#WWDC19

Large Content Viewer

Ensuring readability for everyone

Sommer Panage, Accessibility Engineering Manager

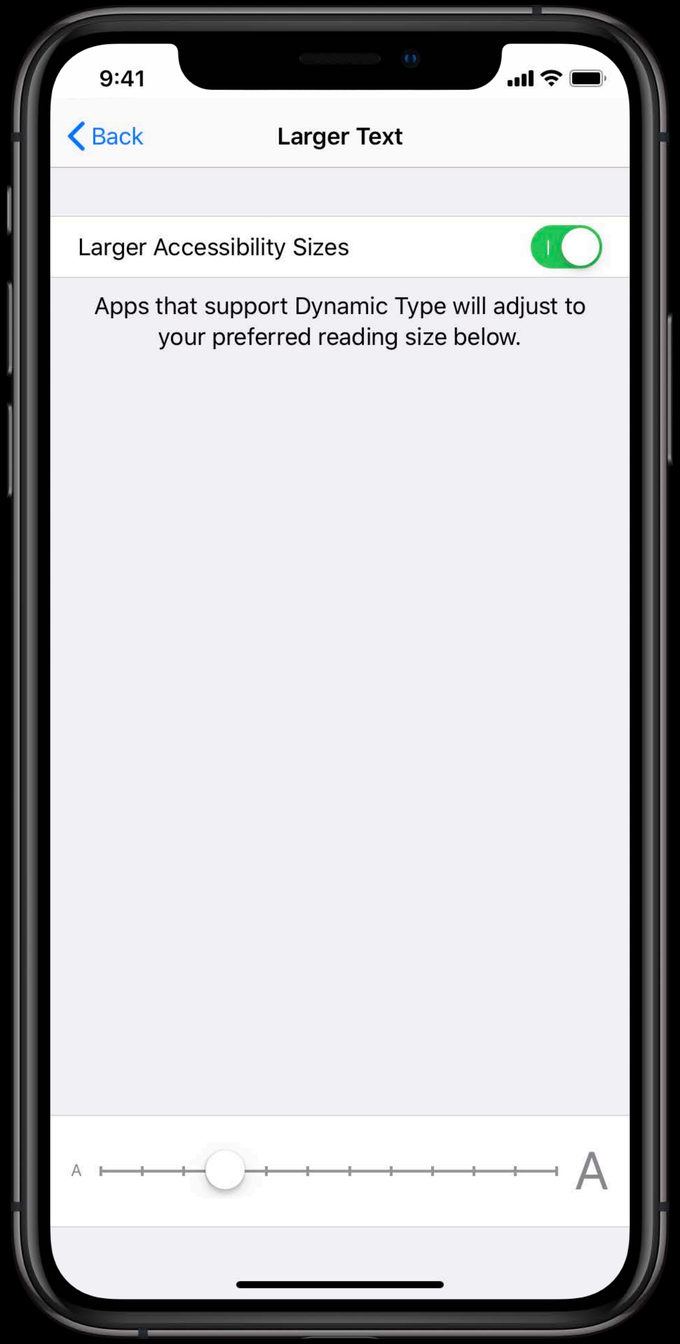
Background

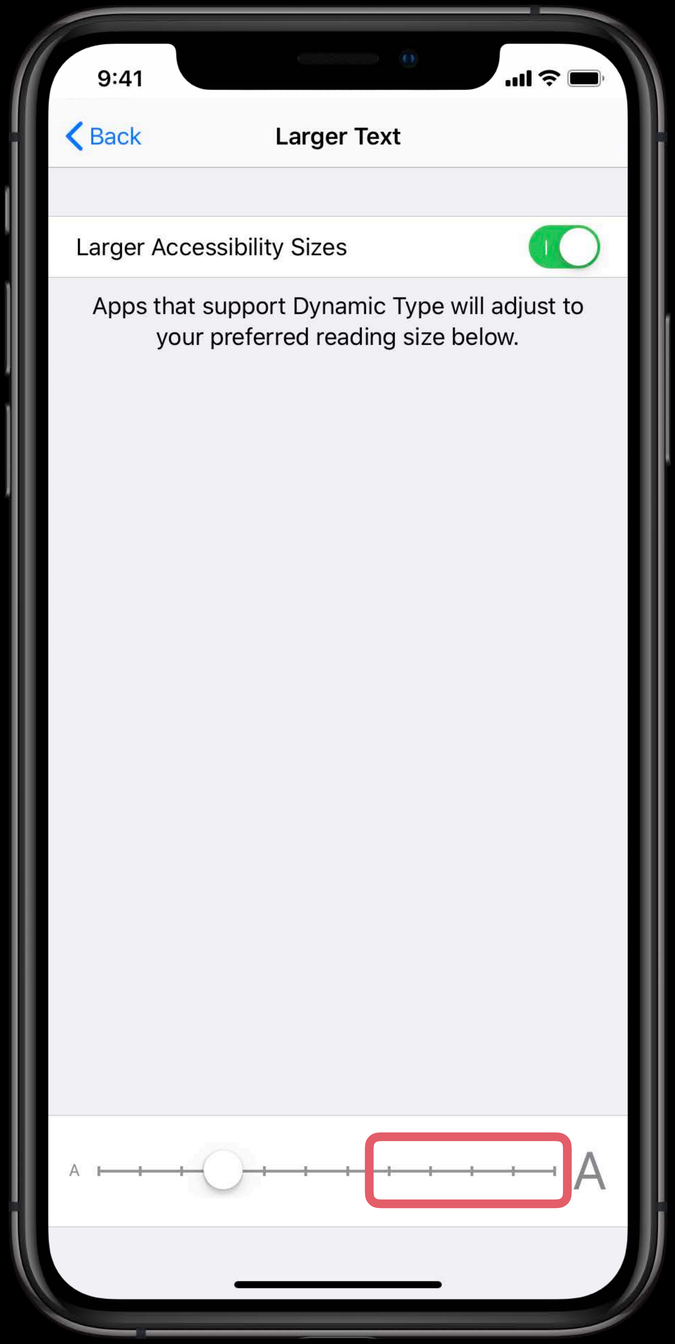
API

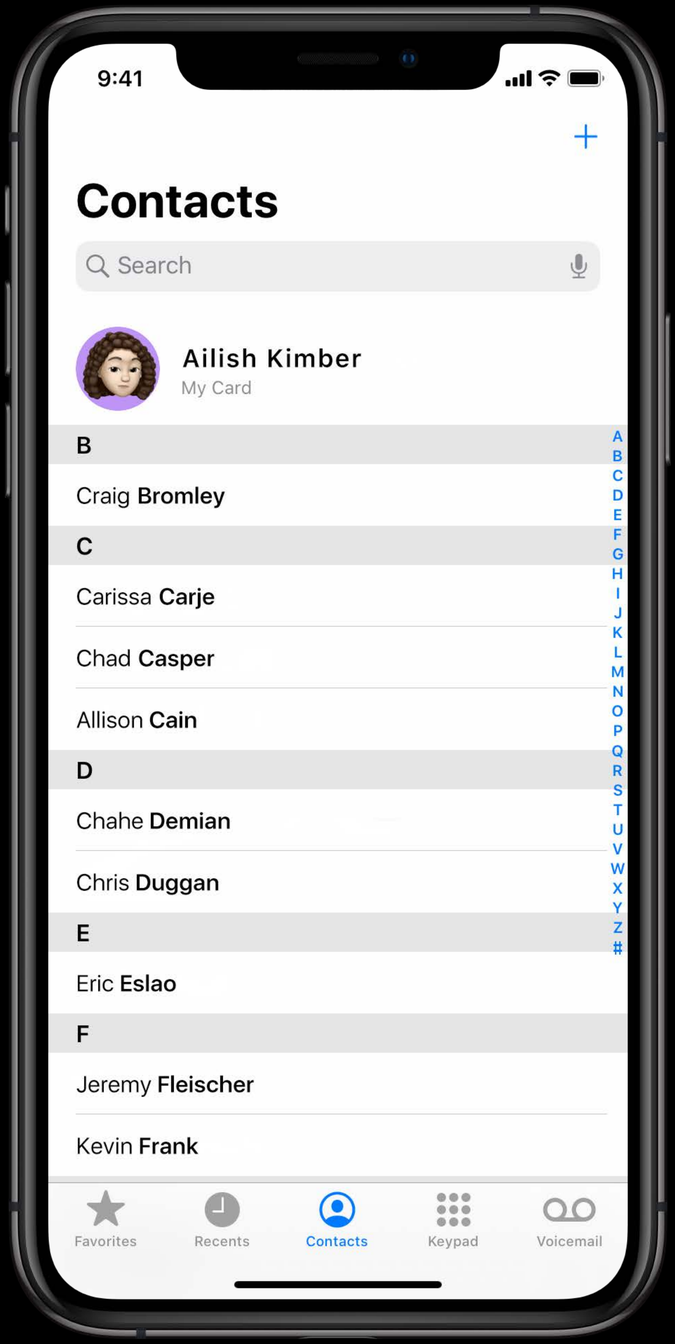
Examples

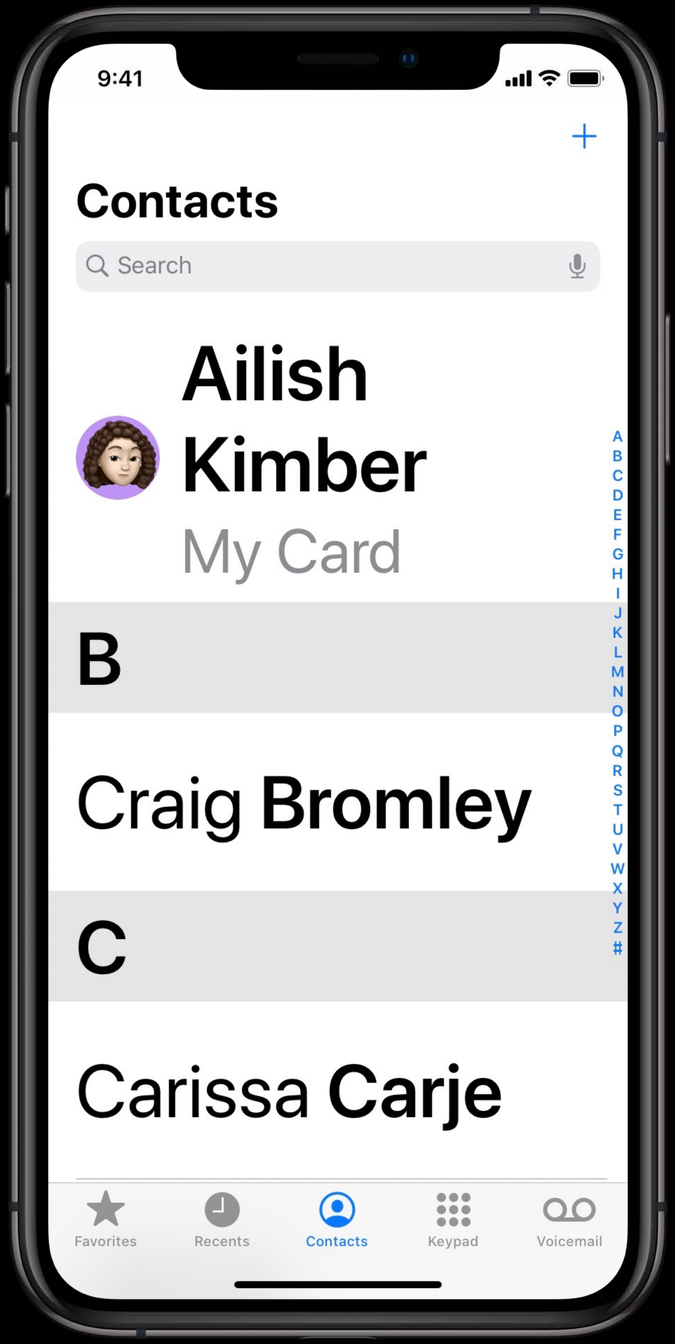
Background

