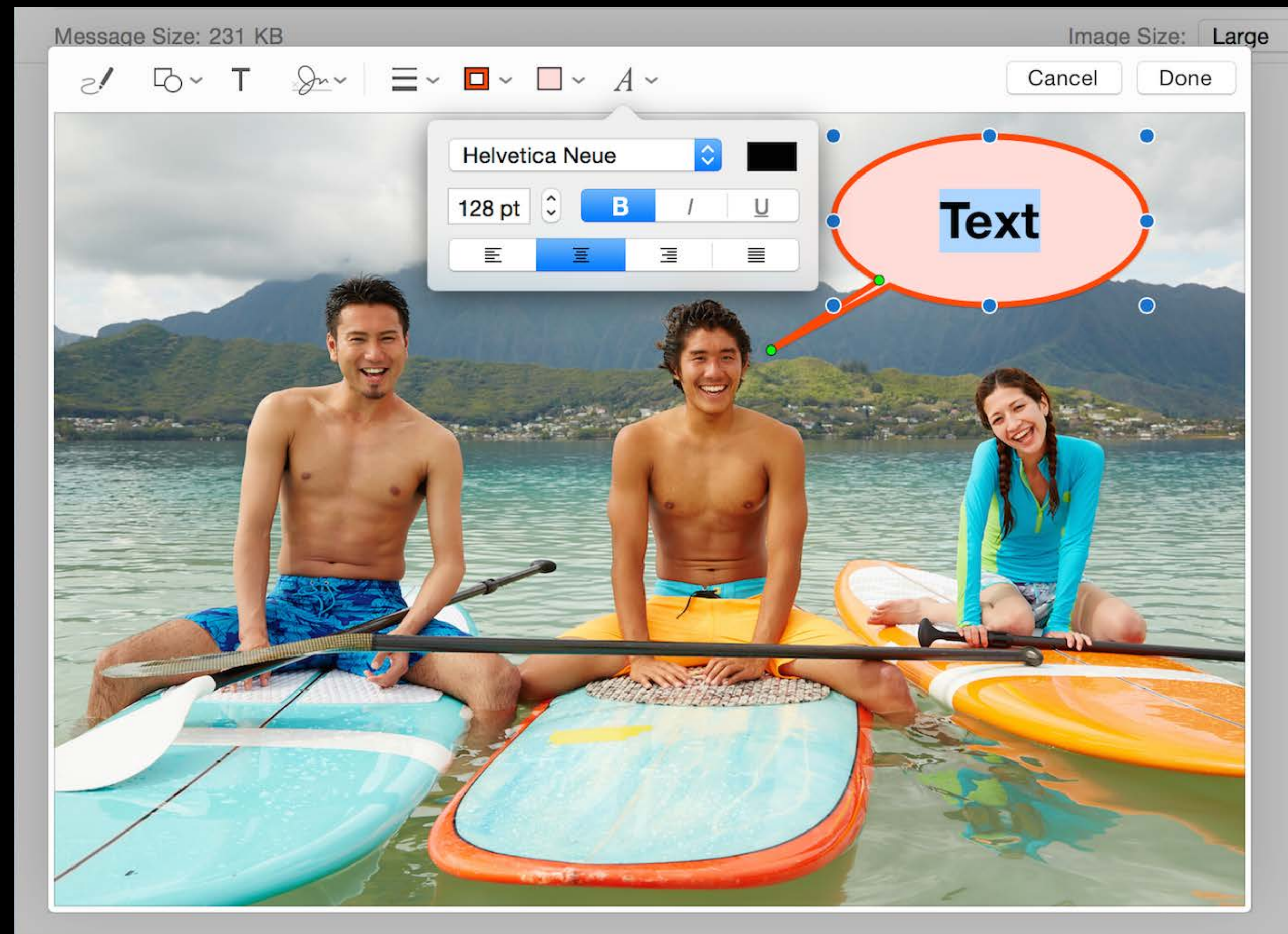
# Reacting to Data Types (continued)

```
NSString *imageType = (NSString *)kUTTypeImage;

// is it image ?
if ([itemProvider hasItemConformingToTypeIdentifier:imageType]) {
    [itemProvider loadItemForTypeIdentifier:imageType options:nil
        completionHandler:^(UIImage *image, NSError *error) {
        if (image) {
            // process image here
        }
    }];
}
```

# Overlay

OS X enhancement



# Specifying Extension Frame

-viewDidLoad

```
CGRect sourceFrame = itemProvider.sourceFrame; // original data frame

if (!NSIsEmptyRect(sourceFrame)) { // the host specifies the source frame
    // Adjust sourceFrame based on extension UI margins
    sourceFrame = NSInsetRect(sourceRect, HORIZ_MARGIN, VERT_MARGIN);

    self.preferredScreenOrigin = sourceFrame.origin;
    self.preferredContentSize = sourceFrame.size;
}
```

