

App Extension Best Practices

Session 224

Sophia Teutschler UIKit Engineer

Ian Baird CoreOS Engineer

Agenda

Agenda

Action and Share Extensions

Agenda

Action and Share Extensions

Today widget enhancements

Agenda

Action and Share Extensions

Today widget enhancements

Real world examples

Action and Share Extensions

Action and Share Extensions



Action



Share

Action and Share Extensions

Action Extensions



Action

Action and Share Extensions

Action Extensions

Acts on the current content



Action

Action and Share Extensions

Action Extensions

Acts on the current content

Uses the content as the user interface



Action

Action and Share Extensions

Action Extensions

Acts on the current content

Uses the content as the user interface

Opportunity to present additional options



Action

Action and Share Extensions

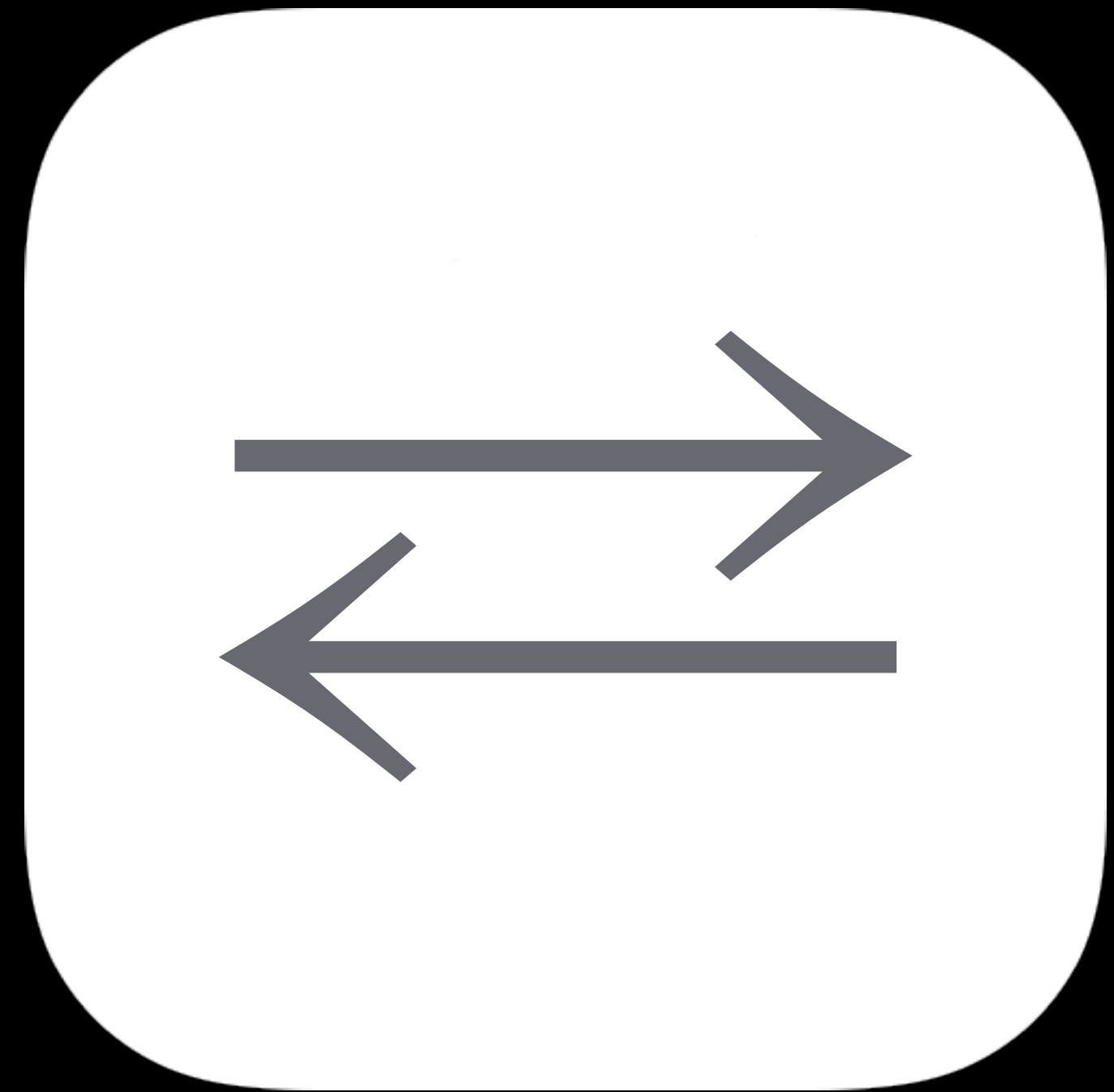
Action Extensions

Acts on the current content

Uses the content as the user interface

Opportunity to present additional options

App can provide multiple Action Extensions



Action

iPad

9:41 AM

100%

<

>

🔖

apple.com

↺

🔗

+

📄

🍏 WATCH


Explore

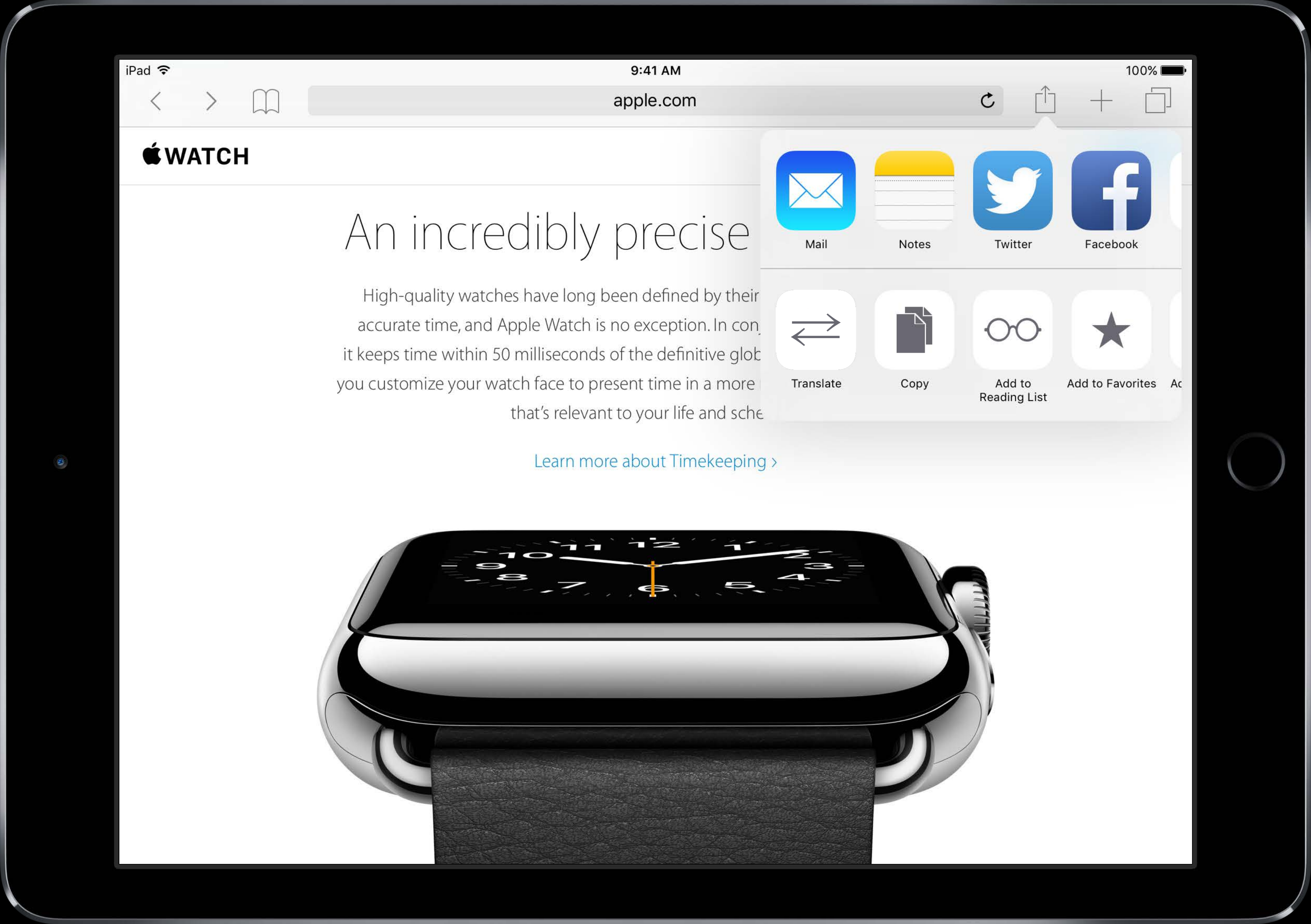
Buy Now

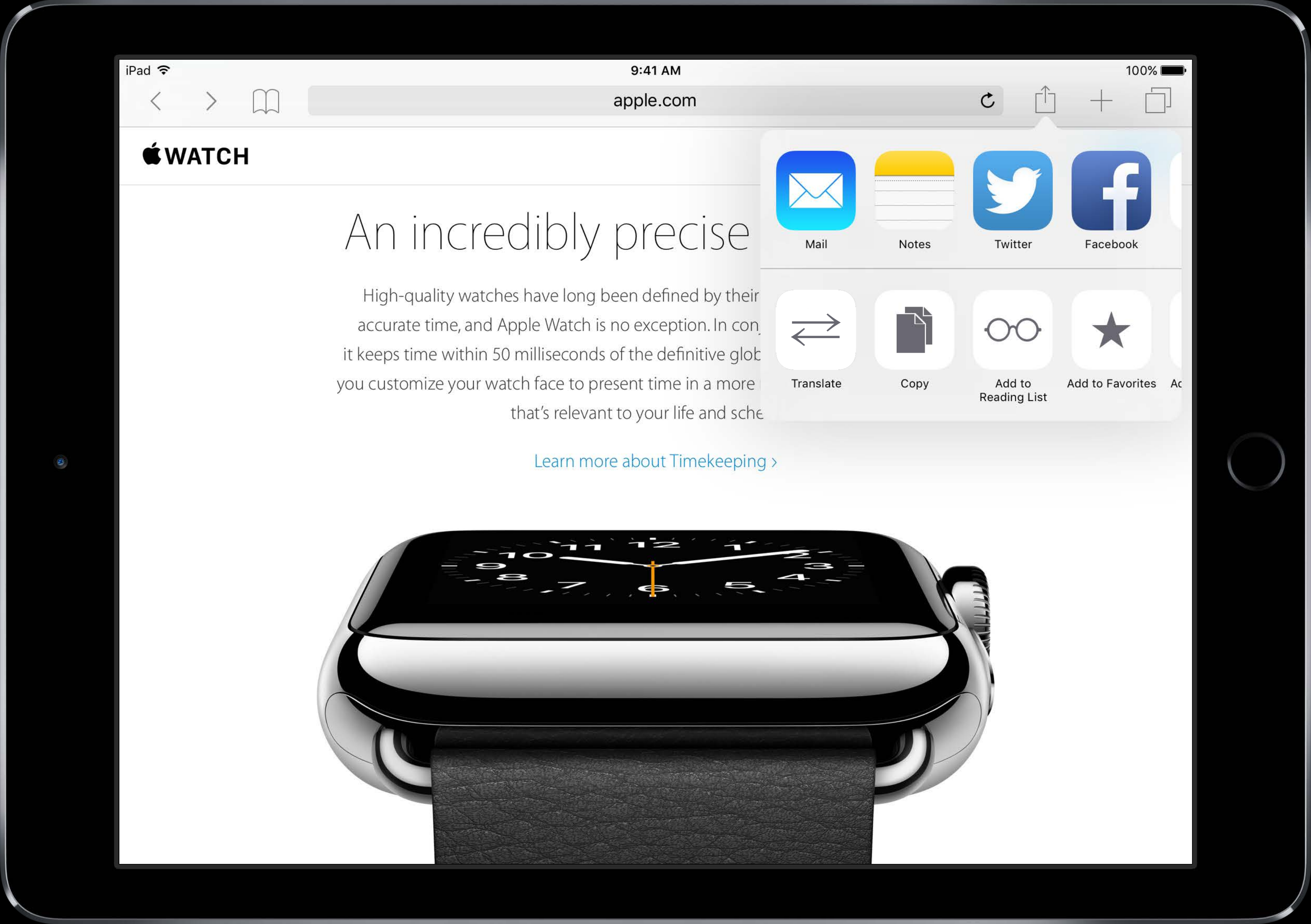
An incredibly precise timepiece.

High-quality watches have long been defined by their ability to keep unfailingly accurate time, and Apple Watch is no exception. In conjunction with your iPhone, it keeps time within 50 milliseconds of the definitive global time standard. It even lets you customize your watch face to present time in a more meaningful, personal context that's relevant to your life and schedule.

[Learn more about Timekeeping >](#)







iPad

9:41 AM

100%

<

>

📖

apple.com

↺

📄

+

📑

Apple WATCH

An incredibly precise

High-quality watches have long been defined by their accurate time, and Apple Watch is no exception. In con, it keeps time within 50 milliseconds of the definitive glob you customize your watch face to present time in a more that's relevant to your life and sche

[Learn more about Timekeeping >](#)

📧

Mail

📝

Notes

🐦

Twitter

📘

Facebook

↔

Translate

📄

Copy

🕶

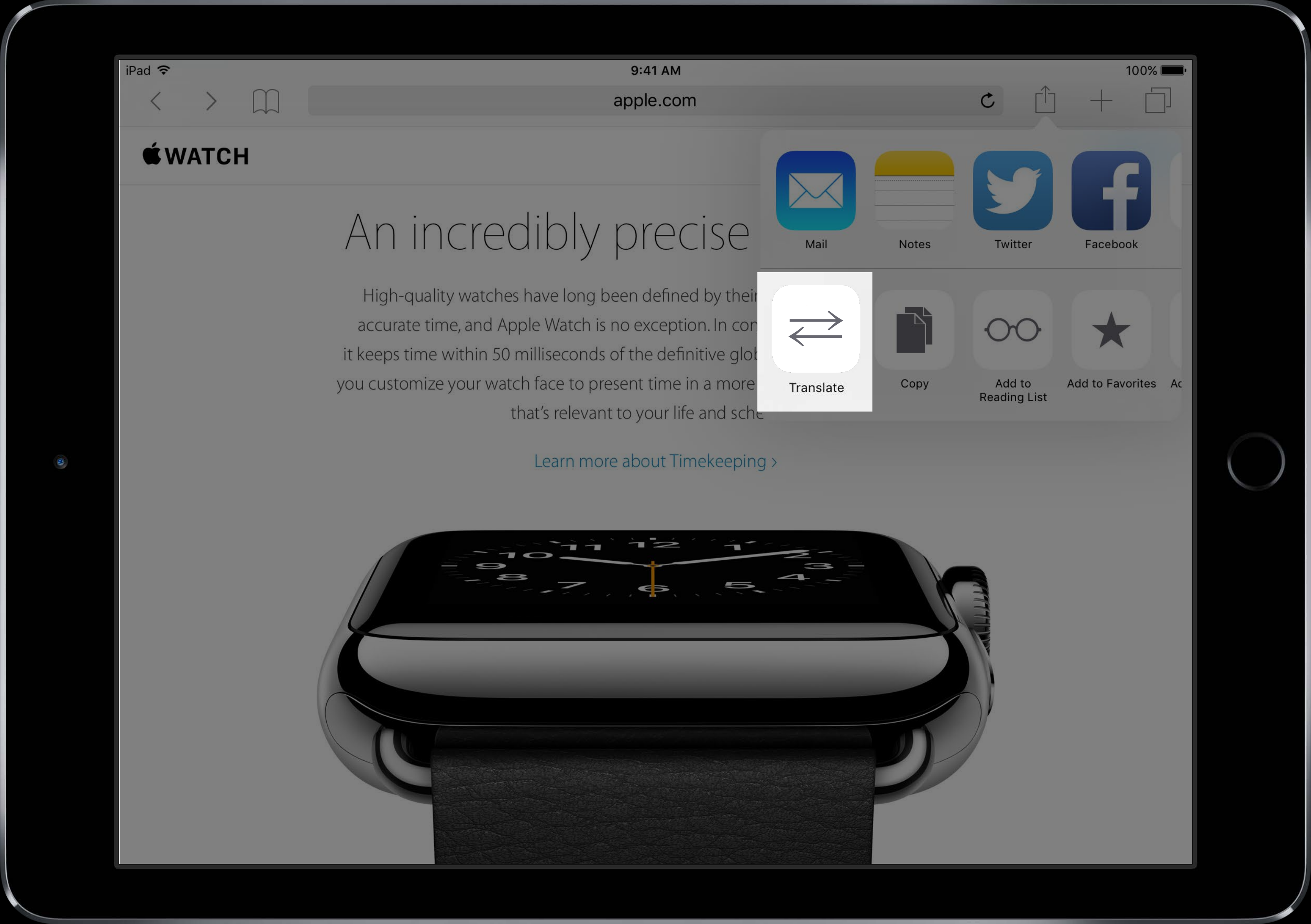
Add to Reading List

★

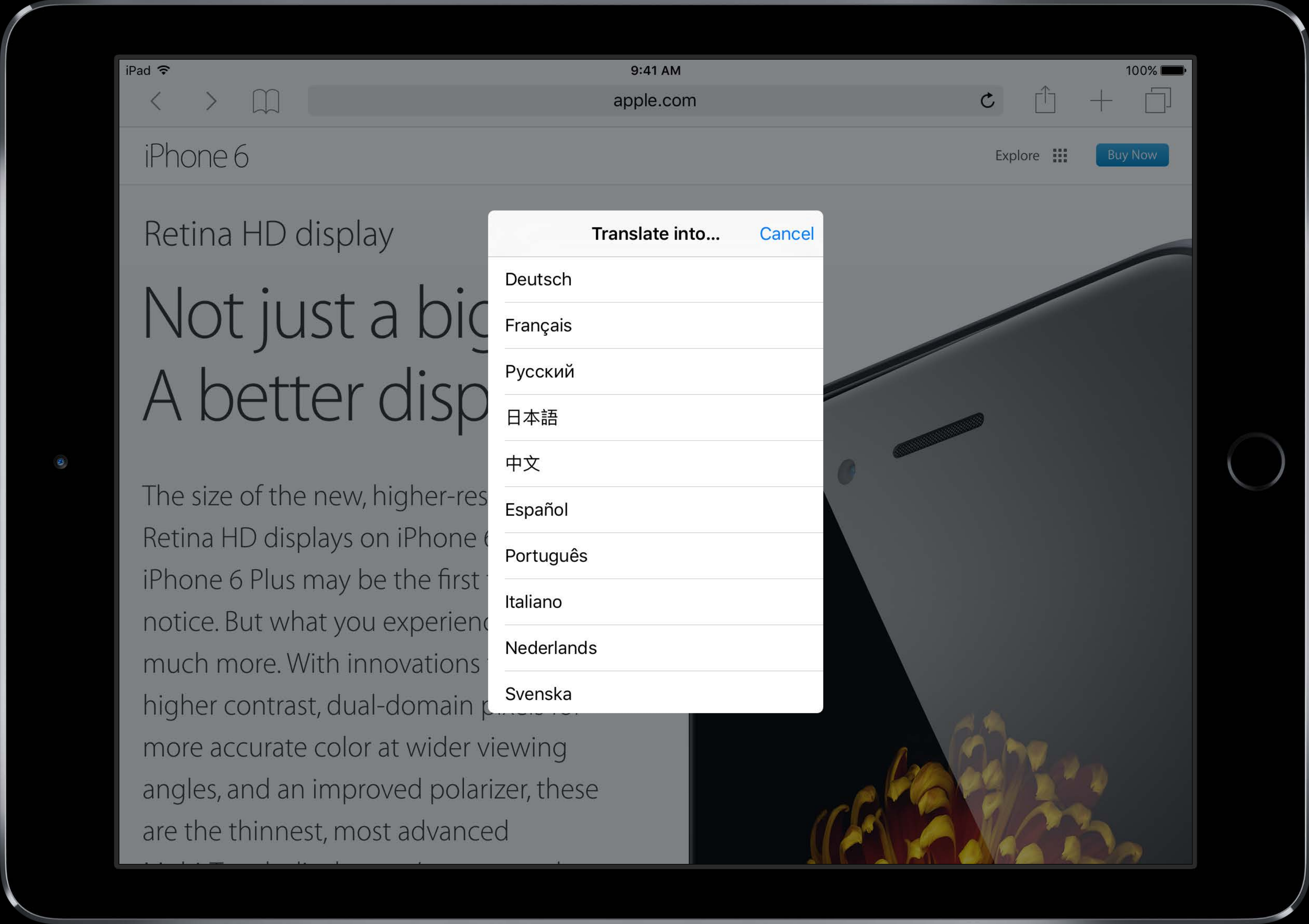
Add to Favorites

🔖

Ac







iPad

9:41 AM

100%



apple.com



iPhone 6

Explore

Buy Now

Retina HD display

Not just a big
A better display

The size of the new, higher-resolution Retina HD displays on iPhone 6 and iPhone 6 Plus may be the first thing you notice. But what you experience is much more. With innovations like higher contrast, dual-domain pixels for more accurate color at wider viewing angles, and an improved polarizer, these are the thinnest, most advanced

Translate into...