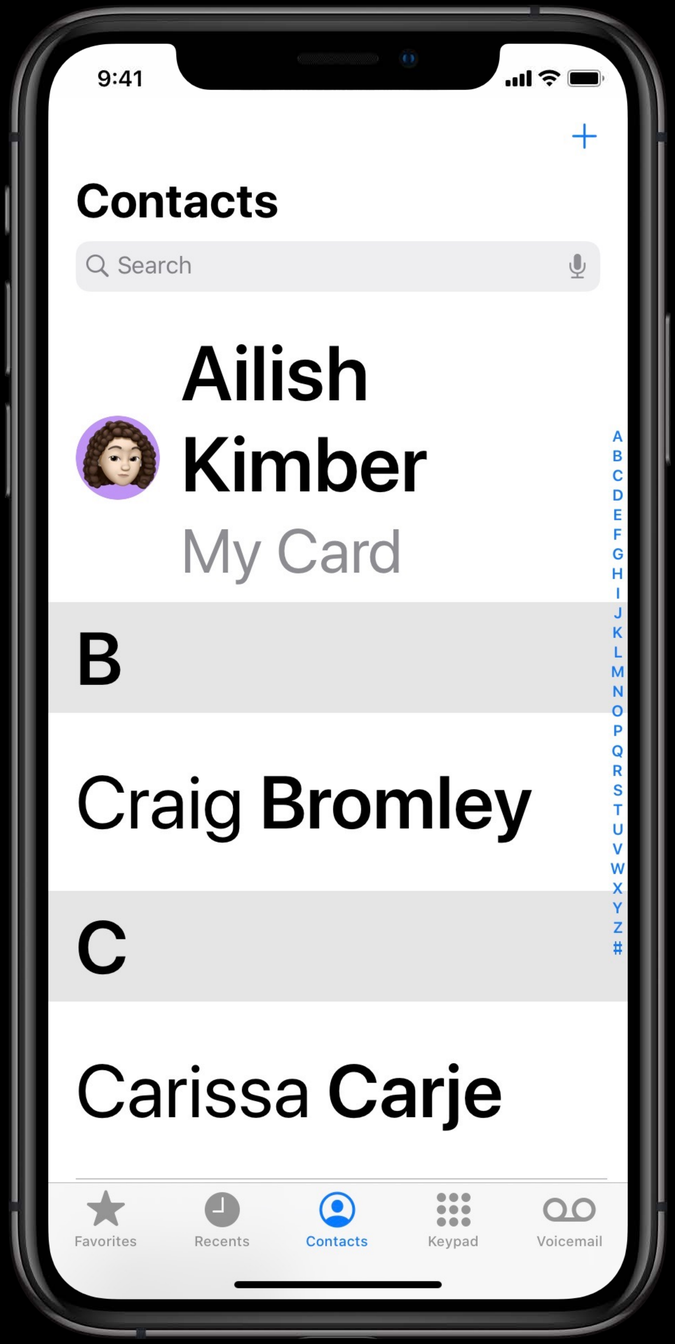

9:41
Back
Text Size
Apps that support Dynamic Type will adjust to your preferred reading size below.
A
A

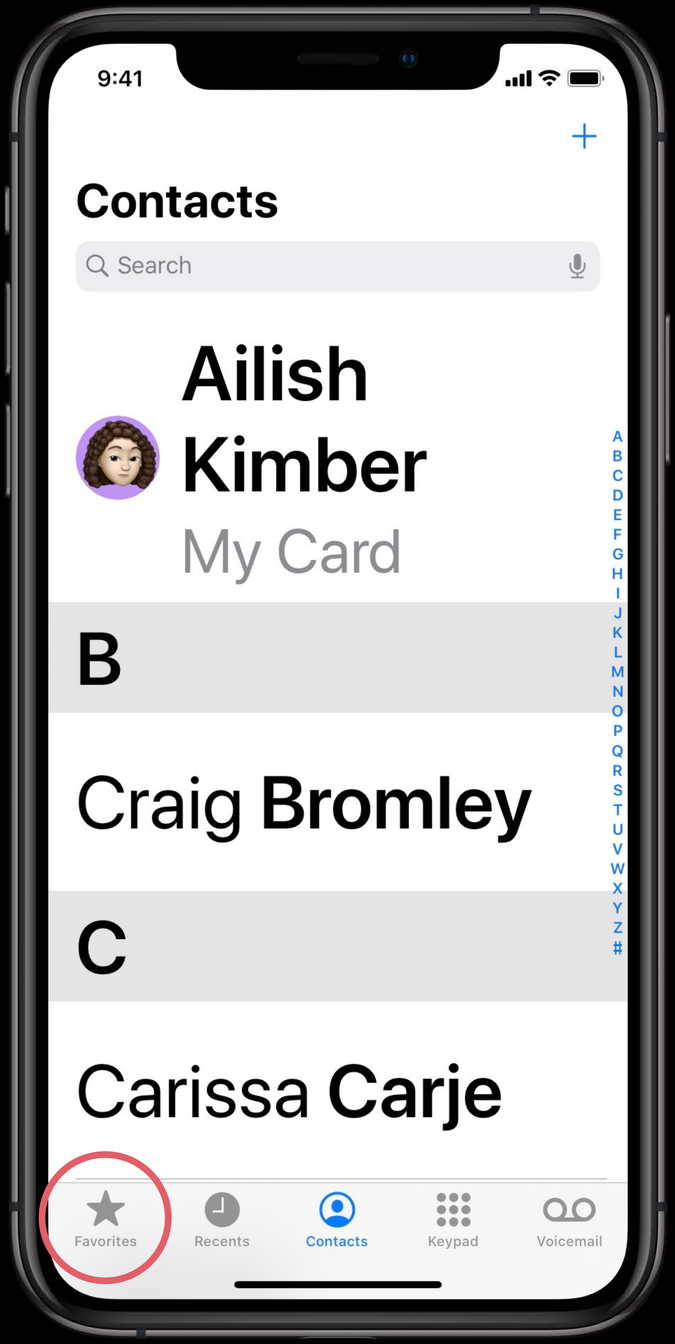

9:41
Back
Larger Text
Larger Accessibility Sizes
Apps that support Dynamic Type will adjust to your preferred reading size below.
A
A