# Specifying Extension Frame

-viewDidLoad

```
CGRect sourceFrame = itemProvider.sourceFrame; // original data frame
```

```
if (!NSIsEmptyRect(sourceFrame)) { // the host specifies the source frame
    // Adjust sourceFrame based on extension UI margins
    sourceFrame = NSInsetRect(sourceRect, HORIZ_MARGIN, VERT_MARGIN);

    self.preferredScreenOrigin = sourceFrame.origin;
    self.preferredContentSize = sourceFrame.size;
}
```

# Specifying Extension Frame

-viewDidLoad

```
CGRect sourceFrame = itemProvider.sourceFrame; // original data frame

if (!NSIsEmptyRect(sourceFrame)) { // the host specifies the source frame
    // Adjust sourceFrame based on extension UI margins
    sourceFrame = NSInsetRect(sourceRect, HORIZ_MARGIN, VERT_MARGIN);

    self.preferredScreenOrigin = sourceFrame.origin;
    self.preferredContentSize = sourceFrame.size;
}
```



# Specifying Extension Frame

-viewDidLoad

```
CGRect sourceFrame = itemProvider.sourceFrame; // original data frame

if (!CGRectIsEmpty(sourceFrame)) { // the host specifies the source frame
    // Adjust sourceFrame based on extension UI margins
    sourceFrame = CGRectInset(sourceRect, HORIZ_MARGIN, VERT_MARGIN);

    self.preferredScreenOrigin = sourceFrame.origin;
    self.preferredContentSize = sourceFrame.size;
}
```

# Specifying Extension Frame

-viewDidLoad

```
CGRect sourceFrame = itemProvider.sourceFrame; // original data frame

if (!NSIsEmptyRect(sourceFrame)) { // the host specifies the source frame
    // Adjust sourceFrame based on extension UI margins
    sourceFrame = NSInsetRect(sourceRect, HORIZ_MARGIN, VERT_MARGIN);

    self.preferredScreenOrigin = sourceFrame.origin;
    self.preferredContentSize = sourceFrame.size;
}
```