Cancel

Deutsch

Français

Русский

日本語

中文

Español

Português

Italiano

Nederlands

Svenska

iPad

9:41 AM

100%

<

>

📖

apple.com

↺

🔗

+

📑

iPhone 6

Подробнее

Купить

HD-дисплей Retina

Этот дисплей не просто больше. Он лучше.

Мы понимаем, что вы сразу же обратите внимание на размеры новых HD-дисплеев Retina повышенного разрешения на iPhone 6 и iPhone 6 Plus. Но это далеко не единственное их достоинство. Это самые тонкие и совершенные среди наших дисплеев Multi-Touch: здесь использованы инновационные технологии для



Action and Share Extensions



Action



Share

Action and Share Extensions



Share

Action and Share Extensions

Share Extensions



Share

Action and Share Extensions

Share Extensions

Shares content to your app or web service



Share

Action and Share Extensions

Share Extensions

Shares content to your app or web service

Allows validation and editing



Share

Action and Share Extensions

Share Extensions

Shares content to your app or web service

Allows validation and editing

Can use `SLComposeServiceViewController`



Share

Action and Share Extensions

Share Extensions

Shares content to your app or web service

Allows validation and editing

Can use `SLComposeServiceViewController`

Your app provides one Share Extension



Share

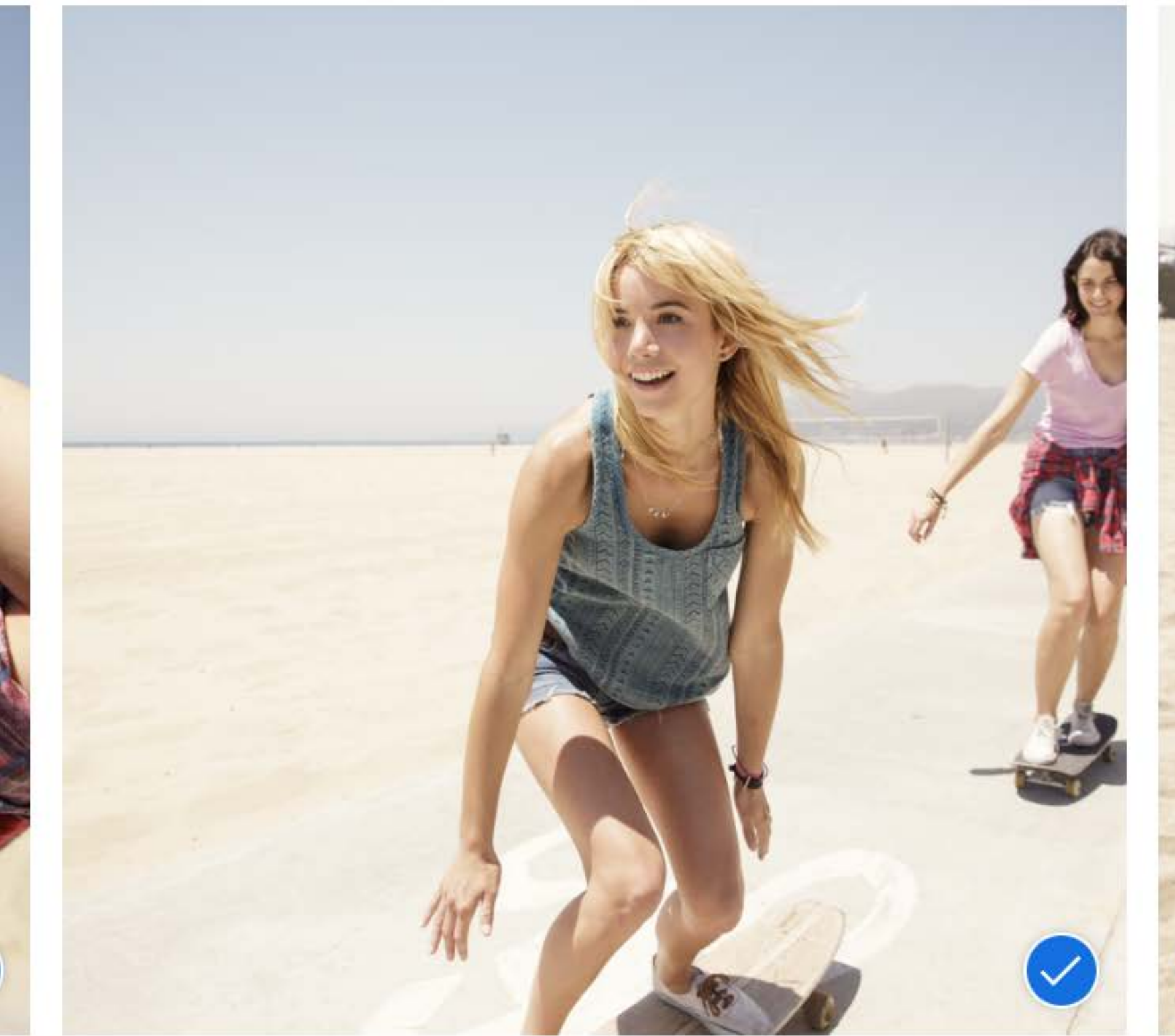


9:41 AM

100%

Cancel

4 Photos Selected



Message



iCloud
Photo Sharing



Mail



Notes



Facebook



Hide



Copy



Slideshow



AirPlay




Print


Cancel

4 Photos Selected







Message




iCloud
Photo Sharing




Mail




Notes




Facebook




Hide




Copy



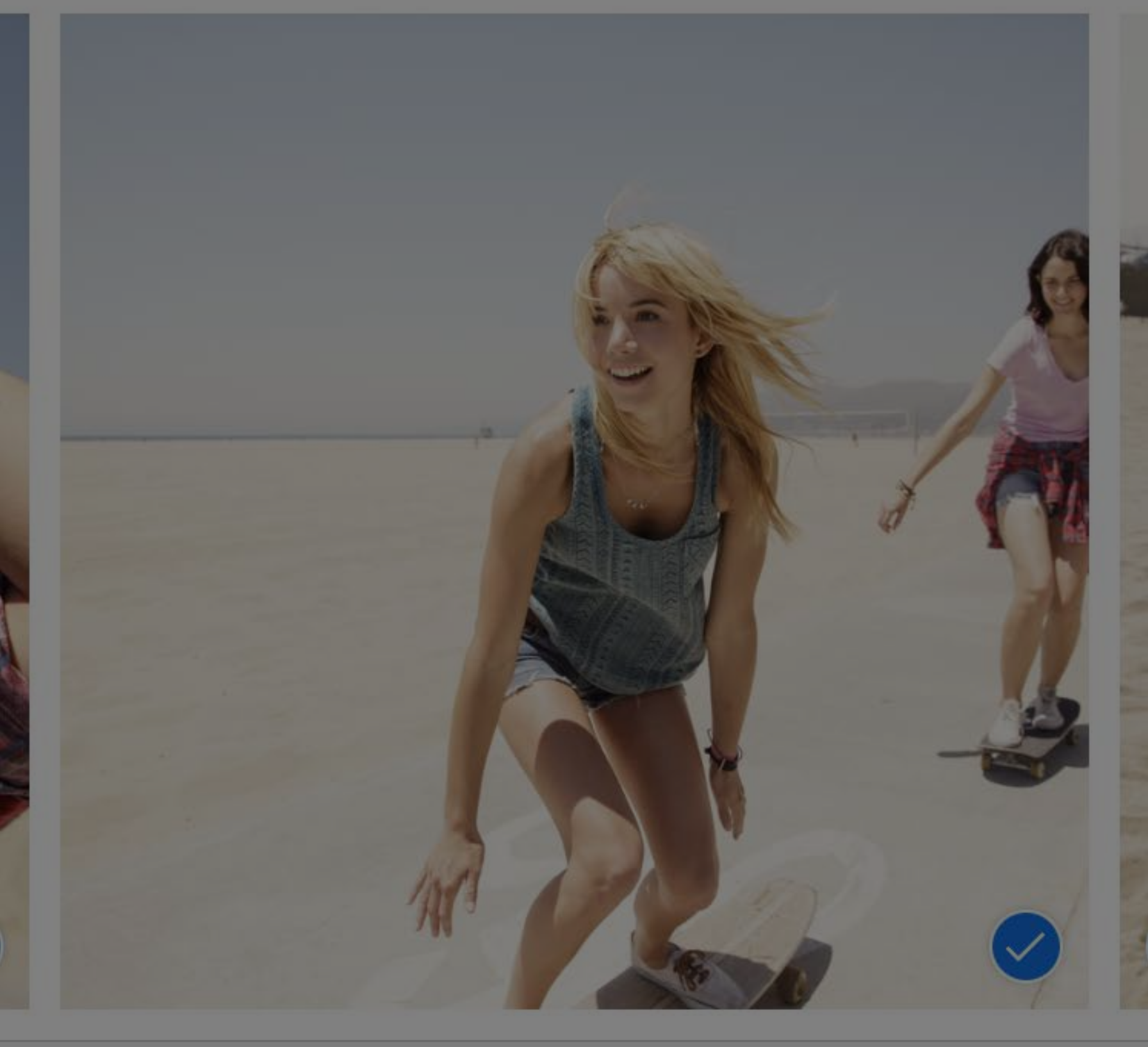
Slideshow





AirPlay





Print




- 


Message
- 


iCloud
Photo Sharing
- 

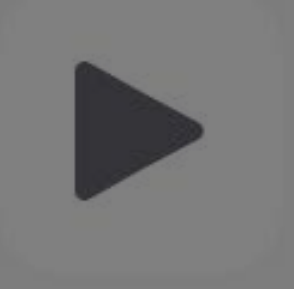
Mail
- 

Notes
- 

Facebook

- 

Hide
- 

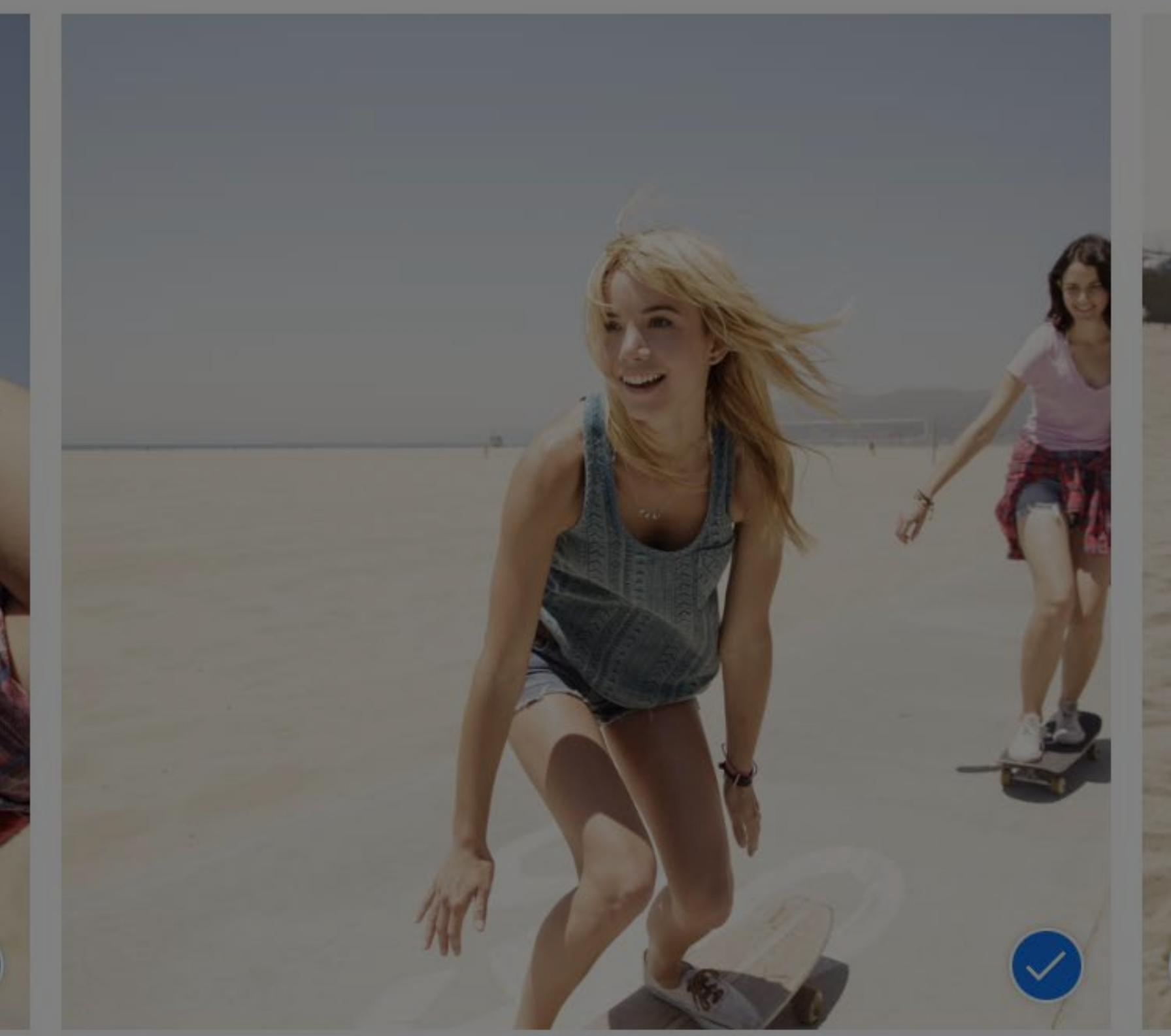
Copy
- 


Slideshow
- 


AirPlay
- 


Print


Cancel4 Photos Selected




- 


Message
- 


iCloud
Photo Sharing
- 


Mail
- 


Notes
- 


Facebook

- 

Hide
- 

Copy
- 

Slideshow
- 

AirPlay
- 

Print

Cancel 4 Photos Selected

Cancel iCloud Post

Comment (optional)



4 Photos

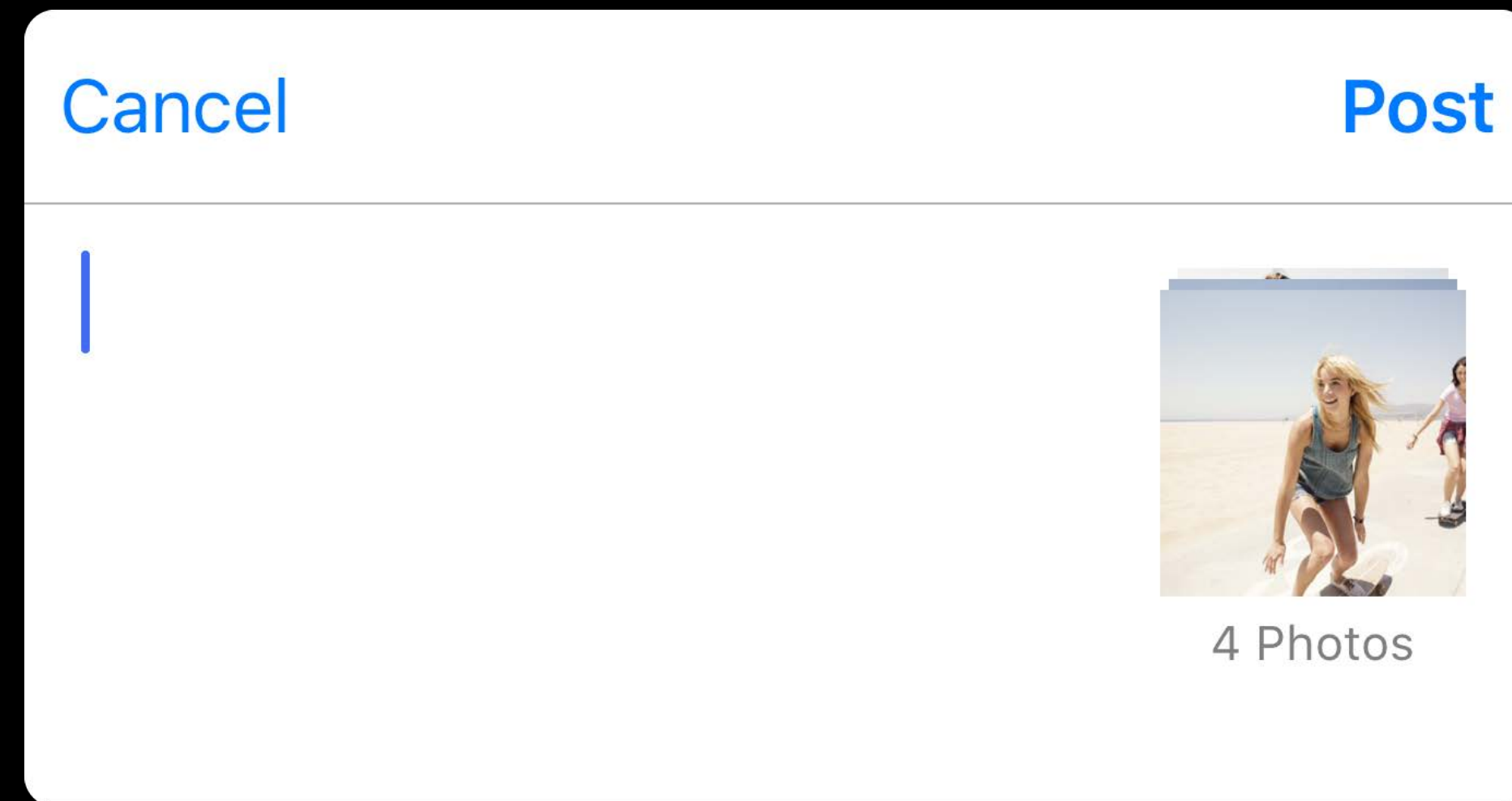
Shared Album

Santa Monica



Action and Share Extensions

SLComposeServiceViewController




Action and Share Extensions

SLComposeServiceViewController

Cancel

Post

Caption (Optional)



80

Blog

My Road Trips

Tags

Santa Monica, Summer >

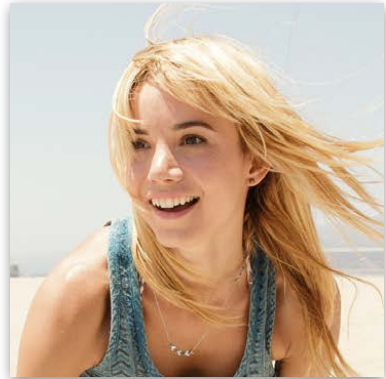
Action and Share Extensions

SLComposeServiceViewController

Cancel

Post

Caption (Optional)