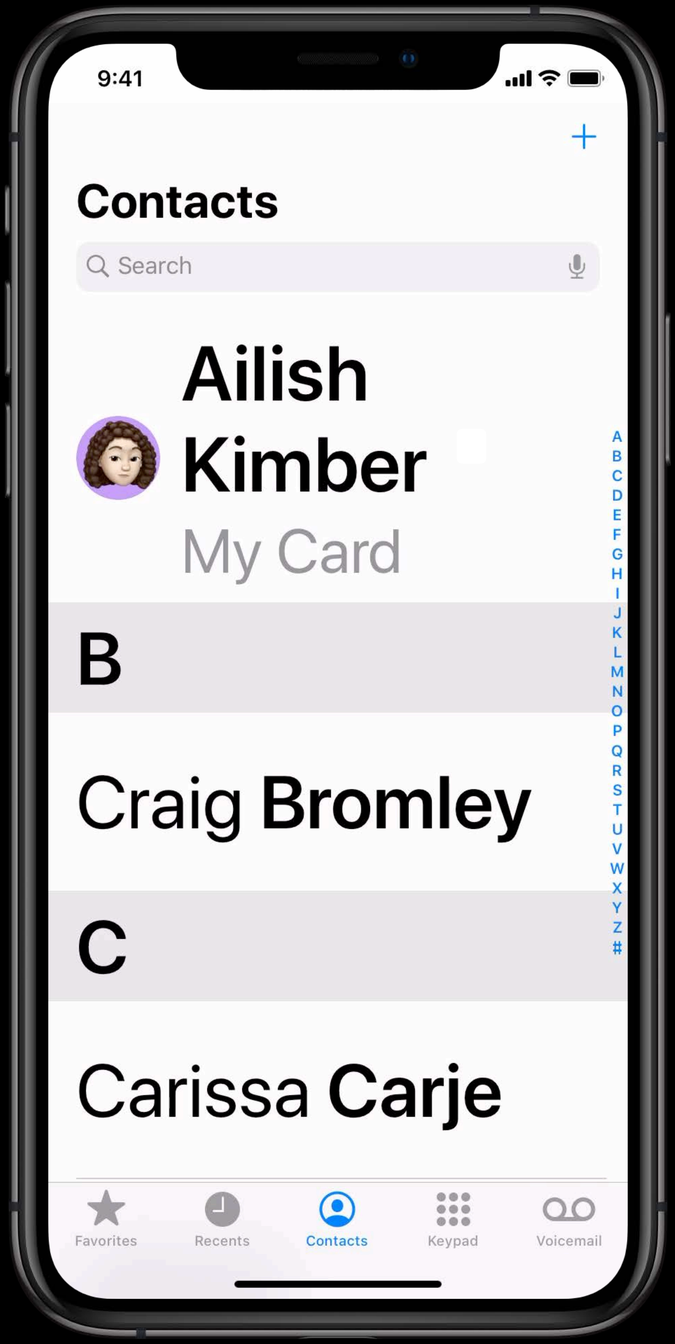

9:41
Back
Larger Text
Larger Accessibility Sizes
Apps that support Dynamic Type will adjust to your preferred reading size below.
A
A

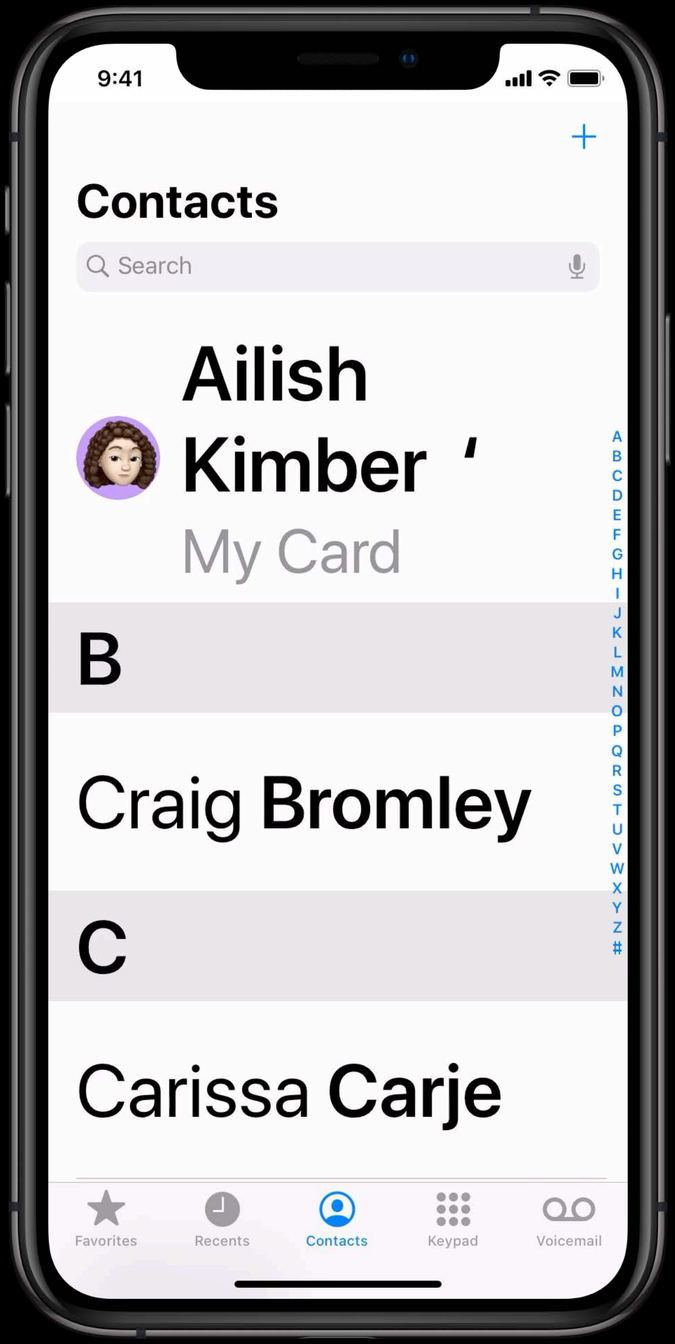

9:41
Contacts
Search
Ailish Kimber
My Card
B
Craig Bromley
C
Carissa Carje
Chad Casper
Allison Cain
D
Chahe Demian
Chris Duggan
E
Eric Eslao
F
Jeremy Fleischer
Kevin Frank
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z #
Favorites
Recents
Contacts
Keypad
Voicemail


9:41
Contacts
Search
Ailish
Kimber
My Card
B
Craig Bromley
C
Carissa Carje
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z #
Favorites
Recents
Contacts
Keypad
Voicemail


9:41
Contacts
Search
Ailish
Kimber
My Card
B
Craig Bromley
C
Carissa Carje
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z #
Favorites
Recents
Contacts
Keypad
Voicemail


9:41
Contacts
Search
Ailish
Kimber
My Card
B
Craig Bromley
C
Carissa Carje
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z #
Favorites
Recents
Contacts
Keypad
Voicemail


9:41
Contacts
Search
Ailish
Kimber
My Card
B
Craig Bromley
C
Carissa Carje
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z #
Favorites
Recents
Contacts
Keypad
Voicemail


9:41
Contacts
Search
Ailish Kimber ‘
My Card
B
Craig Bromley
C
Carissa Carje
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z #
Favorites
Recents
Contacts
Keypad
Voicemail

