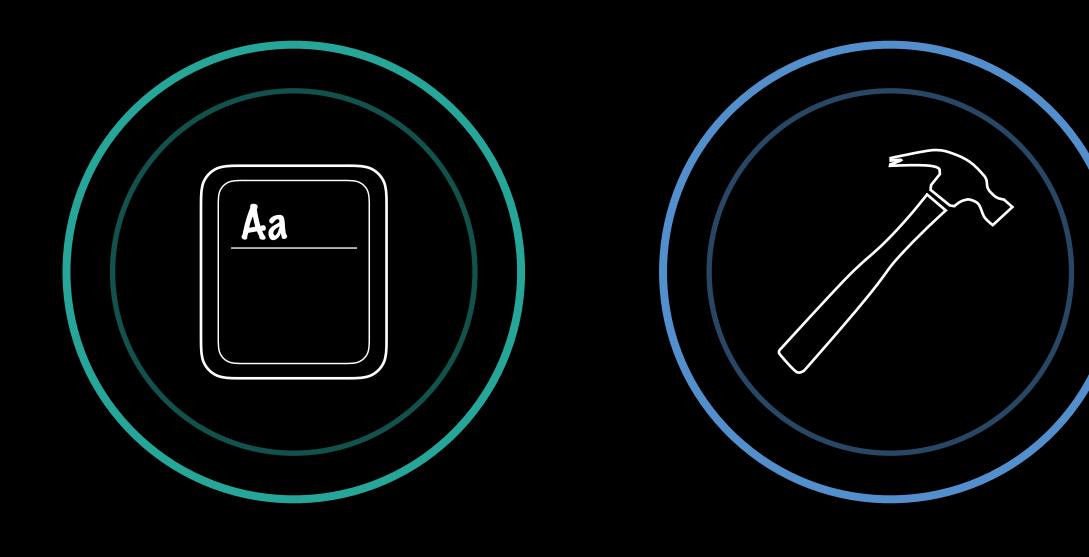
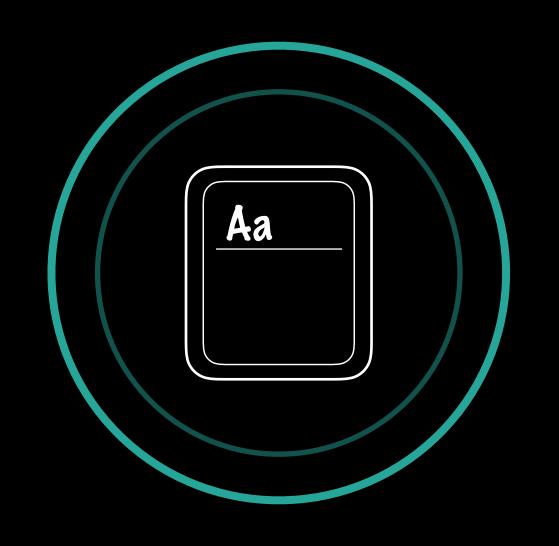
Making Apps Adaptive, Part 2