80

Blog

My Road Trips

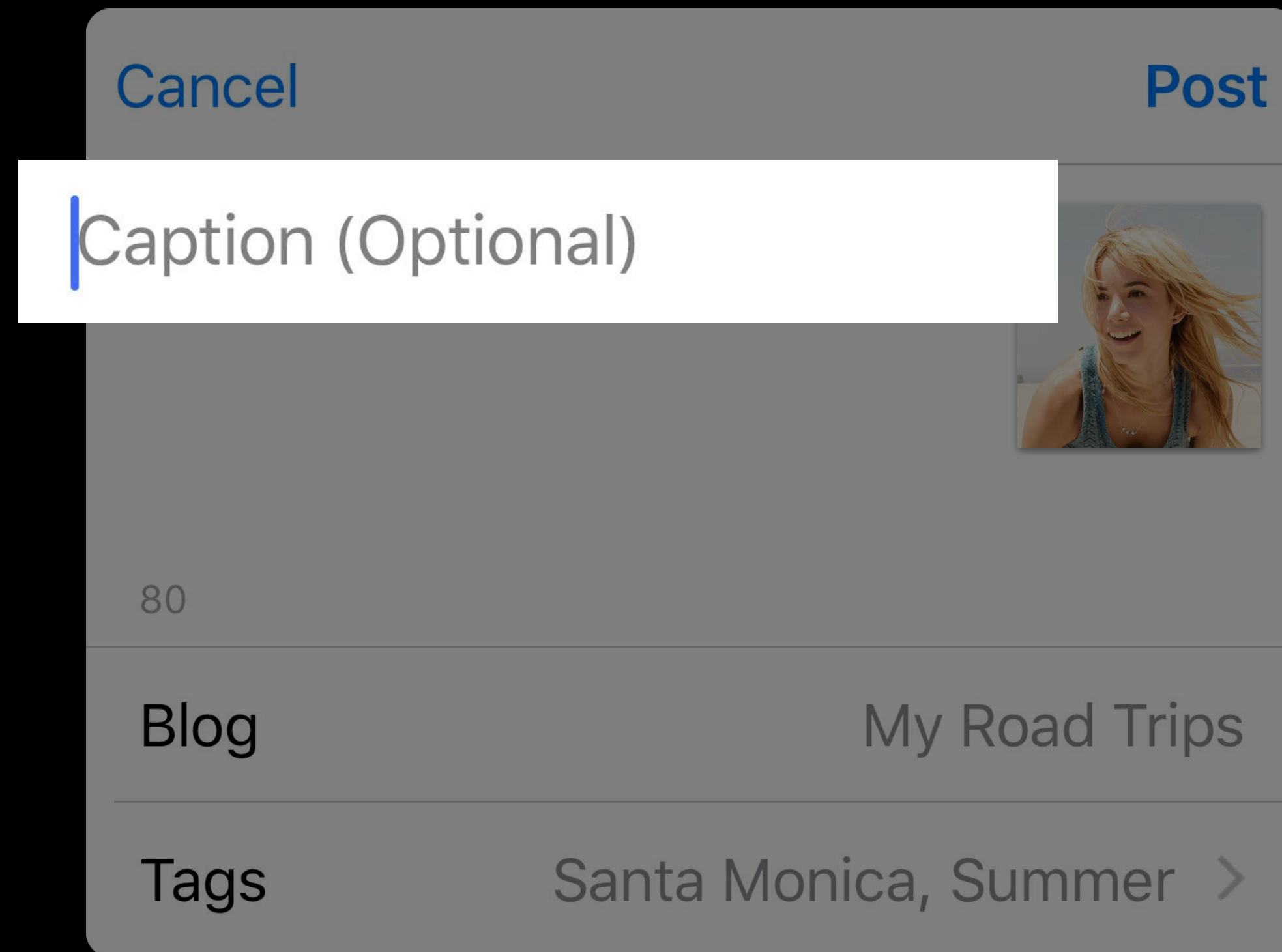
Tags

Santa Monica, Summer >

```
viewController.placeholder = "Caption (Optional)"
```

Action and Share Extensions

SLComposeServiceViewController



```
viewController.placeholder = "Caption (Optional)"
```

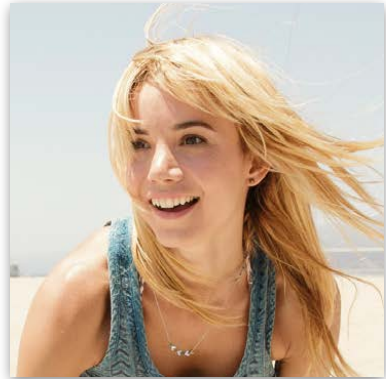
Action and Share Extensions

SLComposeServiceViewController

Cancel

Post

Caption (Optional)



80

Blog

My Road Trips

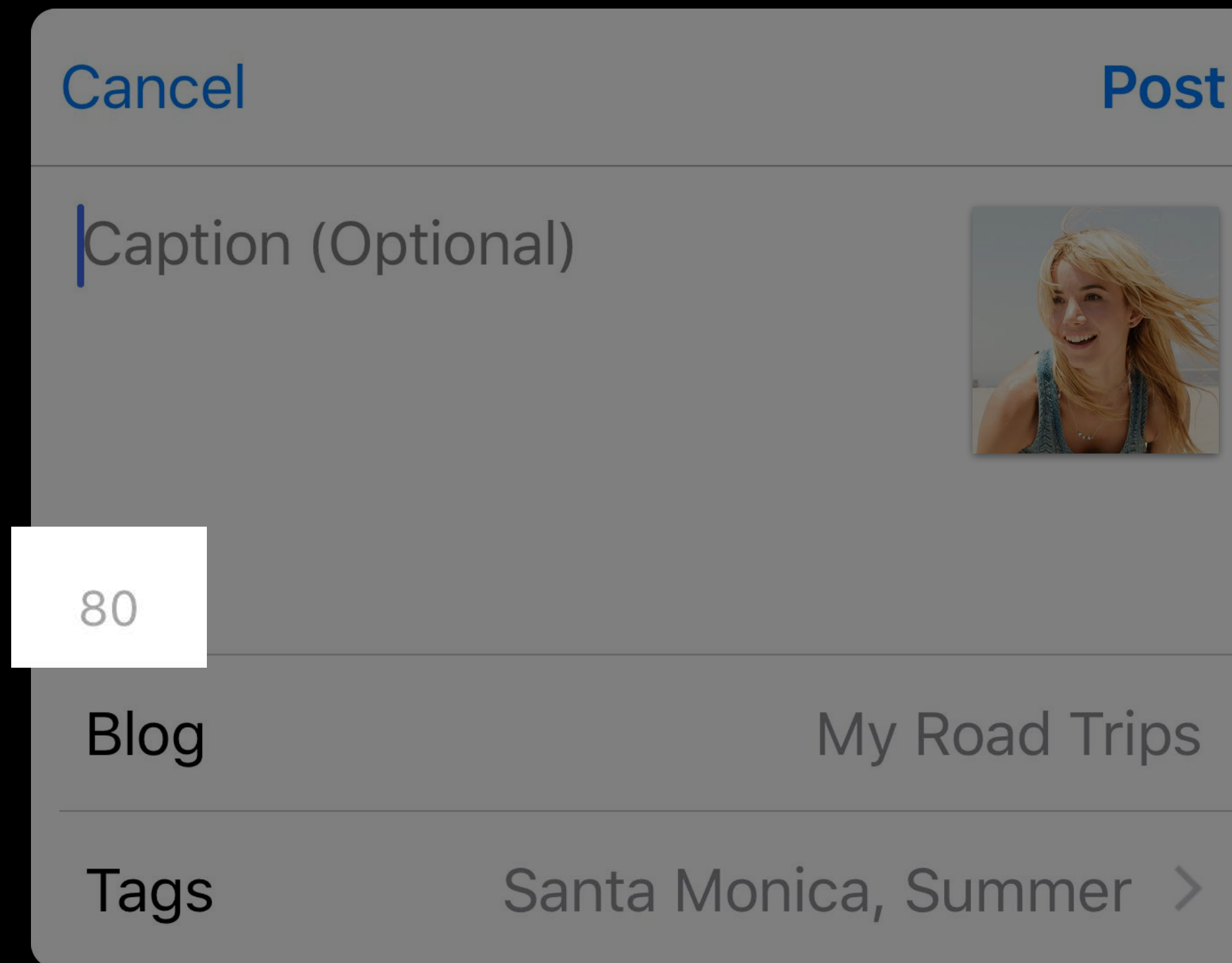
Tags

Santa Monica, Summer >

```
viewController.charactersRemaining = 80
```

Action and Share Extensions

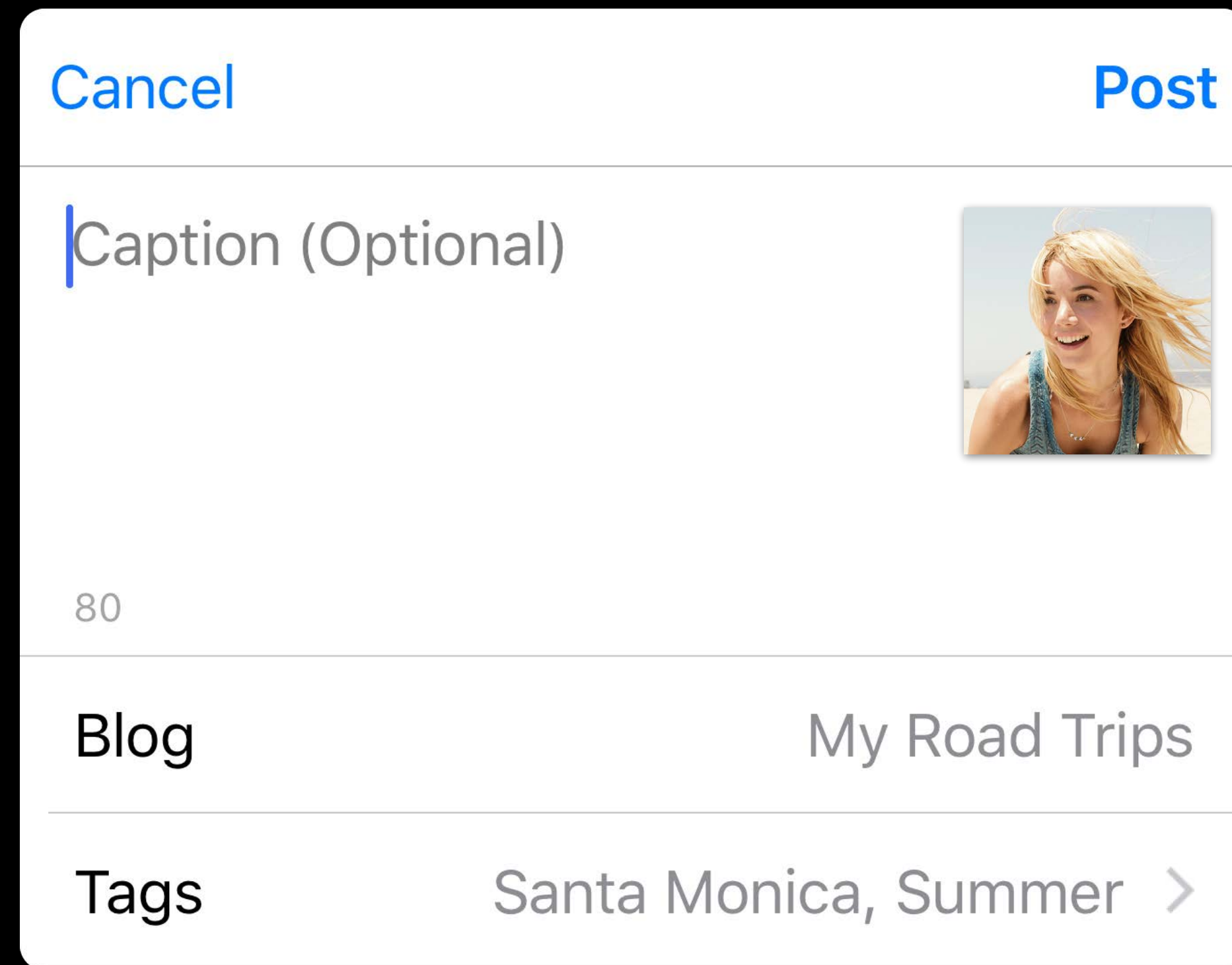
SLComposeServiceViewController



```
viewController.charactersRemaining = 80
```

Action and Share Extensions

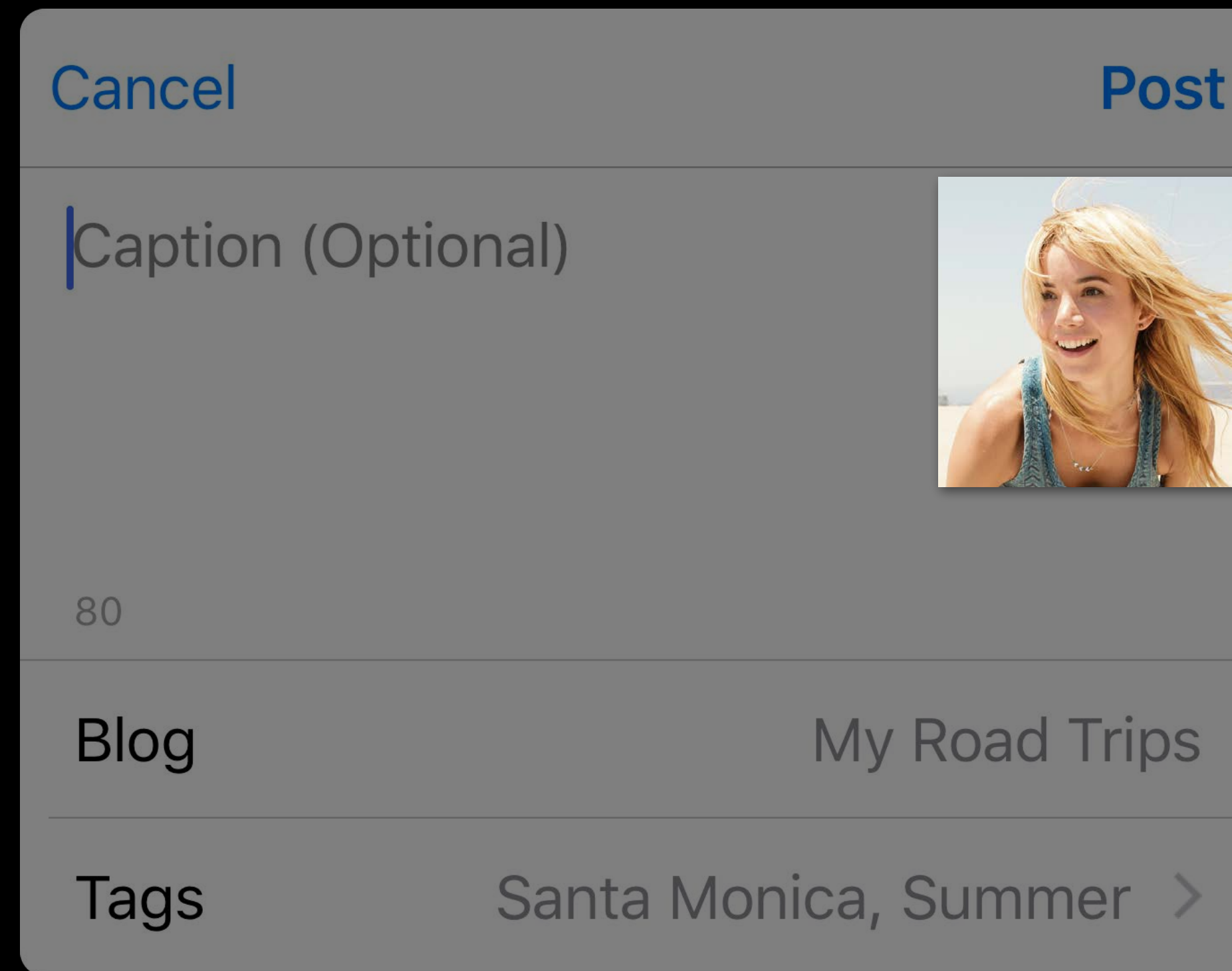
SLComposeServiceViewController



```
override func loadPreviewView() -> UIView! {  
    return FaceDetectingPreviewView(...)  
}
```


Action and Share Extensions

SLComposeServiceViewController



```
override func loadPreviewView() -> UIView! {  
    return FaceDetectingPreviewView(...)  
}
```

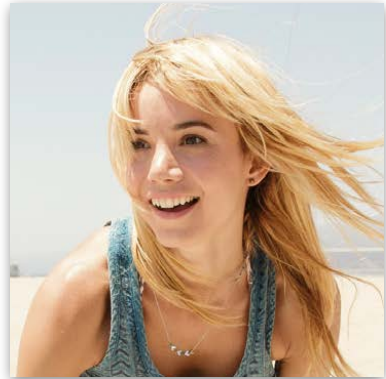
Action and Share Extensions

SLComposeServiceViewController

Cancel

Post

Caption (Optional)



80

Blog

My Road Trips

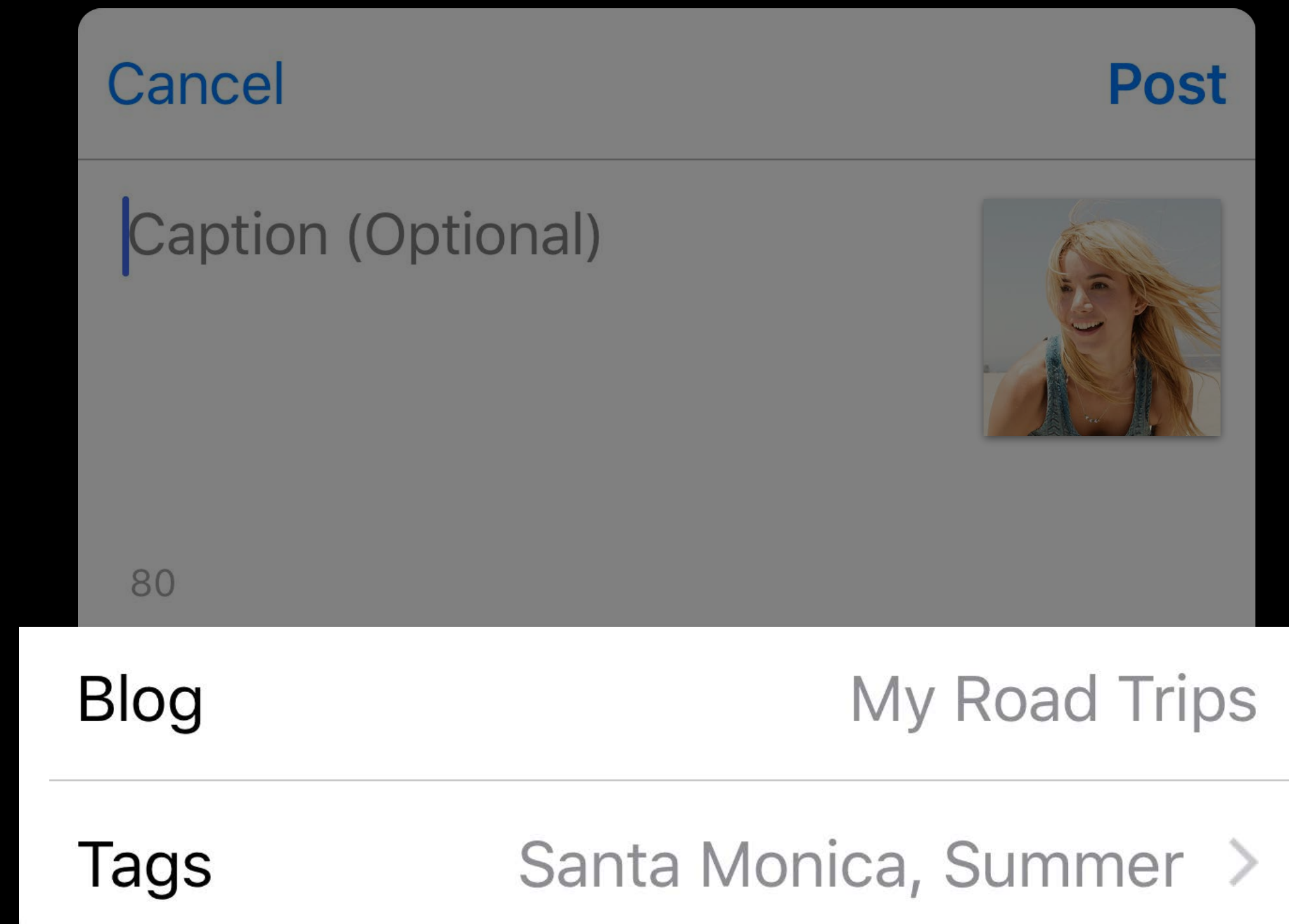
Tags

Santa Monica, Summer >

```
func configurationItems() -> [SLComposeSheetConfigurationItem]
```


Action and Share Extensions

SLComposeServiceViewController



```
func configurationItems() -> [SLComposeSheetConfigurationItem]
```

Action and Share Extensions

SLComposeServiceViewController

```
override func configurationItems() -> [AnyObject]! {  
    let item = SLComposeSheetConfigurationItem()  
  
    item.title = "Blog"  
    item.value = "My Road Trips"  
  
    item.tapHandler = {  
        let viewController = BlogPickerViewController()  
        self.pushConfigurationViewController(viewController)  
    }  
  
    return [ item ]  
}
```

Action and Share Extensions

SLComposeServiceViewController

```
override func configurationItems() -> [AnyObject]! {  
    let item = SLComposeSheetConfigurationItem()  
  
    item.title = "Blog"  
    item.value = "My Road Trips"  
  
    item.tapHandler = {  
        let viewController = BlogPickerViewController()  
        self.pushConfigurationViewController(viewController)  
    }  
  
    return [ item ]  
}
```

Action and Share Extensions

SLComposeServiceViewController

```
override func configurationItems() -> [AnyObject]! {  
    let item = SLComposeSheetConfigurationItem()  
  
    item.title = "Blog"  
    item.value = "My Road Trips"  
  
    item.tapHandler = {  
        let viewController = BlogPickerViewController()  
        self.pushConfigurationViewController(viewController)  
    }  
  
    return [ item ]  
}
```