# Actions and the Web

Safari custom actions

Ian Baird

iOS Frameworks Engineer

iPad 9:41 AM 100%

apple.com

Store Mac iPod

OS X Completely new

**AirDrop**  
Tap to turn on Wi-Fi and Bluetooth to share with people via AirDrop.

Message Mail Twitter Facebook

Annotate Image Add Bookmark Add to Reading List Add to Home Screen

Monday, June 2nd

HEATH CERAMICS

Messages

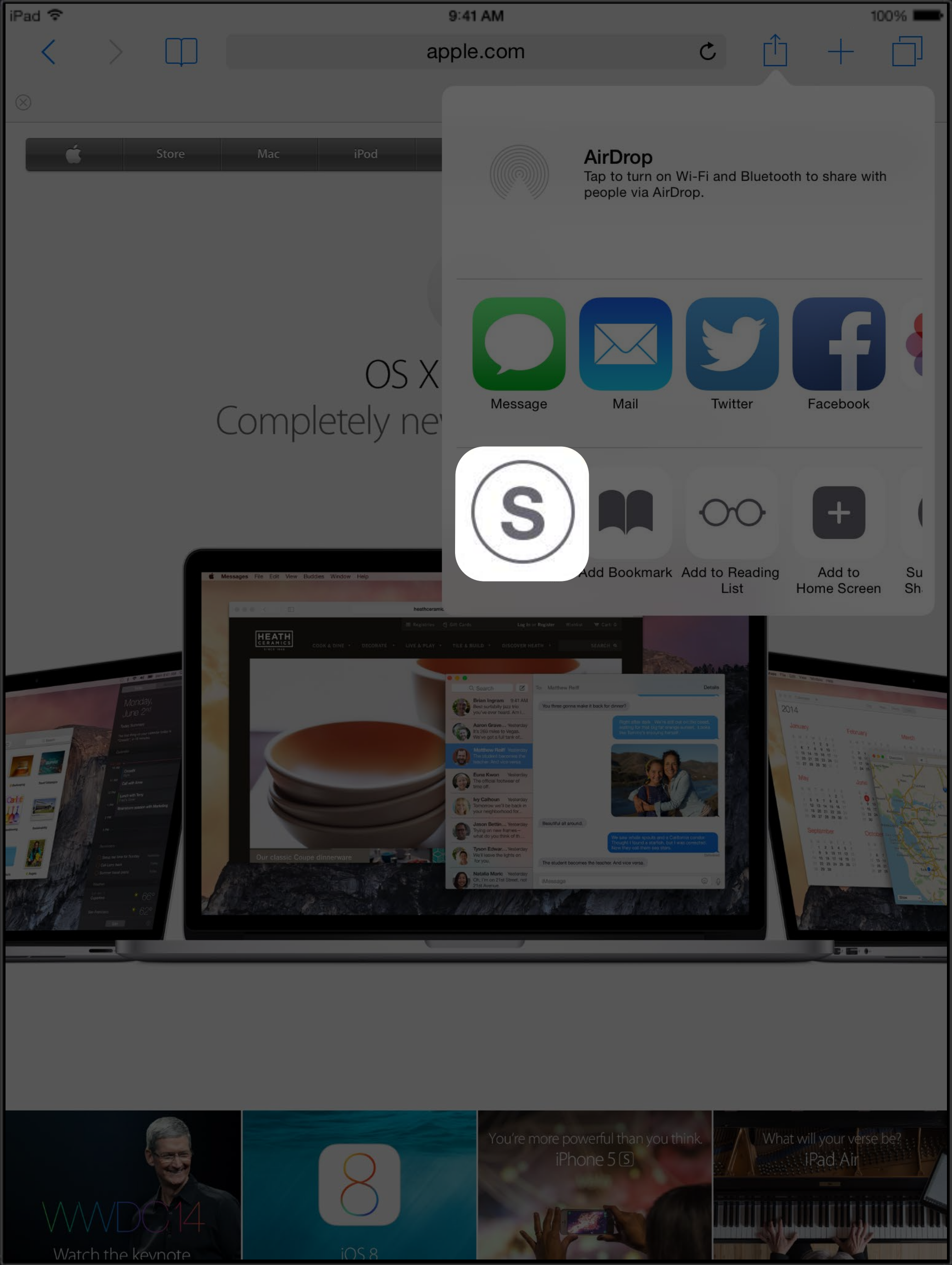
2014

WWDC 14 Watch the keynote

iOS 8

You're more powerful than you think iPhone 5S

What will your verse be? iPad Air



# Safari Custom Actions



Rich new functionality in Safari on iOS



# Safari Custom Actions



Rich new functionality in Safari on iOS

Access the DOM



# Safari Custom Actions



Rich new functionality in Safari on iOS

Access the DOM

Two types





# Safari Custom Actions



Rich new functionality in Safari on iOS

Access the DOM

Two types

- View controller



# Safari Custom Actions



Rich new functionality in Safari on iOS

Access the DOM

Two types

- View controller
- No view



# Safari Custom Actions

JavaScript



# Safari Custom Actions

JavaScript



# Safari Custom Actions

## JavaScript

```
var PreprocessingJavaScript = function() {  
  
};  
  
PreprocessingJavaScript.prototype = {  
  
  run: function(actionArgs) {  
    // Get all image URLs from the document.  
    ...  
    actionArgs.completionFunction({"imageUrls": imageUrls, "baseURI":  
document.baseURI});  
  },  
};
```

# Safari Custom Actions

## JavaScript

```
var PreprocessingJavaScript = function() {  
  
};  
  
PreprocessingJavaScript.prototype = {  
  
  run: function(actionArgs) {  
    // Get all image URLs from the document.  
    ...  
    actionArgs.completionFunction({"imageUrls": imageUrls, "baseURI":  
document.baseURI});  
  },  
};
```

# Safari Custom Actions

JavaScript



# Safari Custom Actions

JavaScript





# Safari Custom Actions

## JavaScript

```
var PreprocessingJavaScript = function() {  
  
};  
  
PreprocessingJavaScript.prototype = {  
...  
  finalize: function(actionArgs) {  
    alert(actionArgs["message"]);  
  }  
};
```

# Safari Custom Actions

## JavaScript

```
var PreprocessingJavaScript = function() {  
  
};  
  
PreprocessingJavaScript.prototype = {  
  ...  
  finalize: function(actionArgs) {  
    alert(actionArgs["message"]);  
  }  
};
```

# Safari Custom Actions

## JavaScript

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

```
PreprocessingJavaScript.prototype = {  
...  
};
```

```
var ExtensionPreprocessingJS = new PreprocessingJavaScript;
```

# Safari Custom Actions

## JavaScript

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

```
PreprocessingJavaScript.prototype = {  
...  
};
```

```
var ExtensionPreprocessingJS = new PreprocessingJavaScript;
```

# Safari Custom Actions

Global variable

# Safari Custom Actions

Global variable

`ExtensionPreprocessingJS`

*Demo*

Safari custom actions

# Safari Custom Actions

## Summary

Flexibility of the web





# Safari Custom Actions

## Summary

Flexibility of the web

Power of extensions



# Safari Custom Actions

## Summary

Flexibility of the web

Power of extensions

Transform web data easily



# Safari Custom Actions

## Summary

Flexibility of the web

Power of extensions

Transform web data easily



# Summary

Rich workflows

# Summary

Rich workflows

Security

# Summary

Rich workflows

Security

Fun!

# More Information

Jake Behrens

App Frameworks Evangelist

[behrens@apple.com](mailto:behrens@apple.com)

Documentation

App Extension Programming Guide

<http://developer.apple.com>

Apple Developer Forums

<http://devforums.apple.com>

# Labs

- 
- Extensions Lab

Frameworks Lab B

Thursday 2:00PM

---



 WWDC14