Session 233

David Duncan iOS Apps and Frameworks
Kurt Revis iOS Apps and Frameworks



PART 1 ————

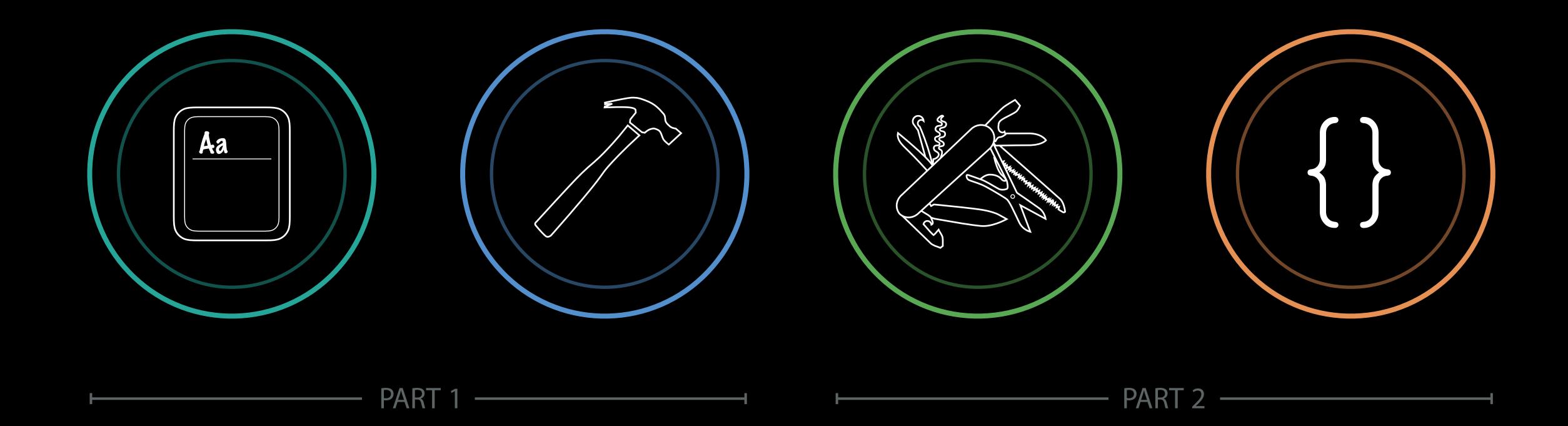






PART 1 ———

—— PART 2 ———————



Basics of Sizes and Size Classes

Basics of Sizes and Size Classes

How to Get the Most Out of UlKit

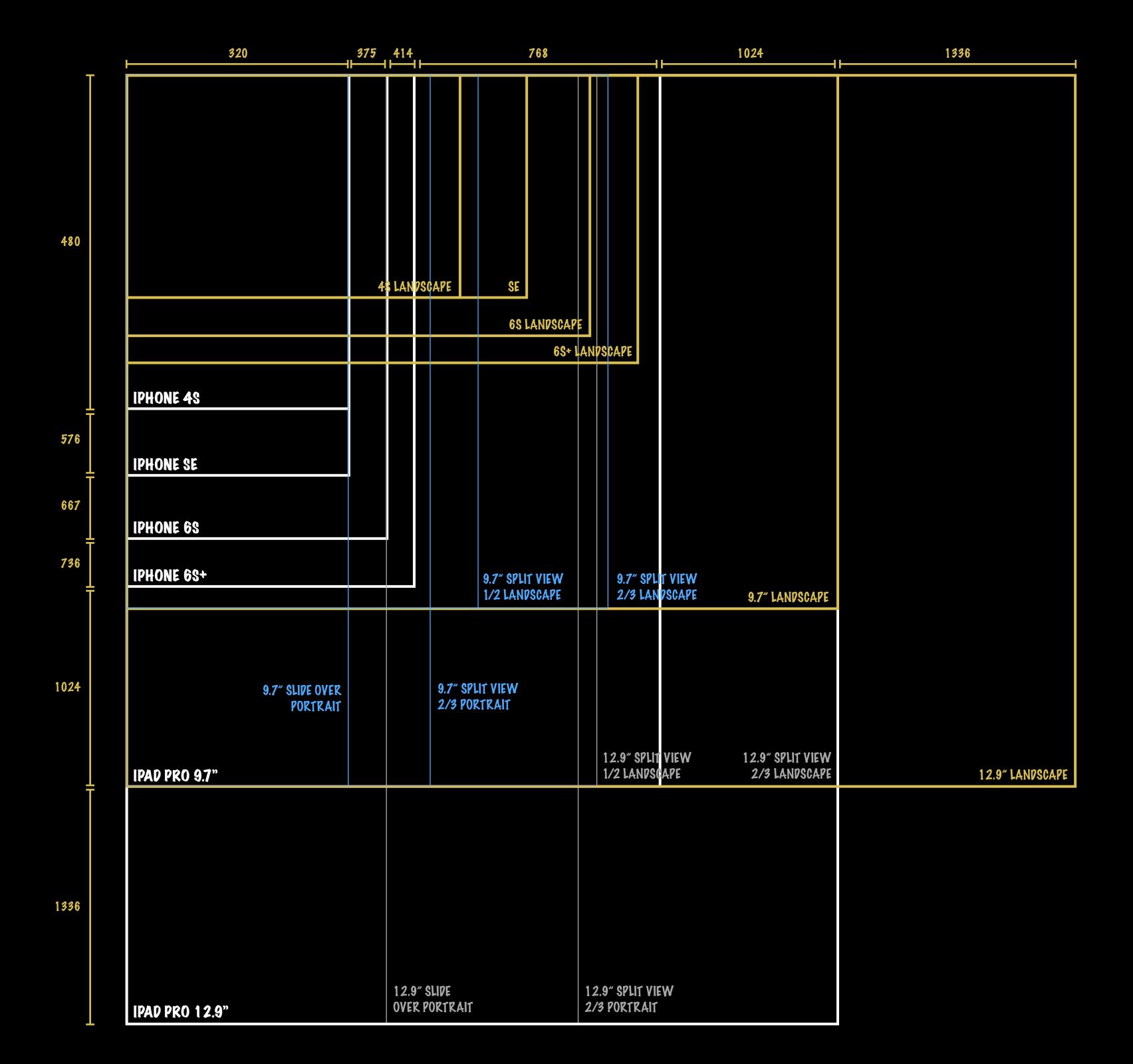
Basics of Sizes and Size Classes

How to Get the Most Out of UlKit

Going Beyond Size Classes to Build Your Custom Experience

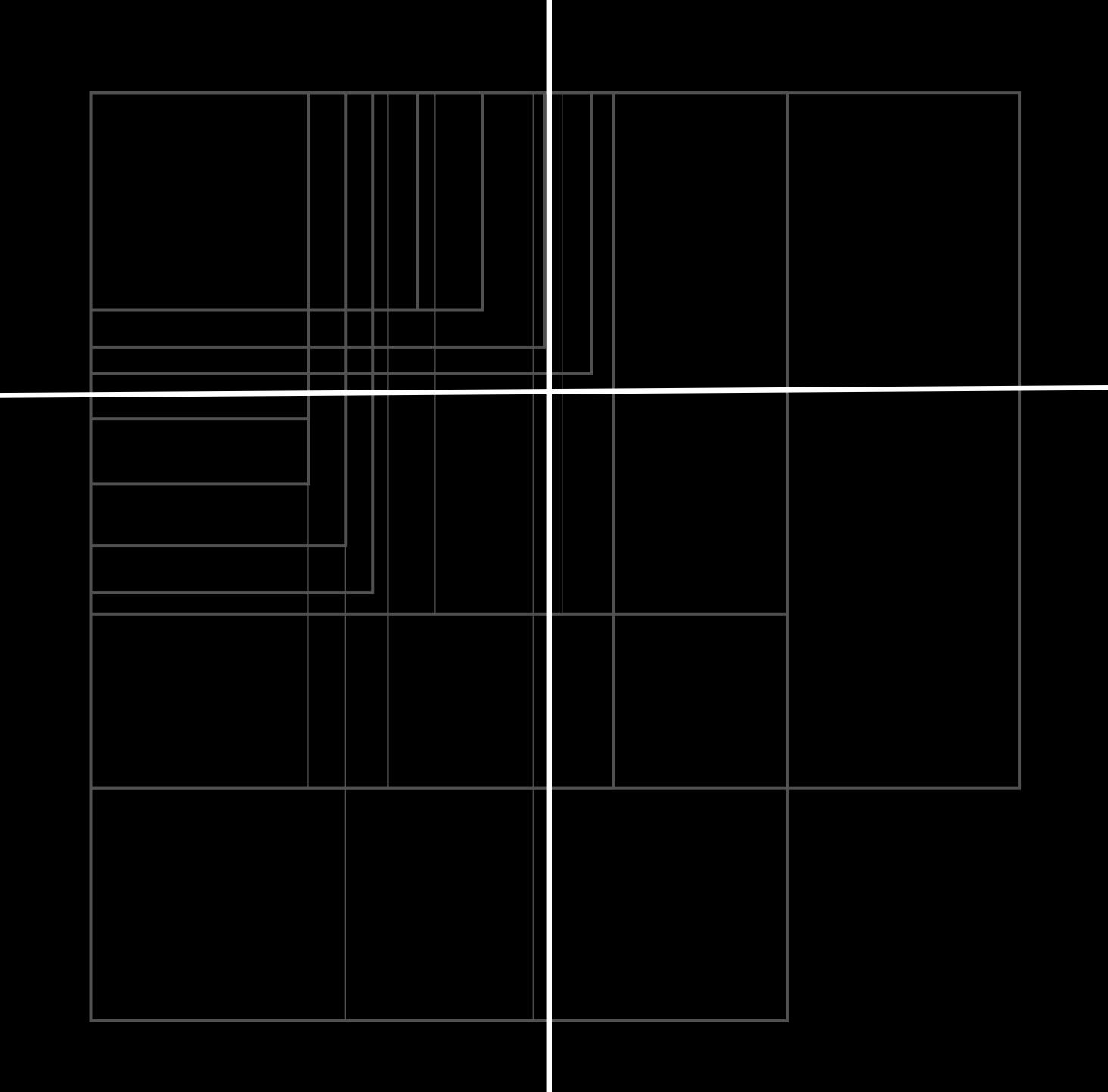
Sizes





			_			
			+			

コ	Т	\Box	$\ \ $				
				4			

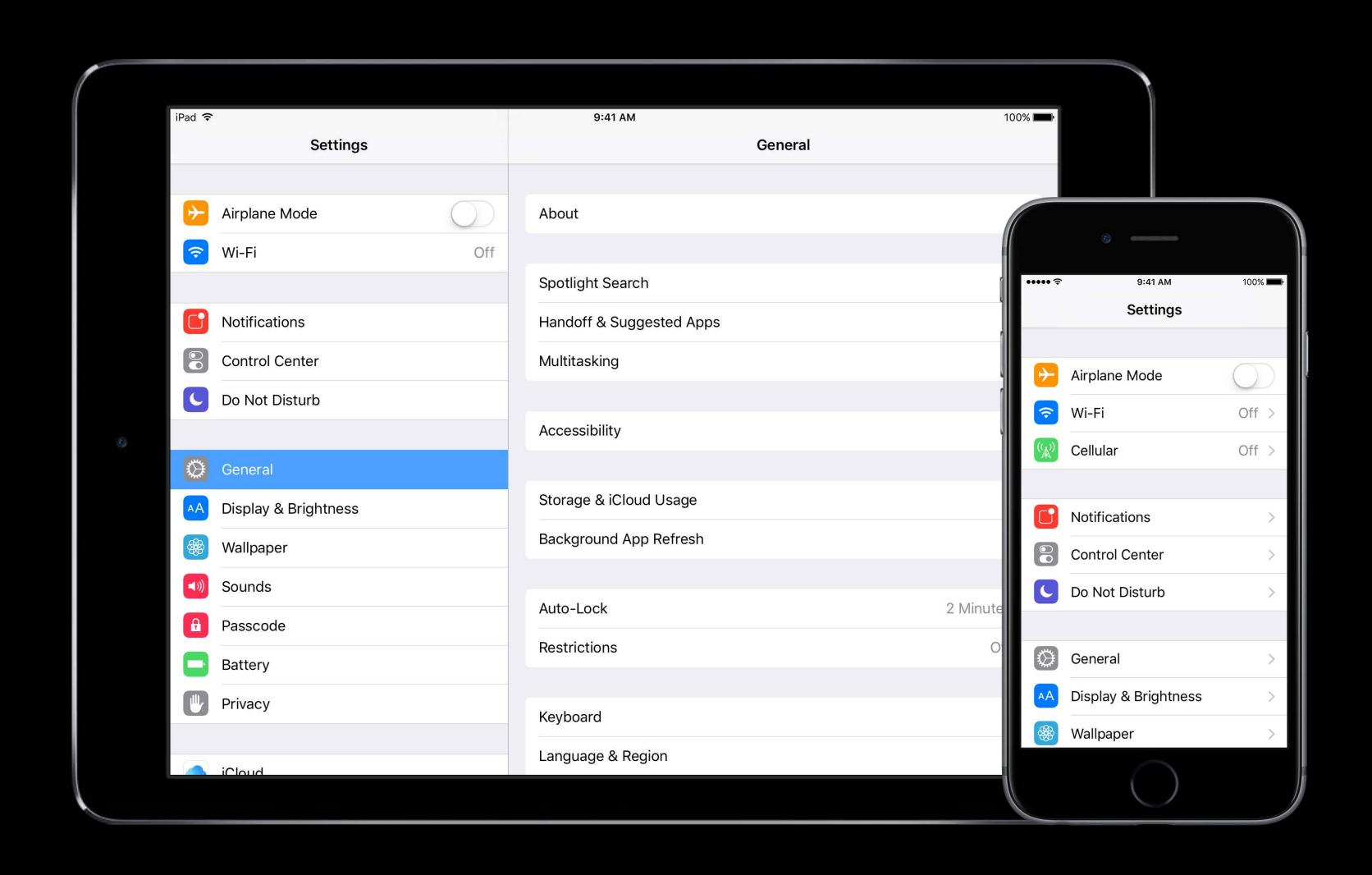


	Compact Width	Regular Width						
Compact Height								
Regular Height								

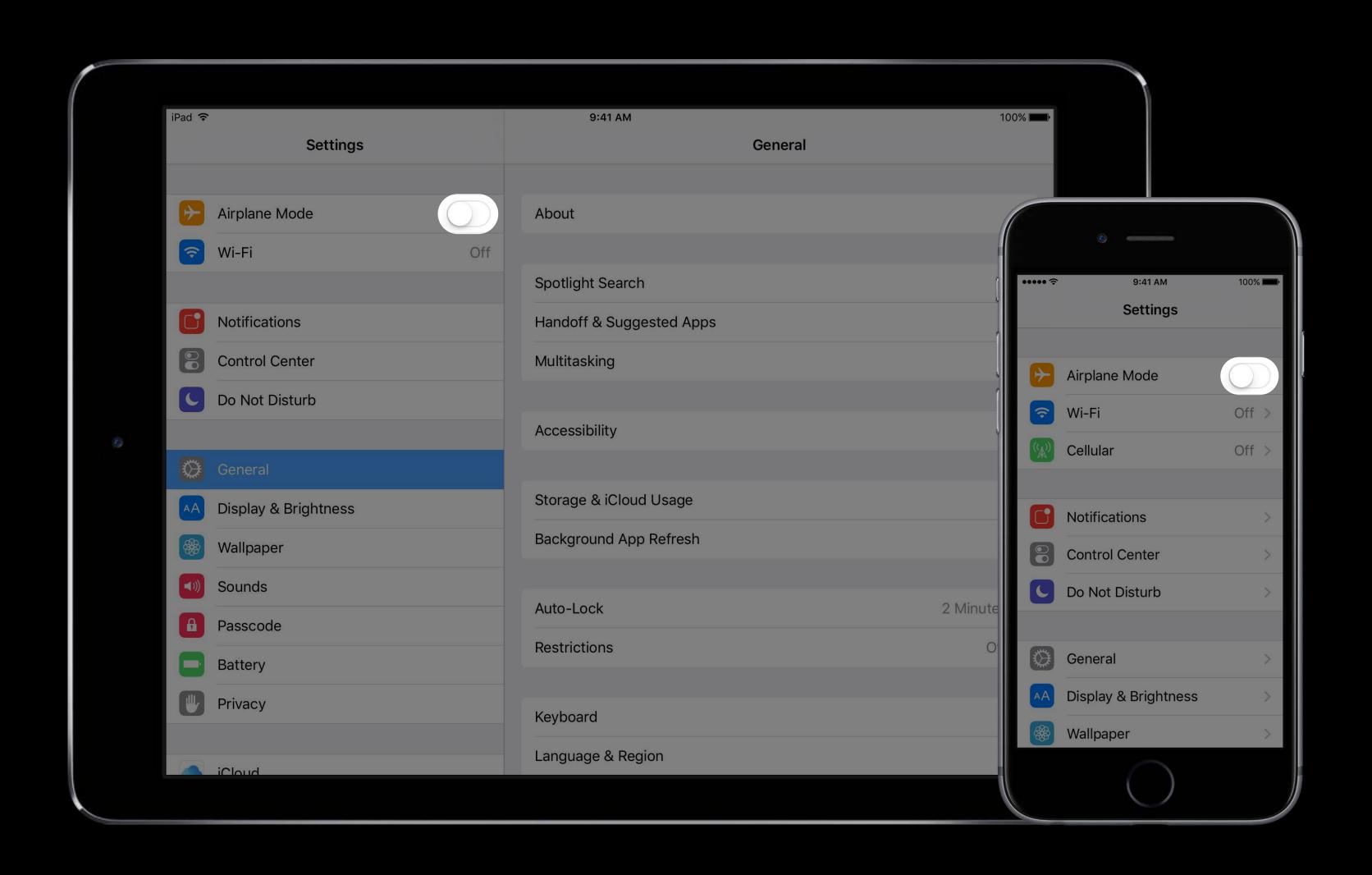
Regular Size Class

Your experience to make

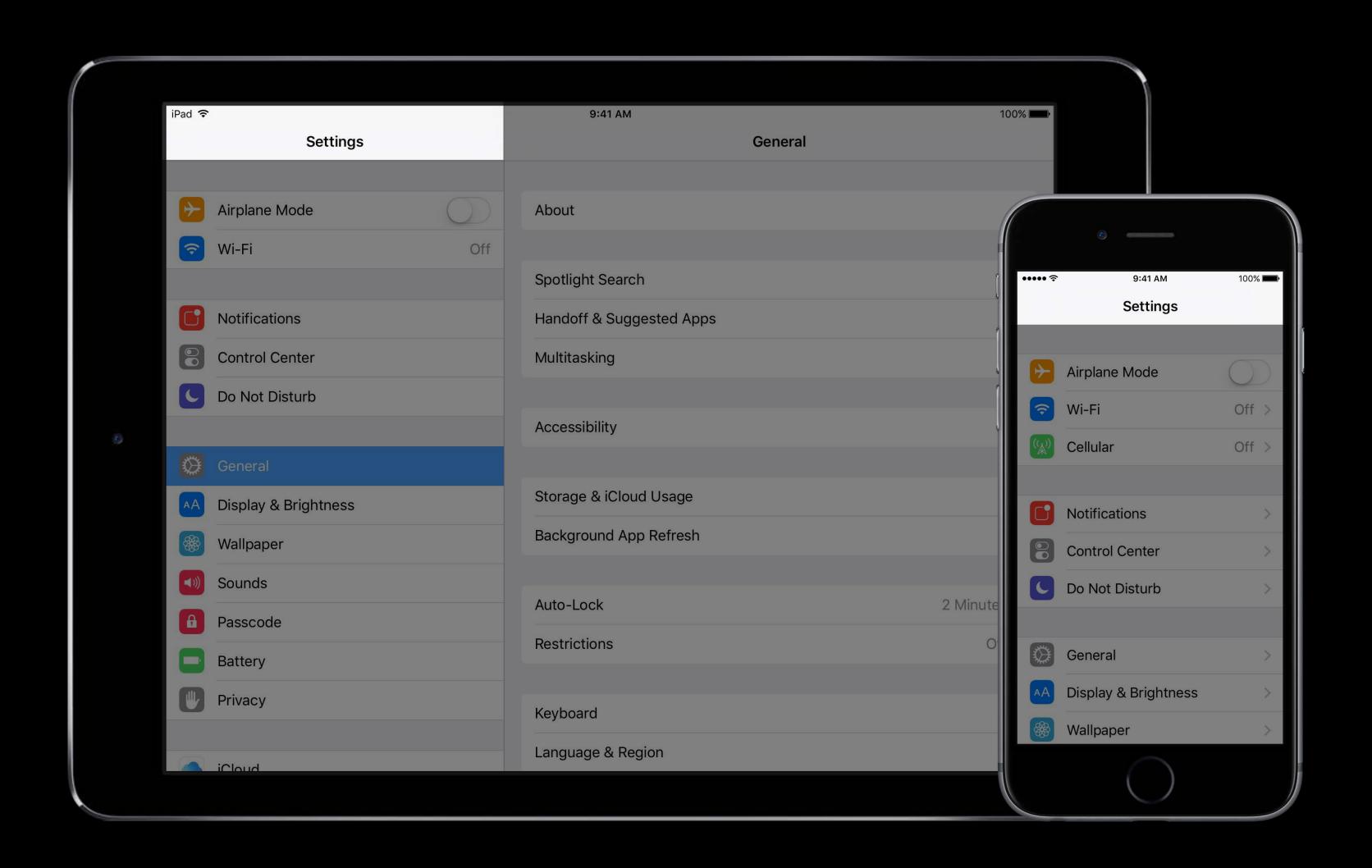
Views and Controls



Views and Controls



Views and Controls

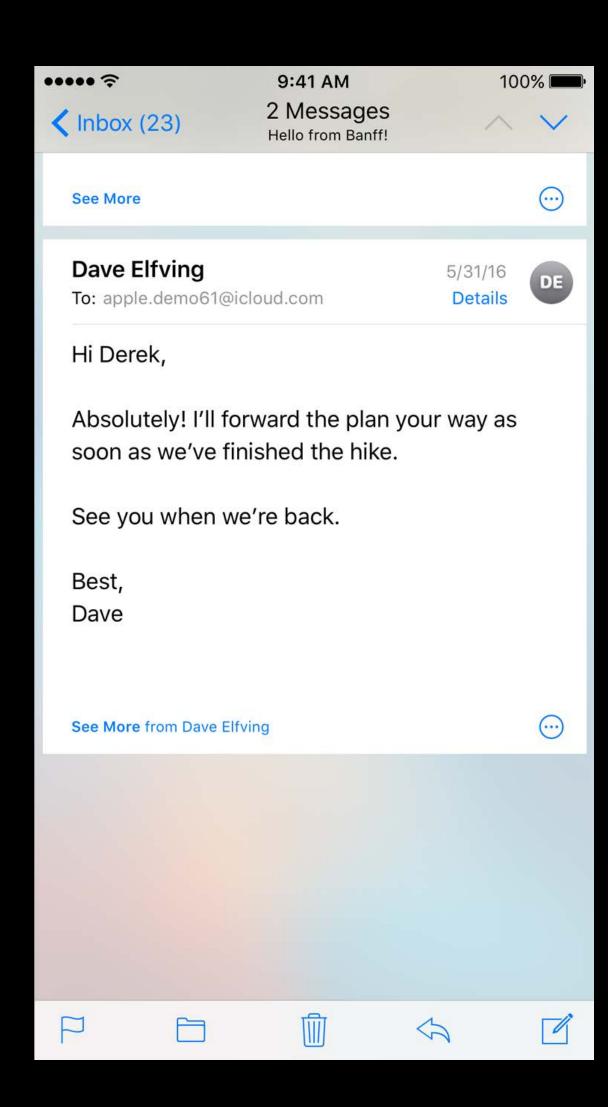


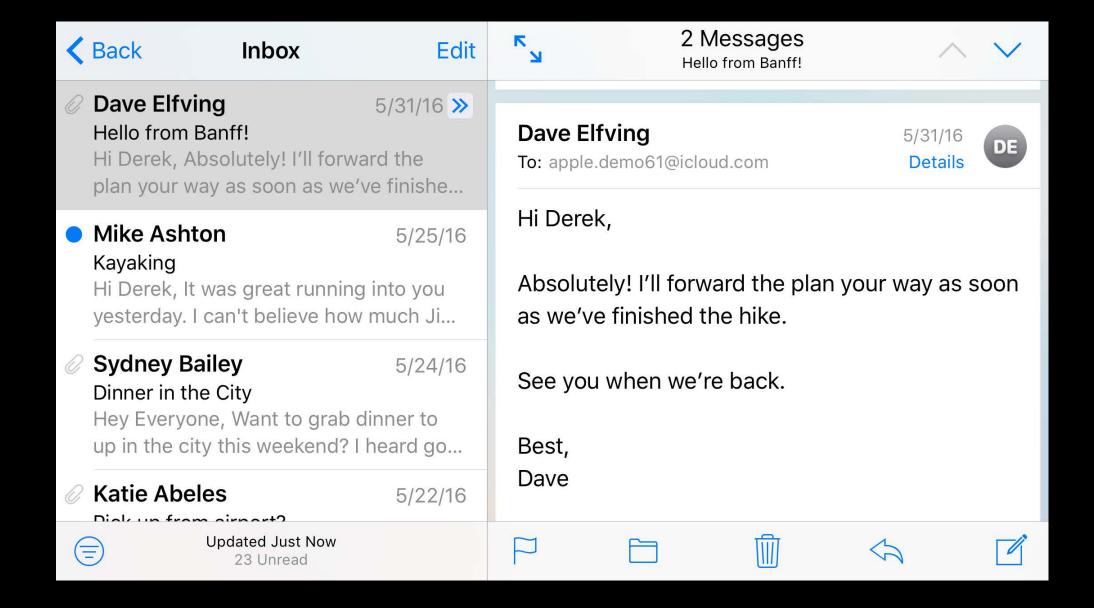
Presentations

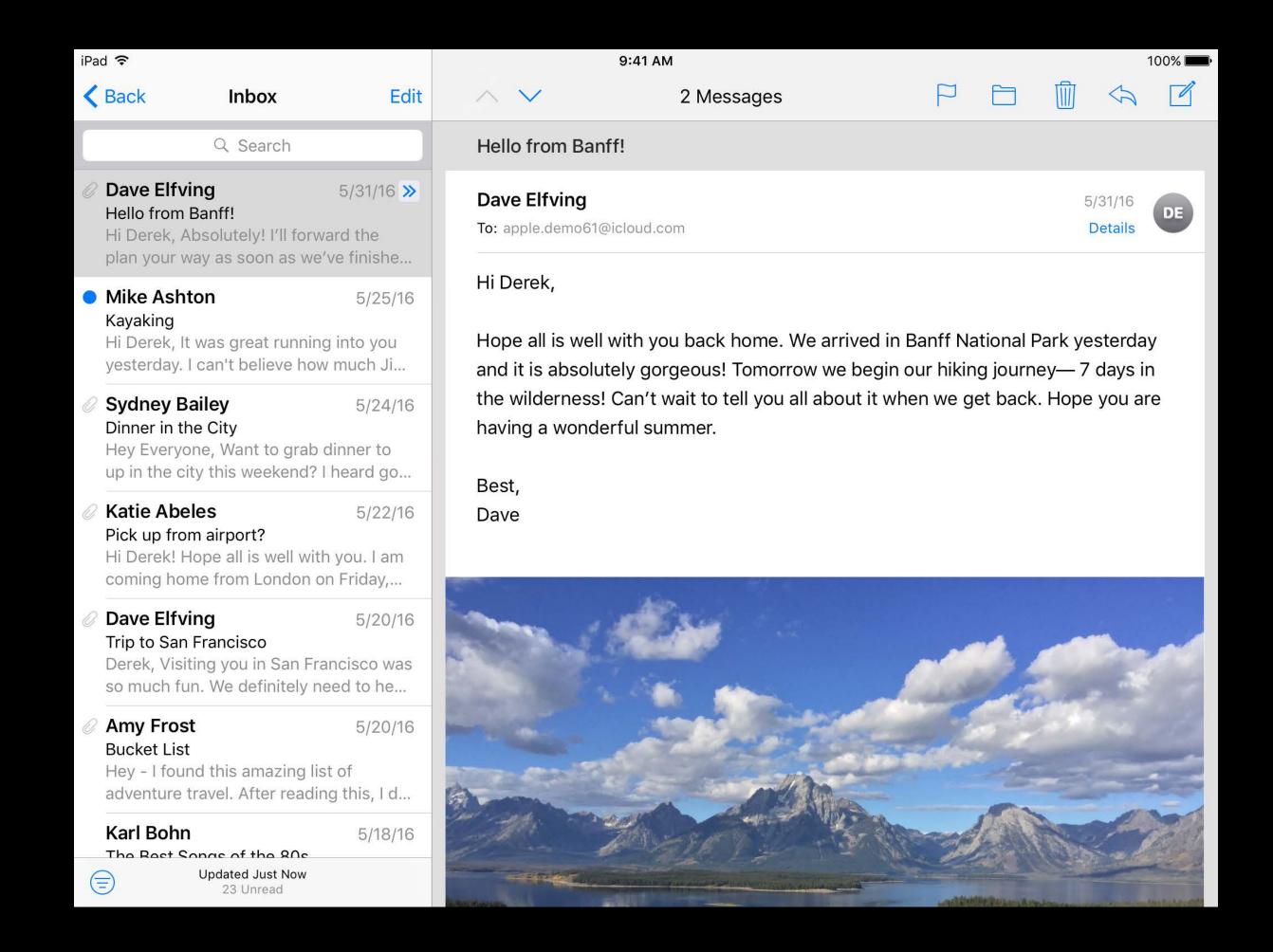
•••• ∻	9:41 AM	100%						
Cancel	Add New Keyboard							
SUGGESTED K	(EYBOARDS							
English								
OTHER IPHON	E KEYBOARDS							
English (Aus	English (Australia)							
English (Ca	nada)							
English (Ind	lia)							
English (Japan)								
English (Sin	gapore)							
English (UK	()							
Arabic								
Bengali								
Bulgarian								
Catalan								
Cherokee								
Obines (Ci								

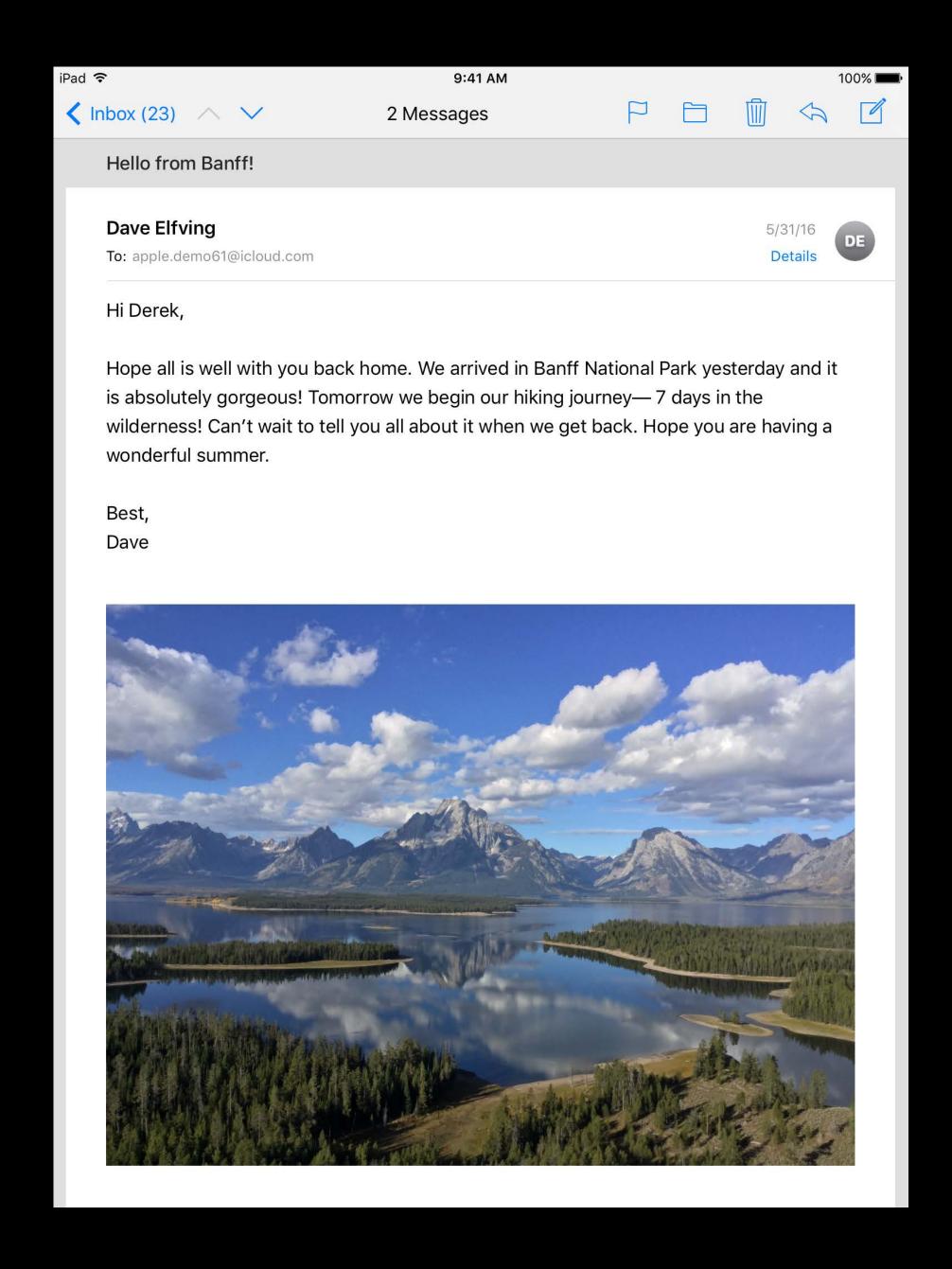
Presentations

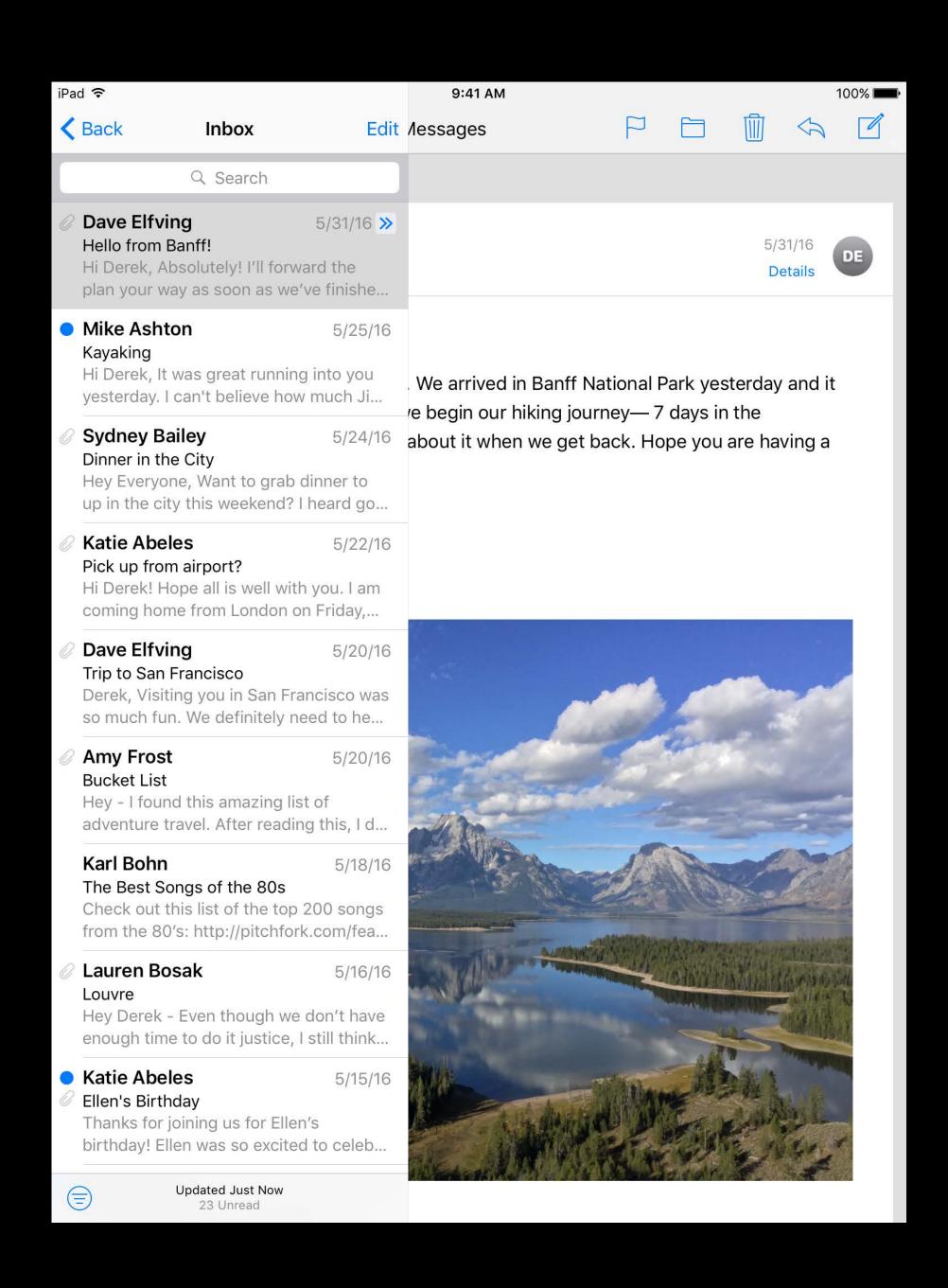


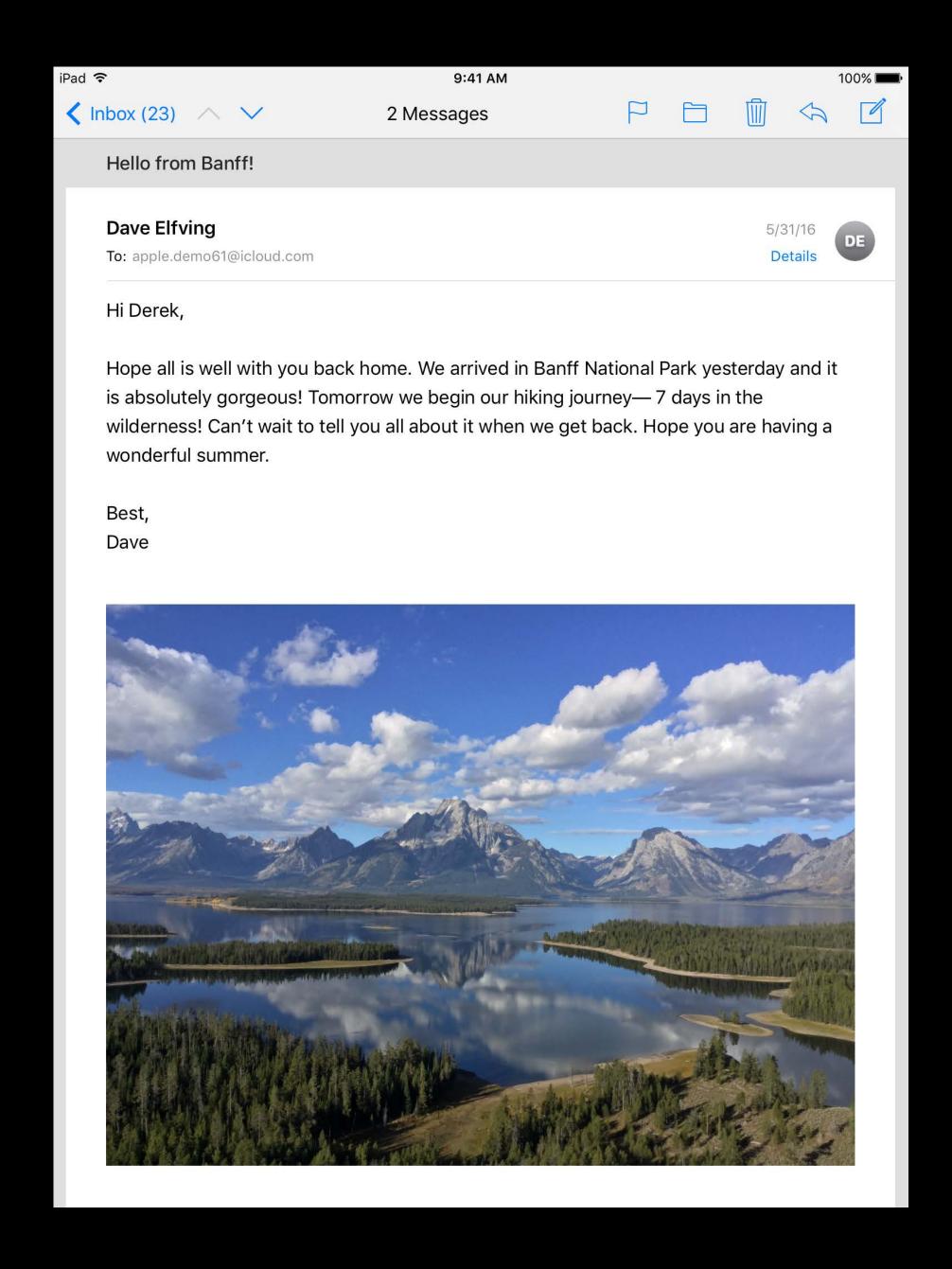












Making the most of UlKit and Xcode

Xcode tools to get your app looking great faster

Xcode tools to get your app looking great faster

Interface Builder

Xcode tools to get your app looking great faster

- Interface Builder
- Asset Catalogs

Xcode tools to get your app looking great faster

- Interface Builder
- Asset Catalogs

UlKit functionality to make it all easier

Xcode tools to get your app looking great faster

- Interface Builder
- Asset Catalogs

UlKit functionality to make it all easier

Auto Layout

Xcode tools to get your app looking great faster

- Interface Builder
- Asset Catalogs

UlKit functionality to make it all easier

- Auto Layout
- UlTraitCollection

Xcode tools to get your app looking great faster

- Interface Builder
- Asset Catalogs

UlKit functionality to make it all easier

- Auto Layout
- UlTraitCollection
- Dynamic Type

Best Practices

Xcode tools to get your app looking great faster

- Interface Builder
- Asset Catalogs

UlKit functionality to make it all easier

- Auto Layout
- UITraitCollection
- Dynamic Type
- Layout Guides

Best Practices

Xcode tools to get your app looking great faster

- Interface Builder
- Asset Catalogs

UlKit functionality to make it all easier

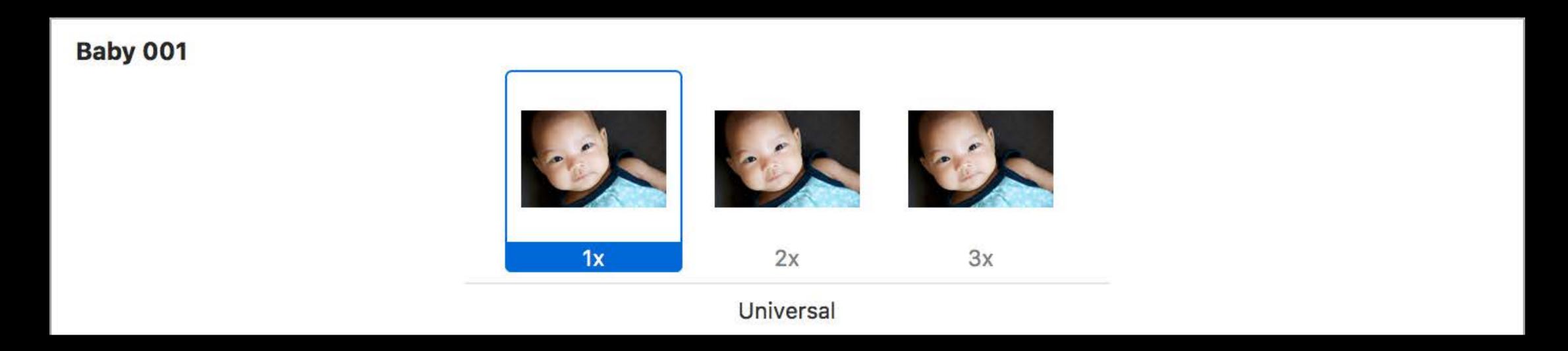
- Auto Layout
- UITraitCollection
- Dynamic Type
- Layout Guides
- UlAppearance

