Cocoa Touch Best Practices

Session 231

Luke Hiesterman UIKit Engineer

Agenda

Agenda

App Lifecycle

Agenda

App Lifecycle

Views and View Controllers

Agenda

App Lifecycle

Views and View Controllers

Auto Layout

Agenda

App Lifecycle

Views and View Controllers

Auto Layout

Table and Collection Views

Goals

Goals

Performance

Goals

Performance

User Experience

Goals

Performance

User Experience

Future Proofing

App Lifecycle

Best practices begin here

Launch Quickly

Launch Quickly

Return quickly from `applicationDidFinishLaunching`

Launch Quickly

Return quickly from `applicationDidFinishLaunching`

- Defer long running work

Launch Quickly

Return quickly from `applicationDidFinishLaunching`

- Defer long running work
- App killed if too much time passes

Beyond App Launch

Being responsive to every input

Beyond App Launch

Being responsive to every input

Not just about asynchrony

Beyond App Launch

Being responsive to every input

Not just about asynchrony

Move long running work to background queues

Beyond App Launch

Being responsive to every input

```
func application(application: UIApplication, didFinishLaunchingWithOptions
                                        launchOptions: [NSObject: AnyObject]?) -> Bool {
    globalDataStructure = MyCoolDataStructure()
    globalDataStructure.fetchDataFromDatabase()

    return true
}
```

Beyond App Launch

Being responsive to every input

```
func application(application: UIApplication, didFinishLaunchingWithOptions
                                        launchOptions: [NSObject: AnyObject]?) -> Bool {
    globalDataStructure = MyCoolDataStructure()
    globalDataStructure.fetchDataFromDatabase()
    return true
}
```

Beyond App Launch

Being responsive to every input

```
func application(application: UIApplication, didFinishLaunchingWithOptions
                                    launchOptions: [NSObject: AnyObject]?) -> Bool {
    globalDataStructure = MyCoolDataStructure()
    dispatch_async(dispatch_get_main_queue()) { // defer work until later
        globalDataStructure.fetchDataFromDatabase()
    }

    return true
}
```

Beyond App Launch

Being responsive to every input

```
func application(application: UIApplication, didFinishLaunchingWithOptions
                                        launchOptions: [NSObject: AnyObject]?) -> Bool {
    globalDataStructure = MyCoolDataStructure()
    let globalQueue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0)
    dispatch_async(globalQueue) { // defer work until later
        globalDataStructure.fetchDataFromDatabase()
    }

    return true
}
```