Action and Share Extensions

SLComposeServiceViewController

```
override func configurationItems() -> [AnyObject]! {  
    let item = SLComposeSheetConfigurationItem()  
  
    item.title = "Blog"  
    item.value = "My Road Trips"  
  
    item.tapHandler = {  
        let viewController = BlogPickerViewController()  
        self.pushConfigurationViewController(viewController)  
    }  
  
    return [ item ]  
}
```


Action and Share Extensions

SLComposeServiceViewController

```
override func configurationItems() -> [AnyObject]! {  
    let item = SLComposeSheetConfigurationItem()  
  
    item.title = "Blog"  
    item.value = "My Road Trips"  
  
    item.tapHandler = {  
        let viewController = BlogPickerViewController()  
        self.pushConfigurationViewController(viewController)  
    }  
  
    return [ item ]  
}
```

Action and Share Extensions

SLComposeServiceViewController

```
override func configurationItems() -> [AnyObject]! {  
    let item = SLComposeSheetConfigurationItem()  
  
    item.title = "Blog"  
    item.value = "My Road Trips"  
  
    item.tapHandler = {  
        let viewController = BlogPickerViewController()  
        self.pushConfigurationViewController(viewController)  
    }  
  
    return [ item ]  
}
```

Action and Share Extensions

SLComposeServiceViewController

```
override func configurationItems() -> [AnyObject]! {  
    let item = SLComposeSheetConfigurationItem()
```

```
    item.title = "Blog"
```

```
    item.value = "My Road Trips"
```

```
    item.tapHandler = {
```

```
        let viewController = BlogPickerViewController()
```

```
        self.pushConfigurationViewController(viewController)
```

```
    }
```

```
    return [ item ]
```

```
}
```

Action and Share Extensions

SLComposeServiceViewController

```
override func configurationItems() -> [AnyObject]! {  
    let item = SLComposeSheetConfigurationItem()  
  
    item.title = "Blog"  
    item.value = "My Road Trips"  
  
    item.tapHandler = {  
        let viewController = BlogPickerViewController()  
        self.pushConfigurationViewController(viewController)  
    }  
  
    return [ item ]  
}
```

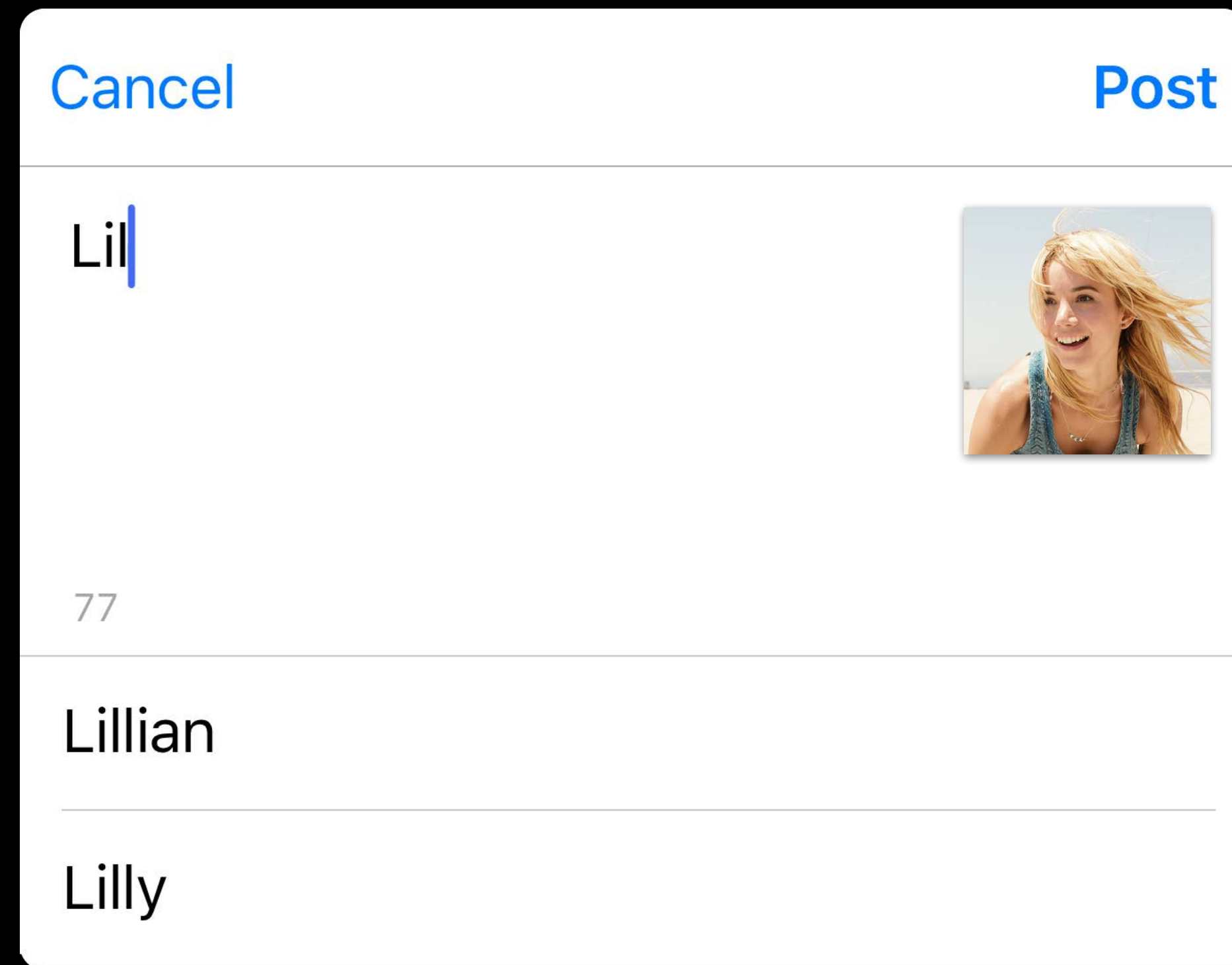
Action and Share Extensions

SLComposeServiceViewController

```
override func configurationItems() -> [AnyObject]! {  
    let item = SLComposeSheetConfigurationItem()  
  
    item.title = "Blog"  
    item.value = "My Road Trips"  
  
    item.tapHandler = {  
        let viewController = BlogPickerViewController()  
        self.pushConfigurationViewController(viewController)  
    }  
  
    return [ item ]  
}
```


Action and Share Extensions

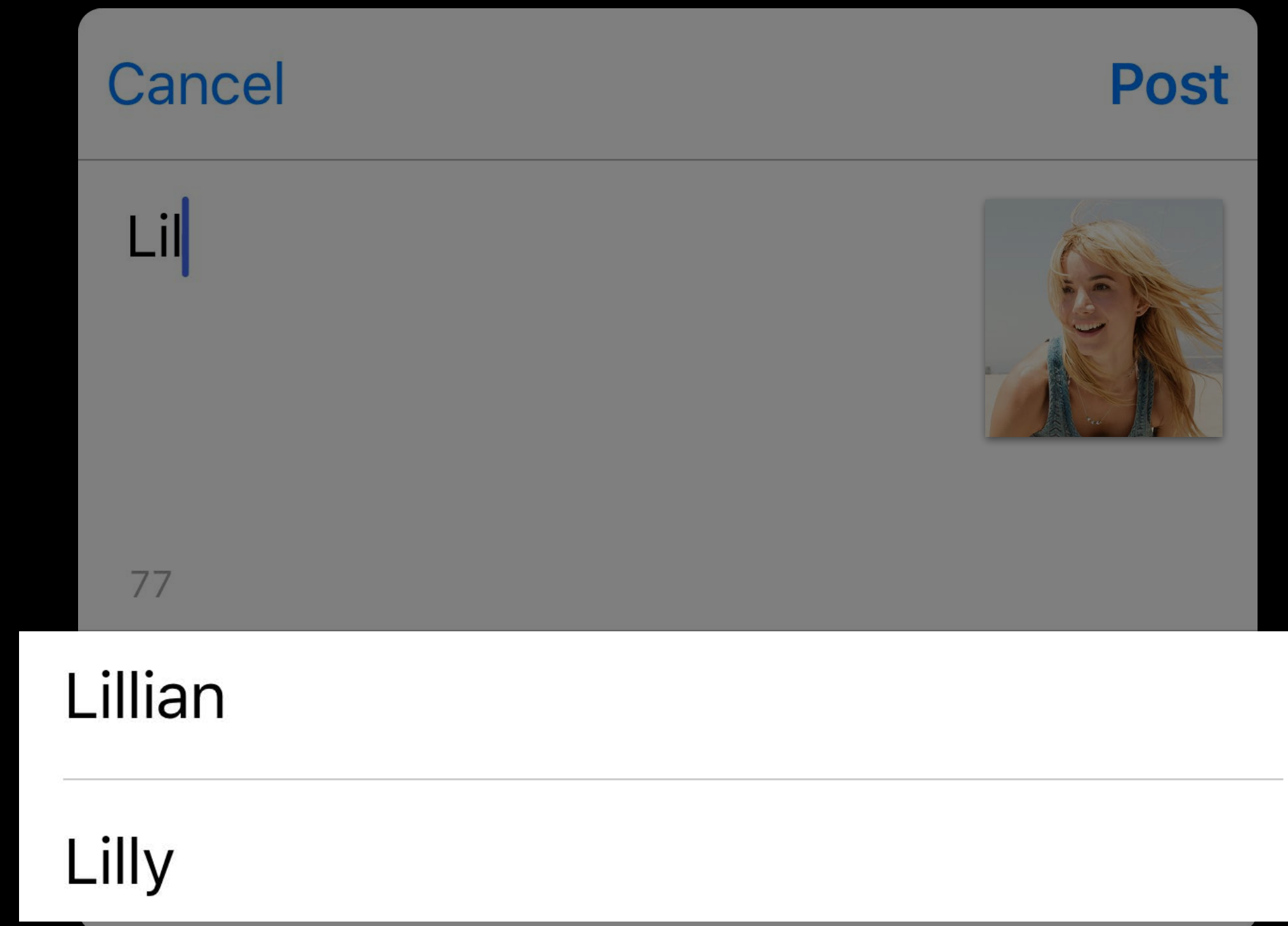
SLComposeServiceViewController



```
viewController.autocompletionViewController = SuggestionViewController(...)
```

Action and Share Extensions

SLComposeServiceViewController



```
viewController.autocompletionViewController = SuggestionViewController(...)
```

Action and Share Extensions

SLComposeServiceViewController

Action and Share Extensions

SLComposeServiceViewController

Provides consistent and familiar UI

Action and Share Extensions

SLComposeServiceViewController

Provides consistent and familiar UI

Customizable

Action and Share Extensions

SLComposeServiceViewController

Provides consistent and familiar UI

Customizable

Completely custom UI using **UIViewController**



Action and Share Extensions

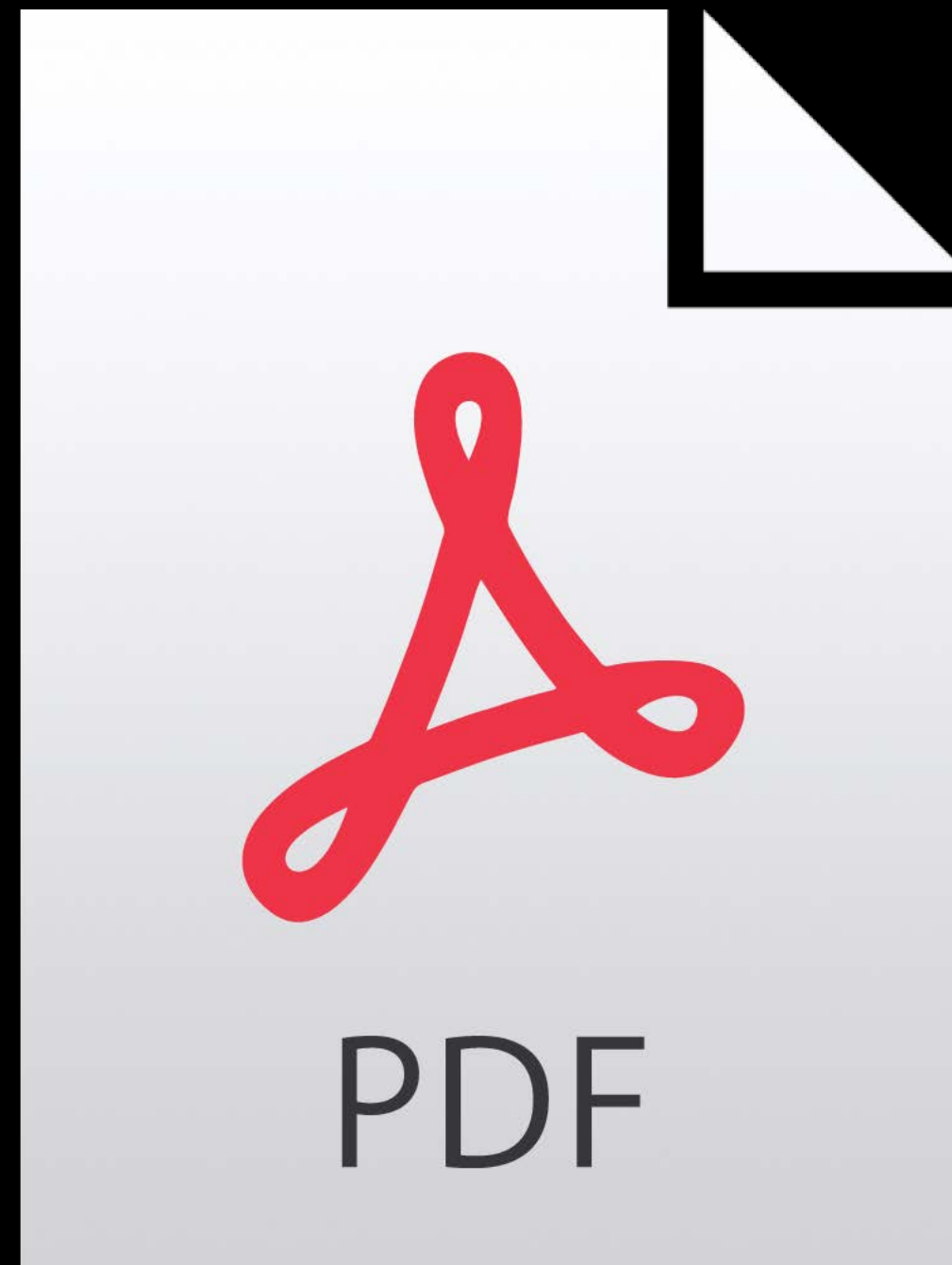
Being a good host



`kUTTypePlainText`

Action and Share Extensions

Being a good host



kUTTypePDF

Action and Share Extensions

Being a good host



kUTTypeHTML

Action and Share Extensions

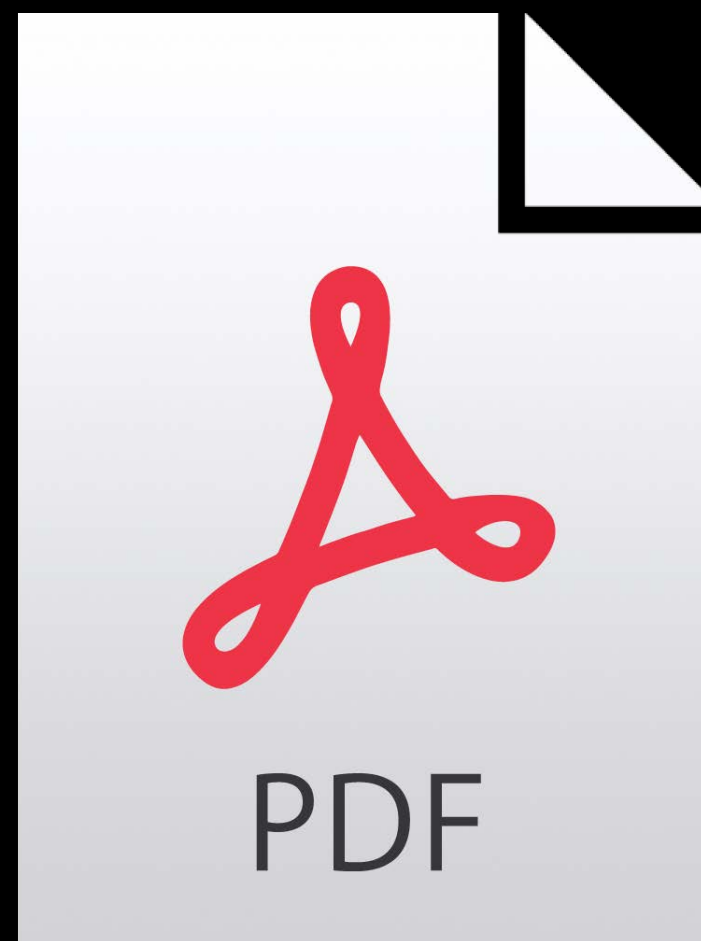
Being a good host

Action and Share Extensions

Being a good host



kUTTypePlainText



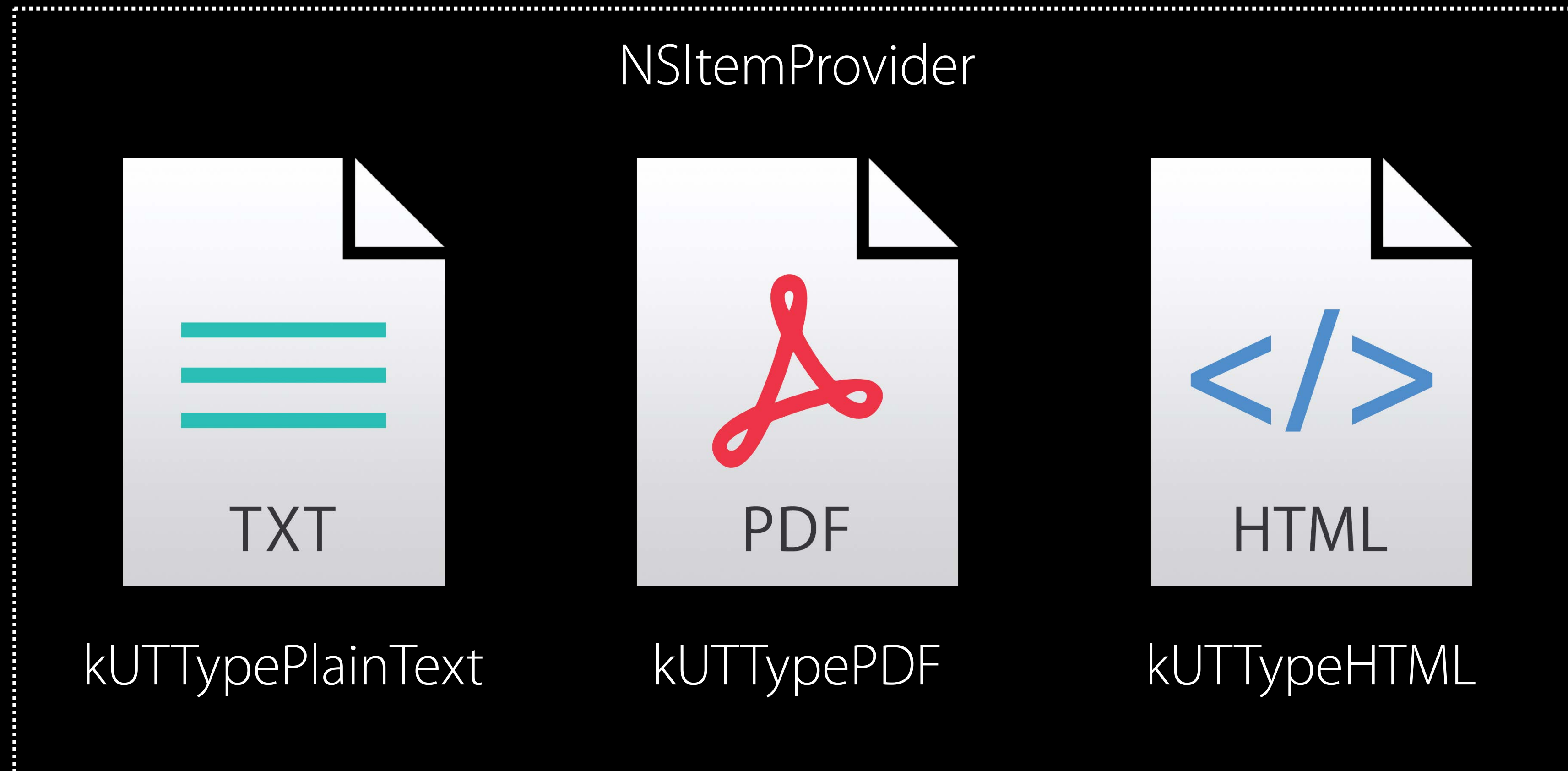
kUTTypePDF



kUTTypeHTML

Action and Share Extensions

Being a good host



Action and Share Extensions

Being a good host

```
let itemProvider = NSItemProvider()

itemProvider.registerItemForTypeIdentifier(kUTTypePlainText as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPlainTextDocumentString(), nil)
}

itemProvider.registerItemForTypeIdentifier(kUTTypePDF as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPDFDocument(), nil)
}

let vc = UIActivityViewController(activityItems: [ itemProvider ],
applicationActivities: nil)
self.presentViewController(vc, animated: true, completion: nil)
```