Asset Catalogs

Images and Traits

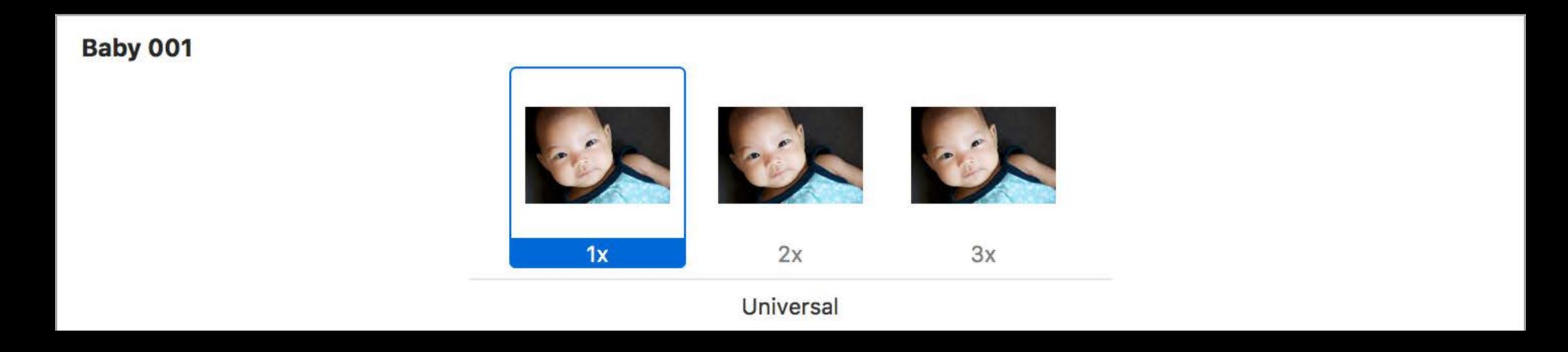
Asset Catalogs Images and Traits

Automatic image selection



Asset Catalogs Images and Traits

Automatic image selection



Design for Application Thinning

Asset Catalogs

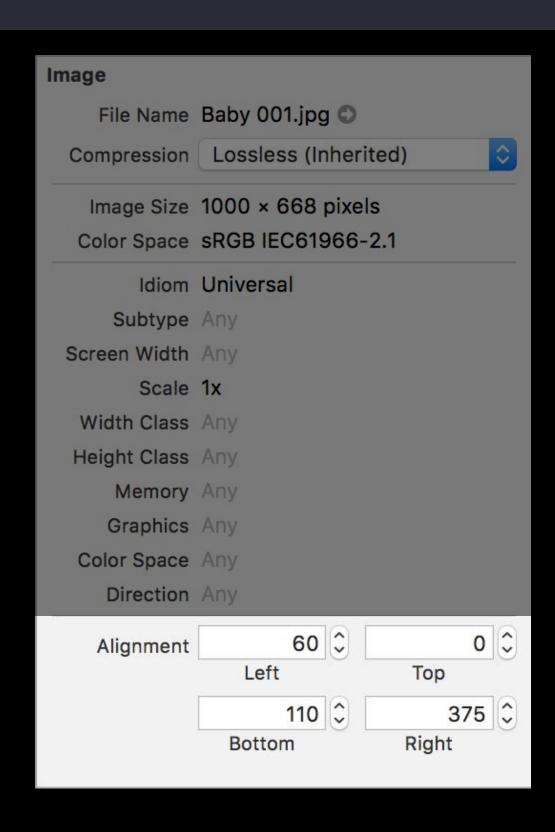
Alignment Insets



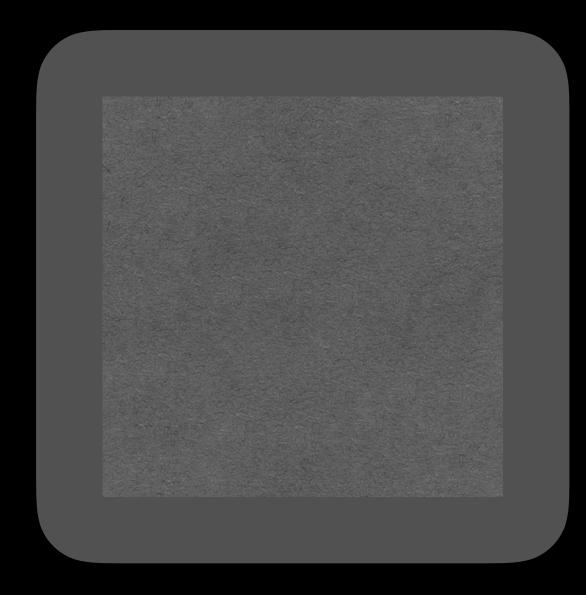


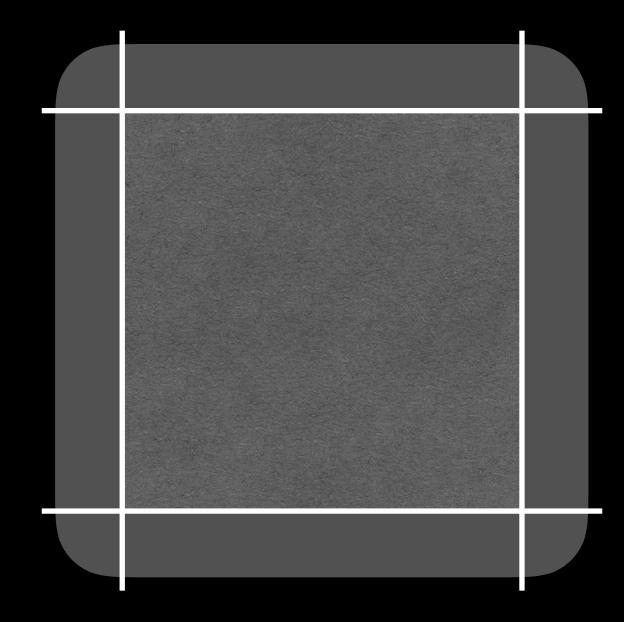


UIImage.withAlignmentRectInsets()
UIImage.alignmentRectInsets

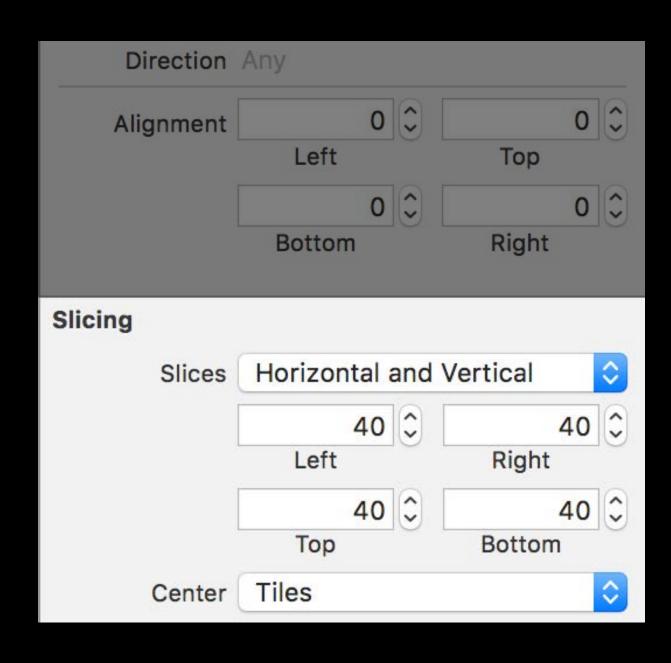


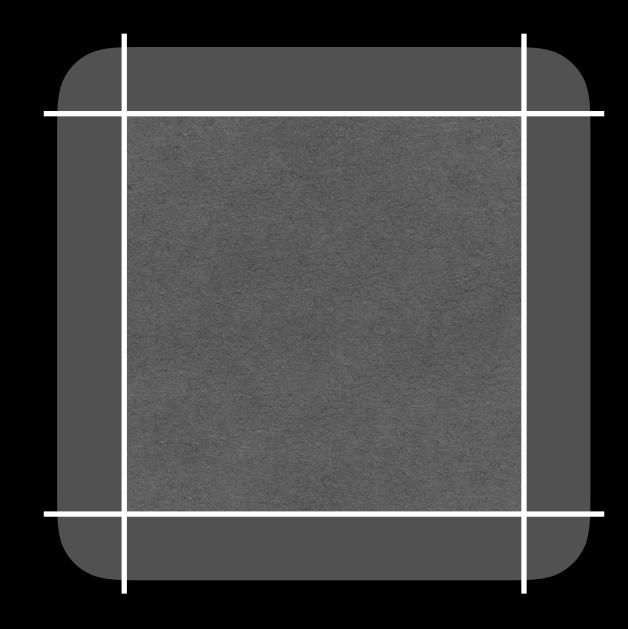




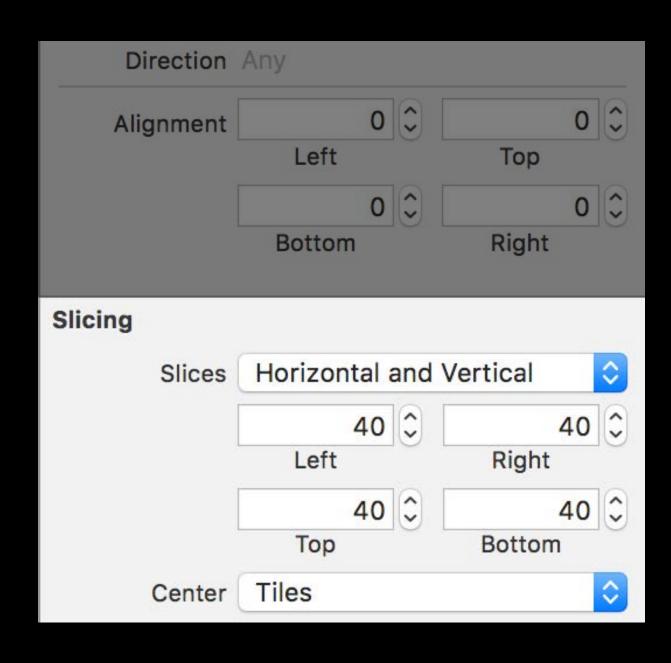


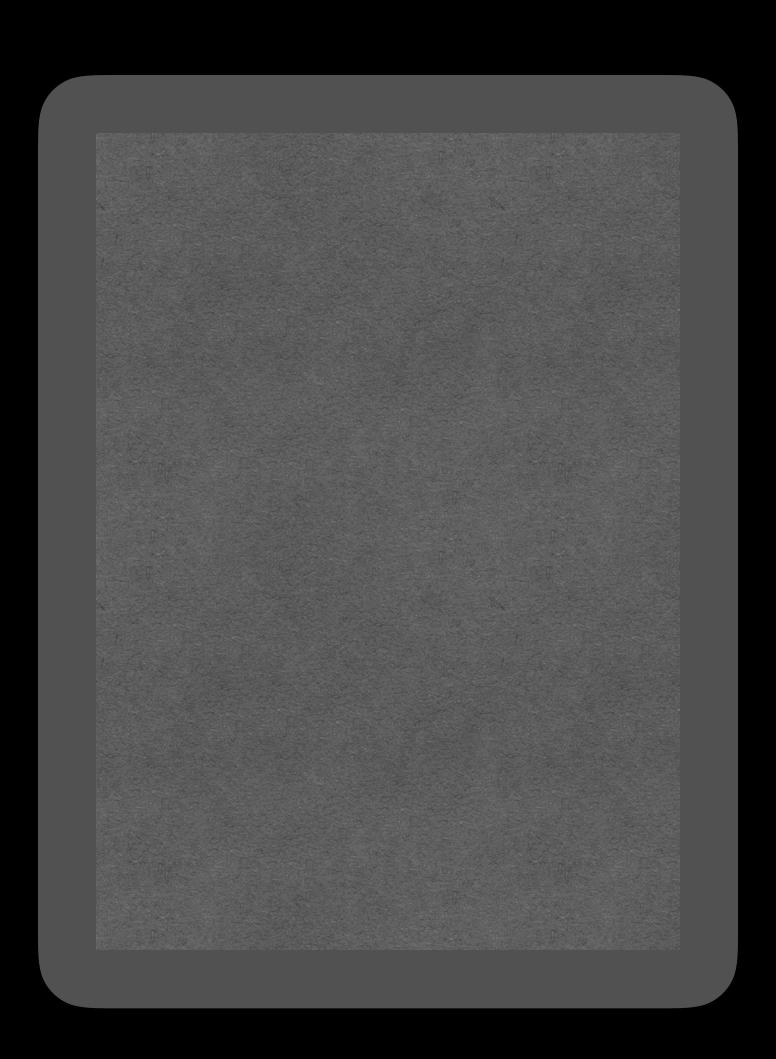
```
UIImage.resizableImage(withCapInsets:)
UIImage.resizableImage(withCapInsets: resizingMode:)
UIImage.capInsets & UIImage.resizingMode
```





```
UIImage.resizableImage(withCapInsets:)
UIImage.resizableImage(withCapInsets: resizingMode:)
UIImage.capInsets & UIImage.resizingMode
```





We've made it easier than ever to adopt



We've made it easier than ever to adopt

UITraitCollection.preferredContentSizeCategory



We've made it easier than ever to adopt

UITraitCollection.preferredContentSizeCategory

Set up your text views

- font = UIFont.preferredFont(forTextStyle:)
- adjustsFontForContentSizeCategory = true



We've made it easier than ever to adopt

UITraitCollection.preferredContentSizeCategory

Set up your text views

- font = UIFont.preferredFont(forTextStyle:)
- adjustsFontForContentSizeCategory = true

Be sure to test with all Dynamic Type sizes!



We've made it easier than ever to adopt

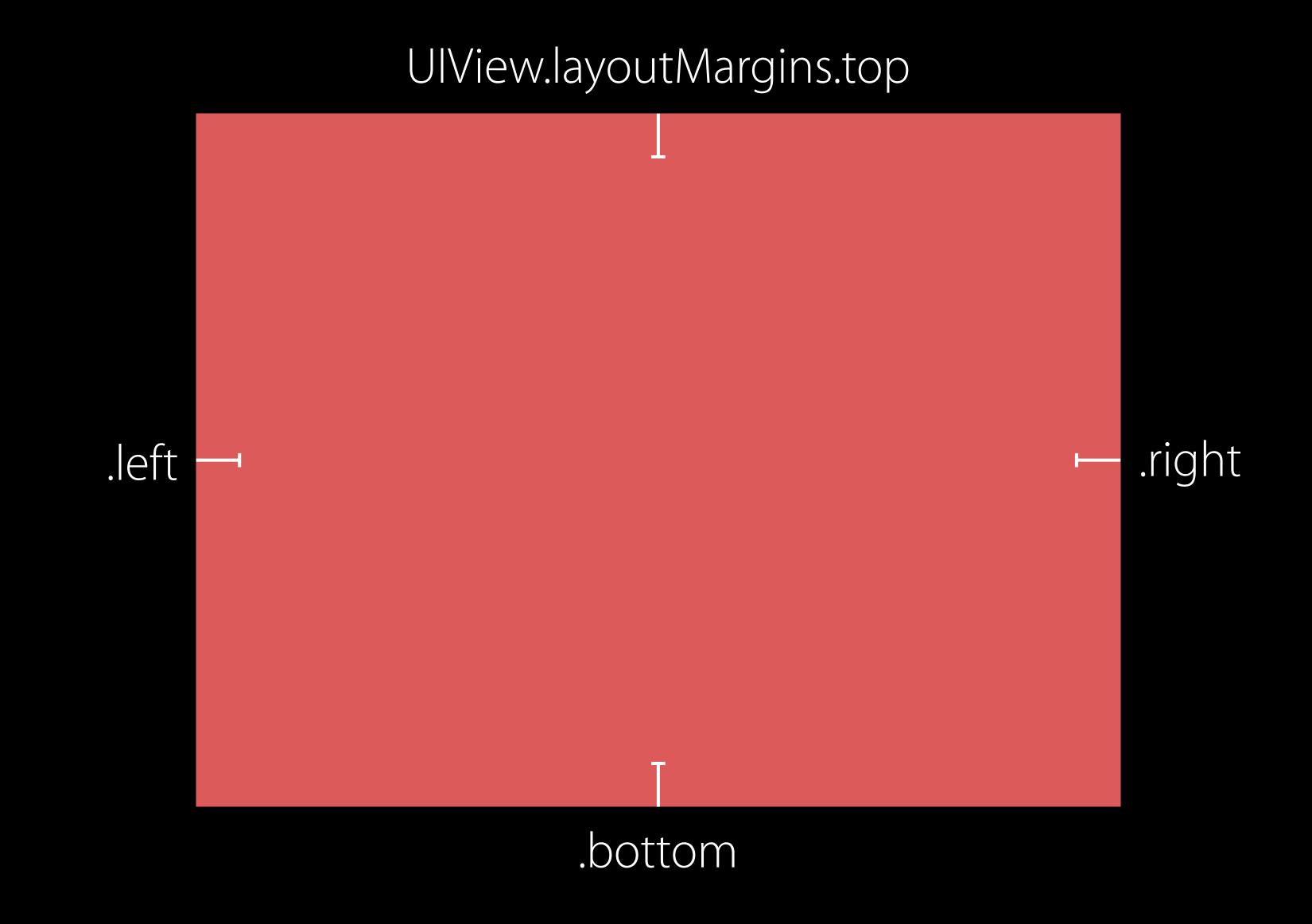
UITraitCollection.preferredContentSizeCategory

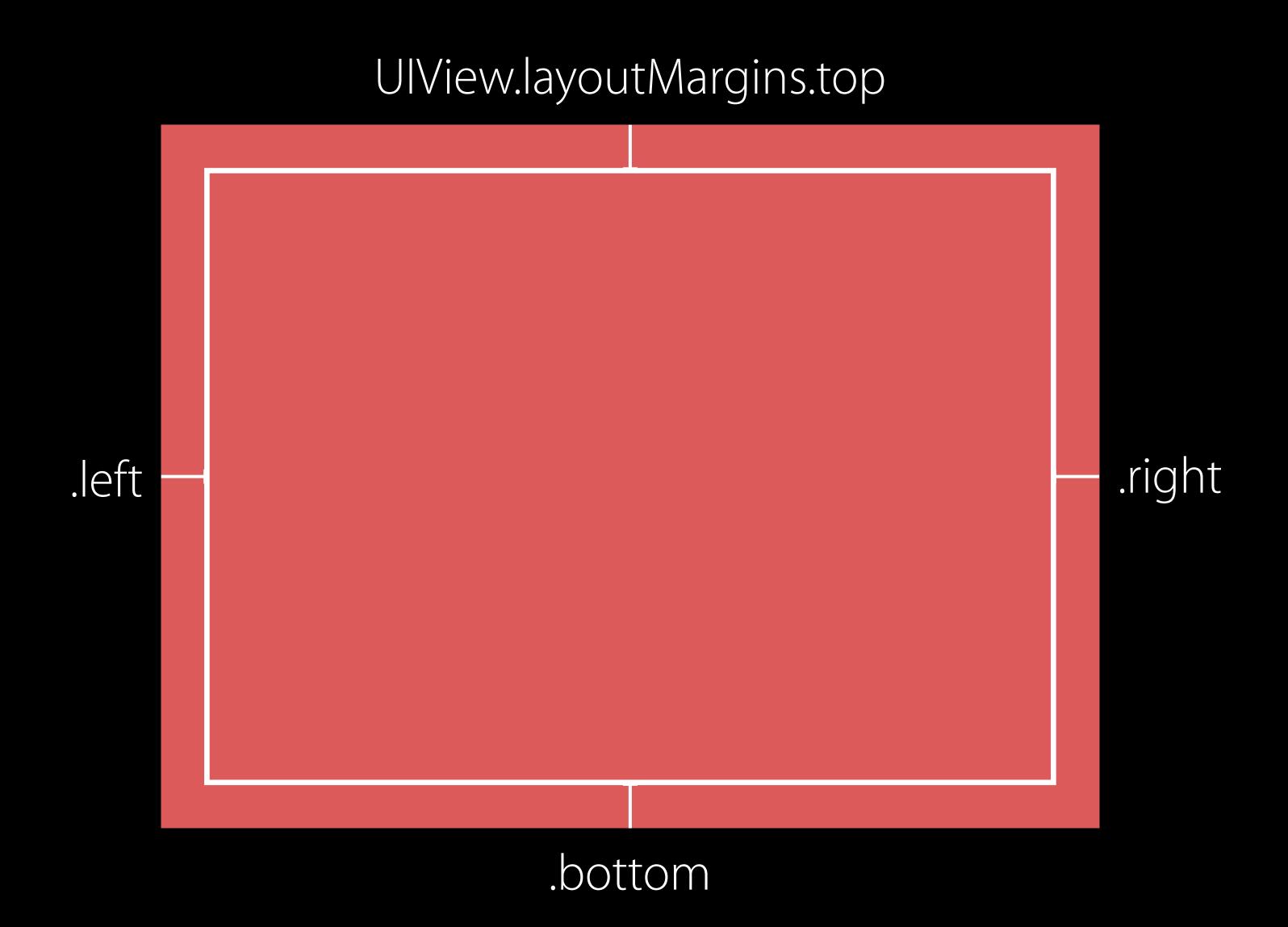
Set up your text views

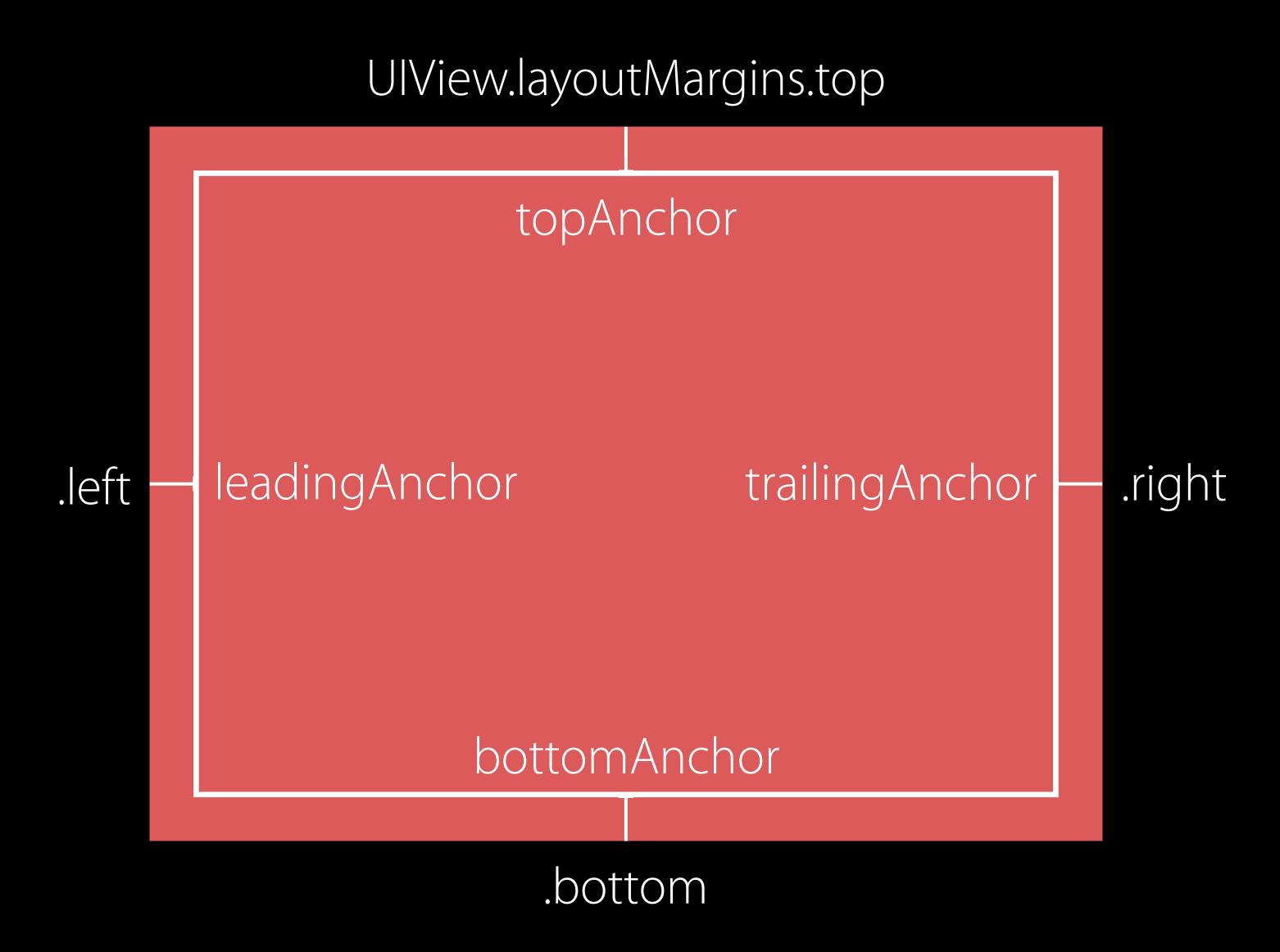
```
• font = UIFont.preferredFont(forTextStyle:)
```

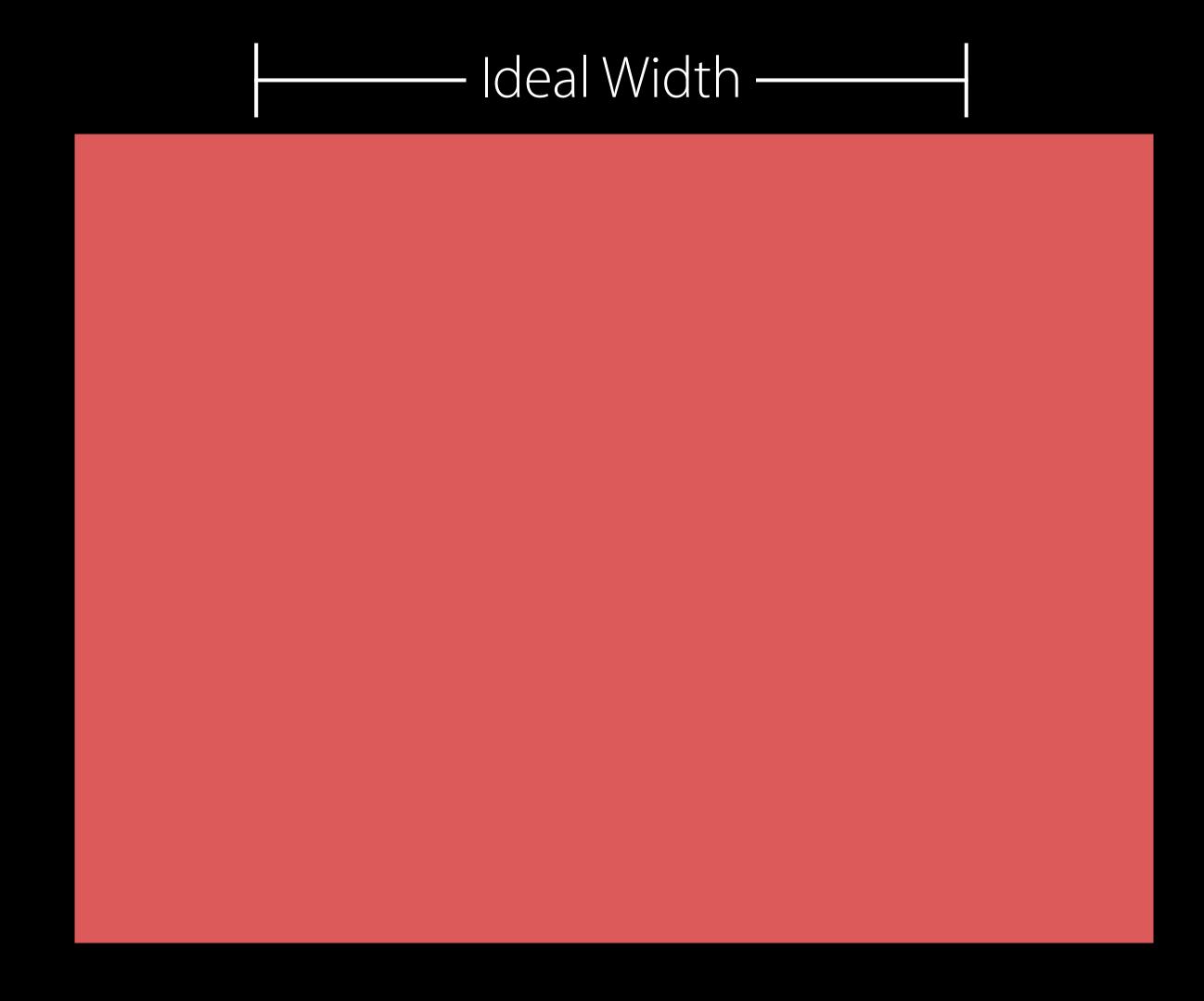
• adjustsFontForContentSizeCategory = true

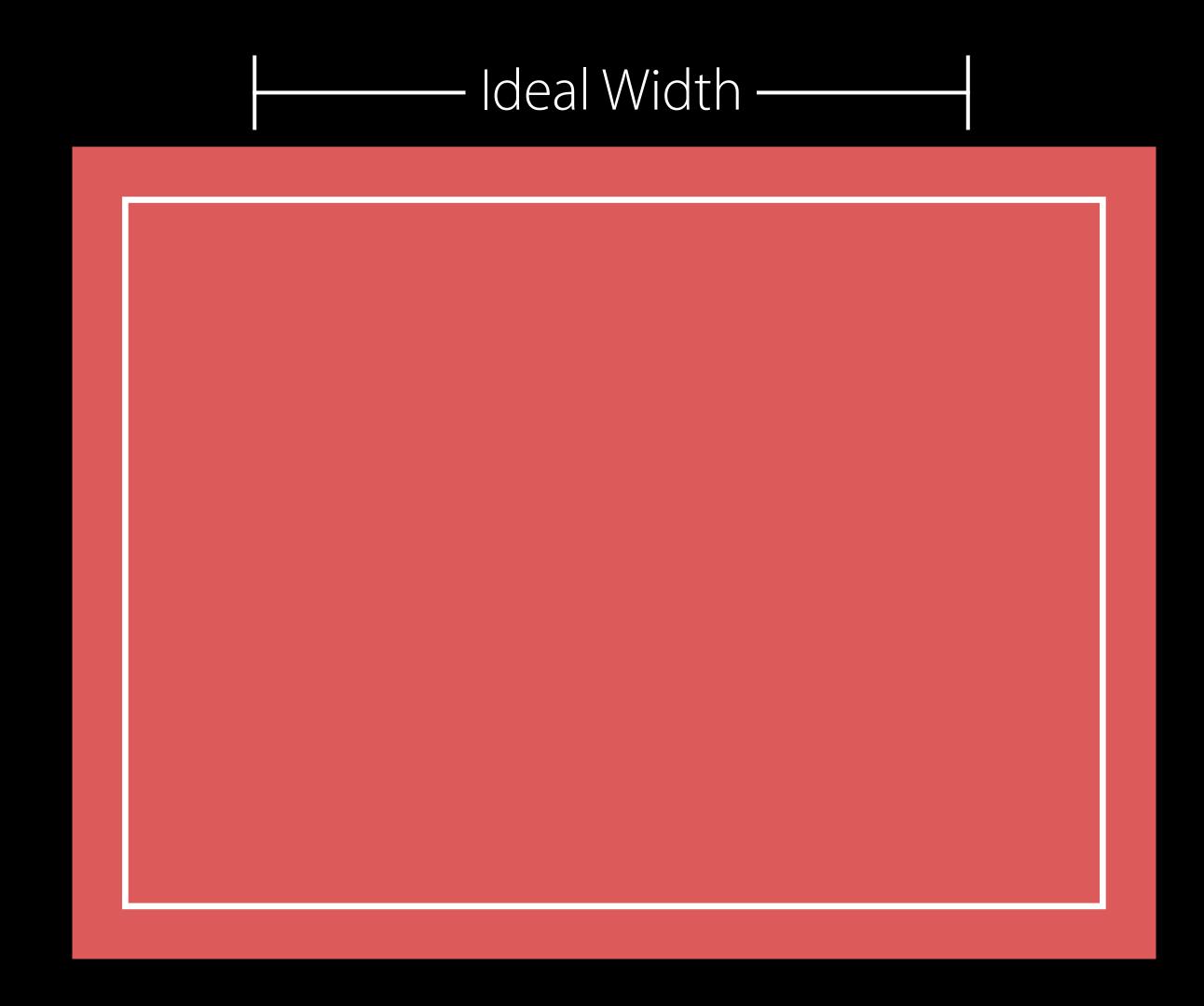
Be sure to test with all Dynamic Type sizes!