Large Content Viewer

Shown by default in standard UIKit bars

- Navigation bars
- Tab bars
- Toolbars
- Status bar

Custom toolbars, navigation bars, and tab bars

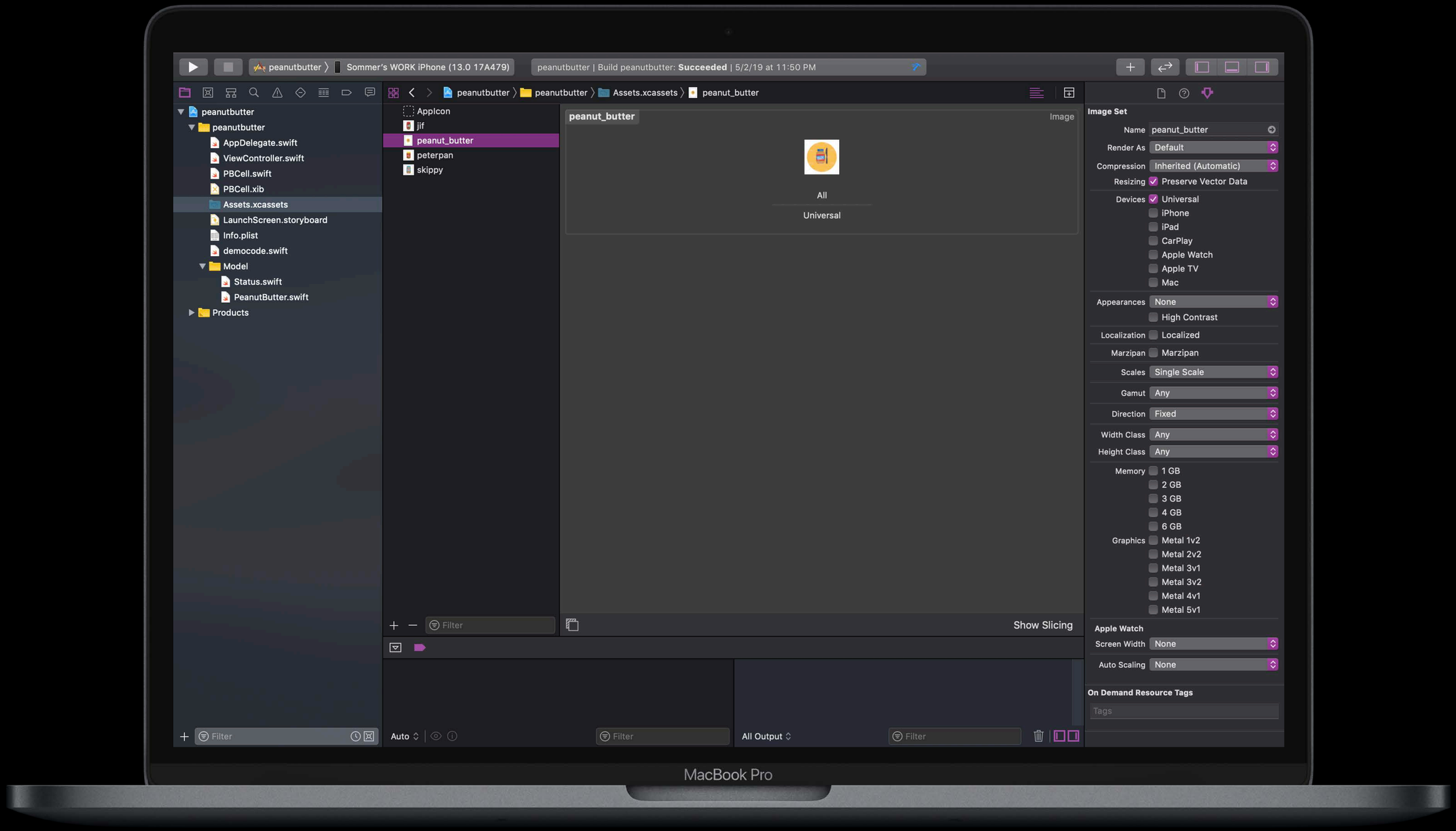
Scaling dynamic type is preferred whenever possible