Action and Share Extensions

Being a good host

```
let itemProvider = NSItemProvider()

itemProvider.registerItemForTypeIdentifier(kUTTypePlainText as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPlainTextDocumentString(), nil)
}

itemProvider.registerItemForTypeIdentifier(kUTTypePDF as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPDFDocument(), nil)
}

let vc = UIActivityViewController(activityItems: [ itemProvider ],
applicationActivities: nil)
self.presentViewController(vc, animated: true, completion: nil)
```

Action and Share Extensions

Being a good host

```
let itemProvider = NSItemProvider()

itemProvider.registerItemForTypeIdentifier(kUTTypePlainText as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPlainTextDocumentString(), nil)
}

itemProvider.registerItemForTypeIdentifier(kUTTypePDF as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPDFDocument(), nil)
}

let vc = UIActivityViewController(activityItems: [ itemProvider ],
applicationActivities: nil)
self.presentViewController(vc, animated: true, completion: nil)
```


Action and Share Extensions

Being a good host

```
let itemProvider = NSItemProvider()

itemProvider.registerItemForTypeIdentifier(kUTTypePlainText as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPlainTextDocumentString(), nil)
}

itemProvider.registerItemForTypeIdentifier(kUTTypePDF as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPDFDocument(), nil)
}

let vc = UIActivityViewController(activityItems: [ itemProvider ],
applicationActivities: nil)
self.presentViewController(vc, animated: true, completion: nil)
```

Action and Share Extensions

Being a good host

```
let itemProvider = NSItemProvider()

itemProvider.registerItemForTypeIdentifier(kUTTypePlainText as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPlainTextDocumentString(), nil)
}

itemProvider.registerItemForTypeIdentifier(kUTTypePDF as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPDFDocument(), nil)
}

let vc = UIActivityViewController(activityItems: [ itemProvider ],
applicationActivities: nil)
self.presentViewController(vc, animated: true, completion: nil)
```

Action and Share Extensions

Being a good host

```
let itemProvider = NSItemProvider()
```

```
itemProvider.registerItemForTypeIdentifier(kUTTypePlainText as String) {  
    completionHandler, expectedClass, options in  
    completionHandler(self.renderPlainTextDocumentString(), nil)  
}
```

```
itemProvider.registerItemForTypeIdentifier(kUTTypePDF as String) {  
    completionHandler, expectedClass, options in  
    completionHandler(self.renderPDFDocument(), nil)  
}
```

```
let vc = UIActivityViewController(activityItems: [ itemProvider ],  
applicationActivities: nil)  
self.presentViewController(vc, animated: true, completion: nil)
```

Action and Share Extensions

Being a good host

```
let itemProvider = NSItemProvider()

itemProvider.registerItemForTypeIdentifier(kUTTypePlainText as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPlainTextDocumentString(), nil)
}

itemProvider.registerItemForTypeIdentifier(kUTTypePDF as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPDFDocument(), nil)
}

let vc = UIActivityViewController(activityItems: [ itemProvider ],
applicationActivities: nil)
self.presentViewController(vc, animated: true, completion: nil)
```

Action and Share Extensions

Being a good host

```
let itemProvider = NSItemProvider()

itemProvider.registerItemForTypeIdentifier(kUTTypePlainText as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPlainTextDocumentString(), nil)
}

itemProvider.registerItemForTypeIdentifier(kUTTypePDF as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPDFDocument(), nil)
}

let vc = UIActivityViewController(activityItems: [ itemProvider ],
applicationActivities: nil)
self.presentViewController(vc, animated: true, completion: nil)
```

Action and Share Extensions

Being a good host

```
let itemProvider = NSItemProvider()

itemProvider.registerItemForTypeIdentifier(kUTTypePlainText as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPlainTextDocumentString(), nil)
}
```

```
itemProvider.registerItemForTypeIdentifier(kUTTypePDF as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPDFDocument(), nil)
}
```

```
let vc = UIActivityViewController(activityItems: [ itemProvider ],
applicationActivities: nil)
self.presentViewController(vc, animated: true, completion: nil)
```


Action and Share Extensions

Being a good host

```
let itemProvider = NSItemProvider()

itemProvider.registerItemForTypeIdentifier(kUTTypePlainText as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPlainTextDocumentString(), nil)
}

itemProvider.registerItemForTypeIdentifier(kUTTypePDF as String) {
    completionHandler, expectedClass, options in
    completionHandler(self.renderPDFDocument(), nil)
}

let vc = UIActivityViewController(activityItems: [ itemProvider ],
applicationActivities: nil)
self.presentViewController(vc, animated: true, completion: nil)
```

Action and Share Extensions

Being a good host

Action and Share Extensions

Being a good host

Host apps should offer content previews

Action and Share Extensions

Being a good host

Host apps should offer content previews

- Represents what will be shared

Action and Share Extensions

Being a good host

Host apps should offer content previews

- Represents what will be shared
- Simple and efficient representation of the content

Action and Share Extensions

Being a good host

```
itemProvider.previewImageHandler = {  
    completionHandler, expectedClass, options in  
    completionHandler(self.renderThumbnail(), nil)  
}
```


Action and Share Extensions

Being a good host

```
itemProvider.previewImageHandler = {  
    completionHandler, expectedClass, options in  
    completionHandler(self.renderThumbnail(), nil)  
}
```

Action and Share Extensions

Being a good host

```
itemProvider.previewImageHandler = {  
    completionHandler, expectedClass, options in  
    completionHandler(self.renderThumbnail(), nil)  
}
```

Action and Share Extensions

Being a good host

```
itemProvider.previewImageHandler = {  
    completionHandler, expectedClass, options in  
    completionHandler(self.renderThumbnail(), nil)  
}
```

Action and Share Extensions

Activation rules

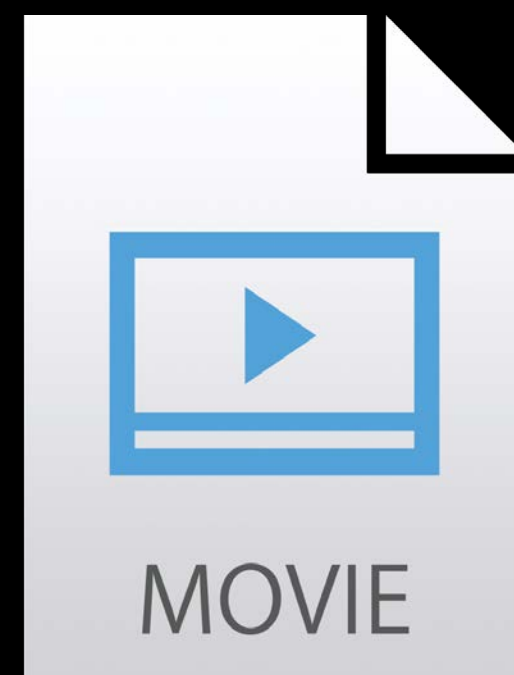
Action and Share Extensions

Activation rules

Host App



kUTTypeImage



kUTTypeMovie

Action and Share Extensions

Activation rules

Host App



kUTTypeImage



kUTTypeMovie

NSExtensionActivationSupportsImage

Extension

NSExtensionActivationSupportsMovie

Extension

NSExtensionActivationSupportsImage

Extension

NSExtensionActivationSupportsMovie

Action and Share Extensions

Activation rules

Host App



kUTTypeImage

NSExtensionActivationSupportsImage

Extension

NSExtensionActivationSupportsMovie

Extension



kUTTypeMovie

NSExtensionActivationSupportsImage

Extension

NSExtensionActivationSupportsMovie

Action and Share Extensions

Activation rules

Host App



kUTTypeImage



kUTTypeMovie

NSExtensionActivationSupportsImage

Extension

NSExtensionActivationSupportsMovie

Extension

NSExtensionActivationSupportsImage

Extension

NSExtensionActivationSupportsMovie

Action and Share Extensions

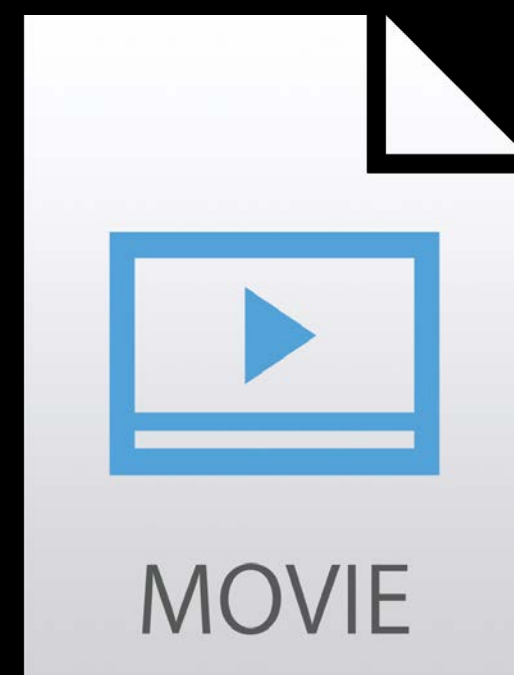


Activation rules

Host App



kUTTypeImage



kUTTypeMovie

NSExtensionActivationSupportsImage

Extension

NSExtensionActivationSupportsMovie

Extension

NSExtensionActivationSupportsImage

NSExtensionActivationSupportsMovie

Extension

Action and Share Extensions

NEW

Activation rules

▼ NSExtension	↕	Dictionary	(3 items)
▼ NSExtensionAttributes		Dictionary	(1 item)
▼ NSExtensionActivationRule		Dictionary	(3 items)
NSExtensionActivationDictionaryVersion		Number	2
NSExtensionActivationSupportsImageWithMaxCount		Number	1
NSExtensionActivationSupportsMovieWithMaxCount		Number	0
NSExtensionPointIdentifier		String	com.apple.services
NSExtensionPrincipalClass		String	ActionRequestHandler

Action and Share Extensions

NEW

Activation rules

▼ NSExtension	⬆	Dictionary	(3 items)
▼ NSExtensionAttributes		Dictionary	(1 item)
▼ NSExtensionActivationRule		Dictionary	(3 items)
NSExtensionActivationDictionaryVersion		Number	2
NSExtensionActivationSupportsImageWithMaxCount		Number	1
NSExtensionActivationSupportsMovieWithMaxCount		Number	0
NSExtensionPointIdentifier		String	com.apple.services
NSExtensionPrincipalClass		String	ActionRequestHandler

Action and Share Extensions

Activation rules



Action and Share Extensions

Activation rules



Host App



kUTTypeImage



kUTTypeMovie

NSExtensionActivationSupportsImage

Extension

NSExtensionActivationSupportsMovie

Extension

NSExtensionActivationSupportsImage

Extension

NSExtensionActivationSupportsMovie

Action and Share Extensions

Activation rules

▼ NSExtension	⬆	Dictionary	(3 items)
▼ NSExtensionAttributes		Dictionary	(1 item)
NSExtensionActivationRule		String	SUBQUERY(extensionItems, \$extensionItem, SUBQUERY (\$extensionItem.attachments,
NSExtensionPointIdentifier	+ -	String	com.apple.services
NSExtensionPrincipalClass		String	ActionRequestHandler

```
SUBQUERY(extensionItems, $extensionItem,  
SUBQUERY($extensionItem.attachments, $attachment, ANY  
$attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.image").@count  
== 1).@count == 1 OR SUBQUERY(extensionItems, $extensionItem,  
SUBQUERY($extensionItem.attachments, $attachment, ANY  
$attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "com.adobe.pdf").@count  
== 1).@count == 1
```

Action and Share Extensions

Activation rules

▼ NSExtension	⬆	Dictionary	(3 items)
▼ NSExtensionAttributes		Dictionary	(1 item)
NSExtensionActivationRule		String	SUBQUERY(extensionItems, \$extensionItem, SUBQUERY (\$extensionItem.attachments,
NSExtensionPointIdentifier	+ -	String	com.apple.services
NSExtensionPrincipalClass		String	ActionRequestHandler

```
SUBQUERY(extensionItems, $extensionItem,  
SUBQUERY($extensionItem.attachments, $attachment, ANY  
$attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.image").@count  
== 1).@count == 1 OR SUBQUERY(extensionItems, $extensionItem,  
SUBQUERY($extensionItem.attachments, $attachment, ANY  
$attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "com.adobe.pdf").@count  
== 1).@count == 1
```