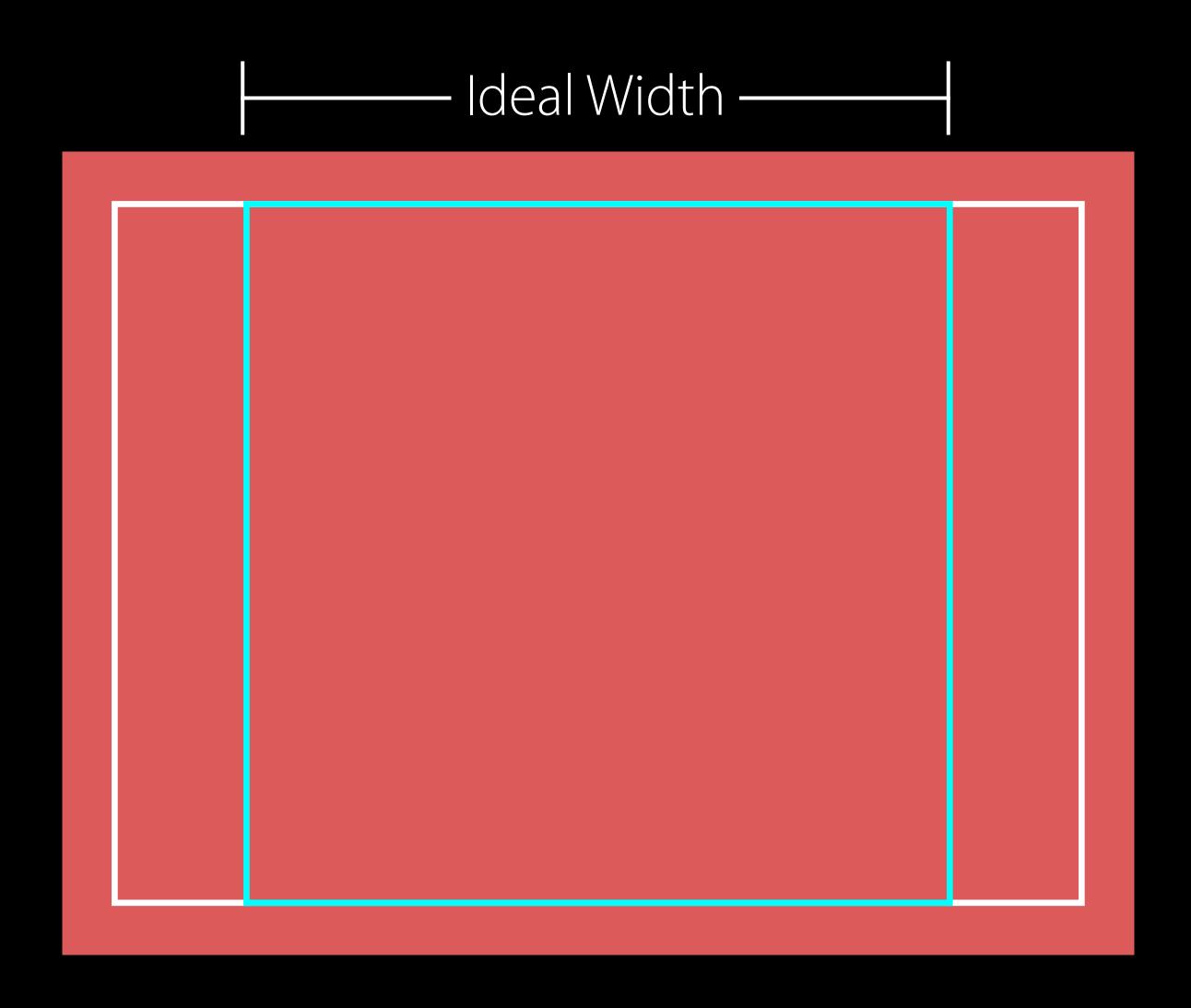


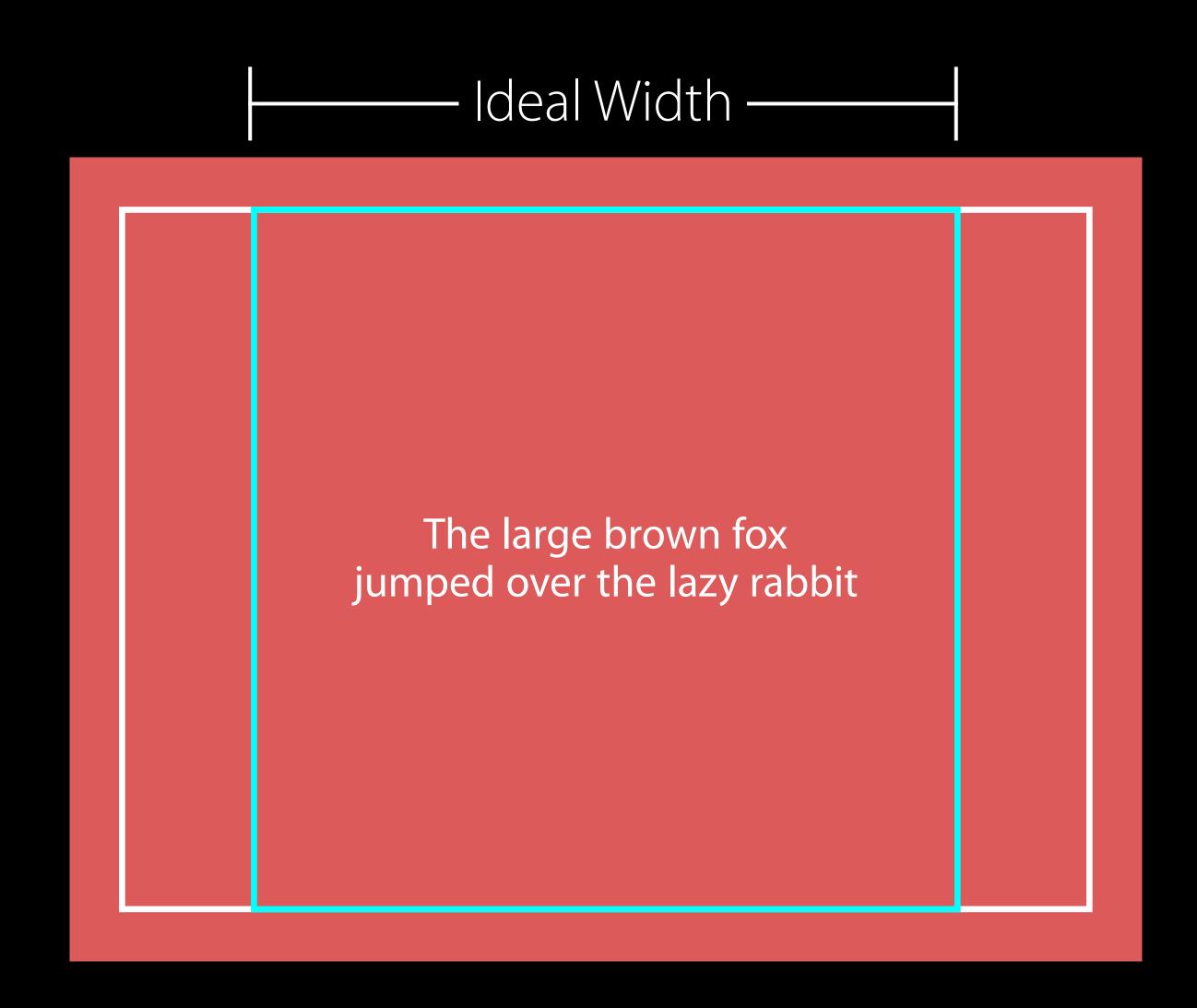


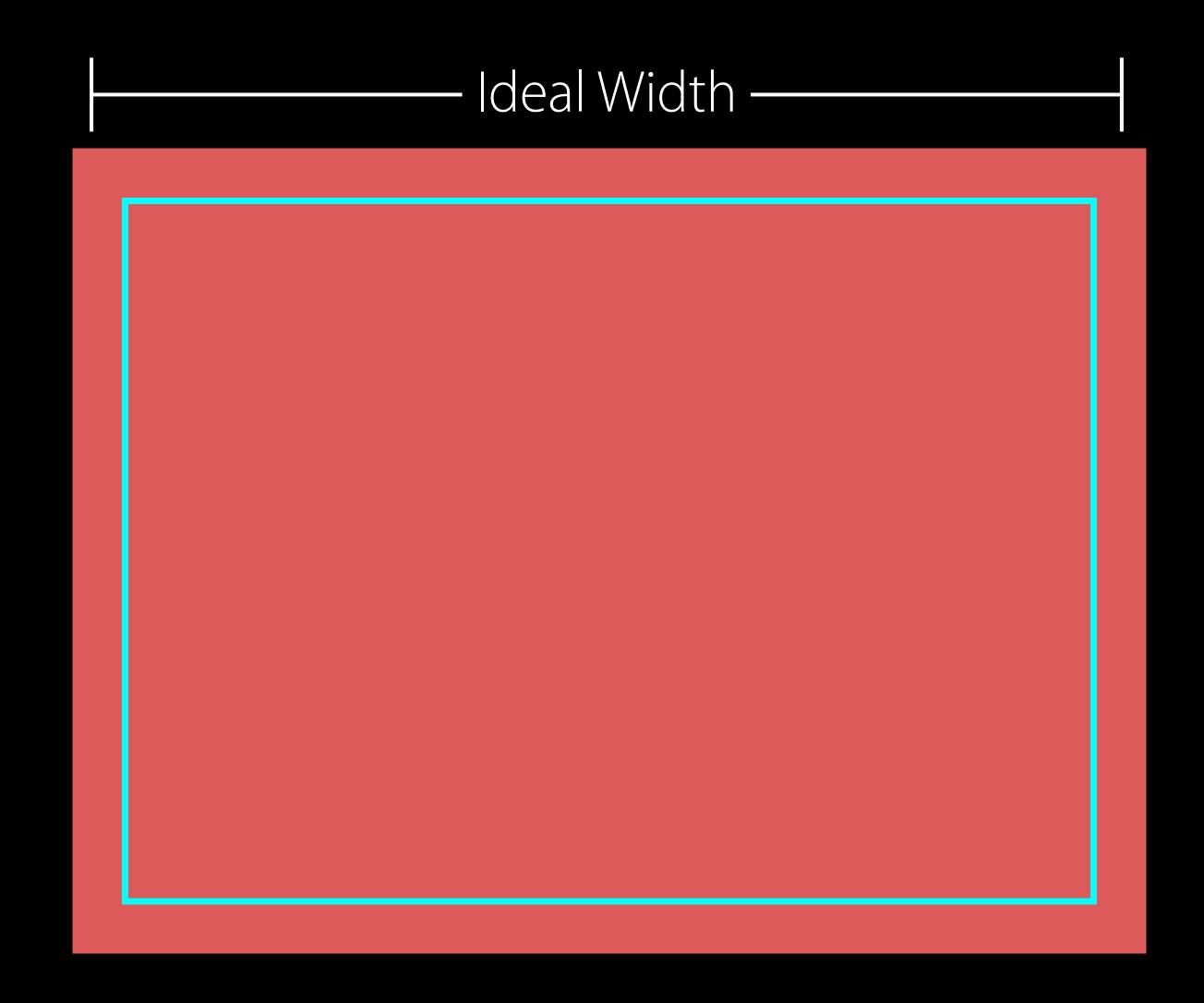












UlView.readableContentGuide

Ideal Width -The large brown fox jumped over the lazy rabbit

UlAppearance

UIAppearance

Declarative

UlAppearance

Declarative

• UITabBar.appearance().unselectedTintColor = UIColor.blue

UlAppearance

Declarative

• UITabBar.appearance().unselectedTintColor = UIColor.blue

Contextual

UlAppearance

Declarative

• UITabBar.appearance().unselectedTintColor = UIColor.blue

Contextual

Trait Collections

UlAppearance

Declarative

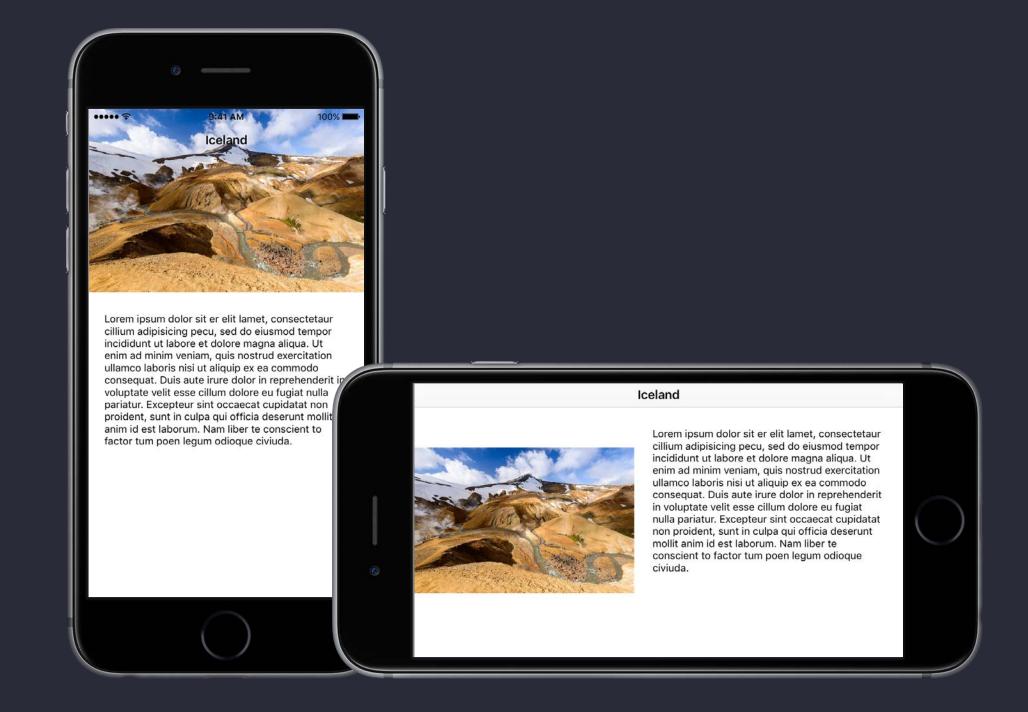
• UITabBar.appearance().unselectedTintColor = UIColor.blue

Contextual

- Trait Collections
- Containment

func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {

}



```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
    let verticalCompactTrait = UITraitCollection(verticalSizeClass: .compact)
```

Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla Iceland pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to Lorem ipsum dolor sit er elit lamet, consectetaur factor tum poen legum odioque civiuda. cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
    let verticalCompactTrait = UITraitCollection(verticalSizeClass: .compact)
    let compactAppearance = UINavigationBar.forTraitCollection(verticalCompactTrait)
```

Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation consequat. Duis aute irure dolor in reprehenderit i voluptate velit esse cillum dolore eu fugiat nulla Iceland pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to Lorem ipsum dolor sit er elit lamet, consectetaur factor tum poen legum odioque civiuda. cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
    let verticalCompactTrait = UITraitCollection(verticalSizeClass: .compact)
    let compactAppearance = UINavigationBar.forTraitCollection(verticalCompactTrait)
    compactAppearance.setBackgroundImage(nil, for: .default)
```

Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation onsequat. Duis aute irure dolor in reprehenderit i Iceland proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to Lorem ipsum dolor sit er elit lamet, consectetaur factor tum poen legum odioque civiuda. cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
    let verticalCompactTrait = UITraitCollection(verticalSizeClass: .compact)
    let compactAppearance = UINavigationBar.forTraitCollection(verticalCompactTrait)
    compactAppearance.setBackgroundImage(nil, for: .default)
```

Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation onsequat. Duis aute irure dolor in reprehenderit i Iceland proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to Lorem ipsum dolor sit er elit lamet, consectetaur factor tum poen legum odioque civiuda. cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
    let verticalCompactTrait = UITraitCollection(verticalSizeClass: .compact)
    let compactAppearance = UINavigationBar.forTraitCollection(verticalCompactTrait)
    compactAppearance.setBackgroundImage(nil, for: .default)
```

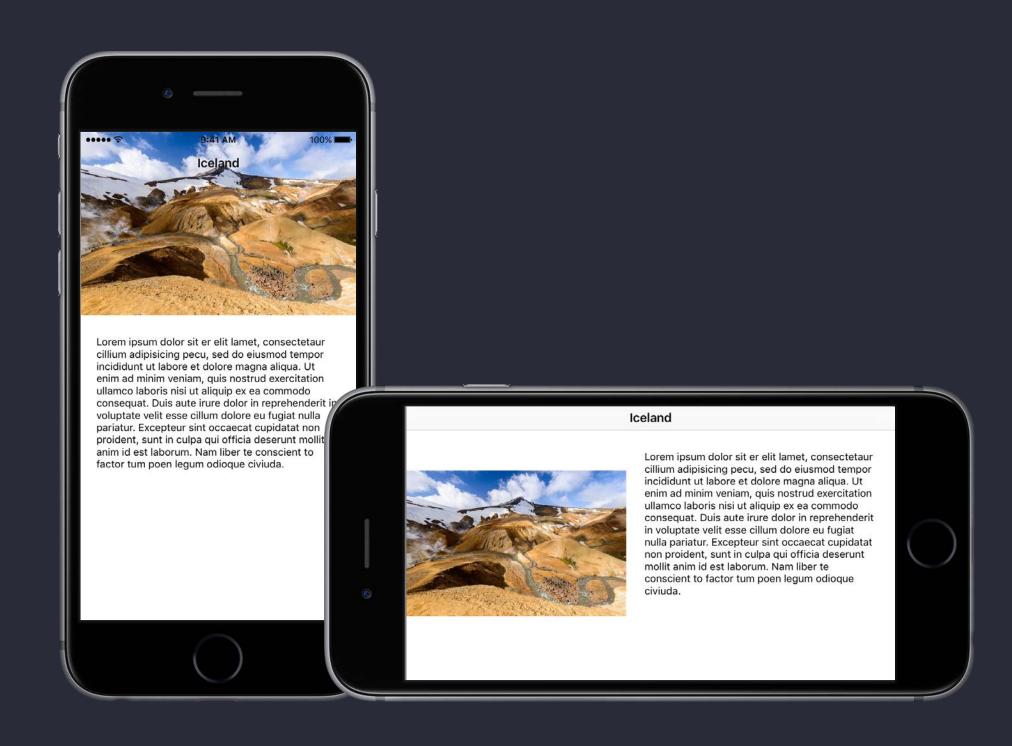
let verticalRegularTrait = UITraitCollection(verticalSizeClass: .regular)

Lorem ipsum dolor sit er eit lamet, consectatur cilium adpisation pecu, sed de situand tempor incididur ut above et dolor mapra aliqua. Ut utalmot et dolor eu trujat nulla pariatur. Exceptur eint occade utugiat nulla pariatur. Exceptur eint occade utugiat nulla pariatur. Exceptur eint occade utugiat nulla pariatur. Exceptur eint occade care dupidatat non protident, sunt in cuisa auf officia desenrant note utugiat nulla pariatur. Exceptur eint occade care dupidatat non protident, sunt in cuisa auf officia desenrant note utugiat nulla pariatur. Exceptur eint occade care dupidatat non protident in cuisa cui officia desenrant note utugiatur. Utugiatur et desenrant note utugiatur et desenrant note utugiatur. Utugiatur et desenrant note utugiatur et desenrant note utugiatur. Utugiatur et desen

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
    let verticalCompactTrait = UITraitCollection(verticalSizeClass: .compact)
    let compactAppearance = UINavigationBar.forTraitCollection(verticalCompactTrait)
    compactAppearance.setBackgroundImage(nil, for: .default)

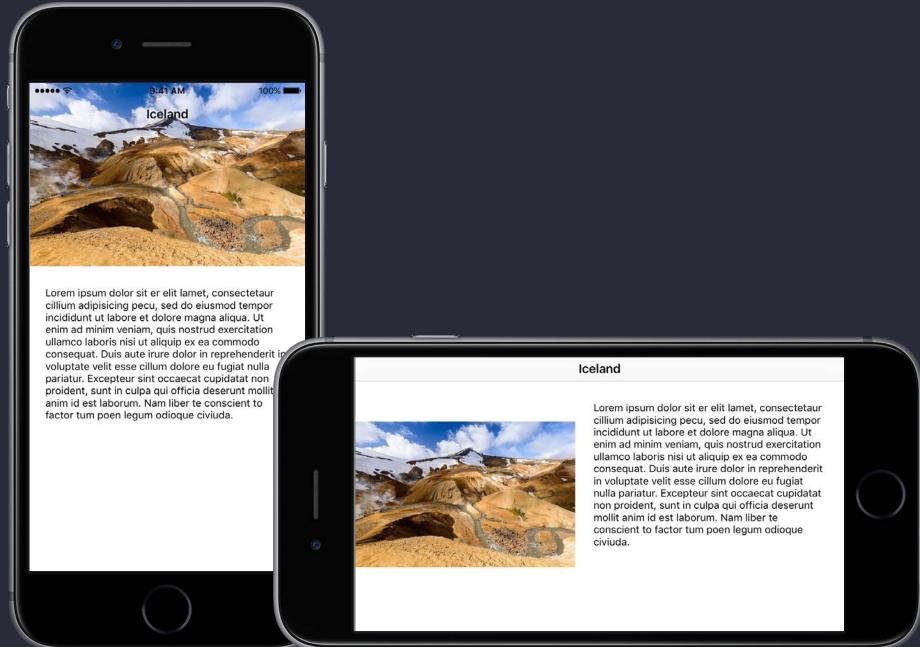
let verticalRegularTrait = UITraitCollection(verticalSizeClass: .regular)
```

let verticalAppearance = UINavigationBar.forTraitCollection(verticalRegularTrait)



```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
    let verticalCompactTrait = UITraitCollection(verticalSizeClass: .compact)
    let compactAppearance = UINavigationBar.forTraitCollection(verticalCompactTrait)
    compactAppearance.setBackgroundImage(nil, for: .default)

let verticalRegularTrait = UITraitCollection(verticalSizeClass: .regular)
    let verticalAppearance = UINavigationBar.forTraitCollection(verticalRegularTrait)
    verticalAppearance.setBackgroundImage(UIImage(), for: .default)
```



Asset Catalogs

Asset Catalogs

Dynamic Type

Asset Catalogs

Dynamic Type

Layout Guides

Asset Catalogs

Dynamic Type

Layout Guides

UlAppearance

Takeaway

The system is going to do most of the work so you don't have to.

If you want to go beyond what the system provides, here's how.

Design for all combinations of device, orientation, and size

Design for all combinations of device, orientation, and size Implement each design and change between them

Design for all combinations of device, orientation, and size Implement each design and change between them

Use reusable elements

My Incredibly Adaptive App

My Incredibly Adaptive App

ltem	Title	Description
1	A	Lorem ipsum dolor sit er elit lamet
2	В	Ut enim ad minim veniam, quis nostrud
3	C	Duis aute irure dolor in reprehenderit



Design 1:"Tall"

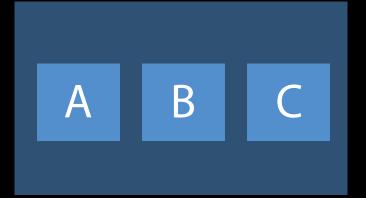
A

В

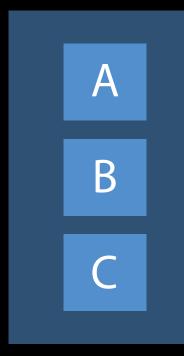
Design 1:"Tall"



Design 2:"Wide"

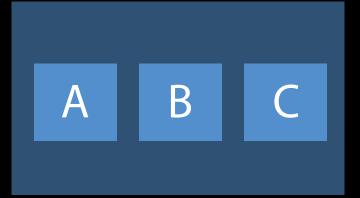


Design 1:"Tall"



Width < Height

Design 2:"Wide"

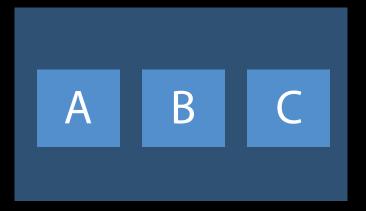


Design 1:"Tall"



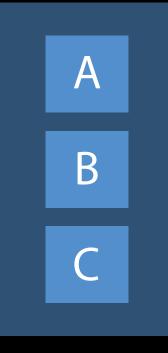
Width < Height

Design 2:"Wide"



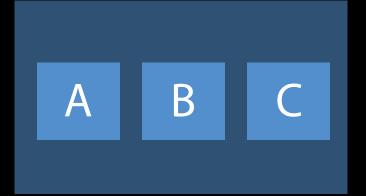
Width >= Height

Design 1:"Tall"



Width < Height

Design 2:"Wide"



Width >= Height



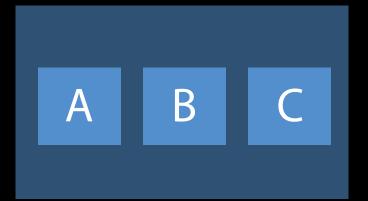
Design 1:"Tall"



Width < Height

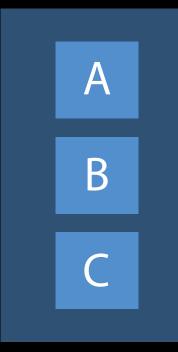


Design 2:"Wide"



Width >= Height

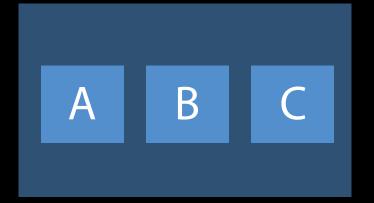
Design 1:"Tall"



Width < Height



Design 2:"Wide"



Width >= Height



Consider the possible *combinations* of device, orientation, and size

Consider the possible *combinations* of device, orientation, and size Make *designs*

Consider the possible *combinations* of device, orientation, and size

Make designs

Define rules for which design to use

Defining Rules

Check if size exactly matches



Check if size exactly matches



Check if size exactly matches



Define Boolean conditions



Check if size exactly matches



Define Boolean conditions

Use size class



Check if size exactly matches



Define Boolean conditions

- Use size class
- Compare value to a threshold



Check if size exactly matches



Define Boolean conditions

- Use size class
- Compare value to a threshold
- Compare values to each other

WWDC16 **ÉWWDC16 WWDC16** WWDC16

WWDC16 WWDC16 WWDC16 **WWDC16**

Find the size of the app

Find the size of the app

Use the rules to decide which design to use

Find the size of the app

Use the rules to decide which design to use

Apply the design to the UI

```
override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
}
```

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
}
```

View is loaded on demand

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
}
```

- View is loaded on demand
- View is not yet in a superview, and layout has not yet happened

- View is loaded on demand
- View is not yet in a superview, and layout has not yet happened

```
Use init(), loadView(), and viewDidLoad() only for code that is the same across designs
```

```
override func viewWillLayoutSubviews() {
    // Add code here
}
```

View is in a superview, and layout of superviews has happened

- View is in a superview, and layout of superviews has happened
- Good time to manipulate things inside your view controller

```
override func viewWillLayoutSubviews() {
    // Add code here
}
```

```
override func viewWillLayoutSubviews() {
    // Add code here
}
```

Do as little work as possible

- Do as little work as possible
- Find out what has changed since last time

- Do as little work as possible
- Find out what has changed since last time
- Don't cause a layout loop

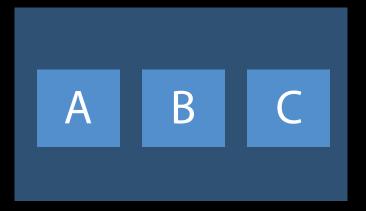
- Do as little work as possible
- Find out what has changed since last time
- Don't cause a layout loop

Design 1:"Tall"



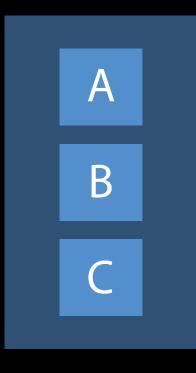
Width < Height

Design 2:"Wide"



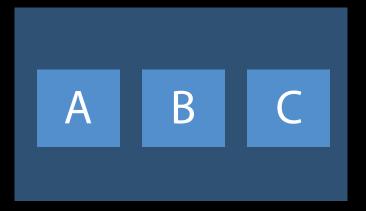
Width >= Height

Design 1:"Tall"



Width < Height

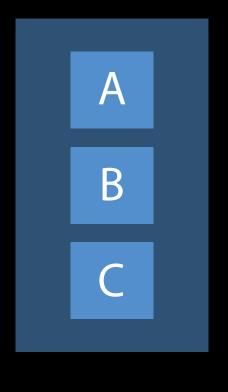
Design 2:"Wide"



Width >= Height

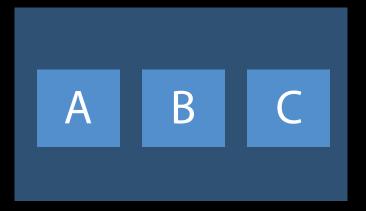
Each item is a view

Design 1:"Tall"



Width < Height

Design 2:"Wide"



Width >= Height

Each item is a view

Use UIStackView for horizontal and vertical arrangement

```
class SimpleExampleViewController: UIViewController {
   @IBOutlet var stackView : UIStackView!
    override func viewWillLayoutSubviews() {
        let size = view.bounds.size
        let useWideDesign = size.width >= size.height
        if useWideDesign {
            stackView.axis = .horizontal
        } else {
            stackView.axis = .vertical
```

```
@IBOutlet var stackView : UIStackView!
override func viewWillLayoutSubviews() {
    let size = view.bounds.size
    let useWideDesign = size.width >= size.height
    if useWideDesign {
        stackView.axis = .horizontal
    } else {
        stackView.axis = .vertical
```