Launch Quickly Again

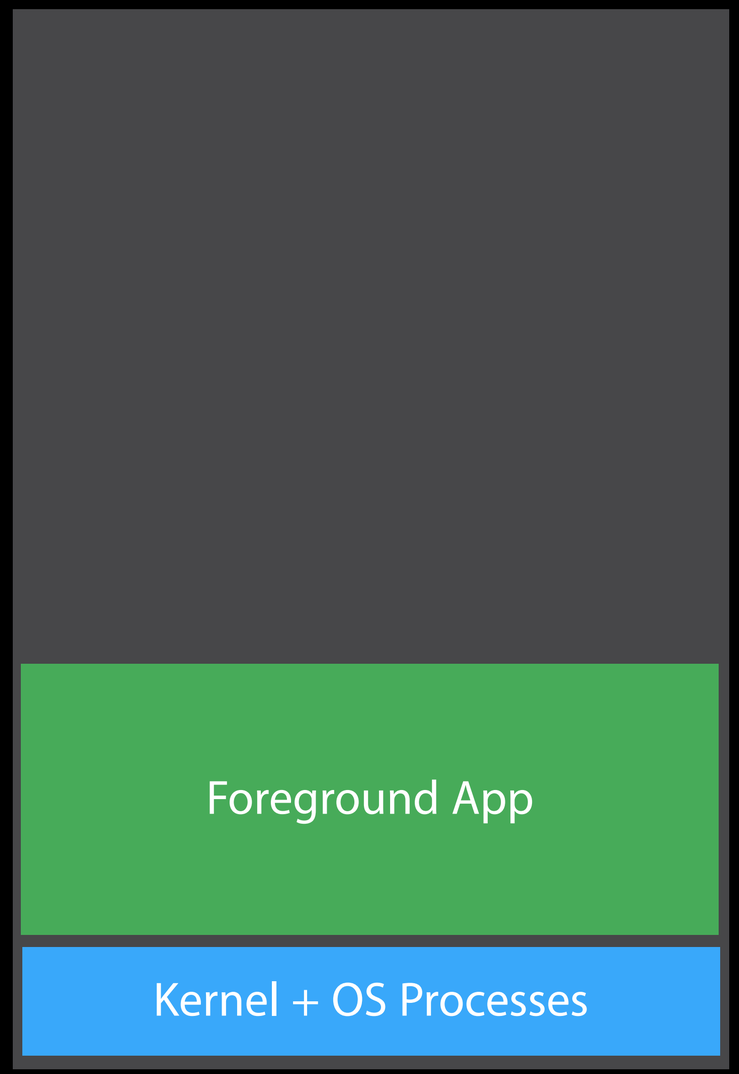
Launch Quickly Again

System Memory

Launch Quickly Again

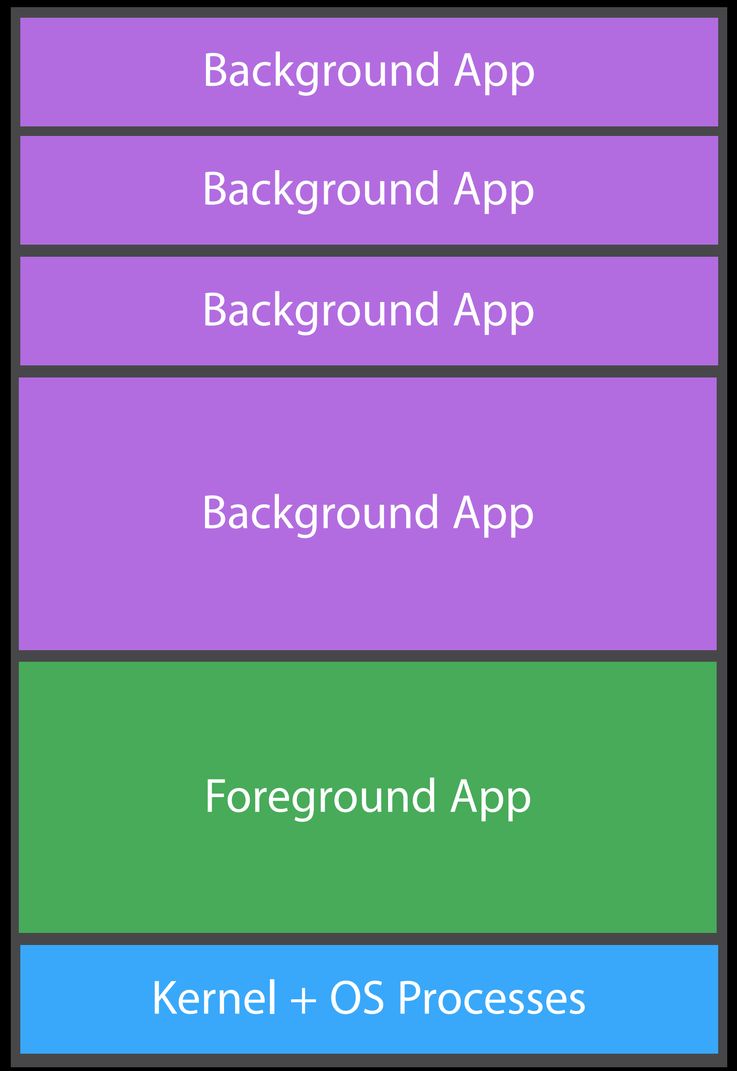
System Memory

Launch Quickly Again

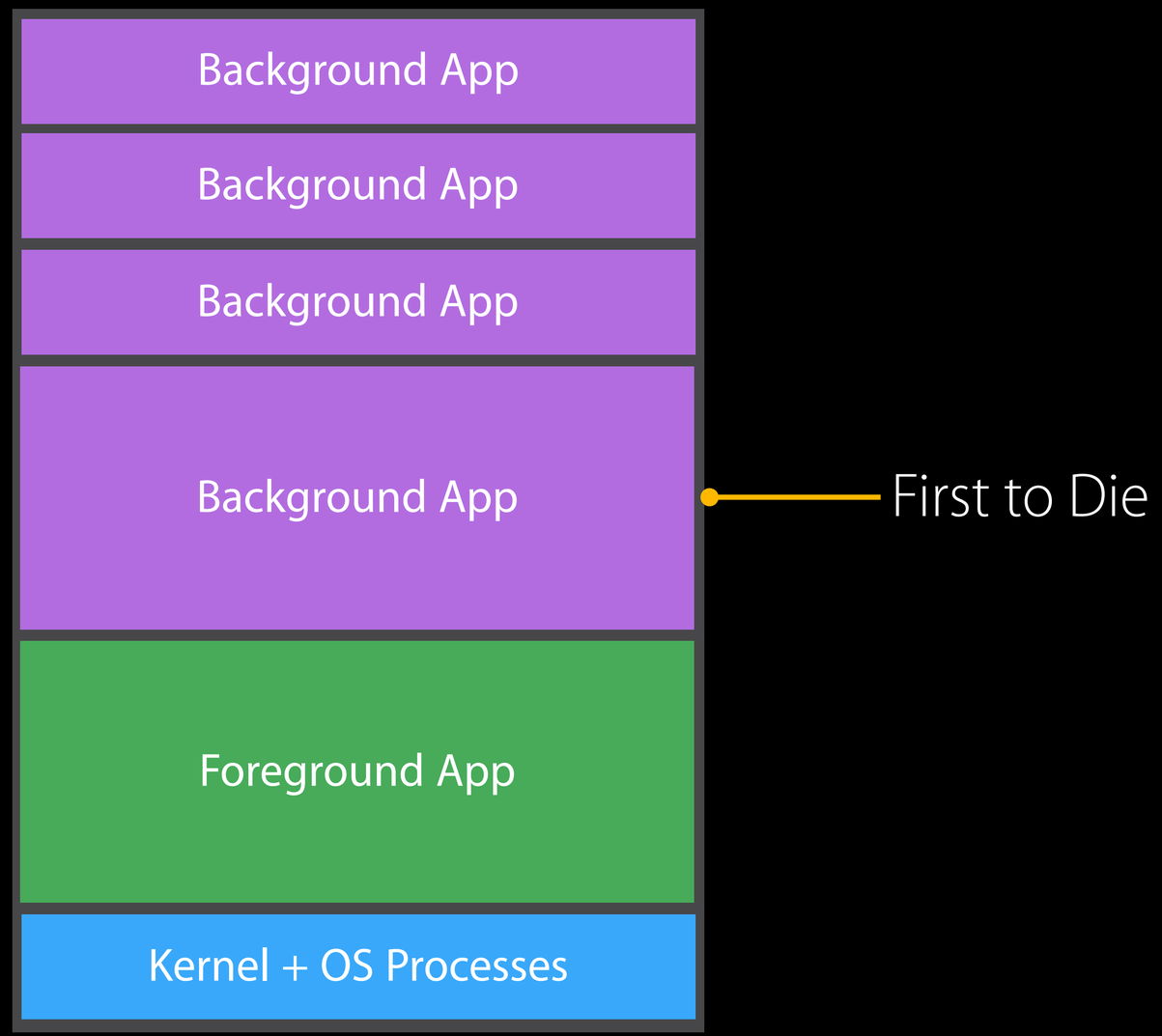


System Memory

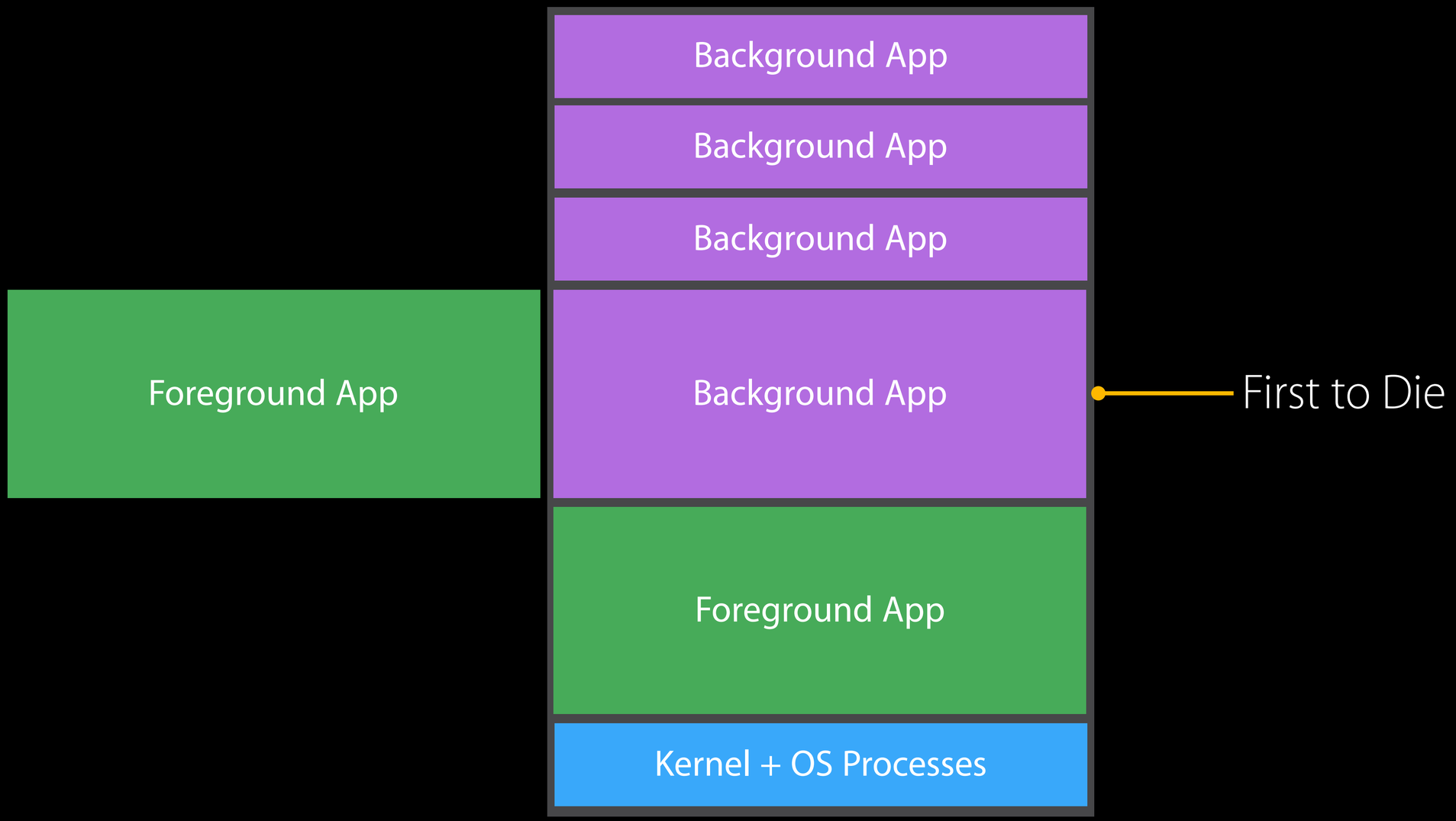
Launch Quickly Again



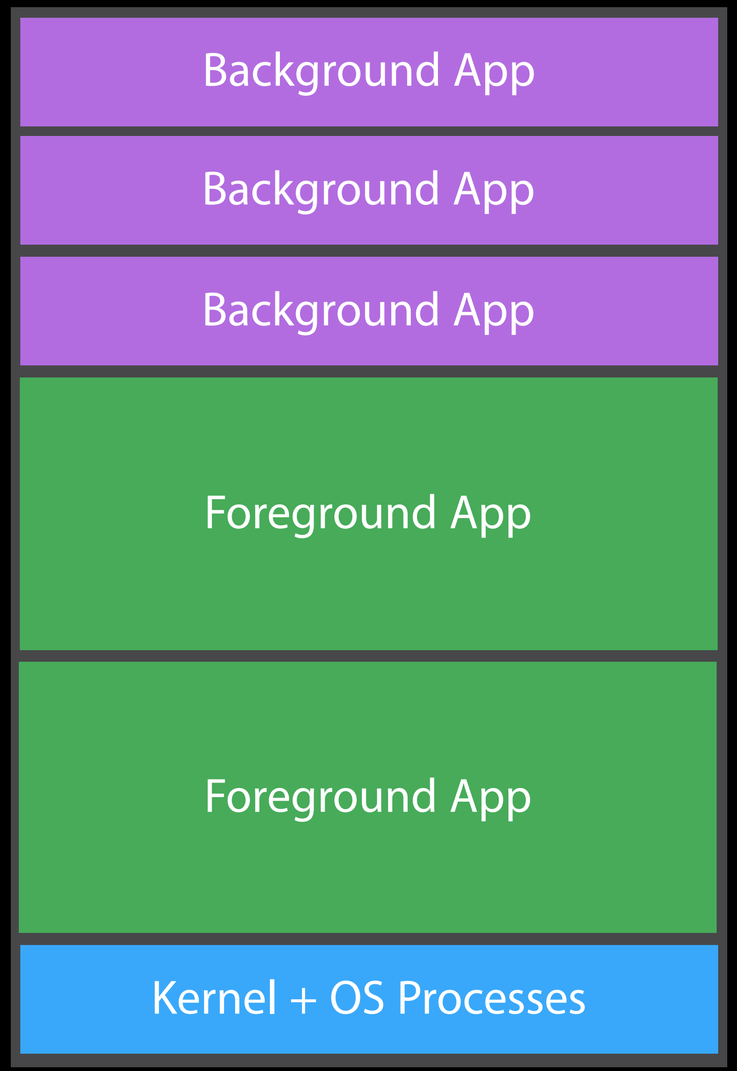
Launch Quickly Again



Launch Quickly Again



Launch Quickly Again



System Memory

Leverage Frameworks

Leverage Frameworks

Reduce maintenance burden

Leverage Frameworks

Reduce maintenance burden

Get improvements for free

Leverage Frameworks

Reduce maintenance burden

Get improvements for free

Focus time on what makes your app special

Leverage Frameworks

Properly manage version change

Leverage Frameworks

Properly manage version change

Target two most recent major releases

Leverage Frameworks

Properly manage version change

Target two most recent major releases

Include version fallbacks

Leverage Frameworks

Properly manage version change

Target two most recent major releases

Include version fallbacks

- e.g., `if systemVersion == 9.0`

Leverage Frameworks

Properly manage version change

Target two most recent major releases

Include version fallbacks

- e.g., `if systemVersion == 9.0`
- e.g., `if systemVersion >= 9.0`

Leverage Frameworks

Properly manage version change

Target two most recent major releases

Include version fallbacks

- e.g., `if systemVersion == 9.0`
- e.g., `if systemVersion >= 9.0`
- e.g., `if #available (iOS 9.0, *) {}`