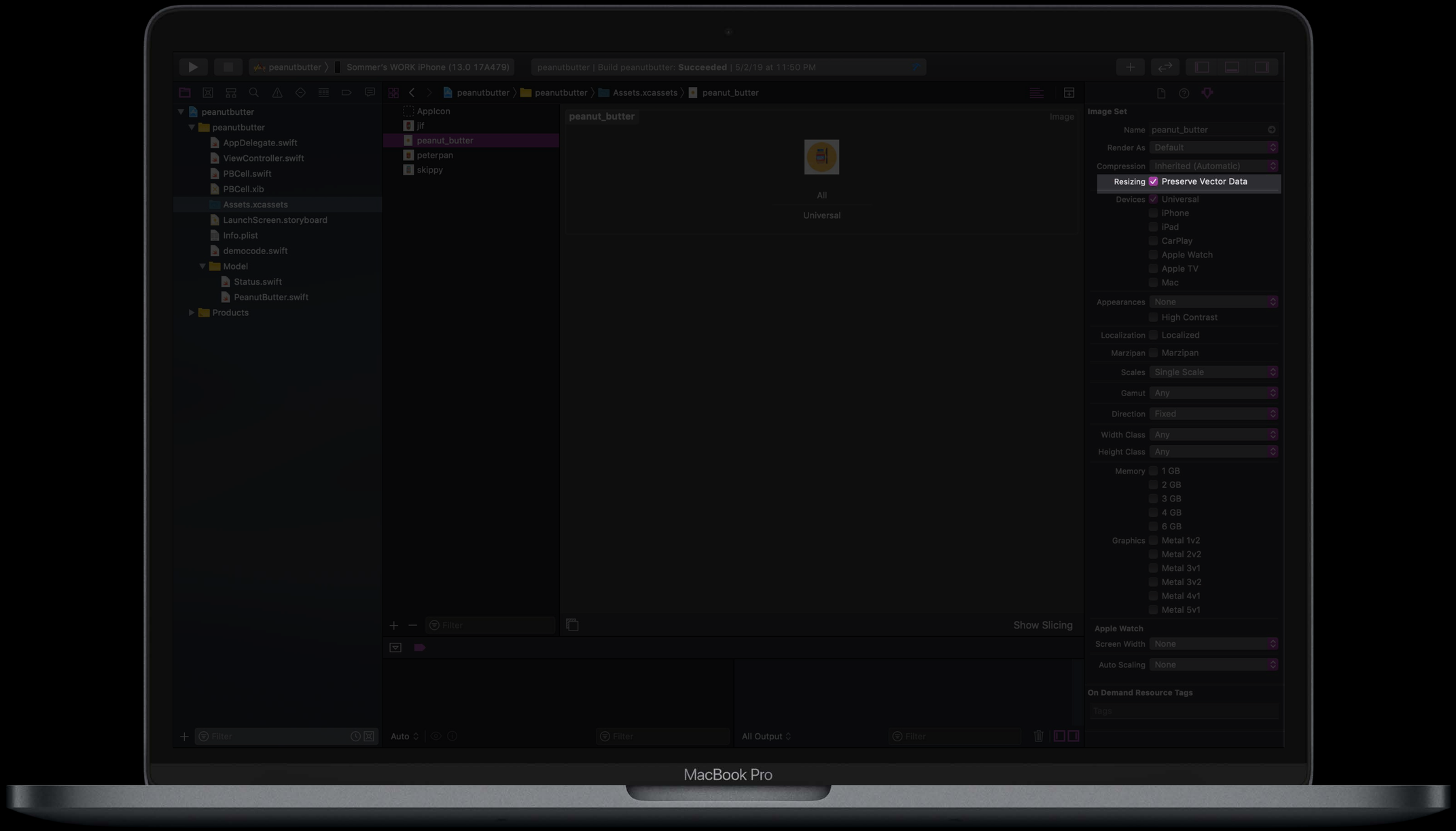
API

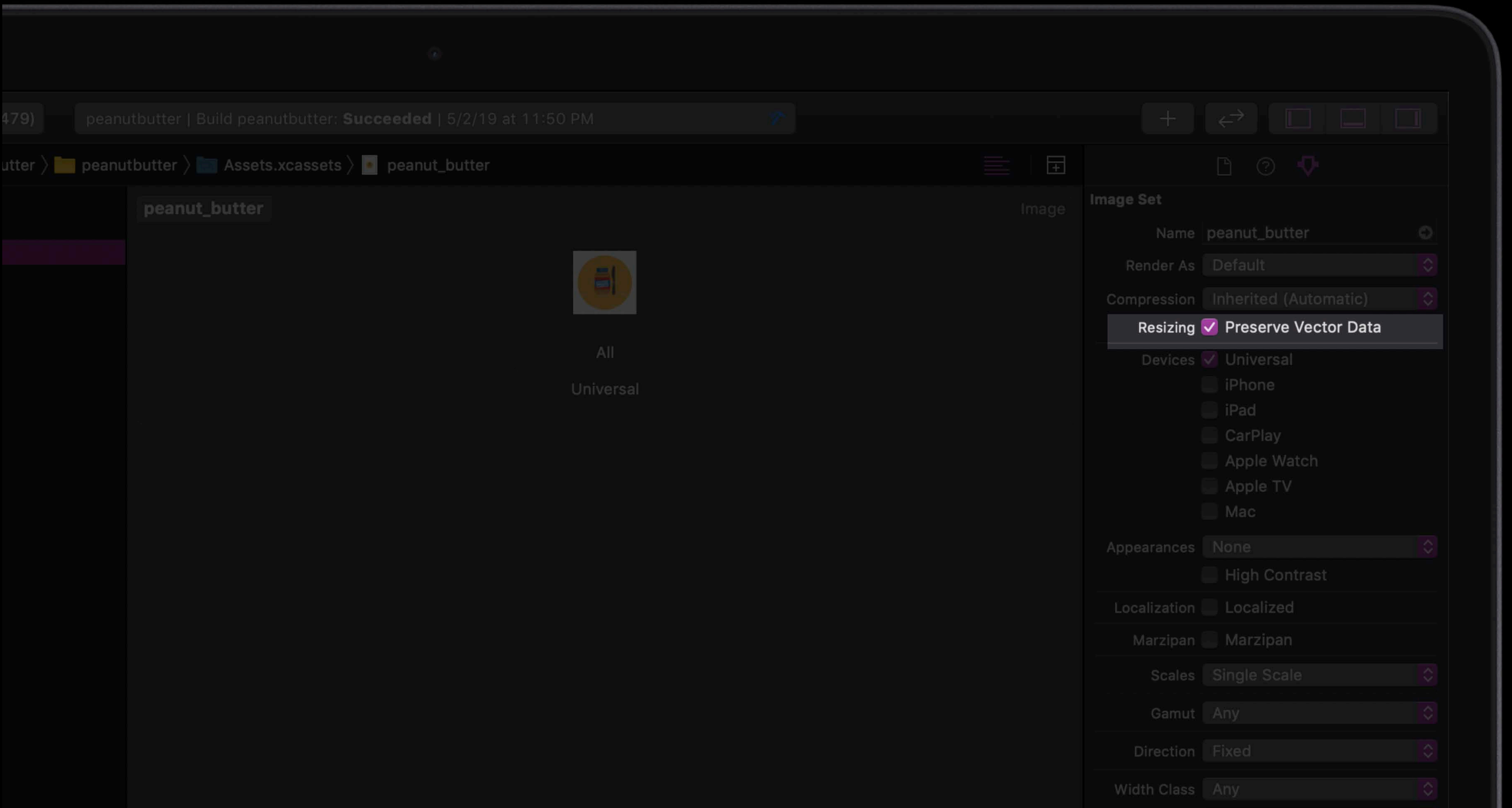
Standard UIKit Controls

PDF images

- Check “Preserves Vector Data” checkbox in asset catalog


peanutbutter 〉 Sommer's WORK iPhone (13.0 17A479)
peanutbutter | Build peanutbutter: Succeeded | 5/2/19 at 11:50 PM
peanutbutter 〉 peanutbutter 〉 Assets.xcassets 〉 peanut_butter
peanutbutter
peanutbutter
AppDelegate.swift
ViewController.swift
PBCell.swift
PBCell.xib
Assets.xcassets
LaunchScreen.storyboard
Info.plist
democode.swift
Model
Status.swift
PeanutButter.swift
Products
AppIcon
jif
peanut_butter
peterpan
skippy
peanut_butter
Image
All
Universal
Show Slicing
Image Set
Name peanut_butter
Render As Default
Compression Inherited (Automatic)
Resizing Preserve Vector Data
Devices Universal
iPhone
iPad
CarPlay
Apple Watch
Apple TV
Mac
Appearances None
High Contrast
Localization Localized
Marzipan Marzipan
Scales Single Scale
Gamut Any
Direction Fixed
Width Class Any
Height Class Any
Memory 1 GB
2 GB
3 GB
4 GB
6 GB
Graphics Metal 1v2
Metal 2v2
Metal 3v1
Metal 3v2
Metal 4v1
Metal 5v1
Apple Watch
Screen Width None
Auto Scaling None
On Demand Resource Tags
Tags
Filter
Auto
All Output
MacBook Pro


peanutbutter 〉 Sommer's WORK iPhone (13.0 17A479)
peanutbutter | Build peanutbutter: Succeeded | 5/2/19 at 11:50 PM
peanutbutter 〉 peanutbutter 〉 Assets.xcassets 〉 peanut_butter
peanutbutter
peanutbutter
AppDelegate.swift
ViewController.swift
PBCell.swift
PBCell.xib
Assets.xcassets
LaunchScreen.storyboard
Info.plist
democode.swift
Model
Status.swift
PeanutButter.swift
Products
AppIcon
jif
peanut_butter
peterpan
skippy
peanut_butter
Image
All
Universal
Show Slicing
Image Set
Name peanut_butter
Render As Default
Compression Inherited (Automatic)
Resizing Preserve Vector Data
Devices Universal
iPhone
iPad
CarPlay
Apple Watch
Apple TV
Mac
Appearances None
High Contrast
Localization Localized
Marzipan Marzipan
Scales Single Scale
Gamut Any
Direction Fixed
Width Class Any
Height Class Any
Memory 1 GB
2 GB
3 GB
4 GB
6 GB
Graphics Metal 1v2
Metal 2v2
Metal 3v1
Metal 3v2
Metal 4v1
Metal 5v1
Apple Watch
Screen Width None
Auto Scaling None
On Demand Resource Tags
Tags
Auto
All Output
MacBook Pro


peanutbutter | Build peanutbutter: Succeeded | 5/2/19 at 11:50 PM
peanutbutter
Assets.xcassets
peanut_butter
peanut_butter
Image
All
Universal
Image Set
Name
peanut_butter
Render As
Default
Compression
Inherited (Automatic)
Resizing
Preserve Vector Data
Devices
Universal
iPhone
iPad
CarPlay
Apple Watch
Apple TV
Mac
Appearances
None
High Contrast
Localization
Localized
Marzipan
Marzipan
Scales
Single Scale
Gamut
Any
Direction
Fixed
Width Class
Any

Standard UIKit Controls

Raster images

- By default, image will be blurry at larger sizes
- Provide a larger version of the image via API

```
// UIBarItem

var largeContentSizeImage: UIImage?

open var largeContentSizeImageInsets: UIEdgeInsets
```

Standard UIKit Controls

Raster images

- By default, image will be blurry at larger sizes
- Provide a larger version of the image via API

```
// UIBarItem

var largeContentSizeImage: UIImage?

open var largeContentSizeImageInsets: UIEdgeInsets
```

Standard UIKit Controls

Raster images

- By default, image will be blurry at larger sizes
- Provide a larger version of the image via API

```
// UIBarItem

var largeContentSizeImage: UIImage?

open var largeContentSizeImageInsets: UIEdgeInsets
```

Custom Views

New API in iOS 13

NEW

Custom Views

New API in iOS 13

NEW

Specify which views should present the viewer

Custom Views

New API in iOS 13

NEW

Specify which views should present the viewer

Provide a title and/or image to show for each view

Custom Views

New API in iOS 13

NEW

Specify which views should present the viewer

Provide a title and/or image to show for each view

Default values provided for standard UIKit classes

Custom Views

New API in iOS 13

NEW

Specify which views should present the viewer

Provide a title and/or image to show for each view

Default values provided for standard UIKit classes

Set up a gesture interaction on the containing view

NEW

```
// UILargeContentViewerItem protocol

public protocol UILargeContentViewerItem : NSObjectProtocol {

    var showsLargeContentViewer: Bool { get }

    var largeContentTitle: String? { get }

    var largeContentImage: UIImage? { get }

    var scalesLargeContentImage: Bool { get }

    var largeContentImageInsets: UIEdgeInsets { get }
}
```