```
class SimpleExampleViewController: UIViewController {
```

```
@IBOutlet var stackView : UIStackView!
```

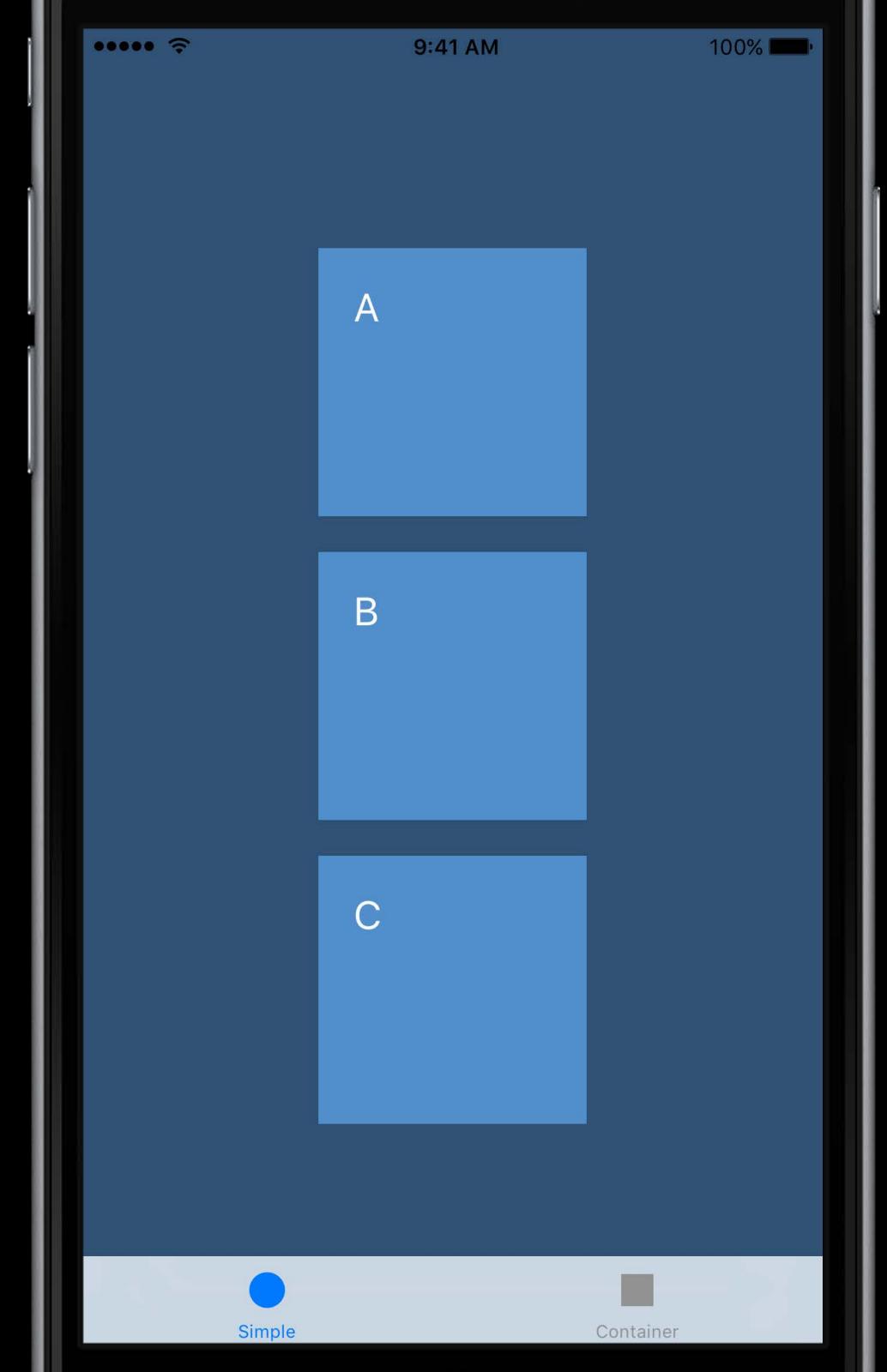
```
override func viewWillLayoutSubviews() {
    let size = view.bounds.size
    let useWideDesign = size.width >= size.height
    if useWideDesign {
        stackView.axis = .horizontal
    } else {
        stackView.axis = .vertical
    }
}
```

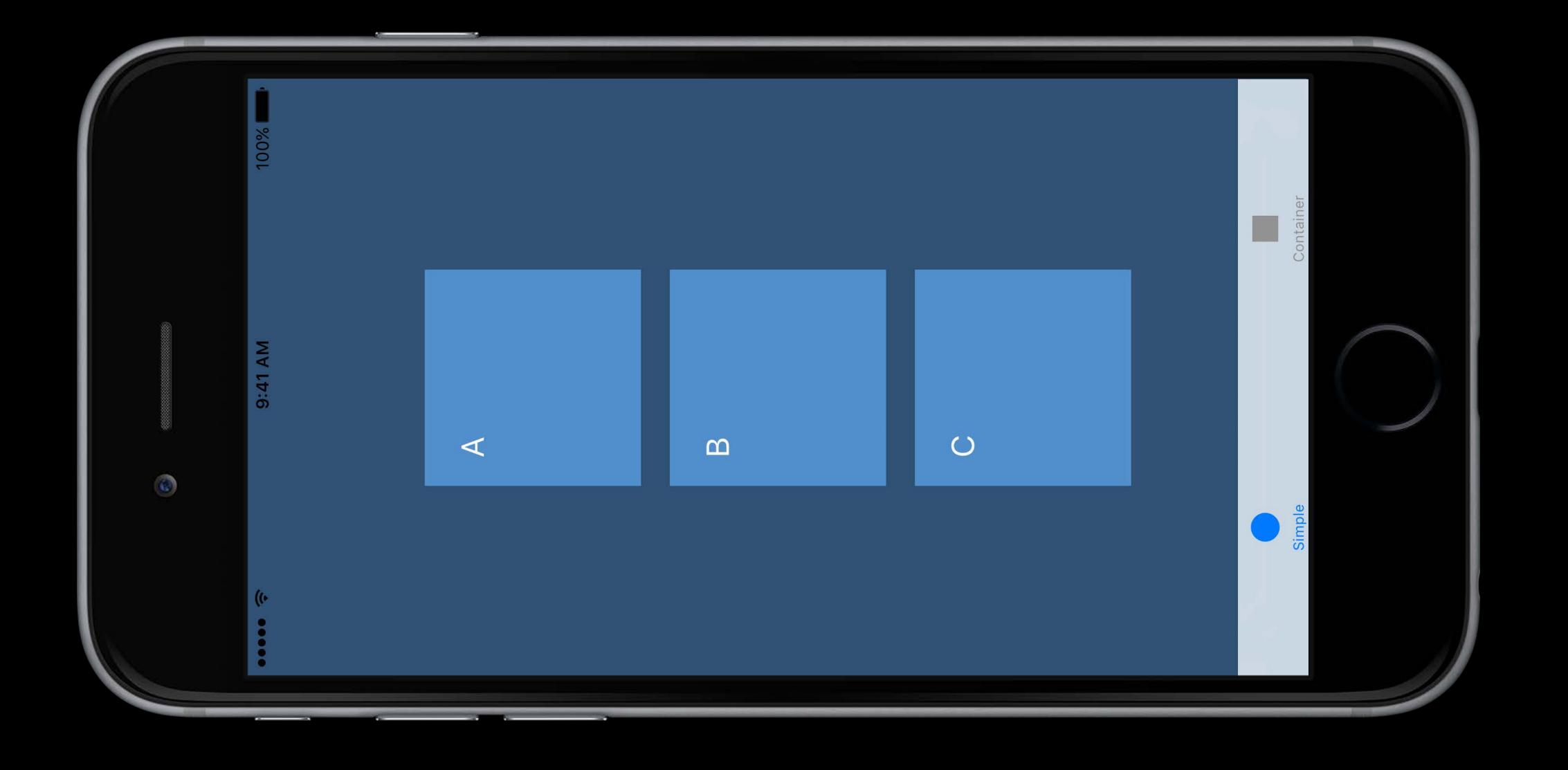
```
class SimpleExampleViewController: UIViewController {
   @IBOutlet var stackView : UIStackView!
    override func viewWillLayoutSubviews() {
        let size = view.bounds.size
        let useWideDesign = size.width >= size.height
        if useWideDesign {
            stackView.axis = .horizontal
        } else {
            stackView.axis = .vertical
```

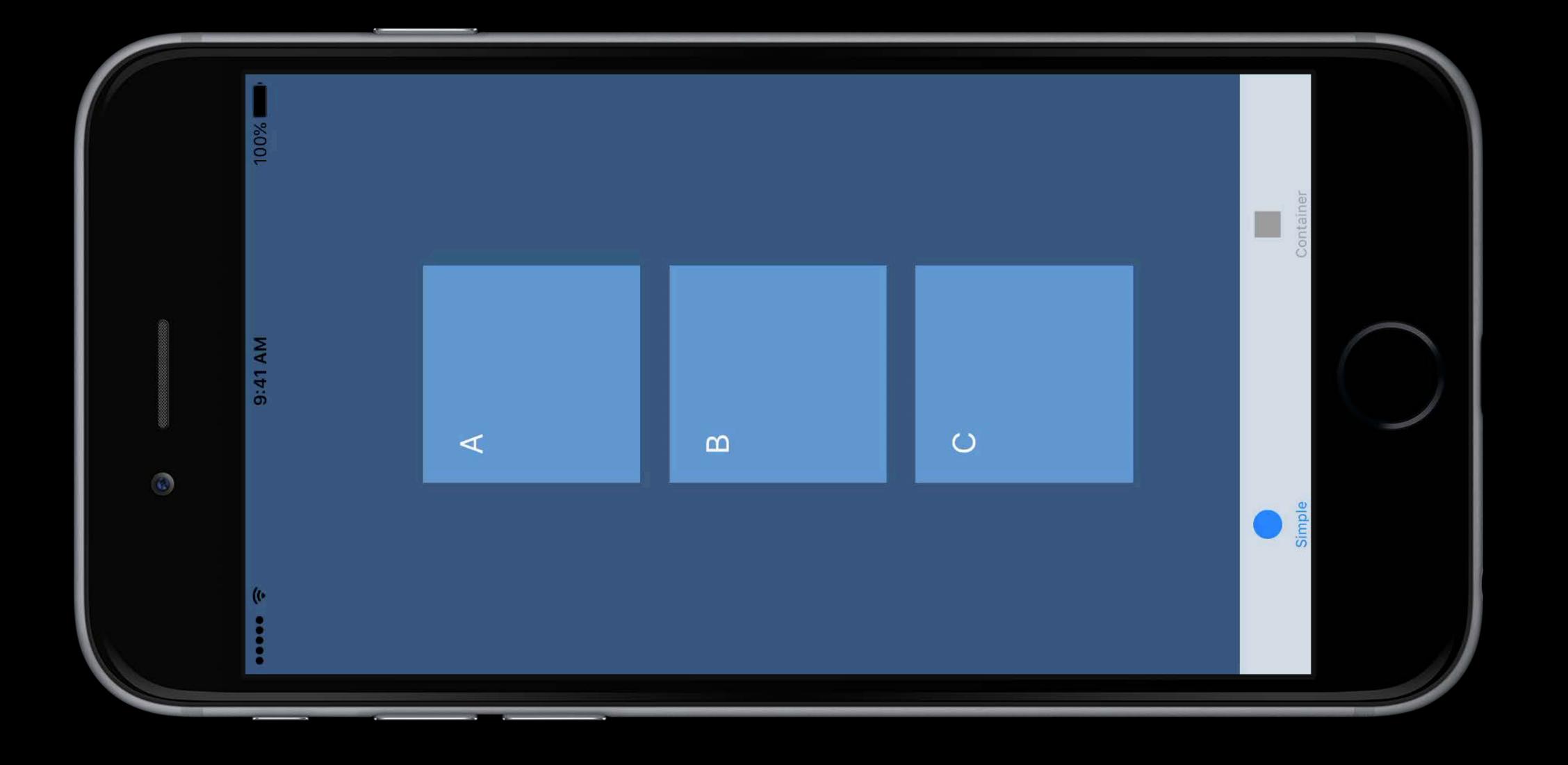
```
class SimpleExampleViewController: UIViewController {
   @IBOutlet var stackView : UIStackView!
    override func viewWillLayoutSubviews() {
        let size = view.bounds.size
        let useWideDesign = size.width >= size.height
        if useWideDesign {
            stackView.axis = .horizontal
        } else {
            stackView.axis = .vertical
```

```
class SimpleExampleViewController: UIViewController {
   @IBOutlet var stackView : UIStackView!
    override func viewWillLayoutSubviews() {
        let size = view.bounds.size
        let useWideDesign = size.width >= size.height
        if useWideDesign {
            stackView.axis = .horizontal
        } else {
            stackView.axis = .vertical
```

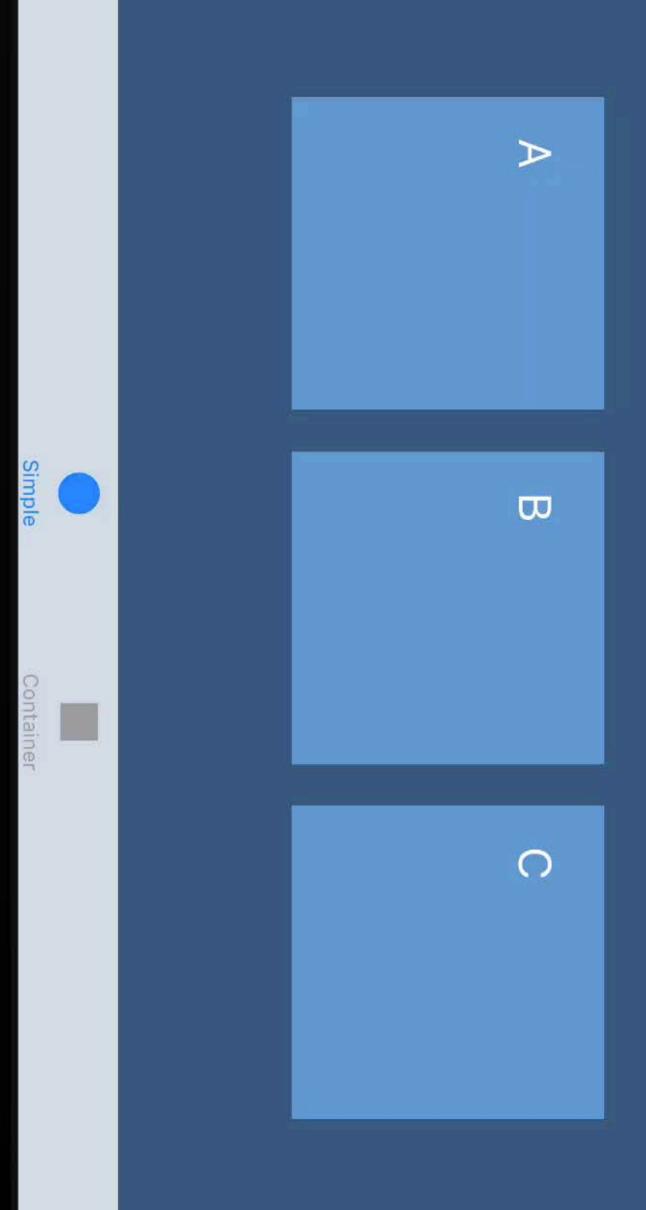
```
class SimpleExampleViewController: UIViewController {
   @IBOutlet var stackView : UIStackView!
    override func viewWillLayoutSubviews() {
        let size = view.bounds.size
        let useWideDesign = size.width >= size.height
        if useWideDesign {
            stackView.axis = .horizontal
        } else {
            stackView.axis = .vertical
```







D \Box 0



"Can you make it pop?"

"Can you make it pop?"

During rotation, make the items grow

"Can you make it pop?"

During rotation, make the items grow

After rotation, go back to normal

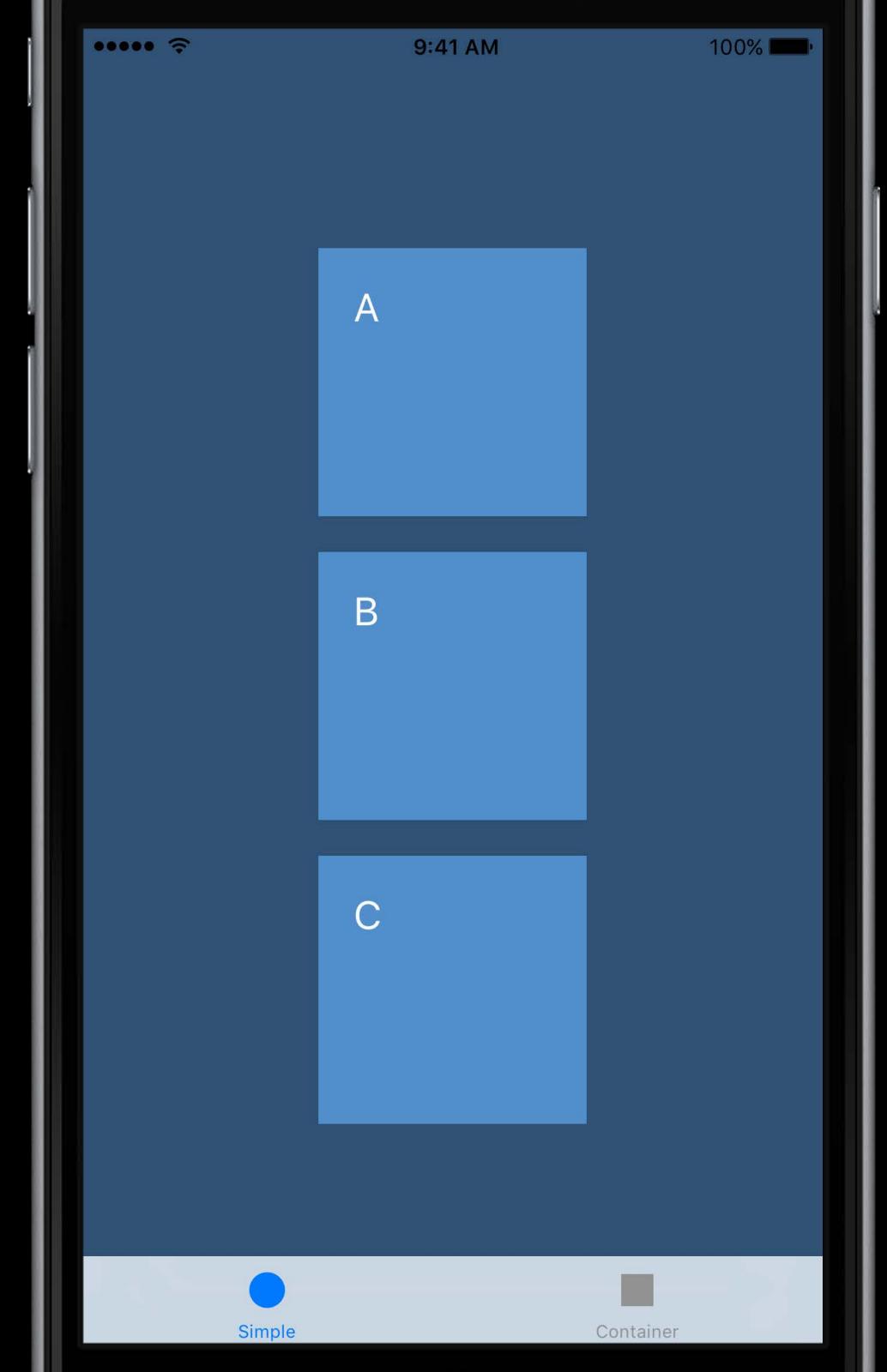
```
override func viewWillTransition(to size: CGSize,
    with coordinator: UIViewControllerTransitionCoordinator) {
    super.viewWillTransition(to: size, with: coordinator)
    coordinator.animate(alongsideTransition:
        { _ in
            stackView.transform = CGAffineTransform(scaleX: 1.4, y: 1.4)
        },
                       completion:
        { _ in
            UIView.animate(withDuration: 0.5, animations: {
                stackView.transform = CGAffineTransform.identity
            })
```

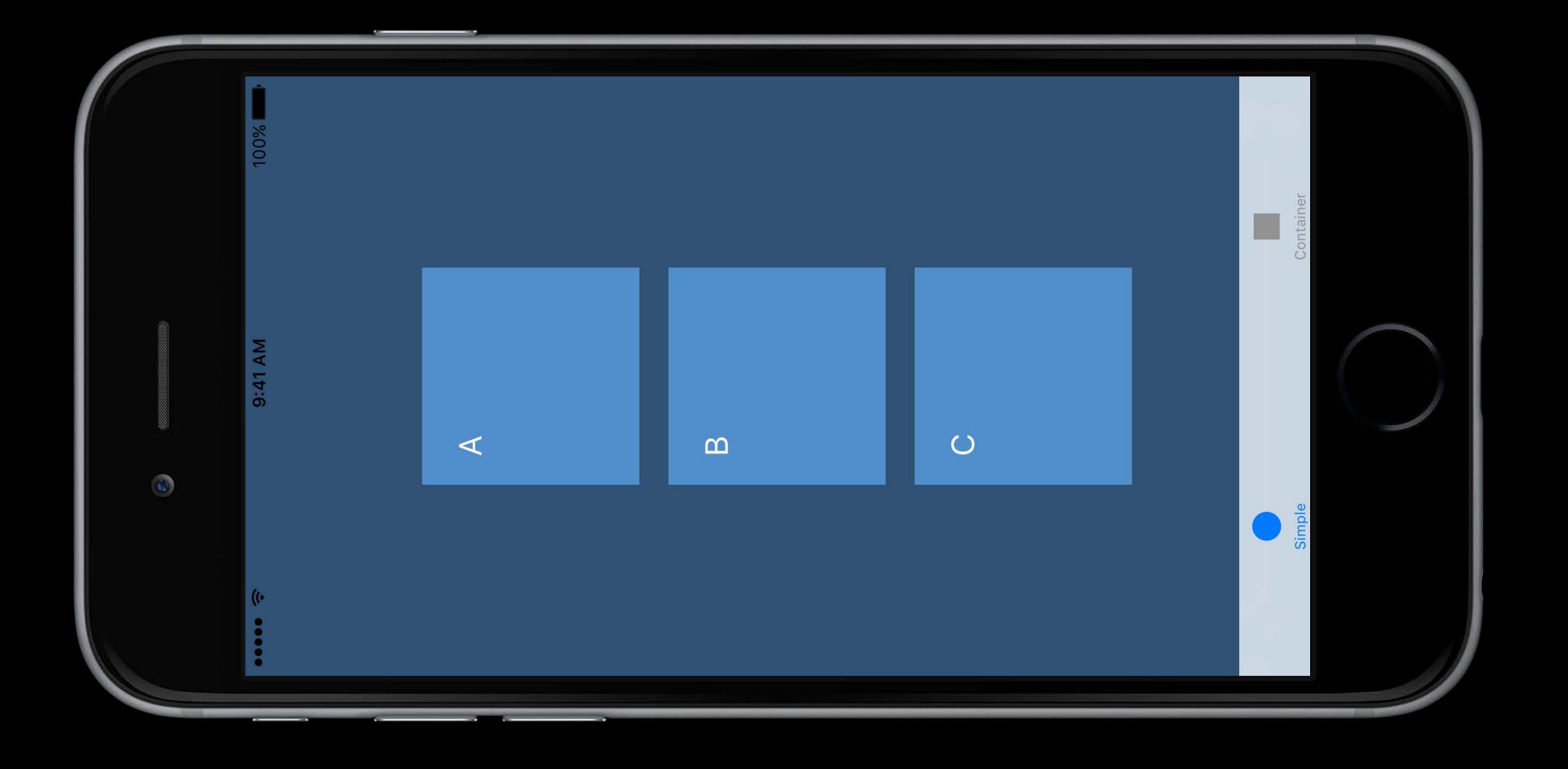
```
override func viewWillTransition(to size: CGSize,
    with coordinator: UIViewControllerTransitionCoordinator) {
    super.viewWillTransition(to: size, with: coordinator)
    coordinator.animate(alongsideTransition:
       { _ in
            stackView.transform = CGAffineTransform(scaleX: 1.4, y: 1.4)
       },
                       completion:
        { _ in
            UIView.animate(withDuration: 0.5, animations: {
                stackView.transform = CGAffineTransform.identity
            })
```

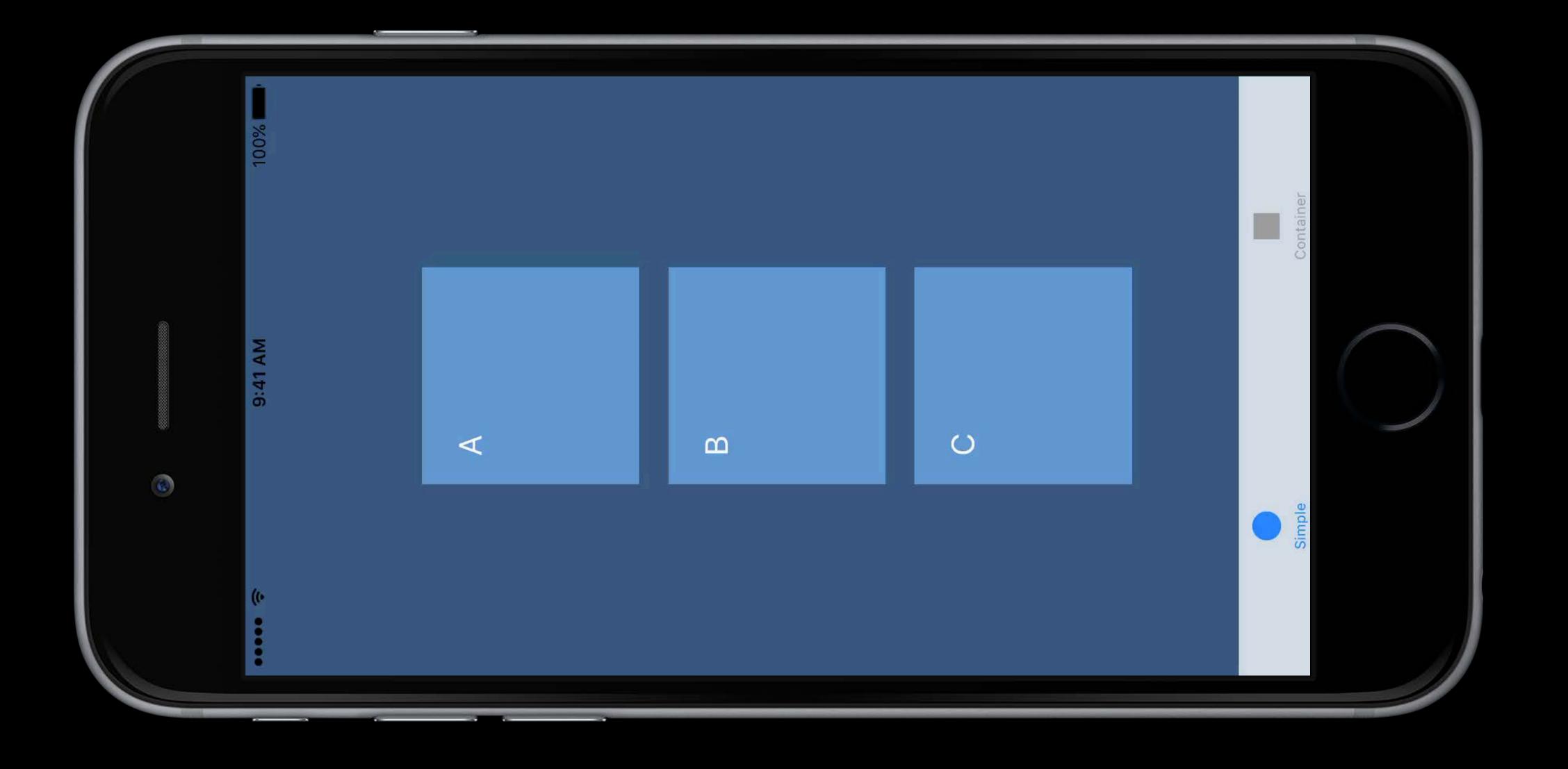
```
override func viewWillTransition(to size: CGSize,
    with coordinator: UIViewControllerTransitionCoordinator) {
    super.viewWillTransition(to: size, with: coordinator)
    coordinator.animate(alongsideTransition:
        { _ in
            stackView.transform = CGAffineTransform(scaleX: 1.4, y: 1.4)
        },
                       completion:
        { _ in
            UIView.animate(withDuration: 0.5, animations: {
                stackView.transform = CGAffineTransform.identity
            })
```