Action and Share Extensions

Icons

Action and Share Extensions

Icons

Share Extensions use the containing app icon

Action and Share Extensions

Icons

Share Extensions use the containing app icon

- No additional work needed

Action and Share Extensions

Icons

Action and Share Extensions

Icons

Action Extensions require template images

Action and Share Extensions

Icons

Action Extensions require template images

- Two sizes for iPhone and iPad

Action and Share Extensions

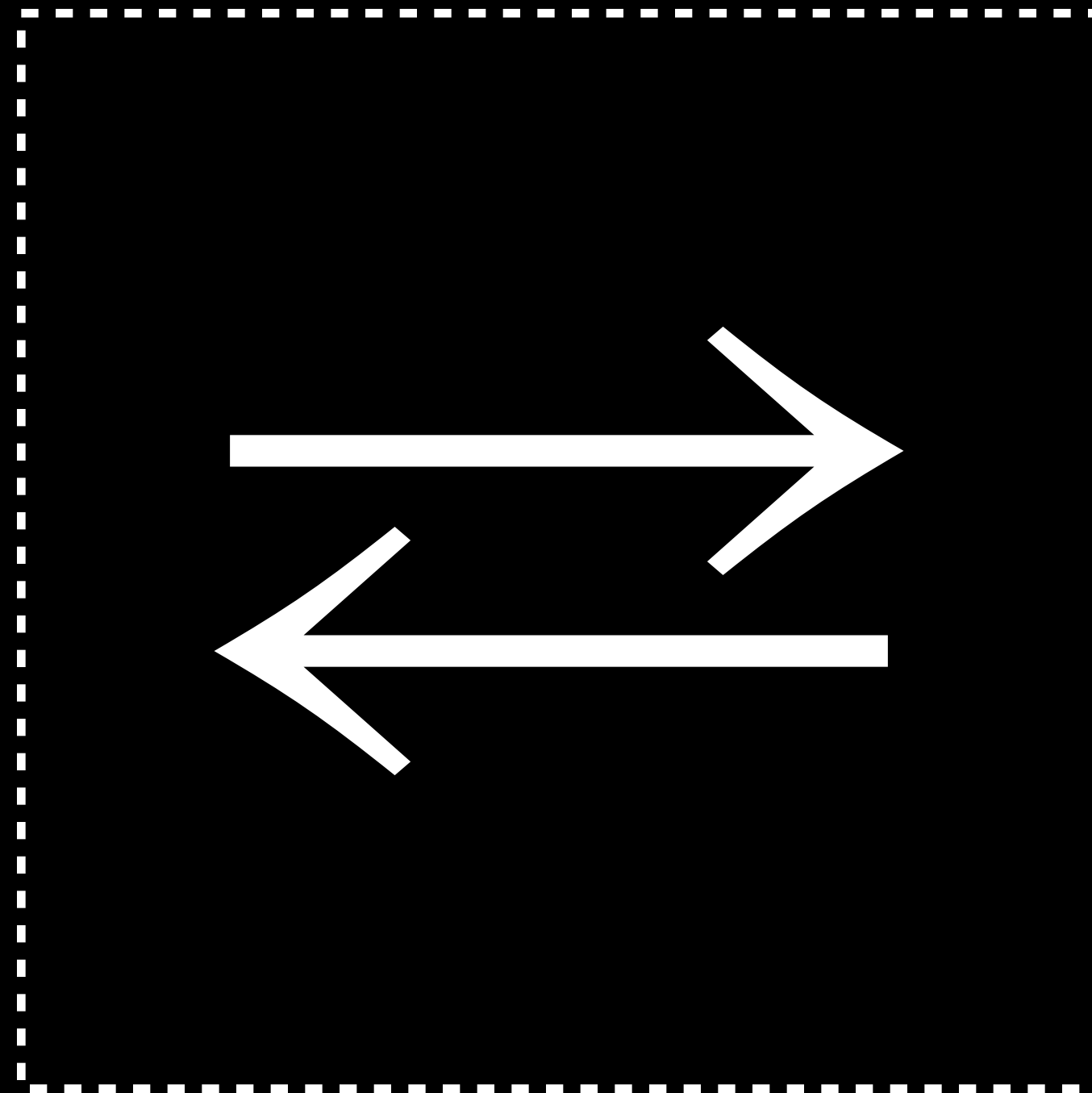
Icons

Action Extensions require template images

- Two sizes for iPhone and iPad
- Reside in the extension bundle

Action and Share Extensions

Icons



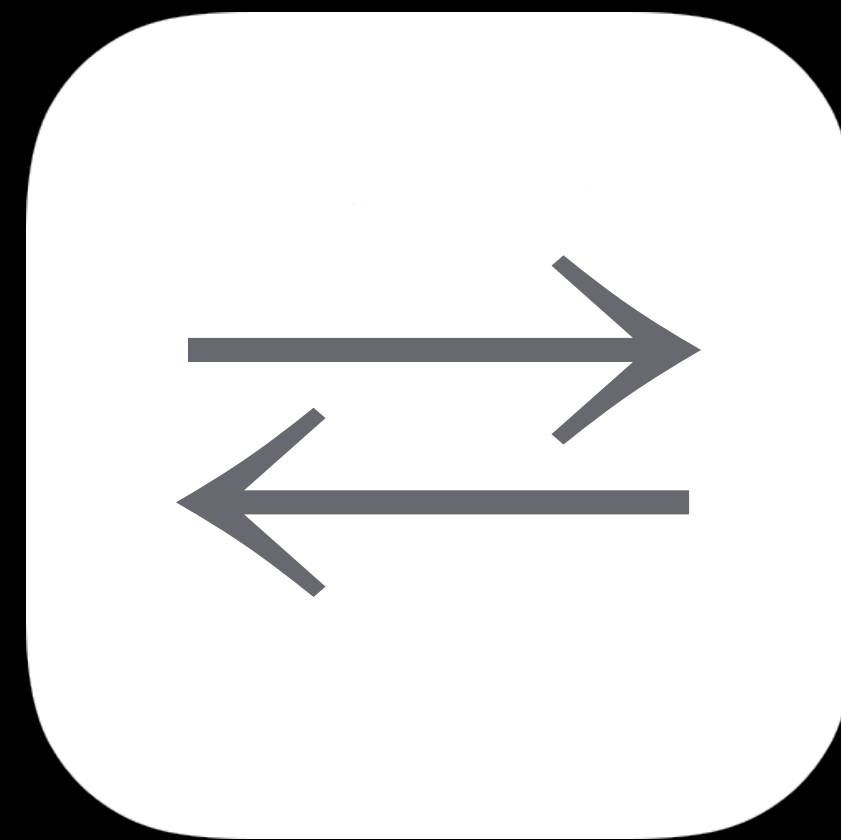
Action and Share Extensions

Action icons

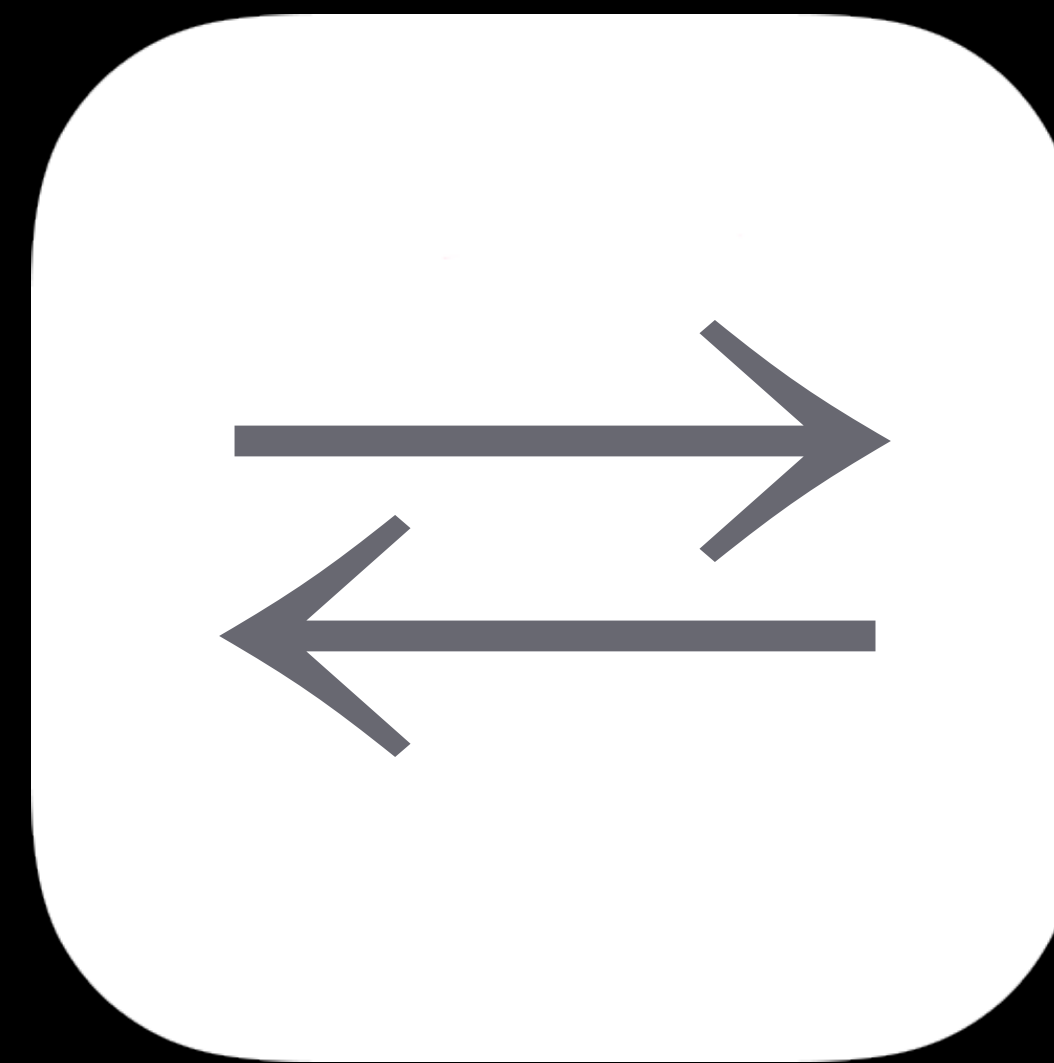


Action and Share Extensions

Action icons



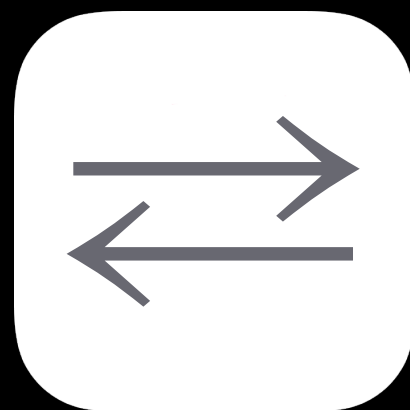
60pt



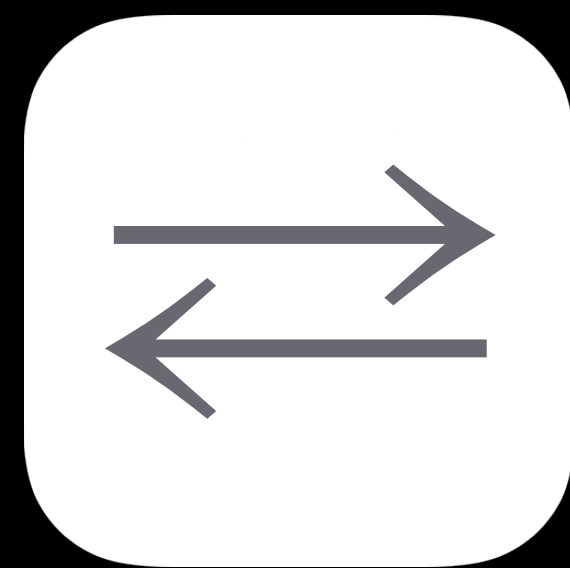
76pt

Action and Share Extensions

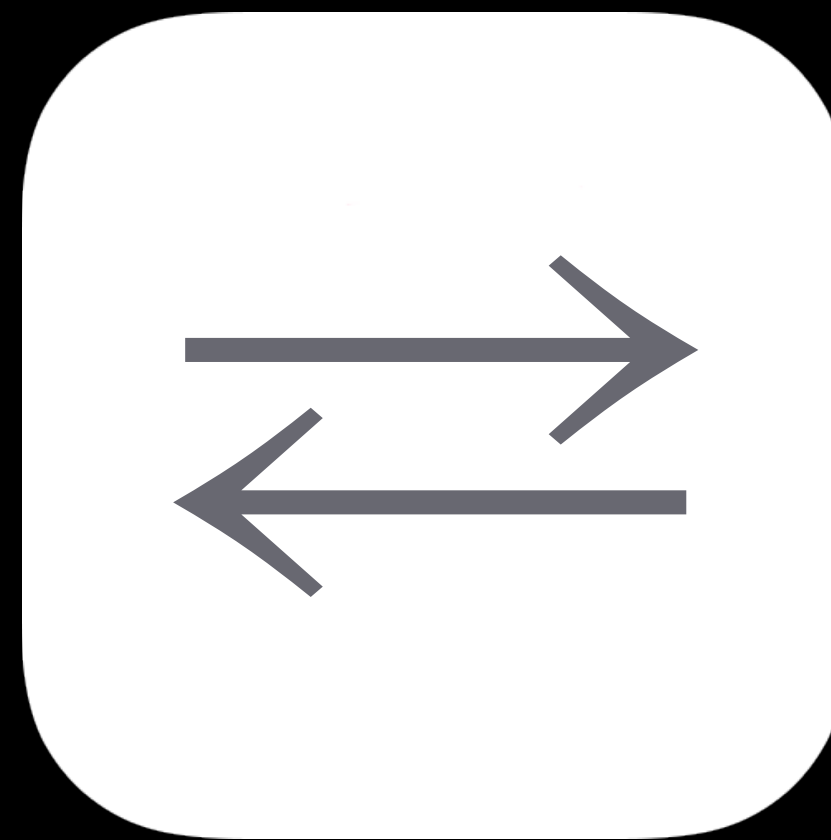
Action icons



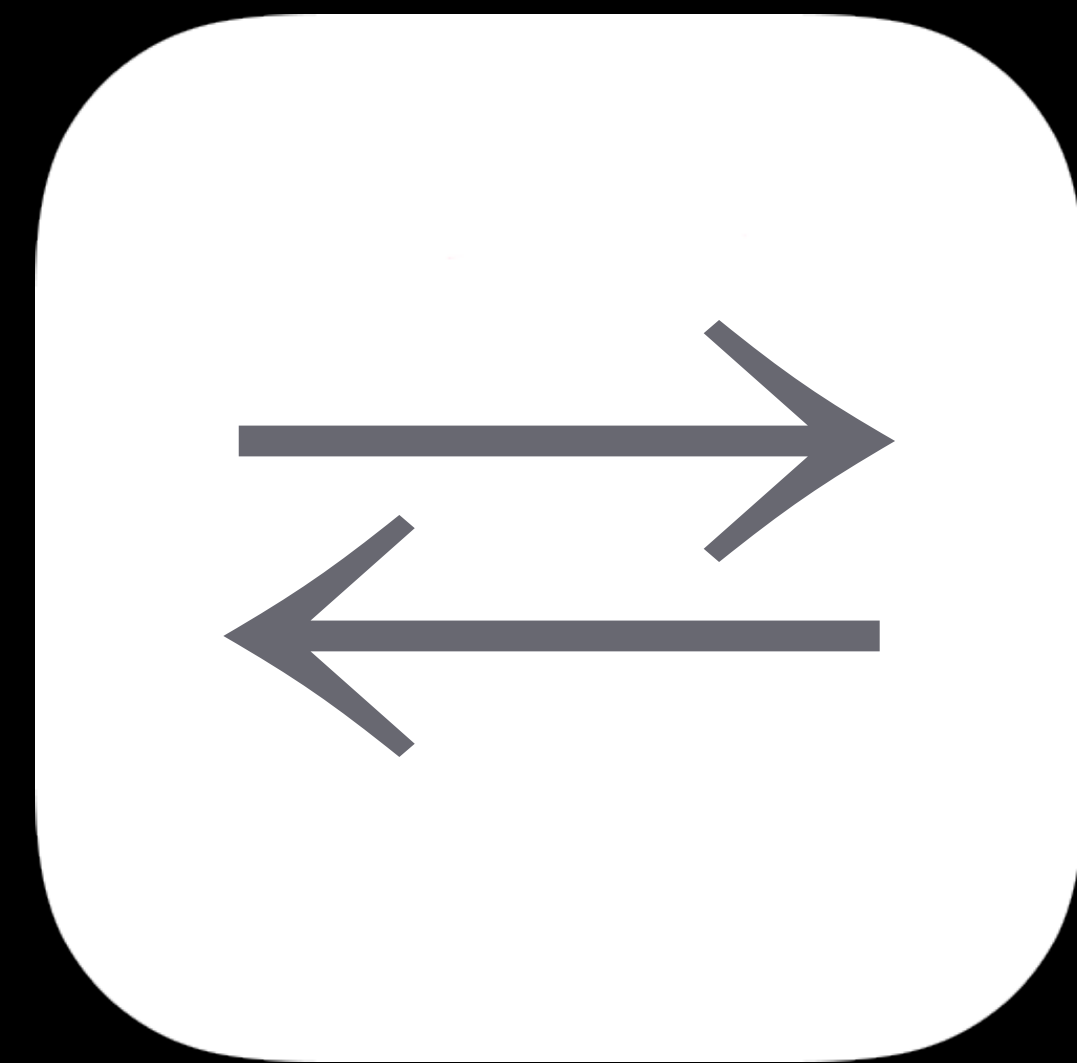
29pt



40pt



60pt



76pt

Today Widget Enhancements

Ian Baird CoreOS Engineer

Today Widget Enhancements

Quick information at a glance

Today Widget Enhancements

Quick information at a glance

Enhance your Today widget

Today Widget Enhancements

Quick information at a glance

Enhance your Today widget

General best practices

Wednesday,
June 3rd



Partly cloudy currently. The high will be 70°. Cloudy tonight with a low of 54°.

The first thing on your calendar today is "Coffee with Sophia", in 17 minutes.



Calendar

3:43 PM

4 PM

Coffee with Sophia

Moscone West

800 Howard St, San Francisco, CA 94103-3010, United States

5 PM



Stocks

NASDAQ

5,099.23

+ 22.71

NYSE

11,107.93

+ 27.04

DOW J

18,076.27

+ 64.33

AAPL

130.12

 $+ 0.16$



The first thing on your calendar today is "Coffee with Sophia", in 17 minutes.



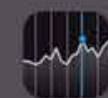
Calendar

3:43 PM

4 PM

Moscone West

5 PM



Stocks

+ 22.71

+ 27.04

+ 64.33

 $+ 0.16$

NASDAQ	5,099.23	+ 22.71
NYSE	11,107.93	+ 27.04
DOW J	18,076.27	+64.33
AAPL	130.12	+ 0.16
SBUX	52.12	+ 0.39
NKE	102.51	+ 0.40
YHOO	43.21	+ 0.06

NASDAQ Composite			
OPEN	5,098.48	MKT CAP	—
HIGH	5,114.60	52W HIGH	5,119.83
LOW	5,084.99	52W LOW	4,116.60
VOL	—	AVG VOL	1.791B
P/E	—	YIELD	—

Working with the Containing App

URL handlers

Interactions use URLs to take user to app

Working with the Containing App

URL handlers

Interactions use URLs to take user to app

Best practices

Working with the Containing App

URL handlers

Interactions use URLs to take user to app

Best practices

- Use your app's registered URL schemes



Working with the Containing App

URL handlers

Interactions use URLs to take user to app

Best practices

- Use your app's registered URL schemes
- System URL schemes



Working with the Containing App

URL handlers

Interactions use URLs to take user to app.

Best practices

- Use your app's registered URL schemes
- System URL schemes

```
func tableView(..., didSelectRowAtIndexPath indexPath: NSIndexPath) {  
    let selectedName = names[indexPath.row]  
    let URL = NSURL(string: "my-app://name=\(selectedName)")!  
    extensionContext!.openURL(URL) { success in  
        print("openURL returned: \(success)")  
    }  
}
```


Working with the Containing App

URL handlers

Interactions use URLs to take user to app.

Best practices

- Use your app's registered URL schemes
- System URL schemes

```
func tableView(..., didSelectRowAtIndexPath indexPath: NSIndexPath) {  
    let selectedName = names[indexPath.row]  
    let URL = NSURL(string: "my-app://name=\(selectedName)")!  
    extensionContext!.openURL(URL) { success in  
        print("openURL returned: \(success)")  
    }  
}
```


Working with the Containing App

URL handlers

Interactions use URLs to take user to app.

Best practices

- Use your app's registered URL schemes
- System URL schemes

```
func tableView(..., didSelectRowAtIndexPath indexPath: NSIndexPath) {  
    let selectedName = names[indexPath.row]  
    let URL = NSURL(string: "my-app://name=\(selectedName)")!  
    extensionContext!.openURL(URL) { success in  
        print("openURL returned: \(success)")  
    }  
}
```

Working with the Containing App

URL handlers

Interactions use URLs to take user to app.

Best practices

- Use your app's registered URL schemes
- System URL schemes

```
func tableView(..., didSelectRowAtIndexPath indexPath: NSIndexPath) {  
    let selectedName = names[indexPath.row]  
    let URL = NSURL(string: "my-app://name=\(selectedName)")!  
    extensionContext!.openURL(URL) { success in  
        print("openURL returned: \(success)")  
    }  
}
```

Working with the Containing App

App groups

Defaults

Working with the Containing App

App groups

Defaults

Containers

Working with the Containing App

App groups

Defaults

Containers

Keychain items

Working with the Containing App

App groups

Defaults

Containers

Keychain items

Framework data

Working with the Containing App

User defaults

Working with the Containing App

User defaults

Used for small pieces of configuration data

Working with the Containing App

User defaults

Used for small pieces of configuration data

```
let defaults = UserDefaults.initWithSuiteName("group.com.example.my-app")
```

Working with the Containing App

What goes in a shared container?

Model data

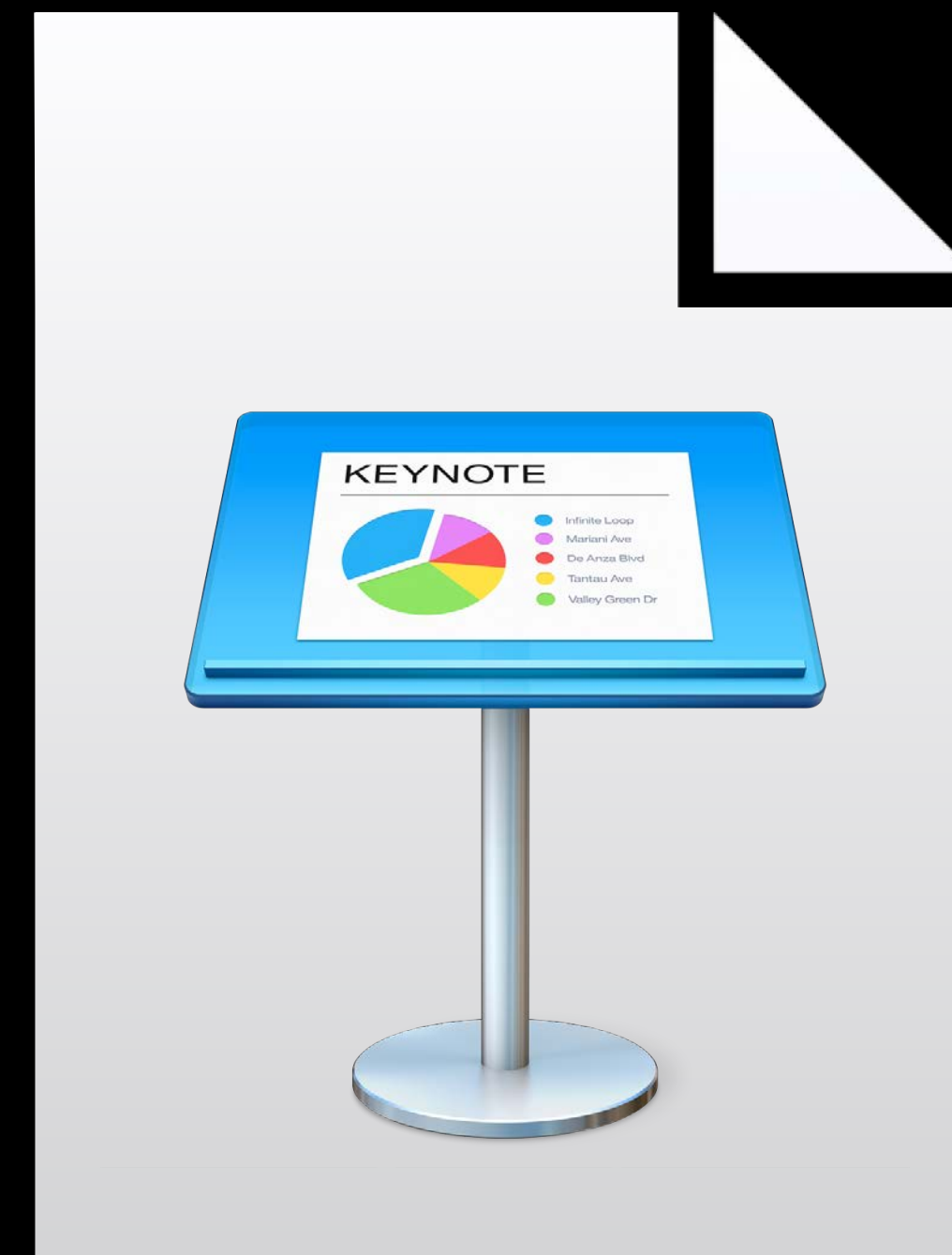


Working with the Containing App

What goes in a shared container?

Model data

Documents



Working with the Containing App

What goes in a shared container?