NEW

```
// UILargeContentViewerItem protocol

public protocol UILargeContentViewerItem : NSObjectProtocol {

    var showsLargeContentViewer: Bool { get }

    var largeContentTitle: String? { get }

    var largeContentImage: UIImage? { get }

    var scalesLargeContentImage: Bool { get }

    var largeContentImageInsets: UIEdgeInsets { get }
}
```

NEW

```
// UILargeContentViewerItem protocol

public protocol UILargeContentViewerItem : NSObjectProtocol {

    var showsLargeContentViewer: Bool { get }

    var largeContentTitle: String? { get }

    var largeContentImage: UIImage? { get }

    var scalesLargeContentImage: Bool { get }

    var largeContentImageInsets: UIEdgeInsets { get }
}
```

NEW

```
// UILargeContentViewerItem protocol

public protocol UILargeContentViewerItem : NSObjectProtocol {

    var showsLargeContentViewer: Bool { get }

    var largeContentTitle: String? { get }

    var largeContentImage: UIImage? { get }

    var scalesLargeContentImage: Bool { get }

    var largeContentImageInsets: UIEdgeInsets { get }
}
```

NEW

```
// UILargeContentViewerItem protocol

public protocol UILargeContentViewerItem : NSObjectProtocol {

    var showsLargeContentViewer: Bool { get }

    var largeContentTitle: String? { get }

    var largeContentImage: UIImage? { get }

    var scalesLargeContentImage: Bool { get }

    var largeContentImageInsets: UIEdgeInsets { get }
}
```

NEW

```
// UILargeContentViewerItem protocol

public protocol UILargeContentViewerItem : NSObjectProtocol {

    var showsLargeContentViewer: Bool { get }

    var largeContentTitle: String? { get }

    var largeContentImage: UIImage? { get }

    var scalesLargeContentImage: Bool { get }

    var largeContentImageInsets: UIEdgeInsets { get }
}
```

NEW

```
// UILargeContentViewerItem protocol

public protocol UILargeContentViewerItem : NSObjectProtocol {

    var showsLargeContentViewer: Bool { get }

    var largeContentTitle: String? { get }

    var largeContentImage: UIImage? { get }

    var scalesLargeContentImage: Bool { get }

    var largeContentImageInsets: UIEdgeInsets { get }
}
```

// UIView implementation

NEW

```
extension UIView : UILargeContentViewerItem {

    var showsLargeContentViewer: Bool

    var largeContentTitle: String?

    var largeContentImage: UIImage?

    var scalesLargeContentImage: Bool

    var largeContentImageInsets: UIEdgeInsets
}
```

NEW

```
extension UIView {

    func addInteraction(_ interaction: UIInteraction)

}
```

NEW

```
class UILargeContentViewerInteraction : NSObject, UIInteraction {

    init(delegate: UILargeContentViewerInteractionDelegate?)

    weak var delegate: UILargeContentViewerInteractionDelegate? { get }

    var gestureRecognizerForExclusionRelationship: UIGestureRecognizer { get }

    class var isEnabled: Bool { get }

}

extension UILargeContentViewerInteraction {
    class let enabledStatusDidChangeNotification: NSNotification.Name
}
```

NEW

```
class UILargeContentViewerInteraction : NSObject, UIInteraction {

    init(delegate: UILargeContentViewerInteractionDelegate?)

    weak var delegate: UILargeContentViewerInteractionDelegate? { get }

    var gestureRecognizerForExclusionRelationship: UIGestureRecognizer { get }

    class var isEnabled: Bool { get }

}

extension UILargeContentViewerInteraction {
    class let enabledStatusDidChangeNotification: NSNotification.Name
}
```

NEW

```
class UILargeContentViewerInteraction : NSObject, UIInteraction {

    init(delegate: UILargeContentViewerInteractionDelegate?)

    weak var delegate: UILargeContentViewerInteractionDelegate? { get }

    var gestureRecognizerForExclusionRelationship: UIGestureRecognizer { get }

    class var isEnabled: Bool { get }

}

extension UILargeContentViewerInteraction {
    class let enabledStatusDidChangeNotification: NSNotification.Name
}
```

NEW

```
class UILargeContentViewerInteraction : NSObject, UIInteraction {

    init(delegate: UILargeContentViewerInteractionDelegate?)

    weak var delegate: UILargeContentViewerInteractionDelegate? { get }

    var gestureRecognizerForExclusionRelationship: UIGestureRecognizer { get }

    class var isEnabled: Bool { get }

}

extension UILargeContentViewerInteraction {
    class let enabledStatusDidChangeNotification: NSNotification.Name
}
```

NEW

```
class UILargeContentViewerInteraction : NSObject, UIInteraction {

    init(delegate: UILargeContentViewerInteractionDelegate?)

    weak var delegate: UILargeContentViewerInteractionDelegate? { get }

    var gestureRecognizerForExclusionRelationship: UIGestureRecognizer { get }

    class var isEnabled: Bool { get }

}

extension UILargeContentViewerInteraction {
    class let enabledStatusDidChangeNotification: NSNotification.Name
}
```

NEW

```
class UILargeContentViewerInteraction : NSObject, UIInteraction {

    init(delegate: UILargeContentViewerInteractionDelegate?)

    weak var delegate: UILargeContentViewerInteractionDelegate? { get }

    var gestureRecognizerForExclusionRelationship: UIGestureRecognizer { get }

    class var isEnabled: Bool { get }

}

extension UILargeContentViewerInteraction {
    class let enabledStatusDidChangeNotification: NSNotification.Name
}
```

NEW

```
class UILargeContentViewerInteraction : NSObject, UIInteraction {

    init(delegate: UILargeContentViewerInteractionDelegate?)

    weak var delegate: UILargeContentViewerInteractionDelegate? { get }

    var gestureRecognizerForExclusionRelationship: UIGestureRecognizer { get }

    class var isEnabled: Bool { get }

}
```

```
extension UILargeContentViewerInteraction {
    class let enabledStatusDidChangeNotification: NSNotification.Name
}
```

NEW

```
protocol UILargeContentViewerInteractionDelegate : NSObjectProtocol {

    optional func largeContentViewerInteraction(_ interaction:
UILargeContentViewerInteraction, didEndOn item: UILargeContentViewerItem?, at point: CGPoint)

    optional func largeContentViewerInteraction(_ interaction:
UILargeContentViewerInteraction, itemAt point: CGPoint) -> UILargeContentViewerItem?

    optional func viewController(for interaction: UILargeContentViewerInteraction) ->
UIViewController

}
```

NEW

```
protocol UILargeContentViewerInteractionDelegate : NSObjectProtocol {

    optional func largeContentViewerInteraction(_ interaction:
UILargeContentViewerInteraction, didEndOn item: UILargeContentViewerItem?, at point: CGPoint)

    optional func largeContentViewerInteraction(_ interaction:
UILargeContentViewerInteraction, itemAt point: CGPoint) -> UILargeContentViewerItem?

    optional func viewController(for interaction: UILargeContentViewerInteraction) ->
UIViewController

}
```

NEW

```
protocol UILargeContentViewerInteractionDelegate : NSObjectProtocol {

    optional func largeContentViewerInteraction(_ interaction:
UILargeContentViewerInteraction, didEndOn item: UILargeContentViewerItem?, at point: CGPoint)

    optional func largeContentViewerInteraction(_ interaction:
UILargeContentViewerInteraction, itemAt point: CGPoint) -> UILargeContentViewerItem?

    optional func viewController(for interaction: UILargeContentViewerInteraction) ->
UIViewController

}
```