```
override func viewWillTransition(to size: CGSize,
    with coordinator: UIViewControllerTransitionCoordinator) {
    super.viewWillTransition(to: size, with: coordinator)
    coordinator.animate(alongsideTransition:
        { _ in
            stackView.transform = CGAffineTransform(scaleX: 1.4, y: 1.4)
        },
                       completion:
        { _ in
            UIView.animate(withDuration: 0.5, animations: {
                stackView.transform = CGAffineTransform.identity
            })
```

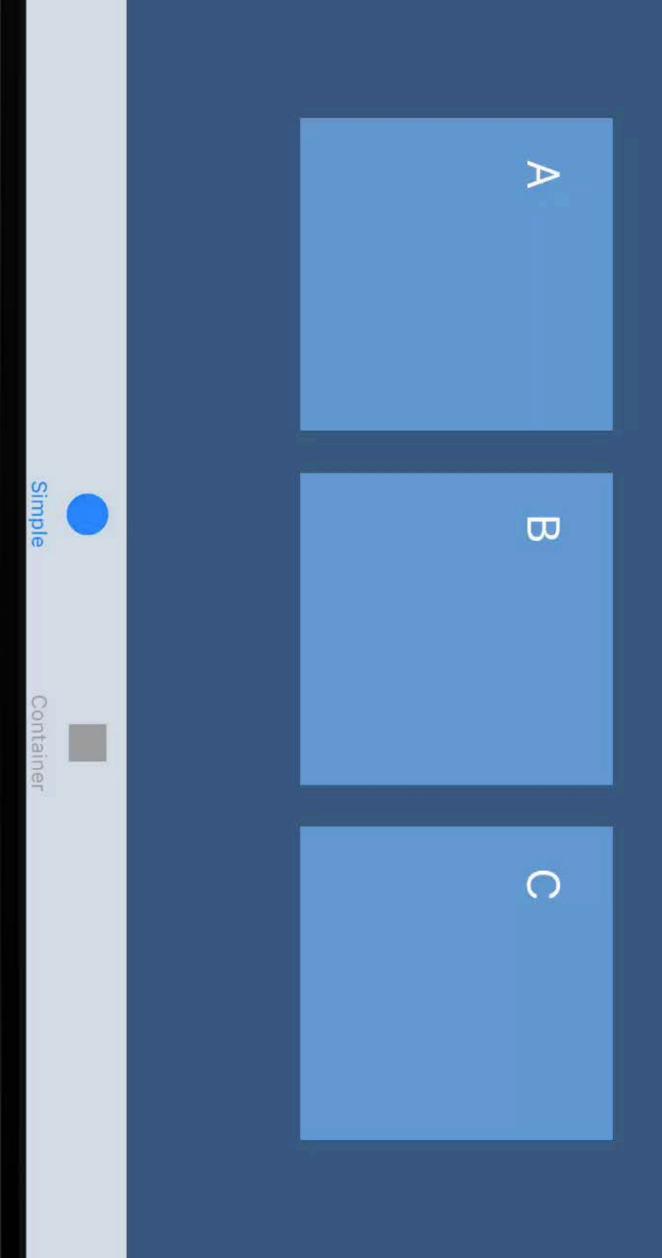
```
override func viewWillTransition(to size: CGSize,
    with coordinator: UIViewControllerTransitionCoordinator) {
    super.viewWillTransition(to: size, with: coordinator)
    coordinator.animate(alongsideTransition:
        { _ in
            stackView.transform = CGAffineTransform(scaleX: 1.4, y: 1.4)
        },
                       completion:
        { _ in
            UIView.animate(withDuration: 0.5, animations: {
                stackView.transform = CGAffineTransform.identity
            })
```







D \Box 0



Build your app out of pieces that can be reused in different designs

Build your app out of pieces that can be reused in different designs

Each piece is typically a view controller

Build your app out of pieces that can be reused in different designs

Each piece is typically a view controller

View tree and constraints

Build your app out of pieces that can be reused in different designs

Each piece is typically a view controller

- View tree and constraints
- Connections to other view controllers

Build your app out of pieces that can be reused in different designs

Each piece is typically a view controller

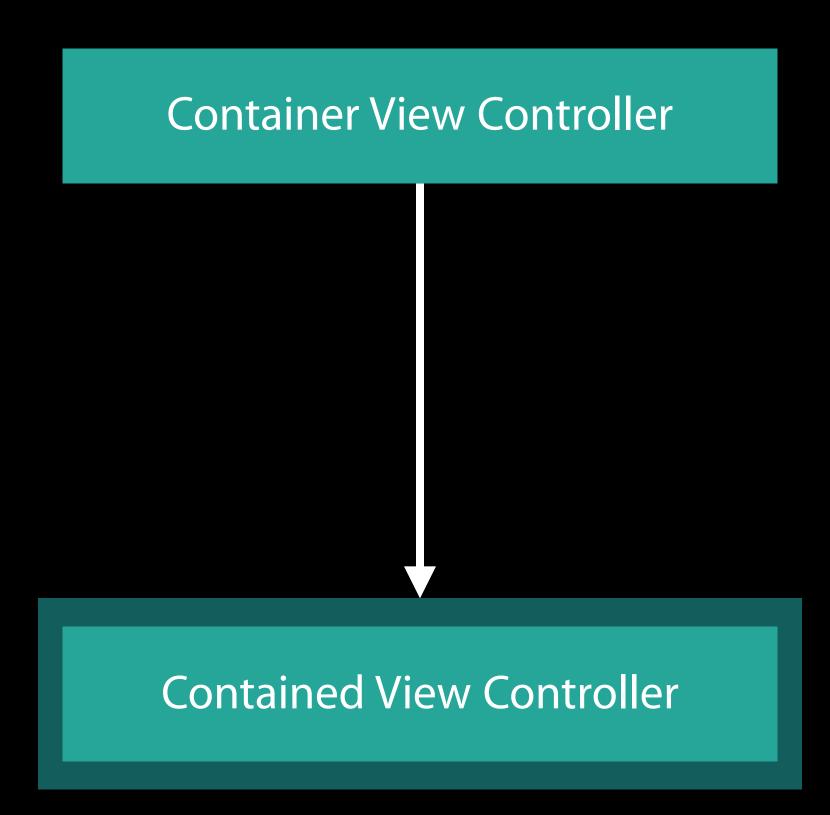
- View tree and constraints
- Connections to other view controllers
- Connections to the rest of your app

View Controller Roles

View Controller Roles

Container View Controller

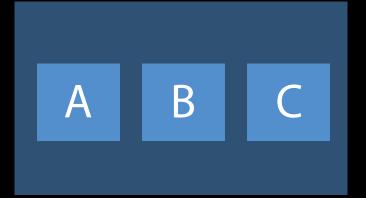
View Controller Roles



Design 1:"Tall"



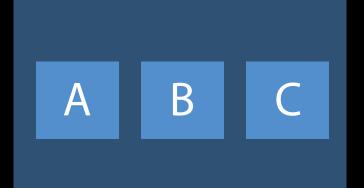
Design 2:"Wide"



Design 1:"Tall"

Design 2:"Wide"

A B

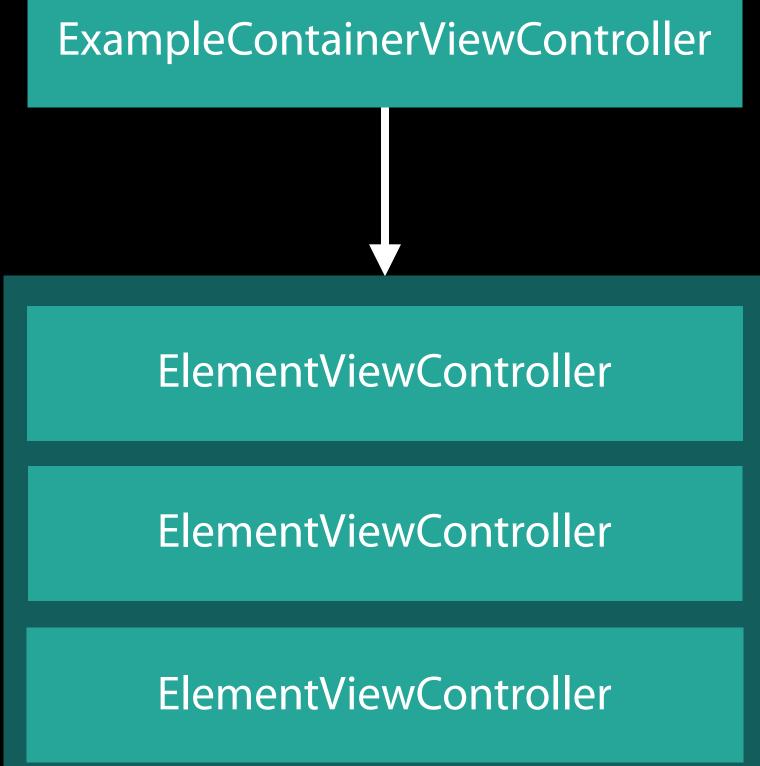


ExampleContainerViewController

Design 1:"Tall"

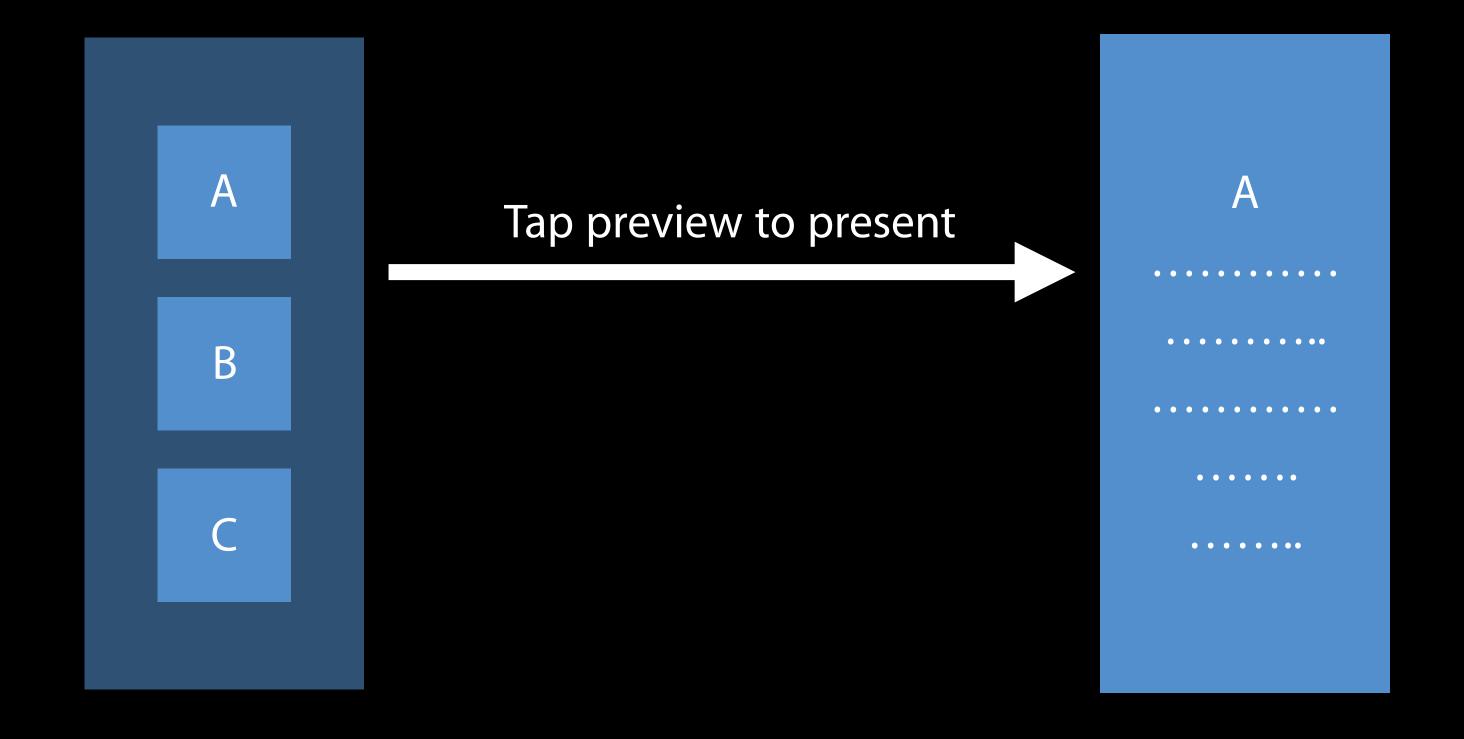
Design 2:"Wide"

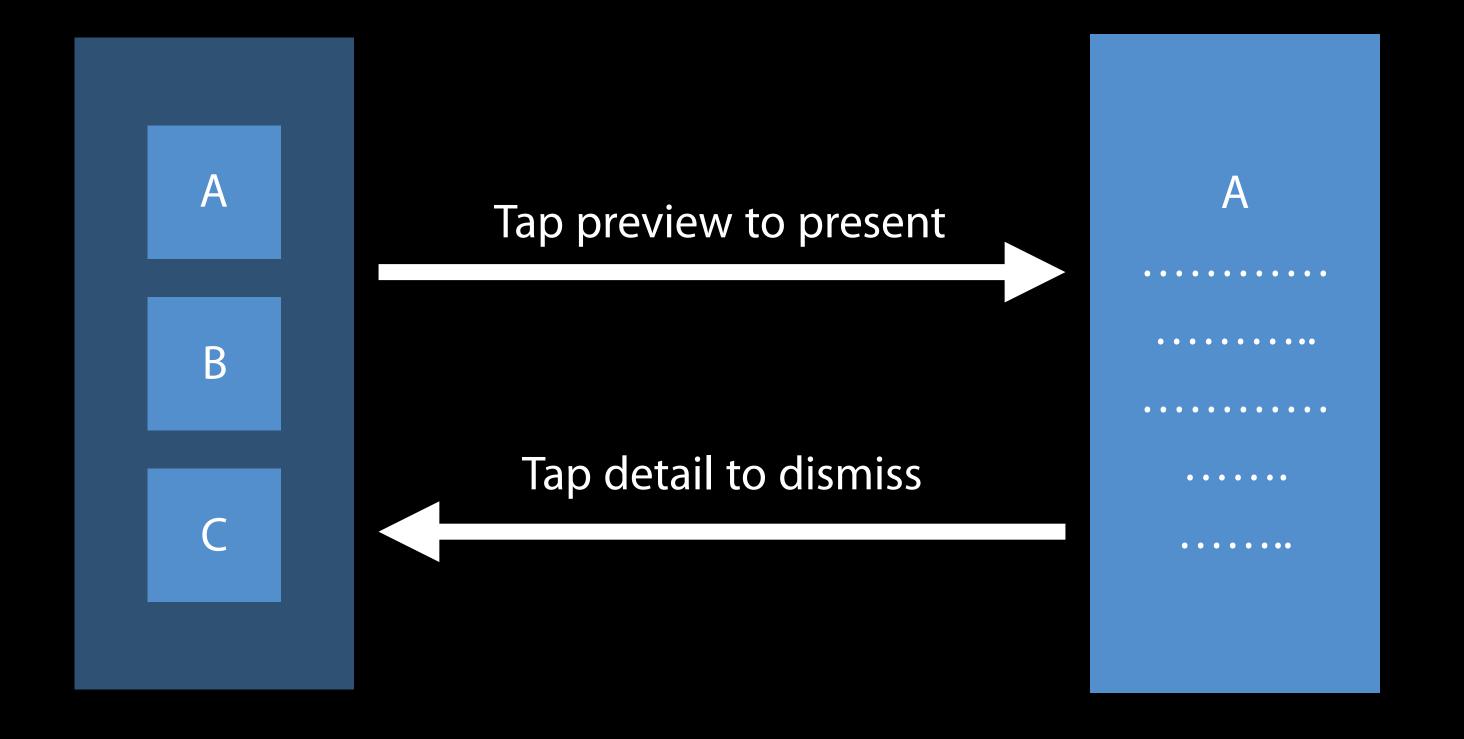


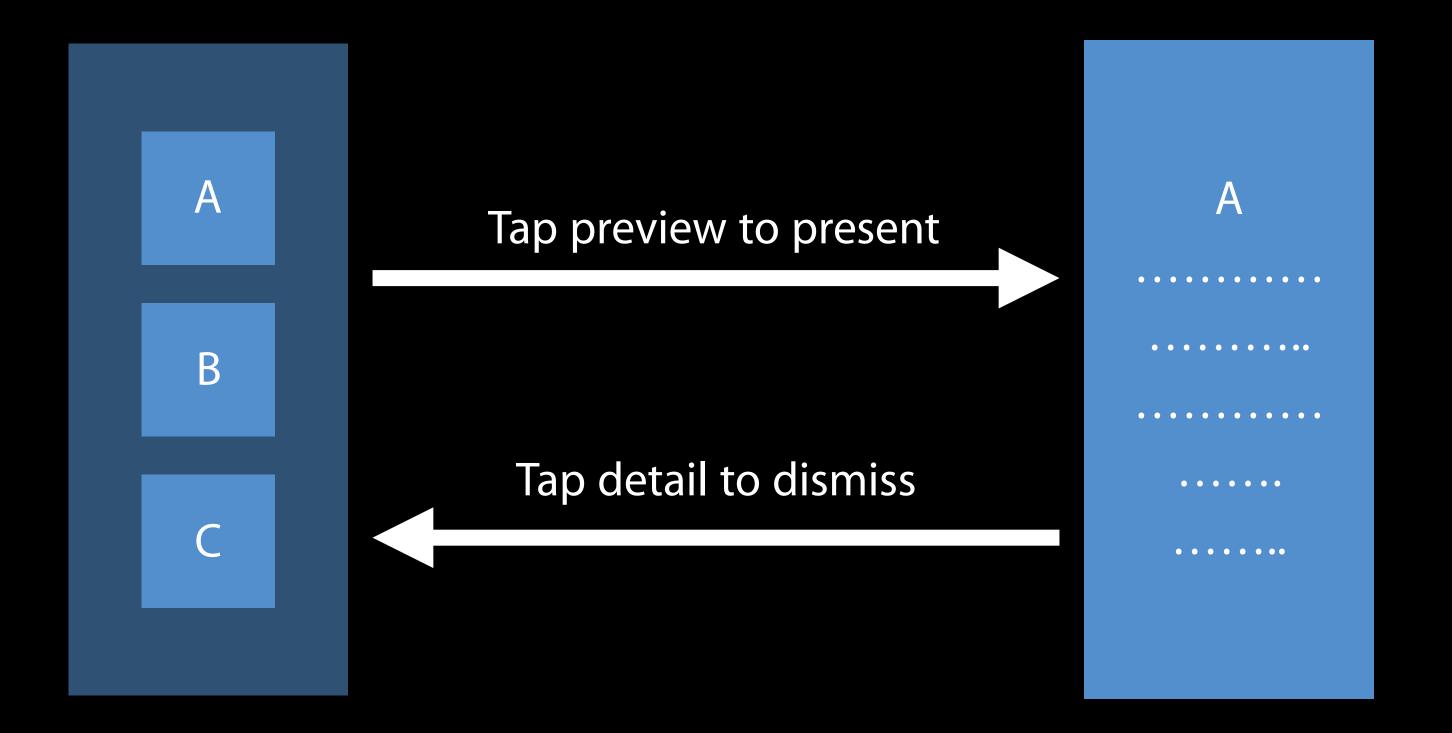


A

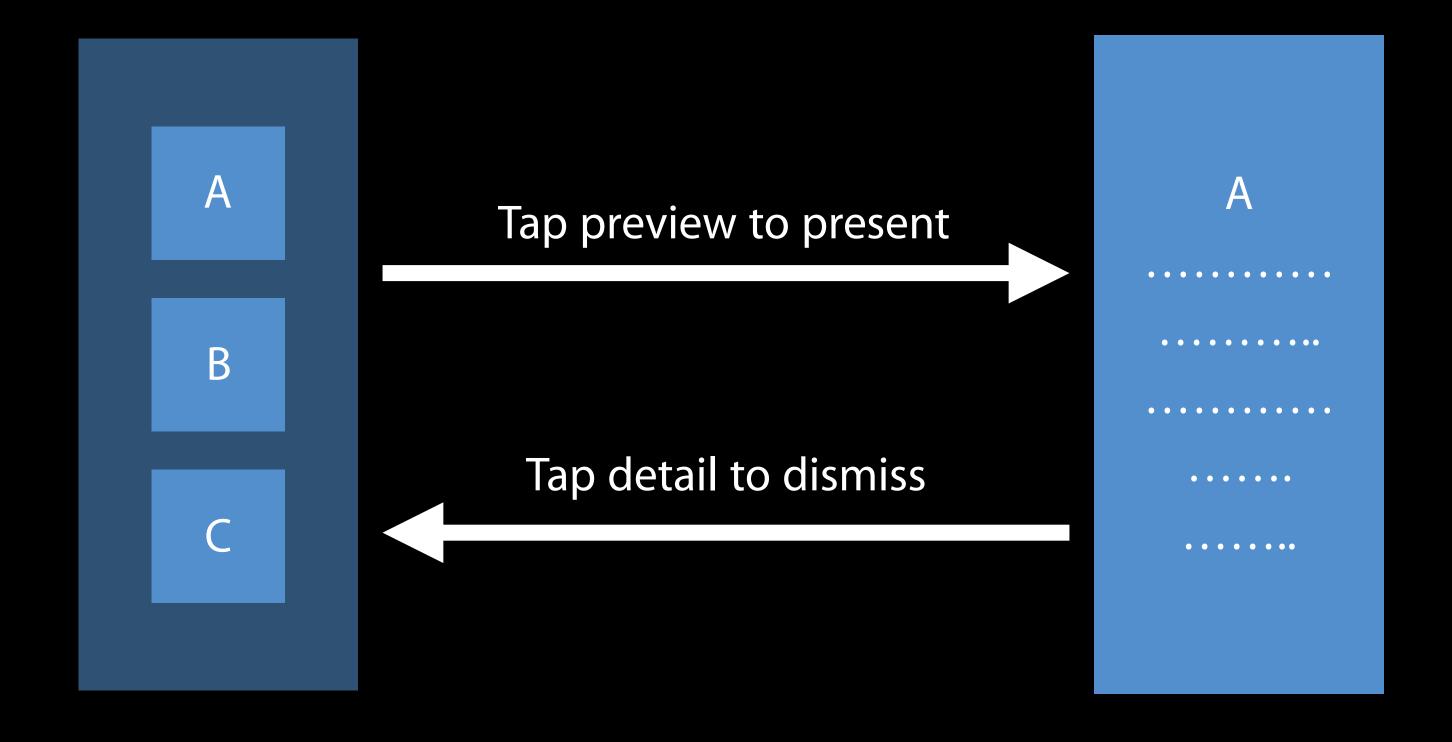
В

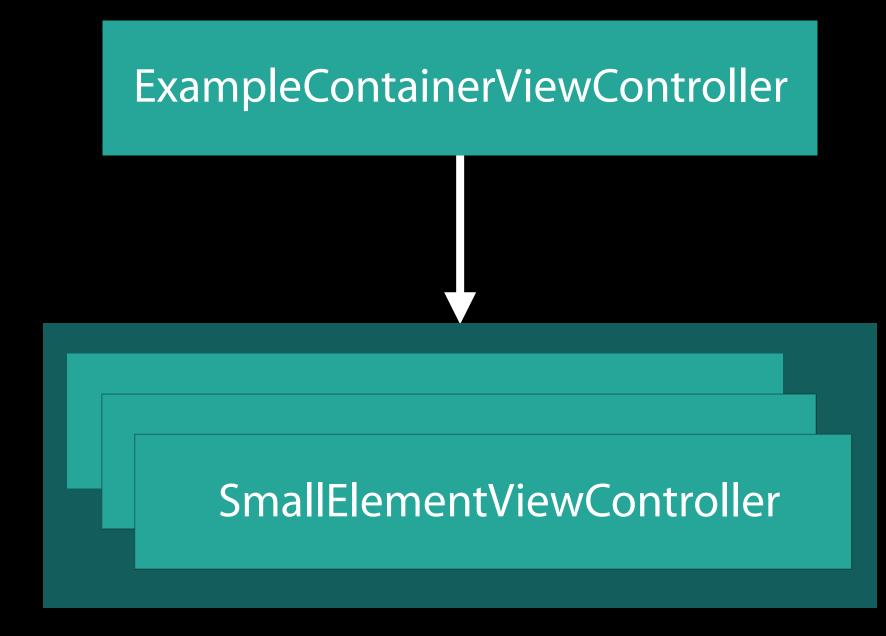


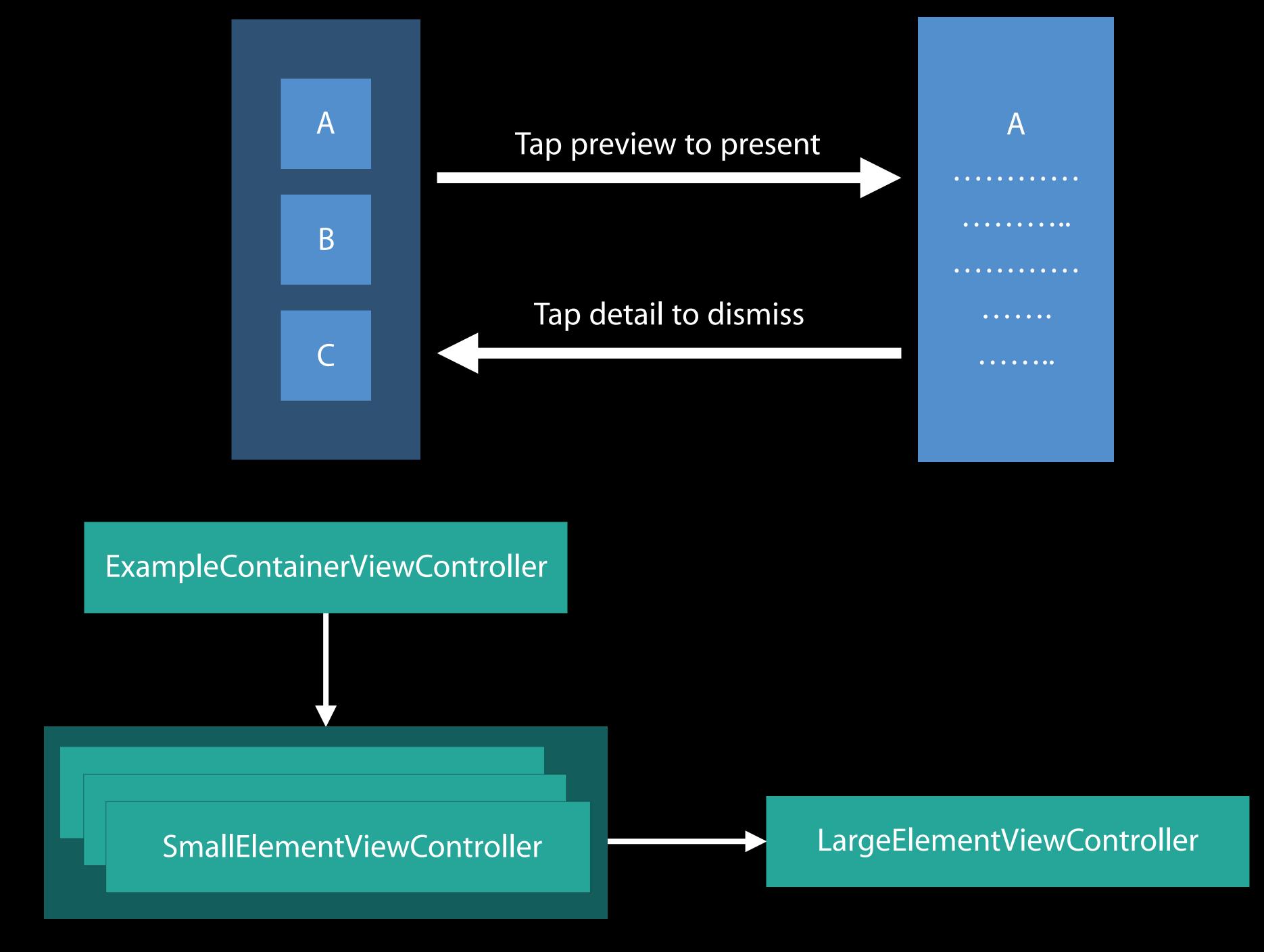




ExampleContainerViewController







A

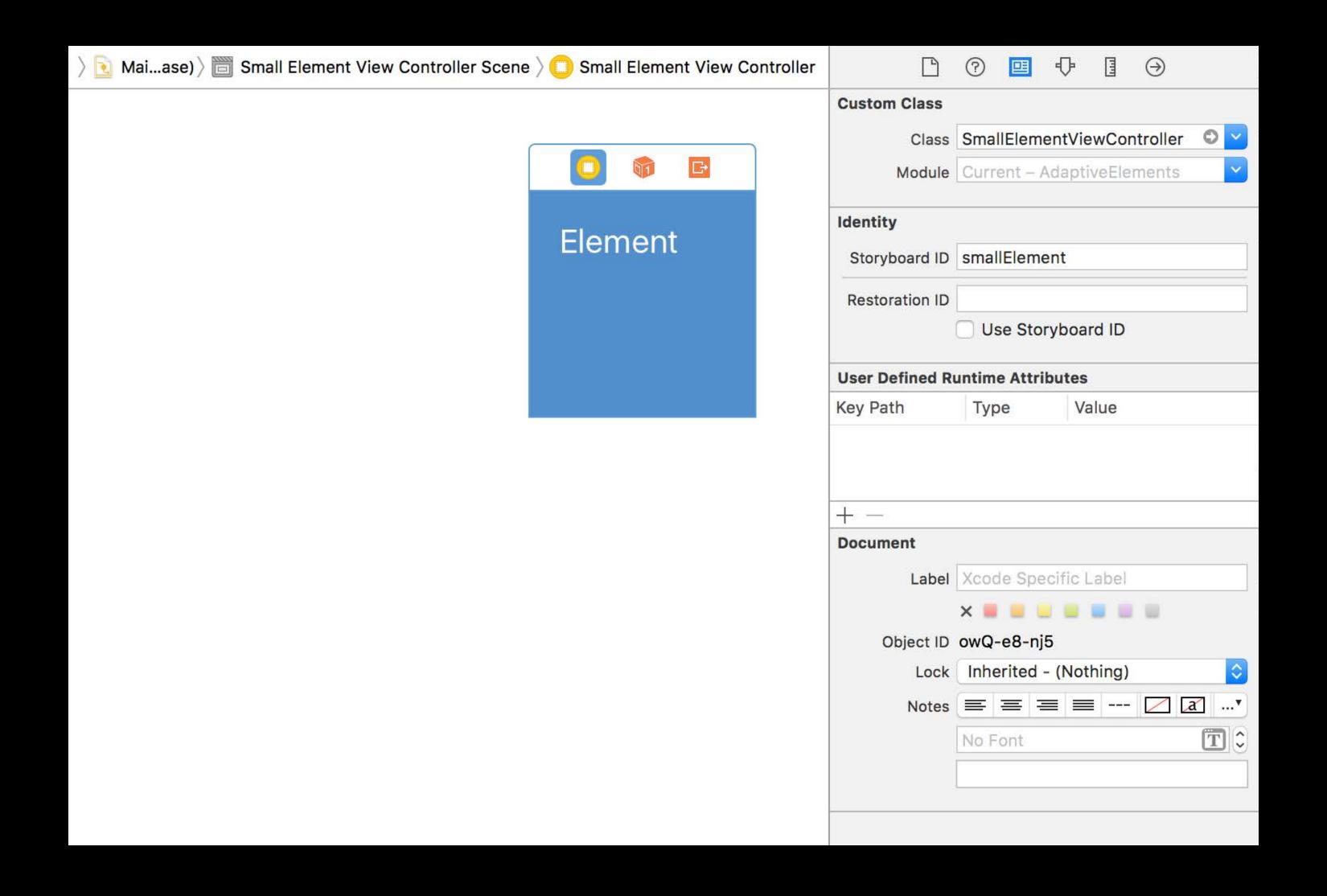
B

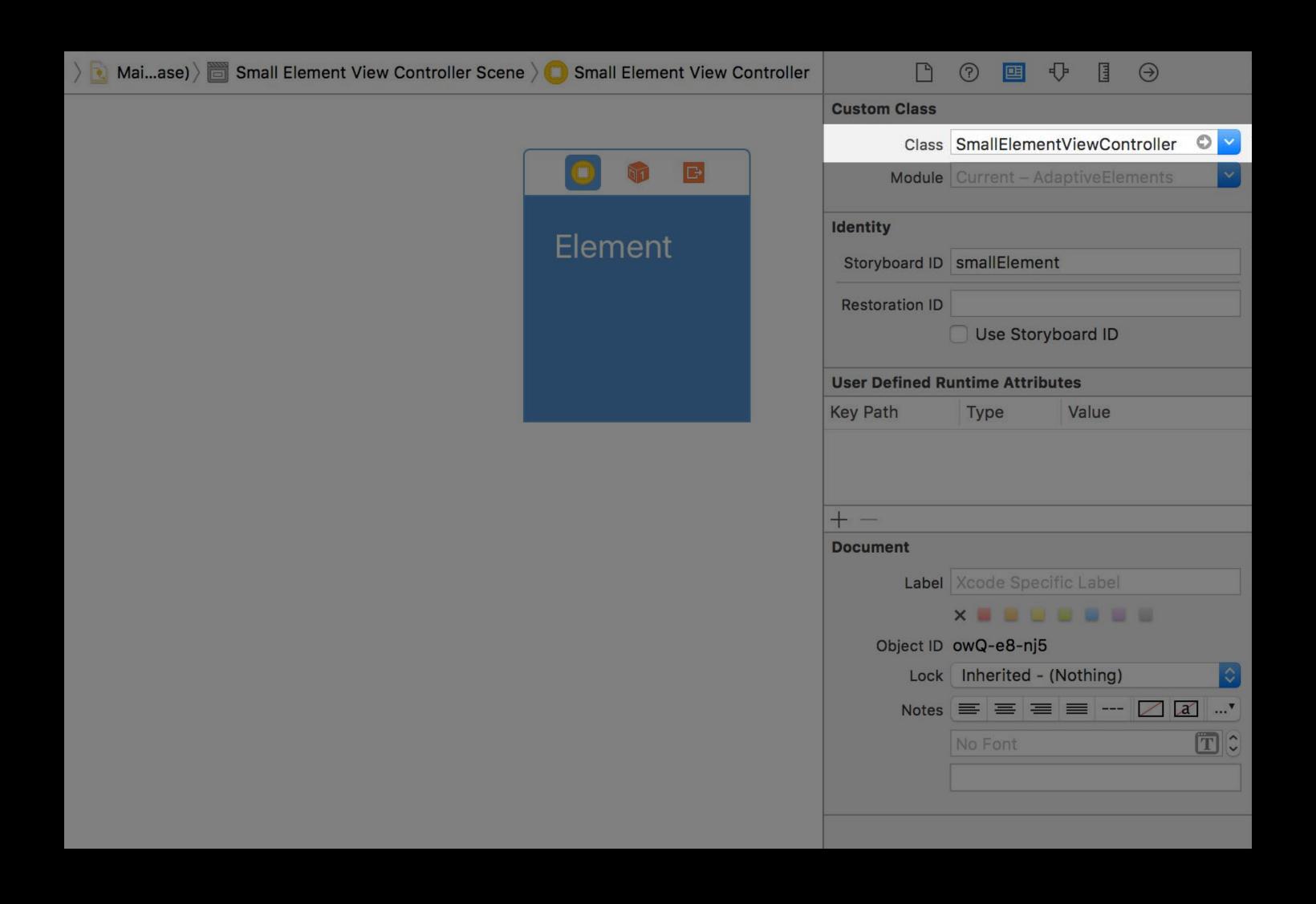
C

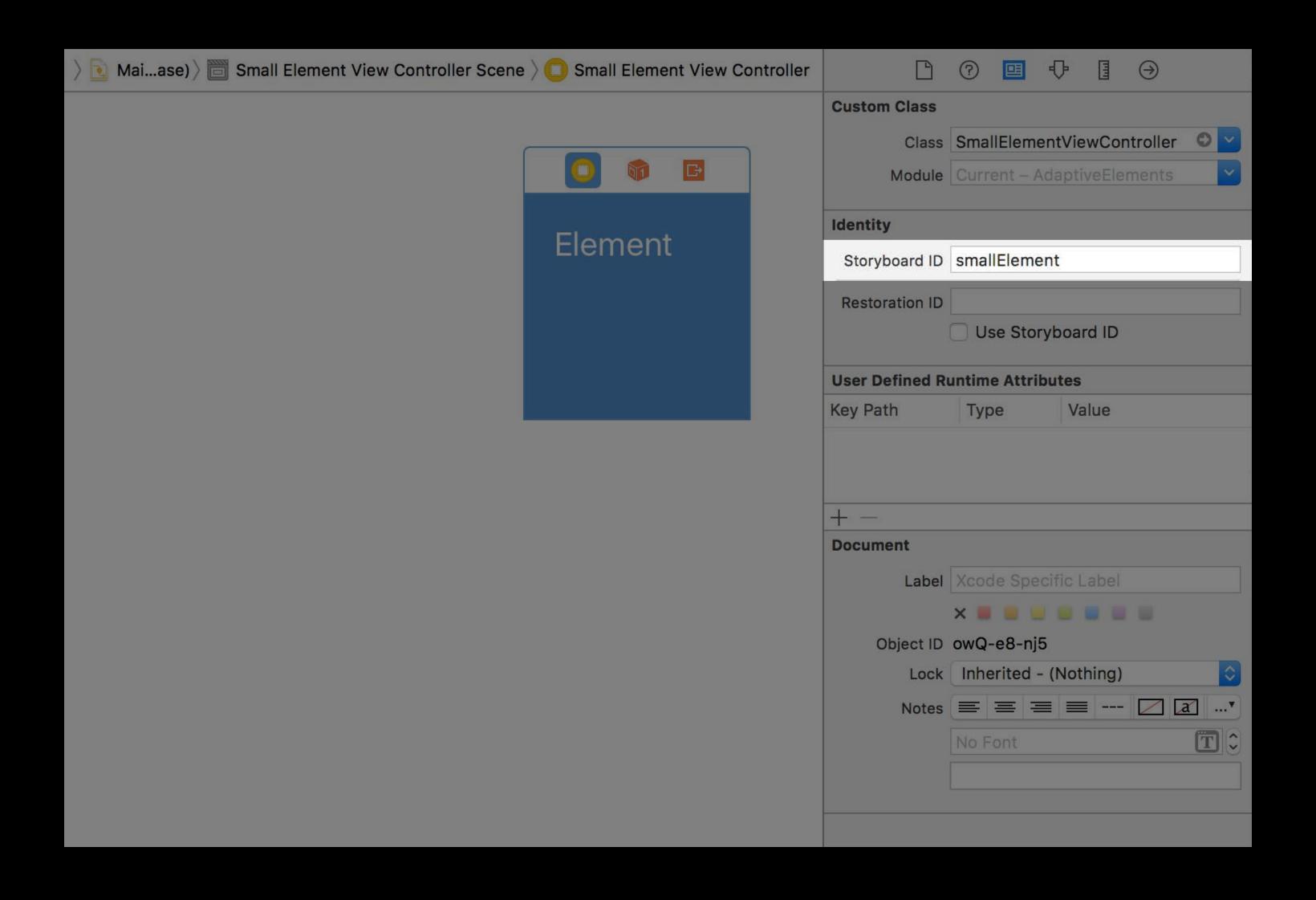
• •••••

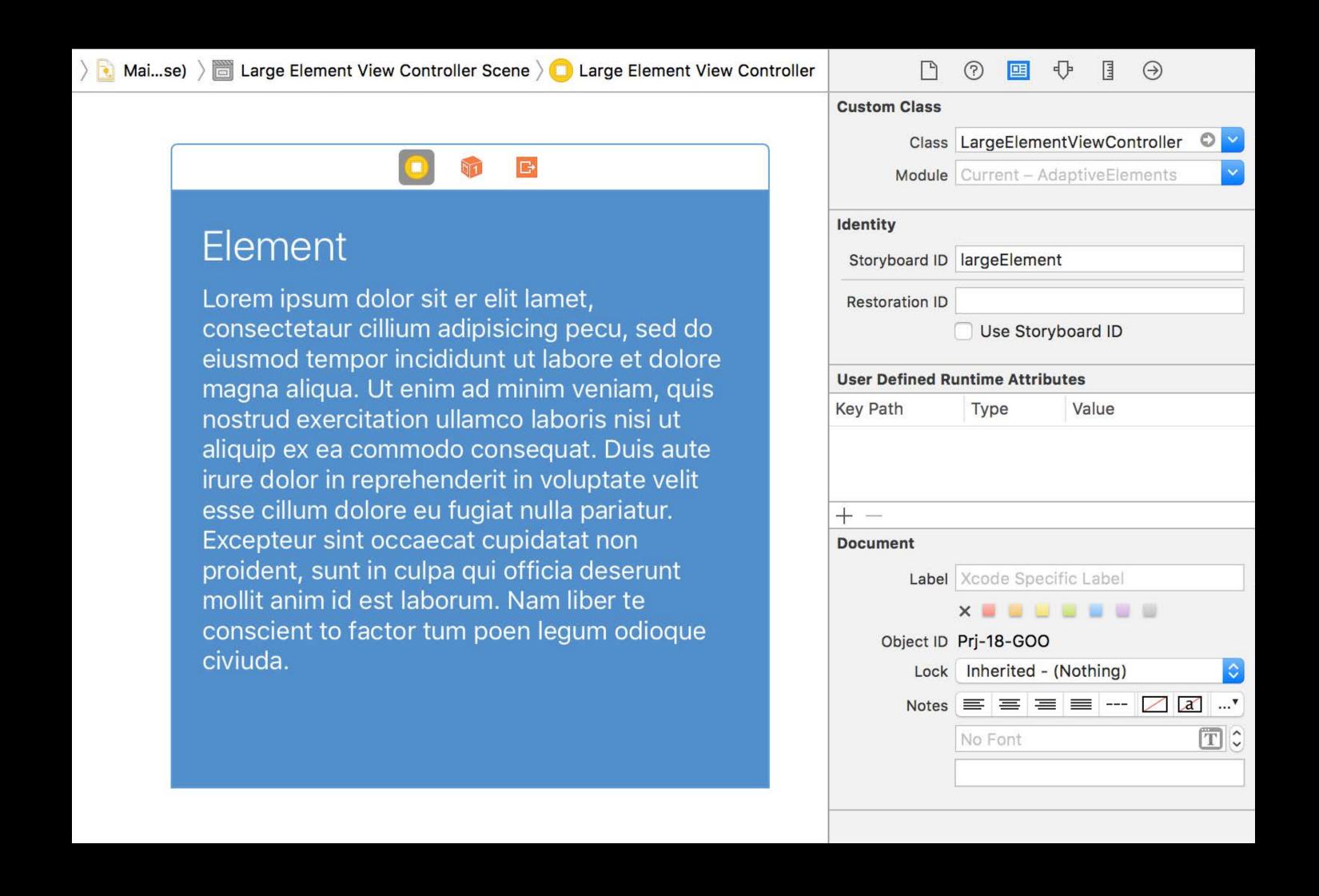
ExampleContainerViewController

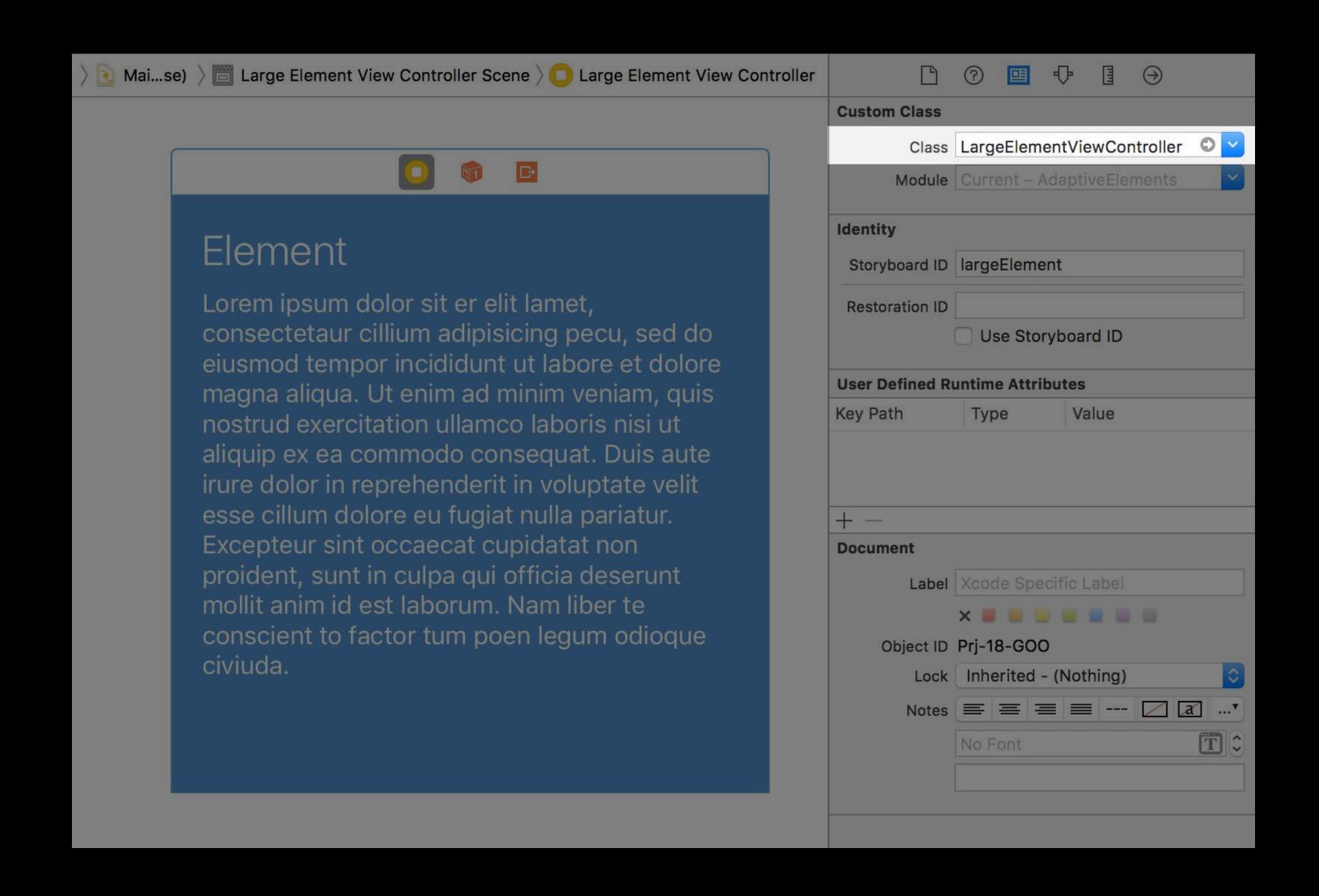
LargeElementViewController

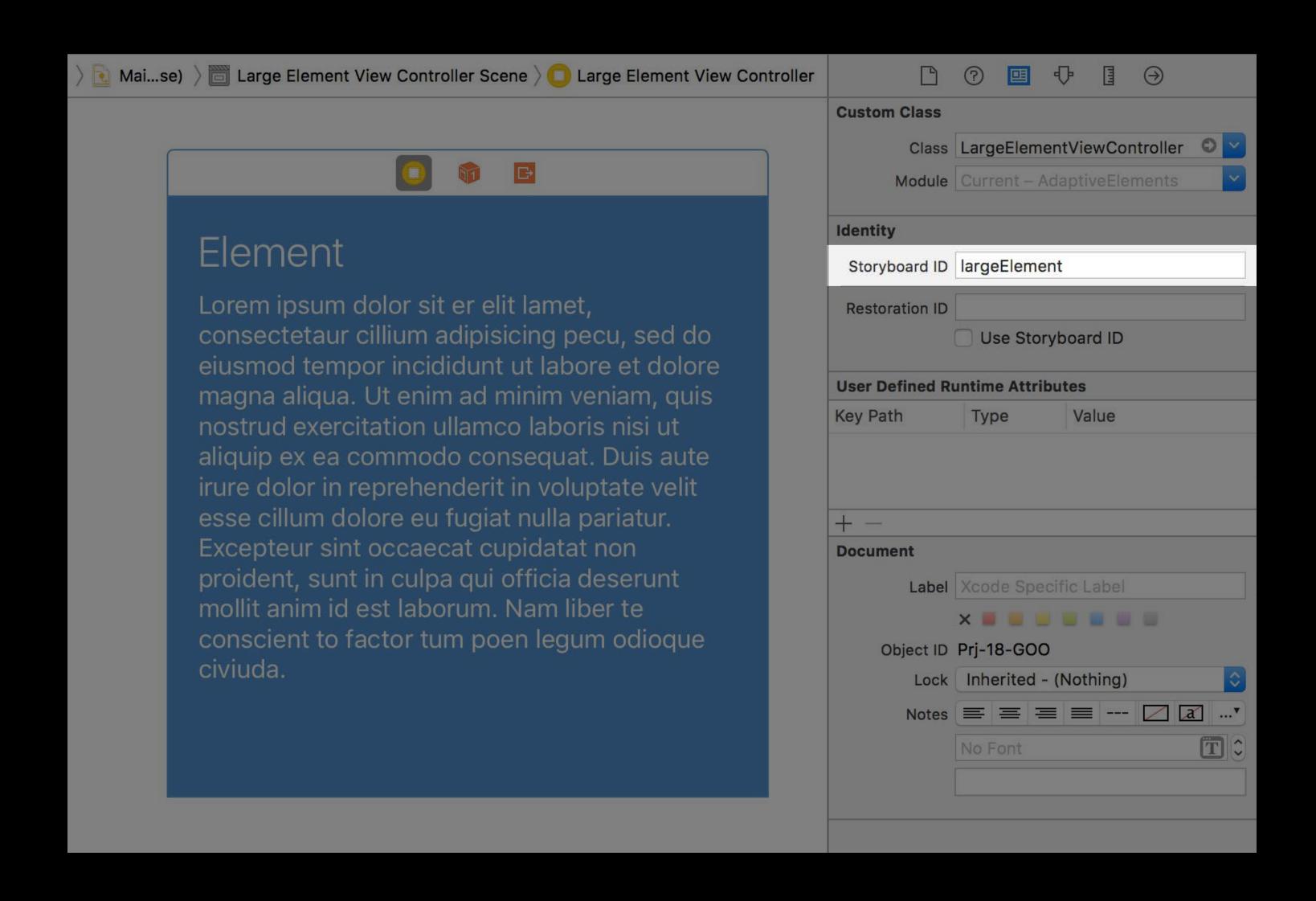