Model data

Documents

Media



Working with the Containing App

Working with model data in the shared container

```
lazy var secureAppGroupPersistentStoreURL: NSURL = {  
    let fm = NSFileManager.defaultManager()  
    let directory =  
fm.containerURLForSecurityApplicationGroupIdentifier("group.com.example.my-  
app")!  
    return directory.URLByAppendingPathComponent("myFileName.sqlite")  
}()
```

Working with the Containing App

Working with model data in the shared container

```
lazy var secureAppGroupPersistentStoreURL: NSURL = {  
    let fm = NSFileManager defaultManager()  
    let directory =  
fm.containerURLForSecurityApplicationGroupIdentifier("group.com.example.my-  
app")!  
    return directory.URLByAppendingPathComponent("myFileName.sqlite")  
}()
```


Working with the Containing App

Working with model data in the shared container

```
lazy var secureAppGroupPersistentStoreURL: NSURL = {  
    let fm = NSFileManager.defaultManager()  
    let directory =  
fm.containerURLForSecurityApplicationGroupIdentifier("group.com.example.my-  
app")!  
    return directory.URLByAppendingPathComponent("myFileName.sqlite")  
}()
```

Working with the Containing App

Working with model data in the shared container

```
lazy var secureAppGroupPersistentStoreURL: NSURL = {  
    let fm = NSFileManager.defaultManager()  
    let directory =  
fm.containerURLForSecurityApplicationGroupIdentifier("group.com.example.my-  
app")!  
    return directory.URLByAppendingPathComponent("myFileName.sqlite")  
}()
```

Working with the Containing App

Working with model data in the shared container

```
lazy var secureAppGroupPersistentStoreURL: NSURL = {  
    let fm = NSFileManager.defaultManager()  
    let directory =  
fm.containerURLForSecurityApplicationGroupIdentifier("group.com.example.my-  
app")!  
    return directory.URLByAppendingPathComponent("myFileName.sqlite")  
}()
```

Working with the Containing App

Working with model data in the shared container

```
lazy var persistentStoreCoordinator: NSPersistentStoreCoordinator = {  
    let storeCoordinator = NSPersistentStoreCoordinator(managedObjectModel:  
self.managedObjectModel)  
    let storeURL = self.secureAppGroupPersistentStoreURL  
    do {  
        try storeCoordinator.addPersistentStoreWithType(NSSQLiteStoreType,  
configuration: nil, URL: storeURL, options:  
[NSMigratePersistentStoresAutomaticallyOption: true,  
NSInferMappingModelAutomaticallyOption, true])  
    } catch let error as NSError {  
        // handle error  
    }  
    return storeCoordinator  
}()
```

Working with the Containing App

Working with model data in the shared container

```
lazy var persistentStoreCoordinator: NSPersistentStoreCoordinator = {
    let storeCoordinator = NSPersistentStoreCoordinator(managedObjectModel:
self.managedObjectModel)
    let storeURL = self.secureAppGroupPersistentStoreURL
    do {
        try storeCoordinator.addPersistentStoreWithType(NSSQLiteStoreType,
configuration: nil, URL: storeURL, options:
[NSMigratePersistentStoresAutomaticallyOption: true,
NSInferMappingModelAutomaticallyOption, true])
    } catch let error as NSError {
        // handle error
    }
    return storeCoordinator
}()
```

Working with the Containing App

Working with model data in the shared container

```
lazy var persistentStoreCoordinator: NSPersistentStoreCoordinator = {  
    let storeCoordinator = NSPersistentStoreCoordinator(managedObjectModel:  
self.managedObjectModel)  
    let storeURL = self.secureAppGroupPersistentStoreURL  
    do {  
        try storeCoordinator.addPersistentStoreWithType(NSSQLiteStoreType,  
configuration: nil, URL: storeURL, options:  
[NSMigratePersistentStoresAutomaticallyOption: true,  
NSInferMappingModelAutomaticallyOption, true])  
    } catch let error as NSError {  
        // handle error  
    }  
    return storeCoordinator  
}()
```

Working with the Containing App

Working with model data in the shared container

```
lazy var persistentStoreCoordinator: NSPersistentStoreCoordinator = {  
    let storeCoordinator = NSPersistentStoreCoordinator(managedObjectModel:  
self.managedObjectModel)  
    let storeURL = self.secureAppGroupPersistentStoreURL  
    do {  
        try storeCoordinator.addPersistentStoreWithType(NSSQLiteStoreType,  
configuration: nil, URL: storeURL, options:  
[NSMigratePersistentStoresAutomaticallyOption: true,  
NSInferMappingModelAutomaticallyOption, true])  
    } catch let error as NSError {  
        // handle error  
    }  
    return storeCoordinator  
}()
```


Working with the Containing App

Working with model data in the shared container

```
lazy var persistentStoreCoordinator: NSPersistentStoreCoordinator = {
    let storeCoordinator = NSPersistentStoreCoordinator(managedObjectModel:
self.managedObjectModel)
    let storeURL = self.secureAppGroupPersistentStoreURL
    do {
        try storeCoordinator.addPersistentStoreWithType(NSSQLiteStoreType,
configuration: nil, URL: storeURL, options:
[NSMigratePersistentStoresAutomaticallyOption: true,
NSInferMappingModelAutomaticallyOption, true])
    } catch let error as NSError {
        // handle error
    }
    return storeCoordinator
}()
```

Working with the Containing App

Working with model data in the shared container

```
lazy var persistentStoreCoordinator: NSPersistentStoreCoordinator = {
    let storeCoordinator = NSPersistentStoreCoordinator(managedObjectModel:
self.managedObjectModel)
    let storeURL = self.secureAppGroupPersistentStoreURL
    do {
        try storeCoordinator.addPersistentStoreWithType(NSSQLiteStoreType,
configuration: nil, URL: storeURL, options:
[NSMigratePersistentStoresAutomaticallyOption: true,
NSInferMappingModelAutomaticallyOption, true])
    } catch let error as NSError {
        // handle error
    }
    return storeCoordinator
}()
```

Working with the Containing App

Working with model data in the shared container

```
lazy var managedObjectContext: NSManagedObjectContext = {  
    let coordinator = self.persistentStoreCoordinator  
    var managedObjectContext =  
NSManagedObjectContext(concurrencyType: .MainQueueConcurrencyType)  
    managedObjectContext.persistentStoreCoordinator = coordinator  
    return managedObjectContext  
}()
```

Working with the Containing App

Working with model data in the shared container

```
lazy var managedObjectContext: NSManagedObjectContext = {  
    let coordinator = self.persistentStoreCoordinator  
    var managedObjectContext =  
NSManagedObjectContext(concurrencyType: .MainQueueConcurrencyType)  
    managedObjectContext.persistentStoreCoordinator = coordinator  
    return managedObjectContext  
}()
```

Working with the Containing App

Working with model data in the shared container

```
lazy var managedObjectContext: NSManagedObjectContext = {  
    let coordinator = self.persistentStoreCoordinator  
    var managedObjectContext =  
NSManagedObjectContext(concurrencyType: .MainQueueConcurrencyType)  
    managedObjectContext.persistentStoreCoordinator = coordinator  
    return managedObjectContext  
}()
```

Working with the Containing App

Working with model data in the shared container

```
lazy var managedObjectContext: NSManagedObjectContext = {  
    let coordinator = self.persistentStoreCoordinator  
    var managedObjectContext =  
NSManagedObjectContext(concurrencyType: .MainQueueConcurrencyType)  
    managedObjectContext.persistentStoreCoordinator = coordinator  
    return managedObjectContext  
}()
```

Staying Responsive

Locking in the shared container



Staying Responsive

Locking in the shared container

Be careful taking exclusive locks for data
in the shared container



Staying Responsive

Locking in the shared container

Be careful taking exclusive locks for data in the shared container

Extension killed if it's suspended while holding an exclusive lock



Staying Responsive

Task assertions

Extensions suspended when no longer in use

Staying Responsive

Task assertions

Extensions suspended when no longer in use

Protect serialization and other clean-up tasks with background task assertions

Staying Responsive

Task assertions

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        self.serializeData()
    } else {
        self.cleanupSerializingData()
    }
}
```

Staying Responsive

Task assertions

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        self.serializeData()
    } else {
        self.cleanupSerializingData()
    }
}
```

Staying Responsive

Task assertions

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        self.serializeData()
    } else {
        self.cleanupSerializingData()
    }
}
```

Staying Responsive

Task assertions

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        self.serializeData()
    } else {
        self.cleanupSerializingData()
    }
}
```

Staying Responsive

Task assertions

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        self.serializeData()
    } else {
        self.cleanupSerializingData()
    }
}
```

expired == false

Staying Responsive

Task assertions

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        self.serializeData()
    } else {
        self.cleanupSerializingData()
    }
}
```

expired == false → perform protected task

Staying Responsive

Task assertions

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        self.serializeData()
    } else {
        self.cleanupSerializingData()
    }
}
```



expired == true

Staying Responsive

Task assertions

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        self.serializeData()
    } else {
        self.cleanupSerializingData()
    }
}
```



expired == true -> cancel protected task

Staying Responsive

Task assertions

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        self.serializeData()
    } else {
        self.cleanupSerializingData()
    }
}
```

exit the block -> task assertion is released

Staying Responsive

Task assertions

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        self.serializeData()
    } else {
        self.cleanupSerializingData()
    }
}
```

expired == true

Staying Responsive

Task assertions

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        self.serializeData()
    } else {
        self.cleanupSerializingData()
    }
}
```

expired == true -> cancel any work and exit the block

Staying Responsive

Task assertions

Released when your code exits the block

Staying Responsive

Task assertions

Released when your code exits the block

Re-entrant execution of callback for expiration

Staying Responsive

Task assertions

Released when your code exits the block

Re-entrant execution of callback for expiration

Task assertions are **not** always available for your extension

Staying Responsive

Working with other queues

Task assertion scoped to callback block

Staying Responsive

Working with other queues

Task assertion scoped to callback block

Callback block must synchronize with protected work on other queues

Staying Responsive

Working with other queues

Task assertion scoped to callback block

Callback block must synchronize with protected work on other queues

Example

Staying Responsive

Working with other queues

Task assertion scoped to callback block

Callback block must synchronize with protected work on other queues

Example

- Dispatch block to the main queue

Staying Responsive

Working with the main queue

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        dispatch_async(dispatch_get_main_queue(), {
            self.performInterruptibleWork()
        })
    } else {
        self.canceled = true
    }
}
```

Staying Responsive

Working with the main queue



```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        dispatch_async(dispatch_get_main_queue(), {
            self.performInterruptibleWork()
        })
    } else {
        self.canceled = true
    }
}
```

Staying Responsive

Working with the main queue

```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        dispatch_sync(dispatch_get_main_queue(), {
            self.performInterruptibleWork()
        })
    } else {
        self.canceled = true
    }
}
```


Staying Responsive

Working with the main queue



```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        dispatch_sync(dispatch_get_main_queue(), {
            self.performInterruptibleWork()
        })
    } else {
        self.canceled = true
    }
}
```

Staying Responsive

Working with the main queue



```
let pi = NSProcessInfo.processInfo()
pi.performExpiringActivityWithReason("clean-up") { expired in
    if (!expired) {
        dispatch_sync(dispatch_get_main_queue(), {
            self.performInterruptibleWork()
        })
    } else {
        self.canceled = true
    }
}
```

Callback block is executing on a private system queue

Staying Up to Date

Darwin Notification Center

Similar to NotificationCenter

Staying Up to Date

Darwin Notification Center

Similar to NotificationCenter



Staying Up to Date

Darwin Notification Center

Similar to NotificationCenter

Small number of use-cases



Staying Up to Date

Darwin Notification Center

Similar to NotificationCenter

Small number of use-cases

Example



Staying Up to Date

Darwin Notification Center

Similar to NotificationCenter

Small number of use-cases

Example

- Hint to your extension to reload the model



Staying Up to Date

Containing app API

```
let nc = CFNotificationCenterGetDarwinNotifyCenter()  
CFNotificationCenterPostNotification(nc,  
    "com.example.app-model-updated",  
    nil,  
    nil,  
    CFBooleanGetValue(true))
```


Staying Up to Date

Containing app API

```
let nc = CFNotificationCenterGetDarwinNotifyCenter()  
CFNotificationCenterPostNotification(nc,  
    "com.example.app-model-updated",  
    nil,  
    nil,  
    CFBooleanGetValue(true))
```

Staying Up to Date

Containing app API

```
let nc = CFNotificationCenterGetDarwinNotifyCenter()  
CFNotificationCenterPostNotification(nc,  
    "com.example.app-model-updated",  
    nil,  
    nil,  
    CFBooleanGetValue(true))
```

Staying Up to Date

Containing app API

```
let nc = CFNotificationCenterGetDarwinNotifyCenter()  
CFNotificationCenterPostNotification(nc,  
    "com.example.app-model-updated",  
    nil,  
    nil,  
    CFBooleanGetValue(true))
```

Staying Up to Date

Extension API

```
let nc = CFNotificationCenterGetDarwinNotifyCenter()  
CFNotificationCenterAddObserver(nc,  
    nil,  
    { _ in self.reloadModel() },  
    "com.example.app-model-updated",  
    nil,  
    .DeliverImmediately)
```

Staying Up to Date

Extension API

```
let nc = CFNotificationCenterGetDarwinNotifyCenter()
CFNotificationCenterAddObserver(nc,
    nil,
    { _ in self.reloadModel() },
    "com.example.app-model-updated",
    nil,
    .DeliverImmediately)
```

Staying Up to Date

Extension API

```
let nc = CFNotificationCenterGetDarwinNotifyCenter()
CFNotificationCenterAddObserver(nc,
    nil,
    { _ in self.reloadModel() },
    "com.example.app-model-updated",
    nil,
    .DeliverImmediately)
```

Staying Up to Date

Extension API

```
let nc = CFNotificationCenterGetDarwinNotifyCenter()
CFNotificationCenterAddObserver(nc,
    nil,
    { _ in self.reloadModel() },
    "com.example.app-model-updated",
    nil,
    .DeliverImmediately)
```

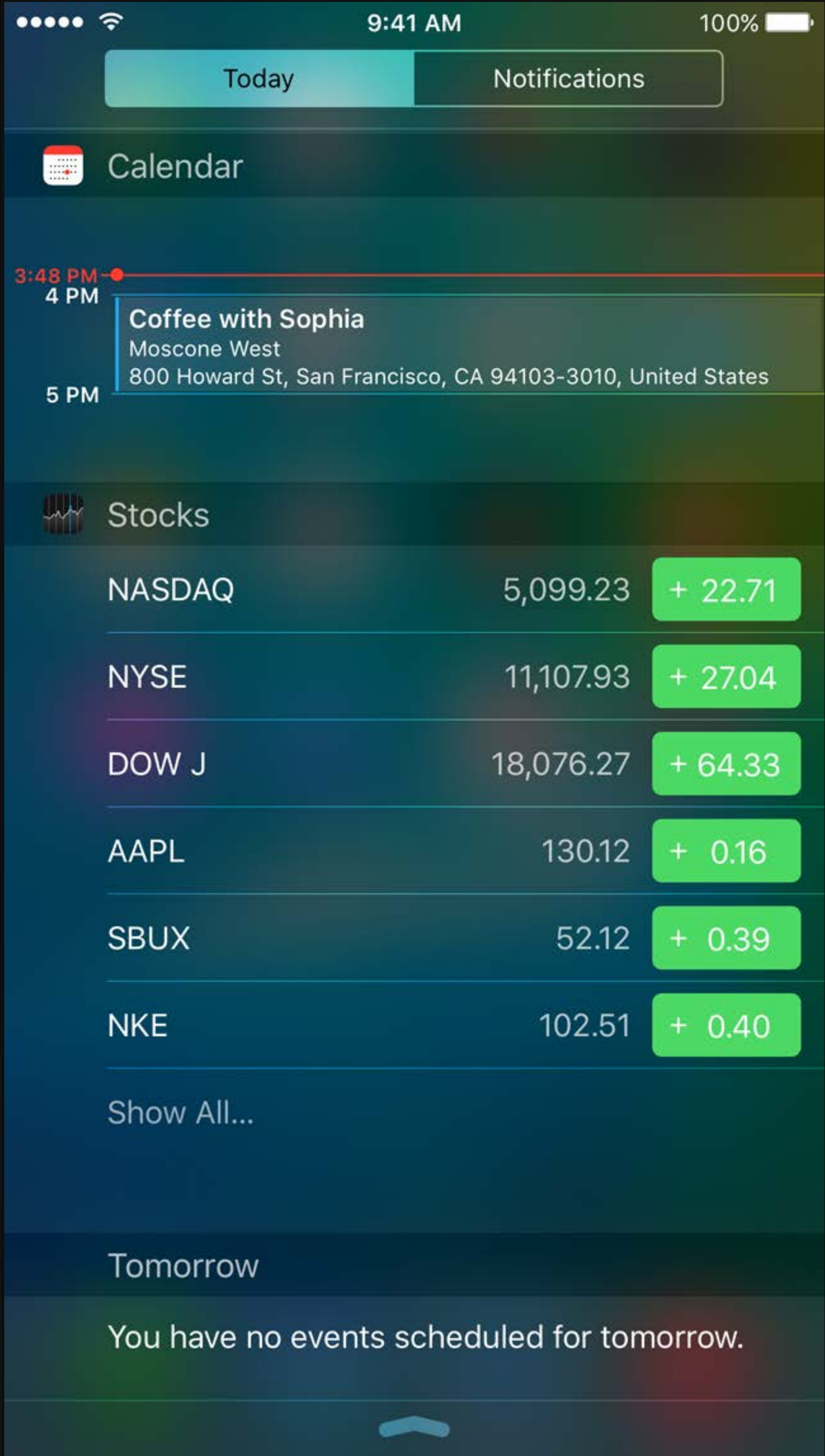
Staying Up to Date

Extension API

```
let nc = CFNotificationCenterGetDarwinNotifyCenter()
CFNotificationCenterAddObserver(nc,
    nil,
    { _ in self.reloadModel() },
    "com.example.app-model-updated",
    nil,
    .DeliverImmediately)
```


Background Refresh

Keeping your widget up to date



Background Refresh

Background Refresh



AAPL: \$132.12



Notifications



Calendar

3:48 PM ●
4 PM I

Coffee with Sophia

Moscone West

800 Howard St, San Francisco, CA 94103-3010, United States



Stocks

NASDAQ

5,099.23

+ 22.71

NYSE

11,107.93

+ 27.04

DOW J

18,076.27

+ 64.33

AAPL

130.12

+ 0.16

SBUX

52.12

+ 0.39

NKE

102.51

+ 0.40

[Show All...](#)

Tomorrow

You have no events scheduled for tomorrow.

Today

Notifications



Calendar

3:48 PM ●
4 PM |

Coffee with Sophia

Moscone West

Stocks

NASDAQ

5,099.23

NYSE

11,107.93

DOW J

18,076.27

AAPL

130.12

0.16

SBUX

52.12

NKE

102.51

[Show All...](#)

Tomorrow

You have no events scheduled for tomorrow.

Background Refresh

Updating the widget

System opportunistically refreshes content

Background Refresh

Updating the widget

System opportunistically refreshes content

View controller conforms to NCWidgetProviding

Background Refresh

Updating the widget

System opportunistically refreshes content

View controller conforms to NCWidgetProviding

Implements update delegate method

Background Refresh

Updating the widget

System opportunistically refreshes content

View controller conforms to NCWidgetProviding

Implements update delegate method

```
func widgetPerformUpdateWithCompletionHandler(completionHandler:
(NCUpdateResult) -> Void) {
    let updated = refreshTheModel()
    if (updated) { view.setNeedsDisplay() }
    completionHandler(updated ? .NewData : .NoData)
}
```

Background Refresh

Updating the widget

System opportunistically refreshes content

View controller conforms to NCWidgetProviding

Implements update delegate method

```
func widgetPerformUpdateWithCompletionHandler(completionHandler:
(NCUpdateResult) -> Void) {
    let updated = refreshTheModel()
    if (updated) { view.setNeedsDisplay() }
    completionHandler(updated ? .NewData : .NoData)
}
```

Background Refresh

Updating the widget

System opportunistically refreshes content

View controller conforms to NCWidgetProviding

Implements update delegate method

```
func widgetPerformUpdateWithCompletionHandler(completionHandler:
(NCUpdateResult) -> Void) {
    let updated = refreshTheModel()
    if (updated) { view.setNeedsDisplay() }
    completionHandler(updated ? .NewData : .NoData)
}
```

Background Refresh

Updating the widget

System opportunistically refreshes content

View controller conforms to NCWidgetProviding

Implements update delegate method

```
func widgetPerformUpdateWithCompletionHandler(completionHandler:
(NCUpdateResult) -> Void) {
    let updated = refreshTheModel()
    if (updated) { view.setNeedsDisplay() }
    completionHandler(updated ? .NewData : .NoData)
}
```

Background Refresh

Updating the widget

System opportunistically refreshes content

View controller conforms to NCWidgetProviding

Implements update delegate method

```
func widgetPerformUpdateWithCompletionHandler(completionHandler:
(NCUpdateResult) -> Void) {
    let updated = refreshTheModel()
    if (updated) { view.setNeedsDisplay() }
    completionHandler(updated ? .NewData : .NoData)
}
```


Notifications



Calendar

3:48 PM ●
4 PM

Coffee with Sophia

Moscone West

800 Howard St, San Francisco, CA 94103-3010, United States



Stocks

NASDAQ

5,099.23

+ 22.71

NYSE

11,107.93

+ 27.04

DOW J

18,076.27

+ 64.33

AAPL

132.12

+ 0.16

SBUX

52.12

+ 0.39

NKE

102.51

+ 0.40

[Show All...](#)

Tomorrow


You have no events scheduled for tomorrow.

9:41 AM

100%

Today

Notifications



Calendar

3:48 PM


4 PM

5 PM

Coffee with Sophia

Moscone West

800 Howard St, San Francisco, CA 94103-3010, United States



Stocks

NASDAQ	5,099.23	+ 22.71
NYSE	11,107.93	+ 27.04
DOW J	18,076.27	+ 64.33
AAPL	132.12	+ 0.16
SBUX	52.12	+ 0.39
NKE	102.51	+ 0.40
Show All...		

Tomorrow

You have no events scheduled for tomorrow.