NEW

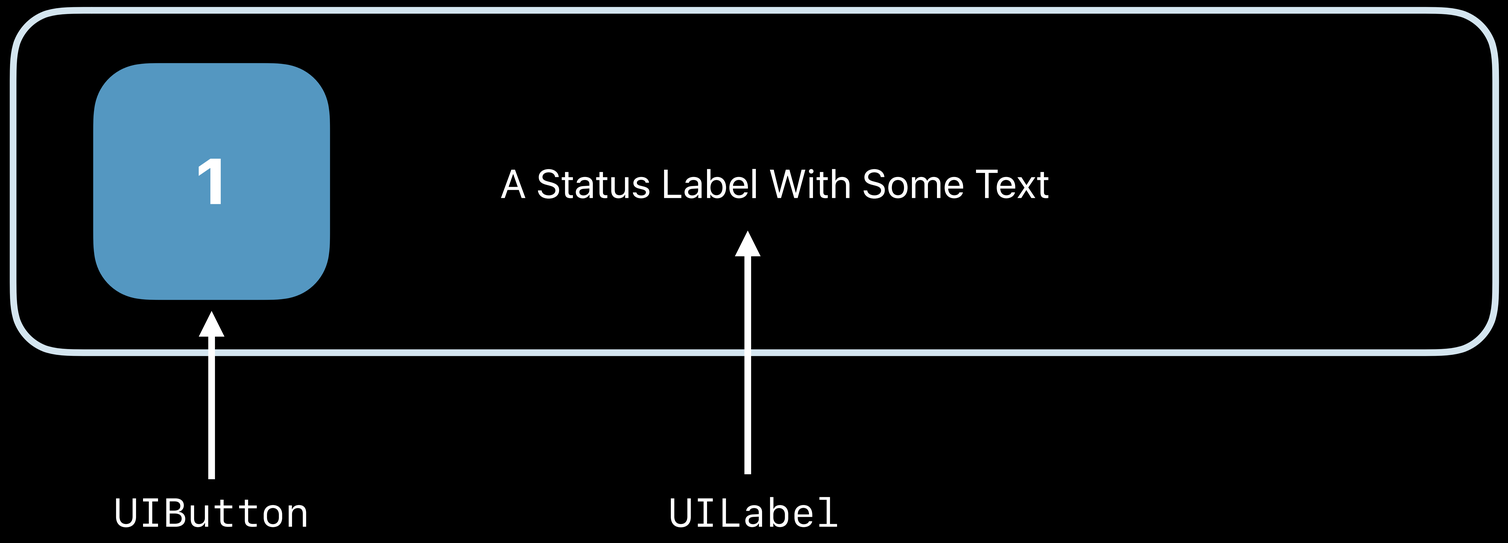
```
protocol UILargeContentViewerInteractionDelegate : NSObjectProtocol {

    optional func largeContentViewerInteraction(_ interaction:
UILargeContentViewerInteraction, didEndOn item: UILargeContentViewerItem?, at point: CGPoint)

    optional func largeContentViewerInteraction(_ interaction:
UILargeContentViewerInteraction, itemAt point: CGPoint) -> UILargeContentViewerItem?

    optional func viewController(for interaction: UILargeContentViewerInteraction) ->
UIViewController

}
```

NEW

```
protocol UILargeContentViewerInteractionDelegate : NSObjectProtocol {

    optional func largeContentViewerInteraction(_ interaction:
UILargeContentViewerInteraction, didEndOn item: UILargeContentViewerItem?, at point: CGPoint)

    optional func largeContentViewerInteraction(_ interaction:
UILargeContentViewerInteraction, itemAt point: CGPoint) -> UILargeContentViewerItem?

    optional func viewController(for interaction: UILargeContentViewerInteraction) ->
UIViewController

}
```

Examples

Custom Bar with Standard Views



Custom Bar with Standard Views

```
button.showsLargeContentViewer = true
label.showsLargeContentViewer = true
customBar.addInteraction(UILargeContentViewerInteraction())
```

Custom Bar with Standard Views

```
button.showsLargeContentViewer = true
label.showsLargeContentViewer = true
customBar.addInteraction(UILargeContentViewerInteraction())
```

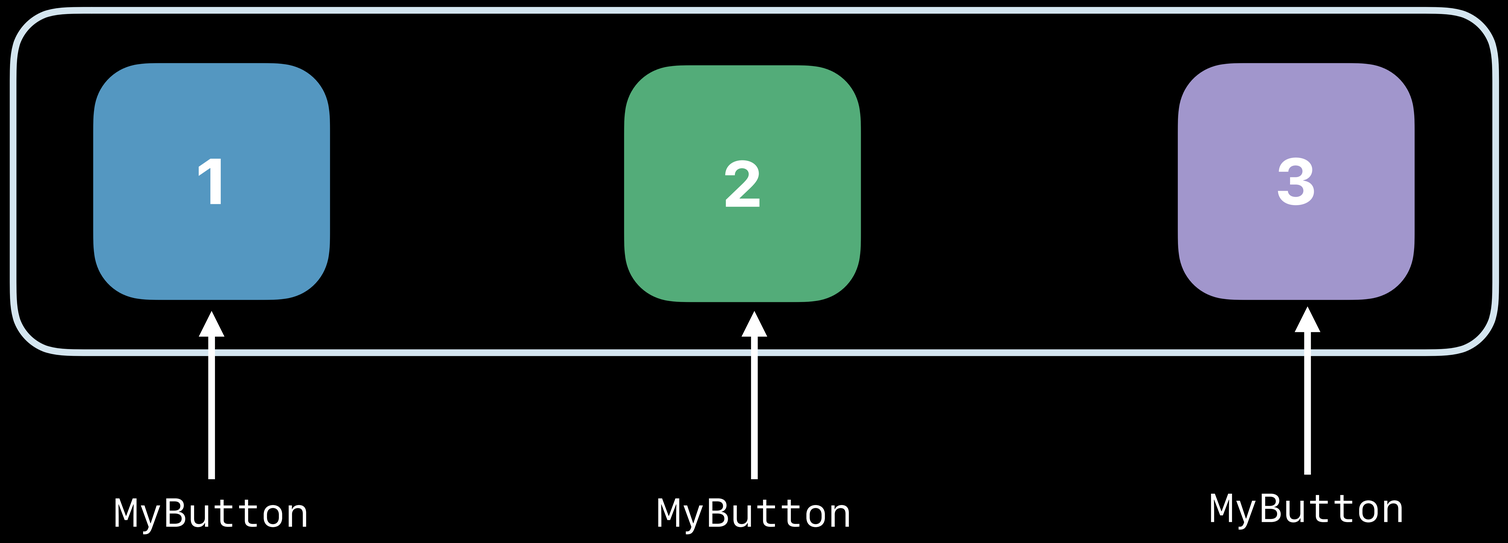
Custom Bar with Standard Views

```
button.showsLargeContentViewer = true
label.showsLargeContentViewer = true
customBar.addInteraction(UILargeContentViewerInteraction())
```

Custom Bar with Standard Views

```
button.showsLargeContentViewer = true
label.showsLargeContentViewer = true
customBar.addInteraction(UILargeContentViewerInteraction())
```

Custom Bar with Custom Views



Custom Bar with Custom Views

```
class MyCustomButton: UIView {
    override var showsLargeContentViewer: Bool {
        get {
            return true
        }
        set { }
    }
    override var largeContentTitle: String? {
        get {
            return myTitle
        }
        set { }
    }
```

Custom Bar with Custom Views

```
class MyCustomButton: UIView {
    override var showsLargeContentViewer: Bool {
        get {
            return true
        }
        set { }
    }
    override var largeContentTitle: String? {
        get {
            return myTitle
        }
        set { }
    }
```

Custom Bar with Custom Views

```
class MyCustomButton: UIView {
    override var showsLargeContentViewer: Bool {
        get {
            return true
        }
        set { }
    }
    override var largeContentTitle: String? {
        get {
            return myTitle
        }
        set { }
    }
```

Custom Bar with Custom Views

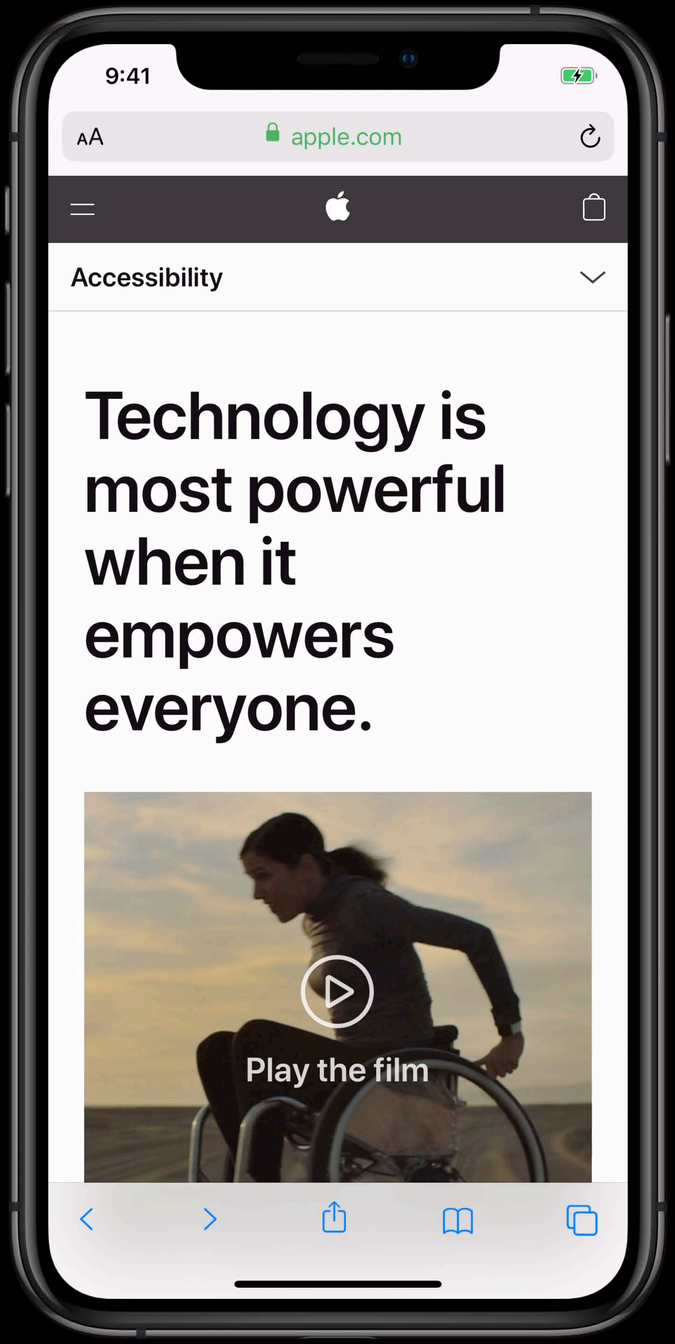
```
    override var largeContentImage: UIImage? {
        get {
            return mySmallImage
        }
        set { }
    }

    override var scalesLargeContentImage: Bool {
        get {
            return true
        }
        set { }
    }
}
```