```
class SmallElementViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        let tapGestureRecognizer = UITapGestureRecognizer(
                target: self, action: #selector(self.tapped))
        view.addGestureRecognizer(tapGestureRecognizer)
    func tapped(_ gestureRecognizer: UITapGestureRecognizer) {
        if gestureRecognizer.state == .ended {
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
                    withIdentifier: "largeElement")
            present(newElementViewController, animated: true, completion: nil)
```

```
class SmallElementViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        let tapGestureRecognizer = UITapGestureRecognizer(
                target: self, action: #selector(self.tapped))
        view.addGestureRecognizer(tapGestureRecognizer)
    func tapped(_ gestureRecognizer: UITapGestureRecognizer) {
        if gestureRecognizer.state == .ended {
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
                    withIdentifier: "largeElement")
            present(newElementViewController, animated: true, completion: nil)
```

```
class SmallElementViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        let tapGestureRecognizer = UITapGestureRecognizer(
                target: self, action: #selector(self.tapped))
        view.addGestureRecognizer(tapGestureRecognizer)
    func tapped(_ gestureRecognizer: UITapGestureRecognizer) {
        if gestureRecognizer.state == .ended {
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
                    withIdentifier: "largeElement")
            present(newElementViewController, animated: true, completion: nil)
```

```
class SmallElementViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        let tapGestureRecognizer = UITapGestureRecognizer(
                target: self, action: #selector(self.tapped))
        view.addGestureRecognizer(tapGestureRecognizer)
    func tapped(_ gestureRecognizer: UITapGestureRecognizer) {
        if gestureRecognizer.state == .ended {
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
                    withIdentifier: "largeElement")
            present(newElementViewController, animated: true, completion: nil)
```

```
class SmallElementViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        let tapGestureRecognizer = UITapGestureRecognizer(
                target: self, action: #selector(self.tapped))
        view.addGestureRecognizer(tapGestureRecognizer)
    func tapped(_ gestureRecognizer: UITapGestureRecognizer) {
        if gestureRecognizer.state == .ended {
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
                    withIdentifier: "largeElement")
            present(newElementViewController, animated: true, completion: nil)
```

```
class SmallElementViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        let tapGestureRecognizer = UITapGestureRecognizer(
                target: self, action: #selector(self.tapped))
        view.addGestureRecognizer(tapGestureRecognizer)
    func tapped(_ gestureRecognizer: UITapGestureRecognizer) {
        if gestureRecognizer.state == .ended {
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
                    withIdentifier: "largeElement")
            present(newElementViewController, animated: true, completion: nil)
```

```
class LargeElementViewController: UIViewController {
    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        if isBeingPresented() {
            let tapGestureRecognizer = UITapGestureRecognizer(
                    target: self, action: #selector(self.tapped))
            view.addGestureRecognizer(tapGestureRecognizer)
    func tapped(_ gestureRecognizer: UITapGestureRecognizer) {
        if gestureRecognizer.state == .ended {
            dismiss(animated: true, completion: nil)
```

```
class LargeElementViewController: UIViewController {
    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        if isBeingPresented() {
            let tapGestureRecognizer = UITapGestureRecognizer()
                    target: self, action: #selector(self.tapped))
            view.addGestureRecognizer(tapGestureRecognizer)
    func tapped(_ gestureRecognizer: UITapGestureRecognizer) {
        if gestureRecognizer.state == .ended {
            dismiss(animated: true, completion: nil)
```

```
class LargeElementViewController: UIViewController {
    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        if isBeingPresented() {
            let tapGestureRecognizer = UITapGestureRecognizer(
                    target: self, action: #selector(self.tapped))
            view.addGestureRecognizer(tapGestureRecognizer)
    func tapped(_ gestureRecognizer: UITapGestureRecognizer) {
        if gestureRecognizer.state == .ended {
            dismiss(animated: true, completion: nil)
```

```
class LargeElementViewController: UIViewController {
    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        if isBeingPresented() {
            let tapGestureRecognizer = UITapGestureRecognizer(
                    target: self, action: #selector(self.tapped))
            view.addGestureRecognizer(tapGestureRecognizer)
    func tapped(_ gestureRecognizer: UITapGestureRecognizer) {
        if gestureRecognizer.state == .ended {
            dismiss(animated: true, completion: nil)
```

Stores information describing a design

Stores information describing a design Immutable struct for safety

Stores information describing a design Immutable struct for safety Allows comparison

```
struct Design {
    let axis: UILayoutConstraintAxis // .vertical or .horizontal
    enum ElementKind {
        case small
        case large
    let elementKind: ElementKind
    var elementIdentifier: String {
        switch elementKind {
            case .small: return "smallElement"
            case .large: return "largeElement"
```

```
struct Design {
    let axis: UILayoutConstraintAxis // .vertical or .horizontal
    enum ElementKind {
        case small
        case large
    let elementKind: ElementKind
    var elementIdentifier: String {
```

case .small: return "smallElement"

case .large: return "largeElement"

switch elementKind {

```
struct Design {
    let axis: UILayoutConstraintAxis // .vertical or .horizontal
    enum ElementKind {
        case small
        case large
    let elementKind: ElementKind
    var elementIdentifier: String {
        switch elementKind {
            case .small: return "smallElement"
            case .large: return "largeElement"
```

```
struct Design {
    let axis: UILayoutConstraintAxis // .vertical or .horizontal
    enum ElementKind {
        case small
        case large
    let elementKind: ElementKind
    var elementIdentifier: String {
        switch elementKind {
            case .small: return "smallElement"
            case .large: return "largeElement"
```

```
extension Design: Equatable { }

func == (left: Design, right: Design) -> Bool {
    return left.axis == right.axis && left.elementKind == right.elementKind
}
```

```
extension Design: Equatable { }

func == (left: Design, right: Design) -> Bool {
    return left.axis == right.axis && left.elementKind == right.elementKind
}
```

Has three child view controllers

Has three child view controllers

Uses rules to decide a design

Has three child view controllers

Uses rules to decide a design

Updates at each layout

```
class ExampleContainerViewController: UIViewController {
    var elementViewControllers: [UIViewController?] = [nil, nil, nil]
    var displayedDesign: Design? = nil
    override func viewWillLayoutSubviews() {
       let size = view.bounds.size
       let newDesign = decideDesign(size)
       if displayedDesign != newDesign {
            applyDesign(newDesign)
            displayedDesign = newDesign
```

```
class ExampleContainerViewController: UIViewController {
    var elementViewControllers: [UIViewController?] = [nil, nil, nil]
    var displayedDesign: Design? = nil
    override func viewWillLayoutSubviews() {
       let size = view.bounds.size
       let newDesign = decideDesign(size)
       if displayedDesign != newDesign {
            applyDesign(newDesign)
            displayedDesign = newDesign
```