Networking

Background sessions

Networking

NSURLSession

Background sessions



Networking

NSURLSession

Background sessions

Task performed by the system



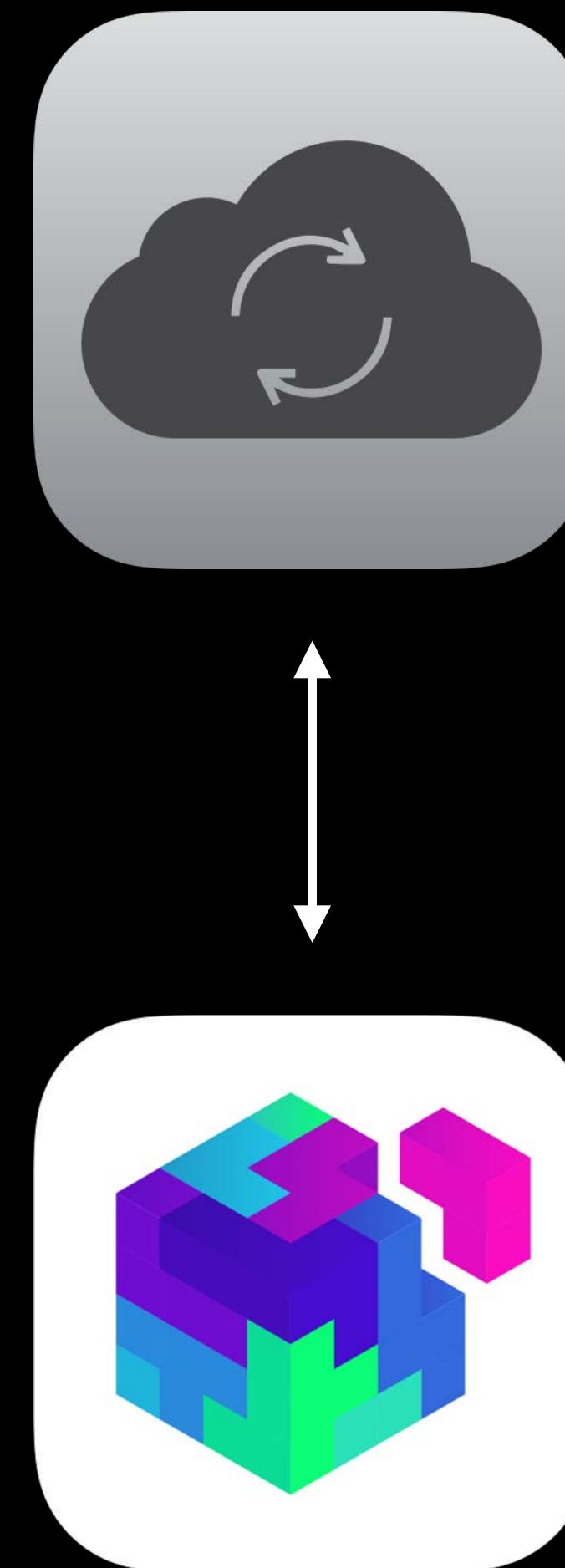
Networking

NSURLSession

Background sessions

Task performed by the system

Updates delivered to your extension
as long as it stays alive



Networking

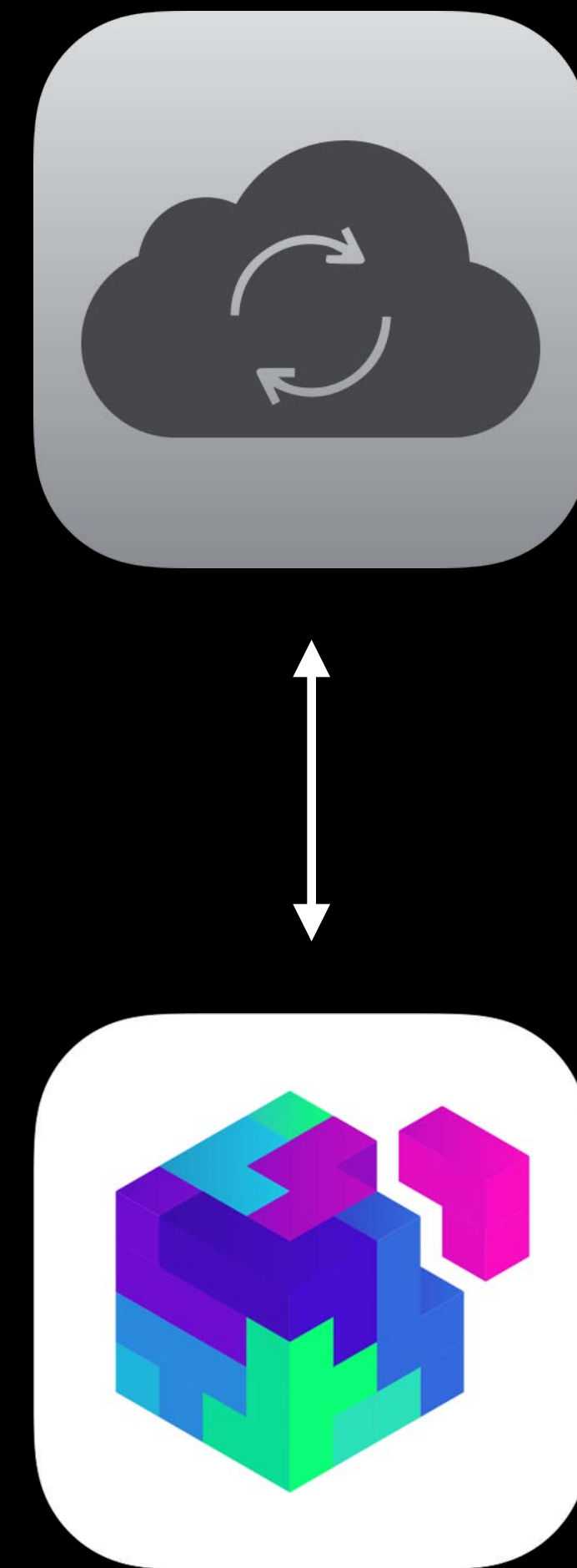
NSURLSession

Background sessions

Task performed by the system

Updates delivered to your extension
as long as it stays alive

Error and event handled by the
containing app



Networking

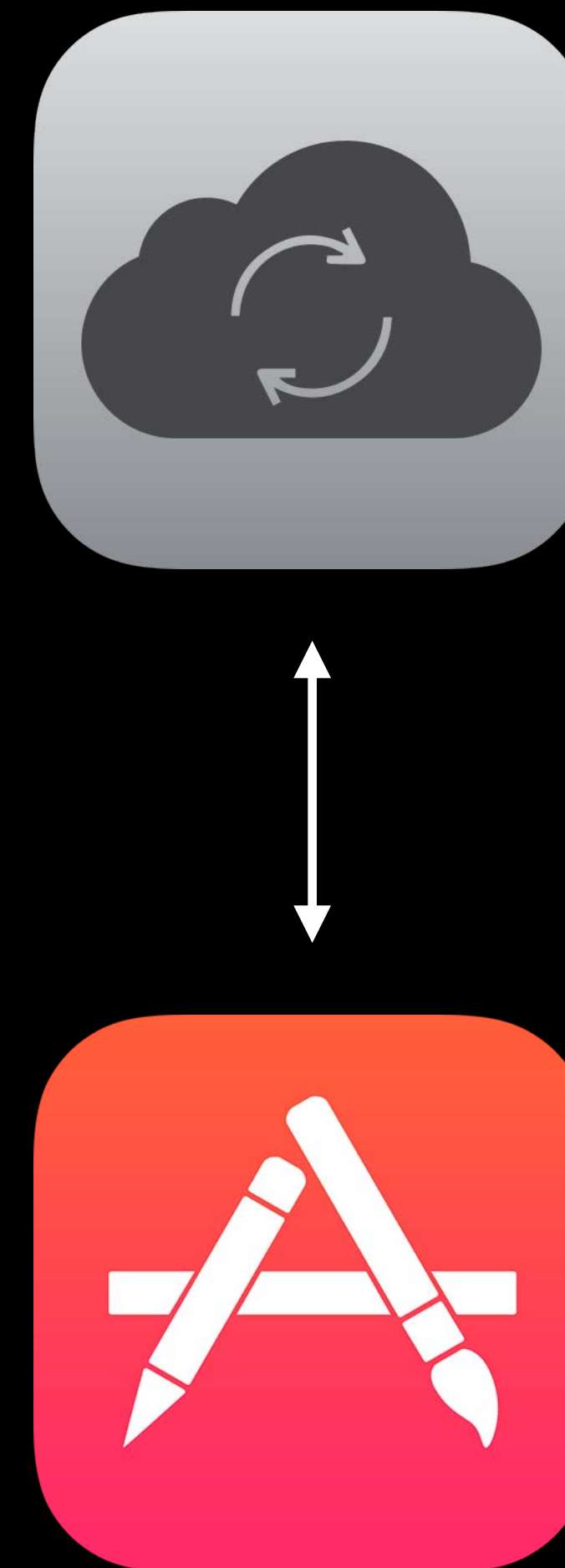
NSURLSession

Background sessions

Task performed by the system

Updates delivered to your extension
as long as it stays alive

Error and event handled by the
containing app



Networking

Extension API

```
let sc =  
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(  
    "com.example.my-download-session")  
sc.sharedContainerIdentifier = "group.com.example.my-app"  
let session = NSURLSession(configuration: sc, delegate: self, delegateQueue:  
    NSOperationQueue.mainQueue())  
let request = NSURLRequest(URL: NSURL(string: "https://www.apple.com/"))!  
let task = session.downloadTaskWithRequest(request)  
task?.resume()
```

Networking

Extension API

```
let sc =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(
    "com.example.my-download-session")
sc.sharedContainerIdentifier = "group.com.example.my-app"
let session = NSURLSession(configuration: sc, delegate: self, delegateQueue:
NSOperationQueue.mainQueue())
let request = NSURLRequest(URL: NSURL(string: "https://www.apple.com/"))
let task = session.downloadTaskWithRequest(request)
task?.resume()
```


Networking

Extension API

```
let sc =  
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(  
    "com.example.my-download-session")  
sc.sharedContainerIdentifier = "group.com.example.my-app"  
let session = NSURLSession(configuration: sc, delegate: self, delegateQueue:  
    NSOperationQueue.mainQueue())  
let request = NSURLRequest(URL: NSURL(string: "https://www.apple.com/"))!  
let task = session.downloadTaskWithRequest(request)  
task?.resume()
```

Networking

Extension API

```
let sc =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(
    "com.example.my-download-session")
sc.sharedContainerIdentifier = "group.com.example.my-app"
let session = NSURLSession(configuration: sc, delegate: self, delegateQueue:
NSOperationQueue.mainQueue())
let request = NSURLRequest(URL: NSURL(string: "https://www.apple.com/"))!
let task = session.downloadTaskWithRequest(request)
task?.resume()
```

Networking

Extension API

```
let sc =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(
    "com.example.my-download-session")
sc.sharedContainerIdentifier = "group.com.example.my-app"
let session = NSURLSession(configuration: sc, delegate: self, delegateQueue:
NSOperationQueue.mainQueue())
let request = NSURLRequest(URL: NSURL(string: "https://www.apple.com/"))
let task = session.downloadTaskWithRequest(request)
task?.resume()
```

Networking

Extension API

```
let sc =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(
    "com.example.my-download-session")
sc.sharedContainerIdentifier = "group.com.example.my-app"
let session = NSURLSession(configuration: sc, delegate: self, delegateQueue:
NSOperationQueue.mainQueue())
let request = NSURLRequest(URL: NSURL(string: "https://www.apple.com/"))
let task = session.downloadTaskWithRequest(request)
task?.resume()
```

Networking

Extension API

```
let sc =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(
    "com.example.my-download-session")
sc.sharedContainerIdentifier = "group.com.example.my-app"
let session = NSURLSession(configuration: sc, delegate: self, delegateQueue:
NSOperationQueue.mainQueue())
let request = NSURLRequest(URL: NSURL(string: "https://www.apple.com/"))
let task = session.downloadTaskWithRequest(request)
task?.resume()
```

Networking

Containing app API

```
func application(application: UIApplication,
    handleEventsForBackgroundURLSession identifier: String,
    completionHandler: () -> Void) {
    guard (identifier == "com.example.my-download-session") else { return }
    let sc =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(identifier)
    sc.sharedContainerIdentifier = "group.com.example.my-app"
    self.session = NSURLSession(configuration: sc,
        delegate: self,
        delegateQueue: NSOperationQueue.mainQueue())
    self.completionHandler = completionHandler
}
```

Networking

Containing app API

```
func application(application: UIApplication,
    handleEventsForBackgroundURLSession identifier: String,
    completionHandler: () -> Void) {
    guard (identifier == "com.example.my-download-session") else { return }
    let sc =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(identifier)
    sc.sharedContainerIdentifier = "group.com.example.my-app"
    self.session = NSURLSession(configuration: sc,
        delegate: self,
        delegateQueue: NSOperationQueue.mainQueue())
    self.completionHandler = completionHandler
}
```

Networking

Containing app API

```
func application(application: UIApplication,
    handleEventsForBackgroundURLSession identifier: String,
    completionHandler: () -> Void) {
    guard (identifier == "com.example.my-download-session") else { return }
    let sc =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(identifier)
    sc.sharedContainerIdentifier = "group.com.example.my-app"
    self.session = NSURLSession(configuration: sc,
        delegate: self,
        delegateQueue: NSOperationQueue.mainQueue())
    self.completionHandler = completionHandler
}
```


Networking

Containing app API

```
func application(application: UIApplication,
    handleEventsForBackgroundURLSession identifier: String,
    completionHandler: () -> Void) {
    guard (identifier == "com.example.my-download-session") else { return }
    let sc =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(identifier)
    sc.sharedContainerIdentifier = "group.com.example.my-app"
    self.session = NSURLSession(configuration: sc,
        delegate: self,
        delegateQueue: NSOperationQueue.mainQueue())
    self.completionHandler = completionHandler
}
```

Networking

Containing app API

```
func application(application: UIApplication,
    handleEventsForBackgroundURLSession identifier: String,
    completionHandler: () -> Void) {
    guard (identifier == "com.example.my-download-session") else { return }
    let sc =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(identifier)
    sc.sharedContainerIdentifier = "group.com.example.my-app"
    self.session = NSURLSession(configuration: sc,
        delegate: self,
        delegateQueue: NSOperationQueue.mainQueue())
    self.completionHandler = completionHandler
}
```

Networking

Containing app API

```
func application(application: UIApplication,
    handleEventsForBackgroundURLSession identifier: String,
    completionHandler: () -> Void) {
    guard (identifier == "com.example.my-download-session") else { return }
    let sc =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(identifier)
    sc.sharedContainerIdentifier = "group.com.example.my-app"
    self.session = NSURLSession(configuration: sc,
        delegate: self,
        delegateQueue: NSOperationQueue.mainQueue())
    self.completionHandler = completionHandler
}
```

Networking

Containing app API

```
func URLSessionDidFinishEventsForBackgroundURLSession(session: NSURLSession){  
    let sc = session.configuration  
    guard (sc.identifier == "com.example.my-download-session") else { return }  
    self.completionHandler!()  
    self.session = nil  
}
```

Networking

Containing app API

```
func URLSessionDidFinishEventsForBackgroundURLSession(session: NSURLSession){  
    let sc = session.configuration  
    guard (sc.identifier == "com.example.my-download-session") else { return }  
    self.completionHandler!()  
    self.session = nil  
}
```

Networking

Containing app API

```
func URLSessionDidFinishEventsForBackgroundURLSession(session: NSURLSession){  
    let sc = session.configuration  
    guard (sc.identifier == "com.example.my-download-session") else { return }  
    self.completionHandler!()  
    self.session = nil  
}
```

Networking

Containing app API

```
func URLSessionDidFinishEventsForBackgroundURLSession(session: NSURLSession){  
    let sc = session.configuration  
    guard (sc.identifier == "com.example.my-download-session") else { return }  
    self.completionHandler()  
    self.session = nil  
}
```

Networking

Containing app API

```
func URLSessionDidFinishEventsForBackgroundURLSession(session: NSURLSession){  
    let sc = session.configuration  
    guard (sc.identifier == "com.example.my-download-session") else { return }  
    self.completionHandler!()  
    self.session = nil  
}
```


Networking

watchOS

Use background task assertions
to protect small tasks



Networking

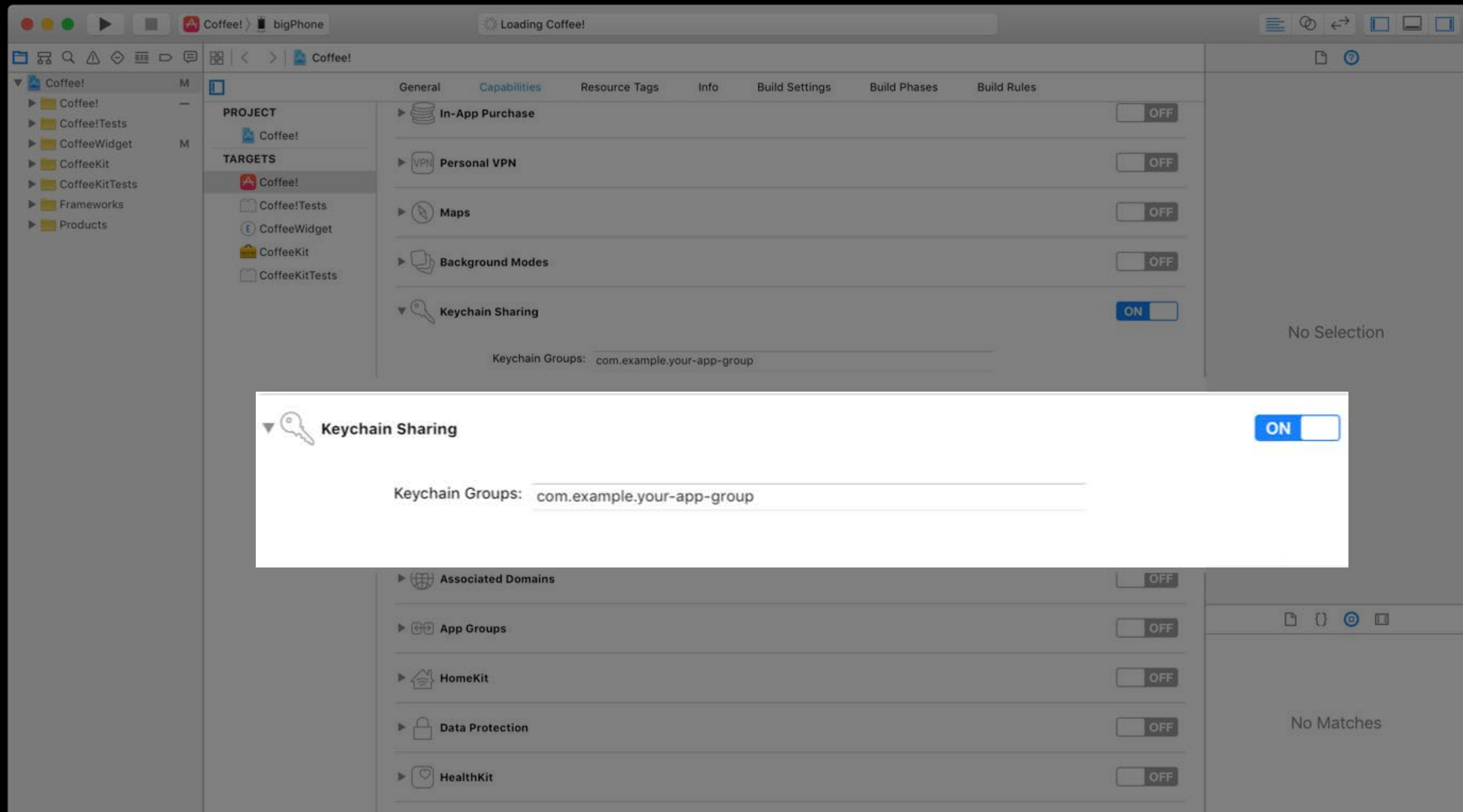
watchOS

Use background task assertions
to protect small tasks



Sharing Secrets

keychain-access-groups entitlement



Sharing Secrets

API

```
// Add an item  
SecItemAdd({ kSecAttrAccessGroup: "com.example.your-app-group"}, result:  
NULL)
```

Sharing Secrets

API

```
// Add an item  
SecItemAdd({ kSecAttrAccessGroup: "com.example.your-app-group"}, result:  
NULL)
```

Sharing Secrets

API

```
// Add an item  
SecItemAdd({ kSecAttrAccessGroup: "com.example.your-app-group"}, result:  
NULL)
```

Sharing Secrets

API

Automatic search behavior for query API

Sharing Secrets

API

Automatic search behavior for query API

- SecItemUpdate

Sharing Secrets

API

Automatic search behavior for query API

- SecItemUpdate
- SecItemDelete

Sharing Secrets

API

Automatic search behavior for query API

- SecItemUpdate
- SecItemDelete
- SecItemCopyMatching

Sharing Secrets

API

Automatic search behavior for query API

- SecItemUpdate
- SecItemDelete
- SecItemCopyMatching

Summary

Action and Share Extensions

Summary

Action and Share Extensions

Enhanced Today widget

Summary

Action and Share Extensions

Enhanced Today widget

General best practices

Related Sessions

WatchKit Tips and Tricks	Presidio	Friday 10:00AM
Security and Your Apps	Mission	Tuesday 4:30PM

More Information

Documentation

App Extension Programming Guide

Sample Code

Lister

Sample Photo Editing Extension

[http://developer.apple.com/
app-extensions](http://developer.apple.com/app-extensions)

Technical Support

Apple Developer Forums

Developer Technical Support

<http://devforums.apple.com>

General Inquiries

Curt Rothert, App Frameworks Evangelist

rothert@apple.com