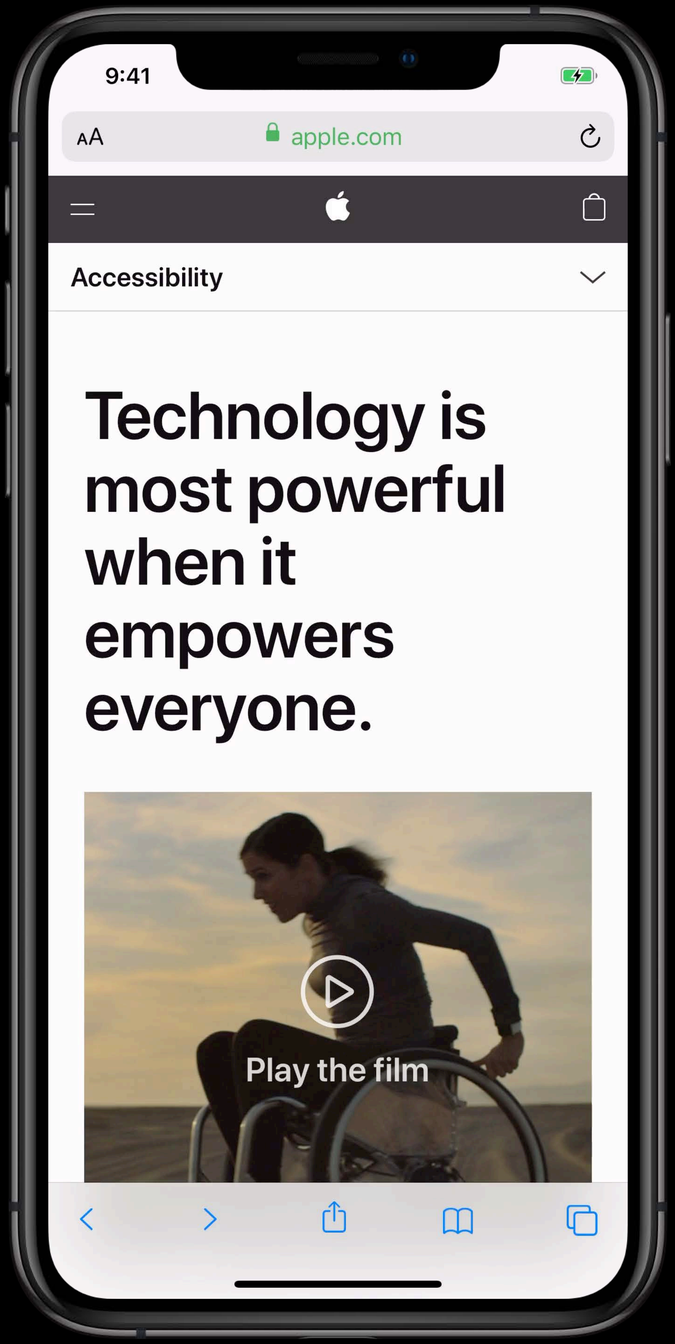
Custom Bar with Custom Views

```
    override var largeContentImage: UIImage? {
        get {
            return mySmallImage
        }
        set { }
    }

    override var scalesLargeContentImage: Bool {
        get {
            return true
        }
        set { }
    }
}
```

Custom Bar with Custom Views

```
    override var largeContentImage: UIImage? {
        get {
            return mySmallImage
        }
        set { }
    }

    override var scalesLargeContentImage: Bool {
        get {
            return true
        }
        set { }
    }
}
```


9:41
AA
apple.com
Accessibility
Technology is most powerful when it empowers everyone.
Play the film


9:41
AA
apple.com
Accessibility
Technology is most powerful when it empowers everyone.
Play the film

Button with Existing Long Press Action

```
longPressRecognizer.minimumPressDuration =
    UILargeContentViewerInteraction.isEnabled ? 3.0 : 1.0;

NotificationCenter.default.addObserver(forName:
    UILargeContentViewerInteraction.enabledStatusDidChangeNotification,
                                       object: nil, queue: nil) { _ in
        longPressRecognizer.minimumPressDuration =
            UILargeContentViewerInteraction.isEnabled ? 3.0 : 1.0;
}

longPressRecognizer.delegate = self
```

Button with Existing Long Press Action

```
longPressRecognizer.minimumPressDuration =
    UILargeContentViewerInteraction.isEnabled ? 3.0 : 1.0;

NotificationCenter.default.addObserver(forName:
    UILargeContentViewerInteraction.enabledStatusDidChangeNotification,
                                       object: nil, queue: nil) { _ in
        longPressRecognizer.minimumPressDuration =
            UILargeContentViewerInteraction.isEnabled ? 3.0 : 1.0;
}

longPressRecognizer.delegate = self
```

Button with Existing Long Press Action

```
longPressRecognizer.minimumPressDuration =
    UILargeContentViewerInteraction.isEnabled ? 3.0 : 1.0;

NotificationCenter.default.addObserver(forName:
    UILargeContentViewerInteraction.enabledStatusDidChangeNotification,
                                       object: nil, queue: nil) { _ in
        longPressRecognizer.minimumPressDuration =
            UILargeContentViewerInteraction.isEnabled ? 3.0 : 1.0;
}

longPressRecognizer.delegate = self
```

Button with Existing Long Press Action

```
longPressRecognizer.minimumPressDuration =
    UILargeContentViewerInteraction.isEnabled ? 3.0 : 1.0;

NotificationCenter.default.addObserver(forName:
    UILargeContentViewerInteraction.enabledStatusDidChangeNotification,
                                       object: nil, queue: nil) { _ in
        longPressRecognizer.minimumPressDuration =
            UILargeContentViewerInteraction.isEnabled ? 3.0 : 1.0;
}

longPressRecognizer.delegate = self
```

Button with Existing Long Press Action

```
func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
    shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer) -> Bool {

    return gestureRecognizer == longPressRecognizer && otherGestureRecognizer ==
        largeContentViewerInteraction.gestureRecognizerForExclusiveRelationship

}
```

Button with Existing Long Press Action

```
func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
    shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer) -> Bool {

    return gestureRecognizer == longPressRecognizer && otherGestureRecognizer ==
        largeContentViewerInteraction.gestureRecognizerForExclusiveRelationship

}
```

9:41

AA

apple.com

iPad Pro

iPad Air
New

iPad

iPad mini
New

Compare

Welcome to the new generation of iPad.

iPad Pro

The most advanced iPad ever

9:41

AA

apple.com

iPad Pro

iPad Air
New

iPad

iPad mini
New

Compare

Welcome to the new generation of iPad.

iPad Pro

The most advanced iPad ever

Summary

Summary

Accommodating customers with low vision

Summary

Accommodating customers with low vision

Using Large Content Viewer when Dynamic Type is not an option

Summary

Accommodating customers with low vision

Using Large Content Viewer when Dynamic Type is not an option

Ensure all interactions with Custom Views

More Information

developer.apple.com/wwdc19/261


WWDC19