```
class ExampleContainerViewController: UIViewController {
    var elementViewControllers: [UIViewController?] = [nil, nil, nil]
    var displayedDesign: Design? = nil
    override func viewWillLayoutSubviews() {
       let size = view.bounds.size
       let newDesign = decideDesign(size)
       if displayedDesign != newDesign {
            applyDesign(newDesign)
            displayedDesign = newDesign
```

```
class ExampleContainerViewController: UIViewController {
   var elementViewControllers: [UIViewController?] = [nil, nil, nil]
   var displayedDesign: Design? = nil

   override func viewWillLayoutSubviews() {
      let size = view.bounds.size

      let newDesign = decideDesign(size)

      if displayedDesign != newDesign {
```

applyDesign(newDesign)

displayedDesign = newDesign

```
class ExampleContainerViewController: UIViewController {
    var elementViewControllers: [UIViewController?] = [nil, nil, nil]
    var displayedDesign: Design? = nil
    override func viewWillLayoutSubviews() {
       let size = view.bounds.size
       let newDesign = decideDesign(size)
       if displayedDesign != newDesign {
            applyDesign(newDesign)
```

displayedDesign = newDesign

```
class ExampleContainerViewController: UIViewController {
    var elementViewControllers: [UIViewController?] = [nil, nil, nil]
    var displayedDesign: Design? = nil
    override func viewWillLayoutSubviews() {
       let size = view.bounds.size
       let newDesign = decideDesign(size)
       if displayedDesign != newDesign {
           applyDesign(newDesign)
            displayedDesign = newDesign
```

```
func decideDesign(_ size: CGSize) -> Design {
    let axis: UILayoutConstraintAxis
    if size width > size height {
        axis = horizontal
   } else {
        axis = .vertical
    let elementKind: Design.ElementKind
    let widthThreshold = CGFloat(750)
    if size.width < widthThreshold {</pre>
        elementKind = .small
   } else {
        elementKind = .large
    return Design(axis: axis, elementKind: elementKind)
```

```
func decideDesign(_ size: CGSize) -> Design {
    let axis: UILayoutConstraintAxis
   if size.width > size.height {
        axis = horizontal
   } else {
        axis = .vertical
    let elementKind: Design.ElementKind
    let widthThreshold = CGFloat(750)
    if size.width < widthThreshold {</pre>
        elementKind = .small
   } else {
```

elementKind = .large

return Design(axis: axis, elementKind: elementKind)

```
func decideDesign(_ size: CGSize) -> Design {
    let axis: UILayoutConstraintAxis
    if size width > size height {
        axis = .horizontal
   } else {
        axis = •vertical
    let elementKind: Design.ElementKind
    let widthThreshold = CGFloat(750)
    if size.width < widthThreshold {</pre>
        elementKind = .small
   } else {
        elementKind = .large
```

return Design(axis: axis, elementKind: elementKind)

```
func decideDesign(_ size: CGSize) -> Design {
    let axis: UILayoutConstraintAxis
    if size width > size height {
        axis = horizontal
   } else {
        axis = .vertical
    let elementKind: Design.ElementKind
    let widthThreshold = CGFloat(750)
    if size.width < widthThreshold {</pre>
        elementKind = .small
   } else {
        elementKind = large
```

```
return Design(axis: axis, elementKind: elementKind)
```

```
func applyDesign(_ newDesign: Design) {
   stackView.axis = newDesign.axis
   if displayedDesign?.elementKind != newDesign.elementKind {
       for (index, elementViewController) in elementViewControllers.enumerated() {
           if let oldElementViewController = elementViewController {
                removeOldElementViewController(oldElementViewController)
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
               withIdentifier: newDesign.elementIdentifier)
           addNewElementViewController(newElementViewController)
           elementViewControllers[index] = newElementViewController
```

```
func applyDesign(_ newDesign: Design) {
   stackView axis = newDesign axis
   if displayedDesign?.elementKind != newDesign.elementKind {
       for (index, elementViewController) in elementViewControllers.enumerated() {
           if let oldElementViewController = elementViewController {
                removeOldElementViewController(oldElementViewController)
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
               withIdentifier: newDesign.elementIdentifier)
           addNewElementViewController(newElementViewController)
           elementViewControllers[index] = newElementViewController
```

```
func applyDesign(_ newDesign: Design) {
   stackView.axis = newDesign.axis
   if displayedDesign?.elementKind != newDesign.elementKind {
       for (index, elementViewController) in elementViewControllers.enumerated() {
           if let oldElementViewController = elementViewController {
                removeOldElementViewController(oldElementViewController)
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
               withIdentifier: newDesign.elementIdentifier)
           addNewElementViewController(newElementViewController)
           elementViewControllers[index] = newElementViewController
```

```
func applyDesign(_ newDesign: Design) {
   stackView.axis = newDesign.axis
   if displayedDesign?.elementKind != newDesign.elementKind {
       for (index, elementViewController) in elementViewControllers.enumerated() {
           if let oldElementViewController = elementViewController {
                removeOldElementViewController(oldElementViewController)
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
               withIdentifier: newDesign.elementIdentifier)
           addNewElementViewController(newElementViewController)
           elementViewControllers[index] = newElementViewController
```

```
func applyDesign(_ newDesign: Design) {
   stackView.axis = newDesign.axis
   if displayedDesign?.elementKind != newDesign.elementKind {
       for (index, elementViewController) in elementViewControllers.enumerated() {
           if let oldElementViewController = elementViewController {
                removeOldElementViewController(oldElementViewController)
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
               withIdentifier: newDesign.elementIdentifier)
           addNewElementViewController(newElementViewController)
           elementViewControllers[index] = newElementViewController
```

```
func applyDesign(_ newDesign: Design) {
   stackView.axis = newDesign.axis
   if displayedDesign?.elementKind != newDesign.elementKind {
       for (index, elementViewController) in elementViewControllers.enumerated() {
           if let oldElementViewController = elementViewController {
                removeOldElementViewController(oldElementViewController)
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
               withIdentifier: newDesign.elementIdentifier)
           addNewElementViewController(newElementViewController)
           elementViewControllers[index] = newElementViewController
```

```
func applyDesign(_ newDesign: Design) {
   stackView.axis = newDesign.axis
   if displayedDesign?.elementKind != newDesign.elementKind {
       for (index, elementViewController) in elementViewControllers.enumerated() {
           if let oldElementViewController = elementViewController {
                removeOldElementViewController(oldElementViewController)
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let newElementViewController = storyboard.instantiateViewController(
               withIdentifier: newDesign.elementIdentifier)
           addNewElementViewController(newElementViewController)
           elementViewControllers[index] = newElementViewController
```

```
// Helper functions to be a well-behaved container view controller:
func addNewElementViewController(_ elementViewController: UIViewController)
    addChildViewController(elementViewController)
    stackView.addArrangedSubview(elementViewController.view)
    elementViewController.didMove(toParentViewController: self)
func removeOldElementViewController(_ elementViewController: UIViewController)
    elementViewController.willMove(toParentViewController: nil)
    elementViewController.view.removeFromSuperview()
    elementViewController.removeFromParentViewController()
```

```
// Helper functions to be a well-behaved container view controller:
func addNewElementViewController(_ elementViewController: UIViewController)
    addChildViewController(elementViewController)
    stackView.addArrangedSubview(elementViewController.view)
    elementViewController.didMove(toParentViewController: self)
func removeOldElementViewController(_ elementViewController: UIViewController)
    elementViewController.willMove(toParentViewController: nil)
    elementViewController.view.removeFromSuperview()
    elementViewController.removeFromParentViewController()
```

```
// Helper functions to be a well-behaved container view controller:
func addNewElementViewController(_ elementViewController: UIViewController)
    addChildViewController(elementViewController)
    stackView.addArrangedSubview(elementViewController.view)
    elementViewController.didMove(toParentViewController: self)
func removeOldElementViewController(_ elementViewController: UIViewController)
    elementViewController.willMove(toParentViewController: nil)
    elementViewController.view.removeFromSuperview()
    elementViewController.removeFromParentViewController()
```

```
// Helper functions to be a well-behaved container view controller:
func addNewElementViewController(_ elementViewController: UIViewController)
{
    addChildViewController(elementViewController)
    stackView.addArrangedSubview(elementViewController.view)
    elementViewController.didMove(toParentViewController: self)
}

func removeOldElementViewController(_ elementViewController: UIViewController)
{
```

elementViewController.willMove(toParentViewController: nil)

elementViewController.removeFromParentViewController()

elementViewController.view.removeFromSuperview()

```
// Helper functions to be a well-behaved container view controller:
func addNewElementViewController(_ elementViewController: UIViewController)
    addChildViewController(elementViewController)
    stackView.addArrangedSubview(elementViewController.view)
    elementViewController.didMove(toParentViewController: self)
func removeOldElementViewController(_ elementViewController: UIViewController)
    elementViewController.willMove(toParentViewController: nil)
    elementViewController.view.removeFromSuperview()
    elementViewController.removeFromParentViewController()
```

```
// Helper functions to be a well-behaved container view controller:
func addNewElementViewController(_ elementViewController: UIViewController)
    addChildViewController(elementViewController)
    stackView.addArrangedSubview(elementViewController.view)
    elementViewController.didMove(toParentViewController: self)
func removeOldElementViewController(_ elementViewController: UIViewController)
    elementViewController.willMove(toParentViewController: nil)
    elementViewController.view.removeFromSuperview()
    elementViewController.removeFromParentViewController()
```

```
// Helper functions to be a well-behaved container view controller:
func addNewElementViewController(_ elementViewController: UIViewController)
    addChildViewController(elementViewController)
    stackView.addArrangedSubview(elementViewController.view)
    elementViewController.didMove(toParentViewController: self)
func removeOldElementViewController(_ elementViewController: UIViewController)
    elementViewController.willMove(toParentViewController: nil)
   elementViewController.view.removeFromSuperview()
    elementViewController.removeFromParentViewController()
```

Element

Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

Element

Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

Element

Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.



Element

Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

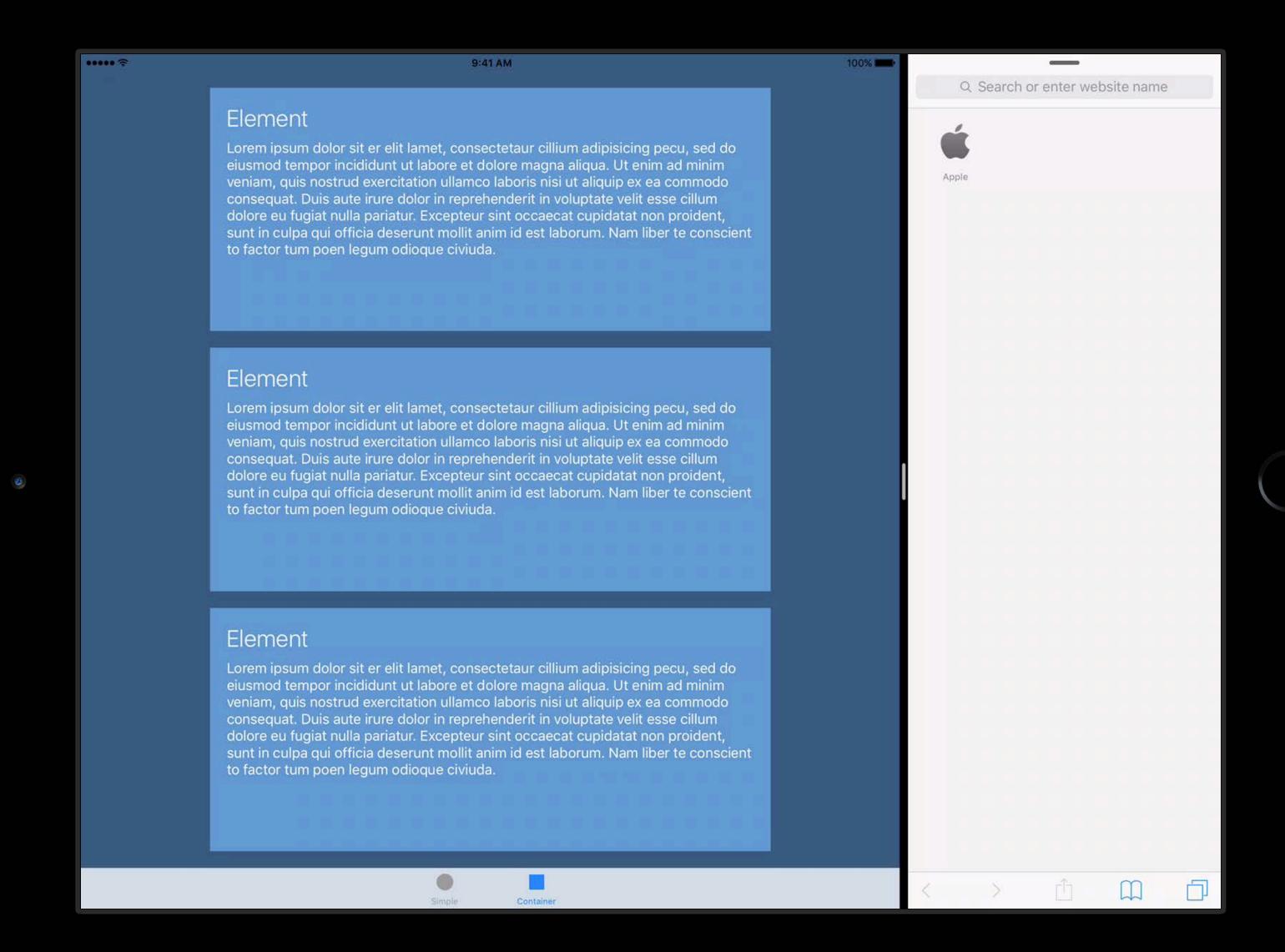
Element

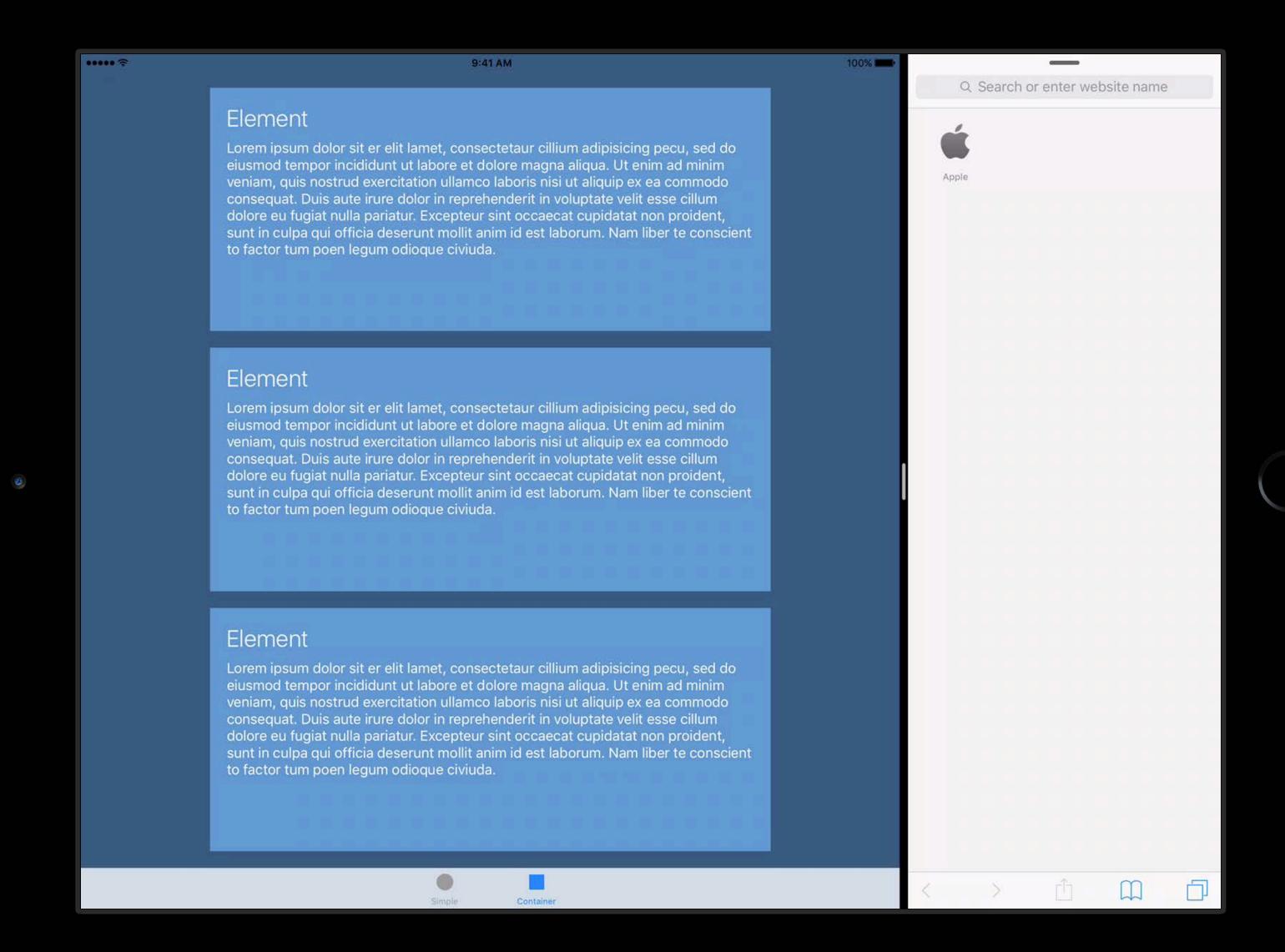
Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

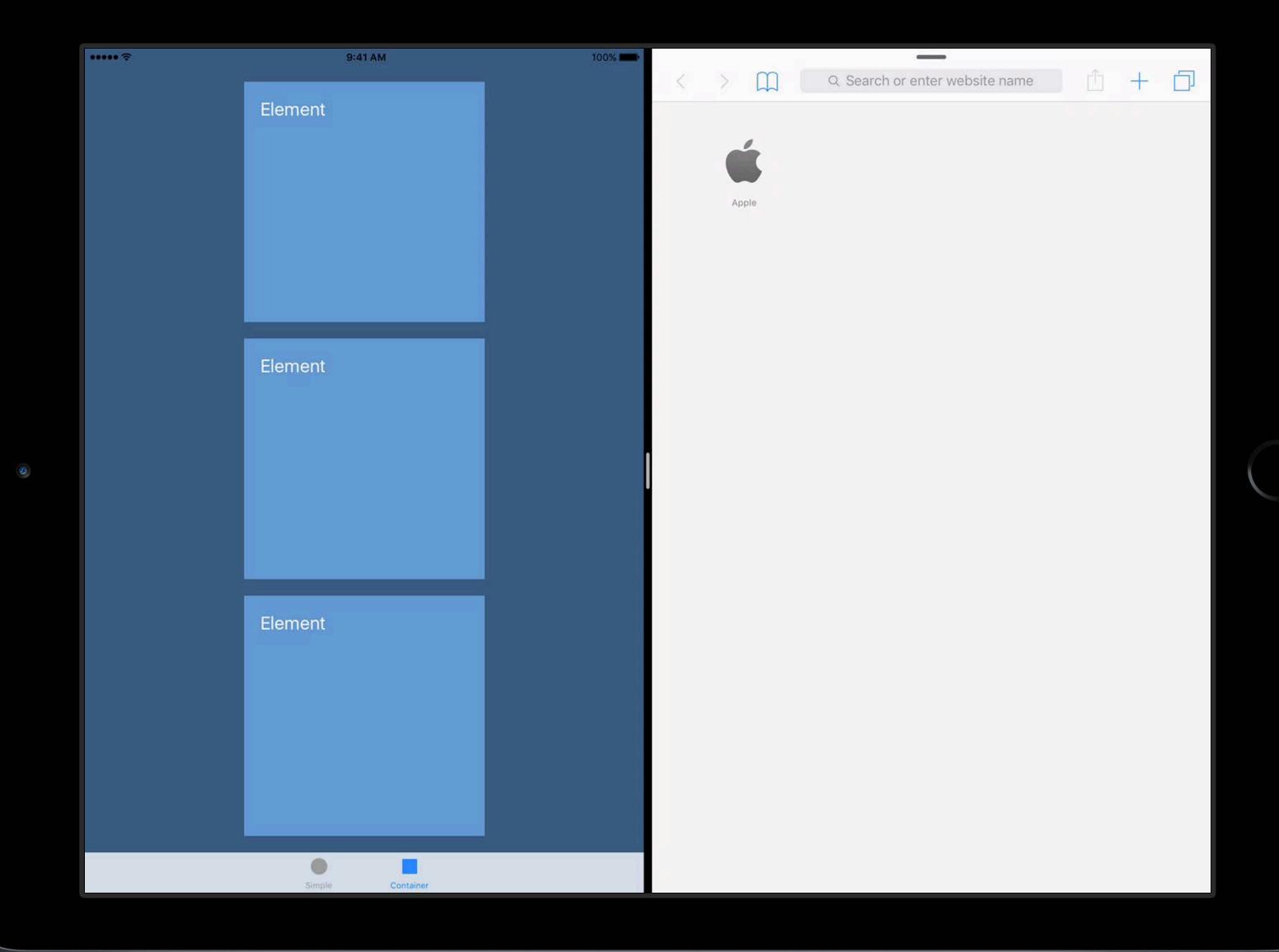
Element

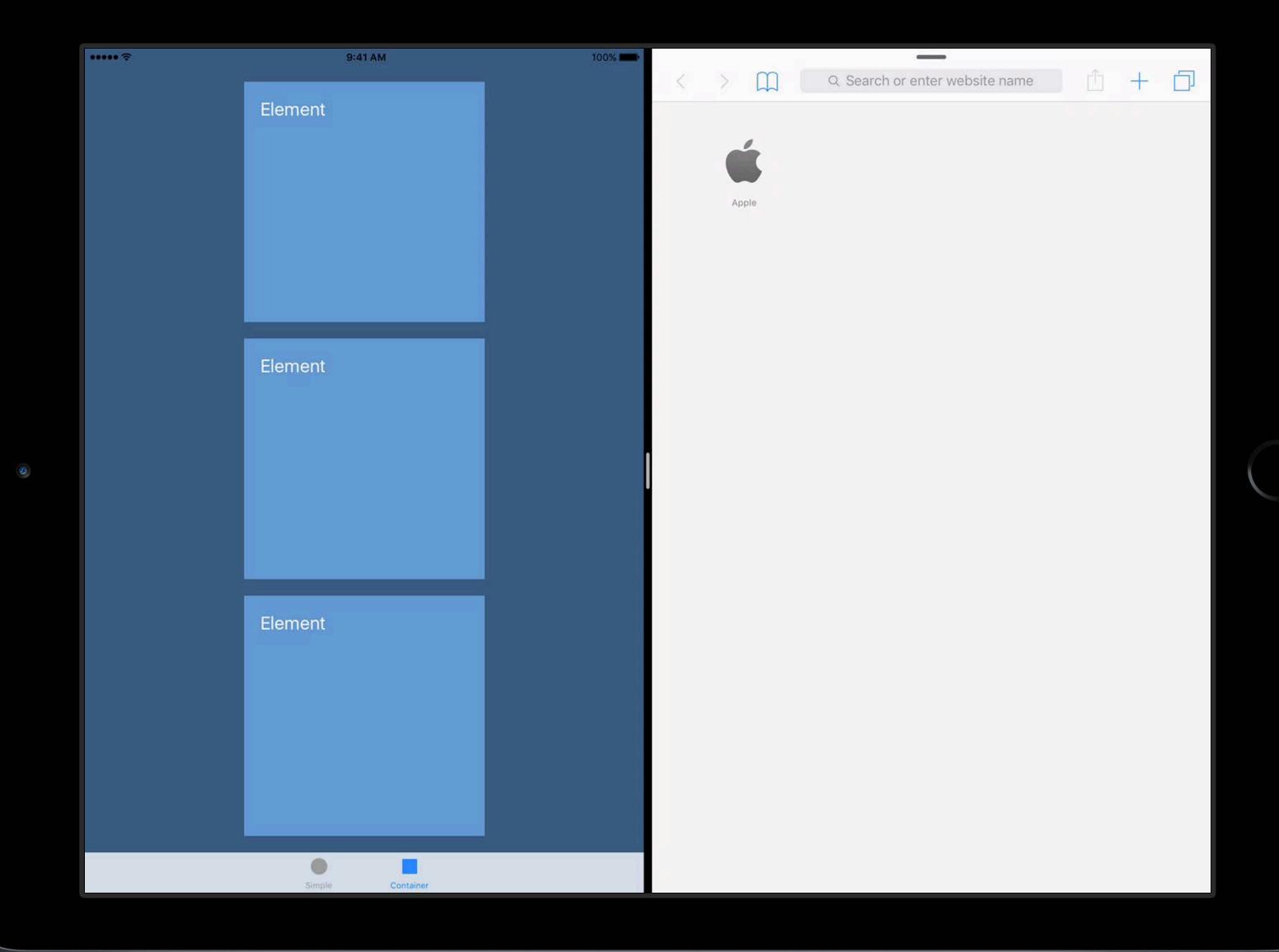
Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.











••••• 100% 9:41 AM Q Search or enter website name Element Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

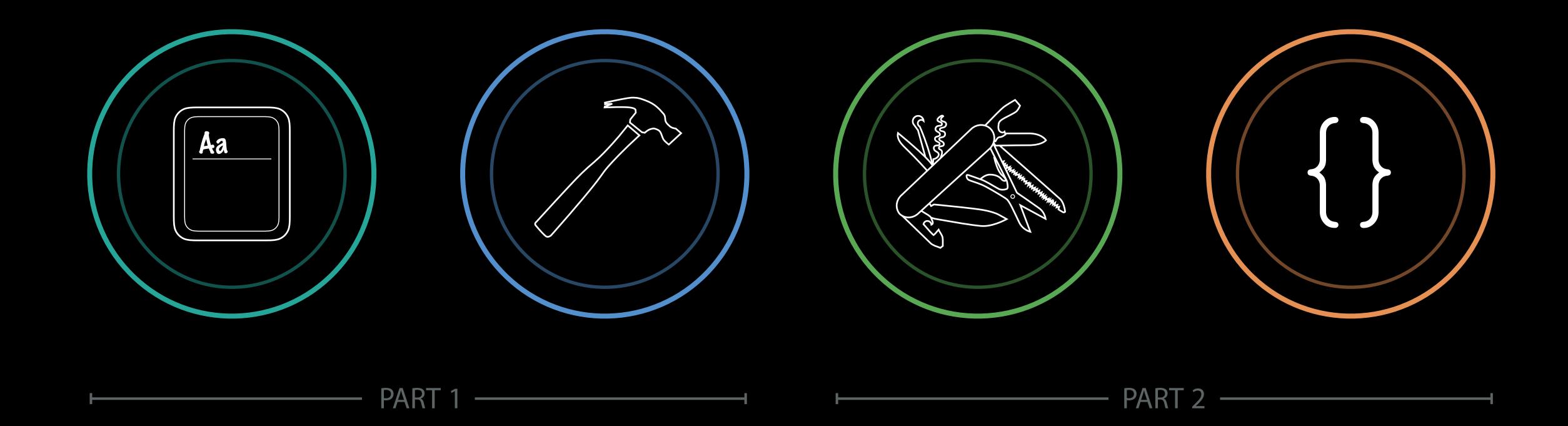
••••• 100% 9:41 AM Q Search or enter website name Element Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

Design for all combinations of device, orientation, and size

Design for all combinations of device, orientation, and size Implement each design and change between them

Design for all combinations of device, orientation, and size Implement each design and change between them

Use reusable elements



More Information

https://developer.apple.com/wwdc16/233

Related Sessions

Making Apps Adaptive, Part 1	Presidio	Wednesday 11:00AM
Inclusive App Design	Pacific Heights	Tuesday 10:00AM
What's New in UlCollectionView in iOS 10	Presidio	Thursday 9:00AM
What's New in Auto Layout	Presidio	Friday 3:00PM
Building Adaptive Apps with UlKit		WWDC 2014
Adopting Multitasking in iOS 9		WWDC 2015

Labs

Cocoa Touch and 3D Touch Lab

Frameworks Lab C Friday 10:30AM

ÓWWDC16