Leverage Frameworks

Properly manage version change

Target two most recent major releases

Include version fallbacks

- e.g., `if systemVersion == 9.0`
- e.g., `if systemVersion >= 9.0`
- e.g., `if #available (iOS 9.0, *) {}`

Have an else clause

Views and View Controllers

A best practice for every screen

Layout on Modern Devices

Layout on Modern Devices

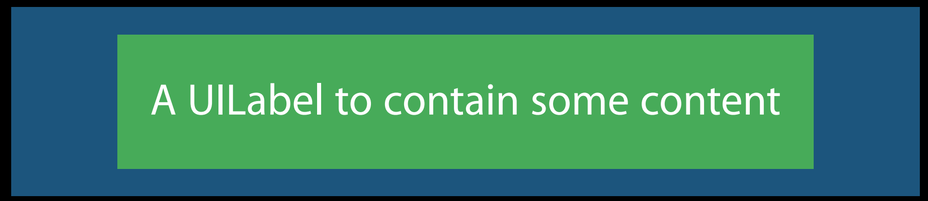
Layout to Proportions

Layout to Proportions

Avoid hard-coded layout values

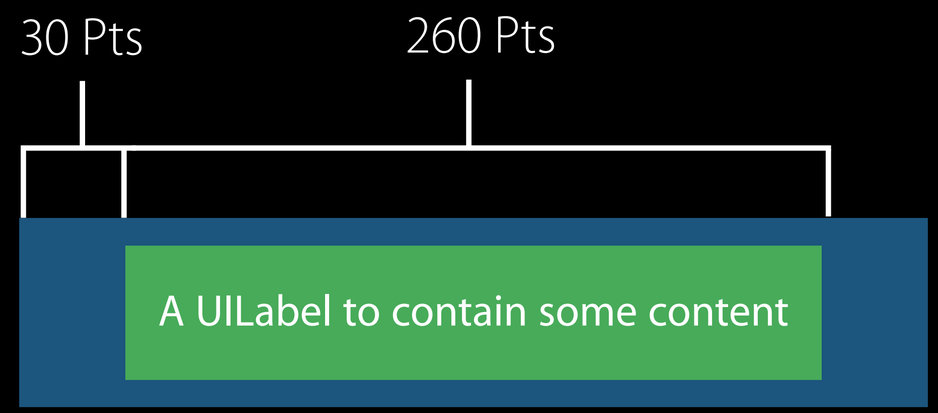
Layout to Proportions

Avoid hard-coded layout values



Layout to Proportions

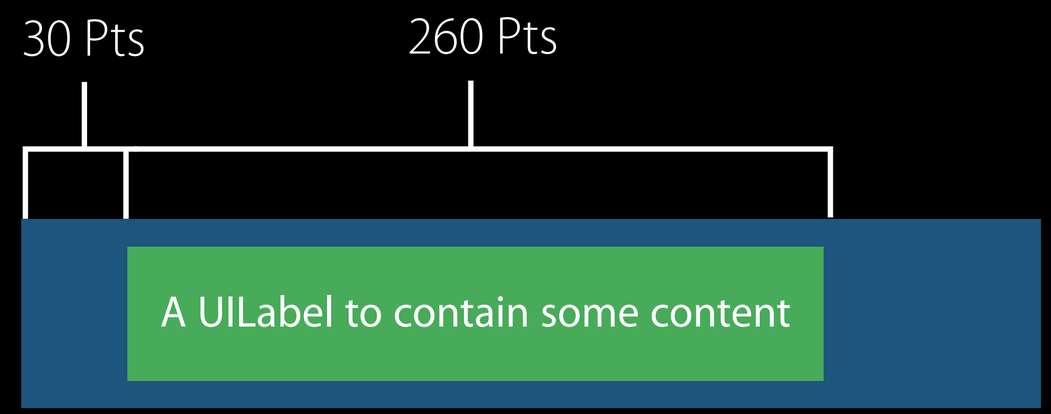
Avoid hard-coded layout values



Layout to Proportions

Avoid hard-coded layout values

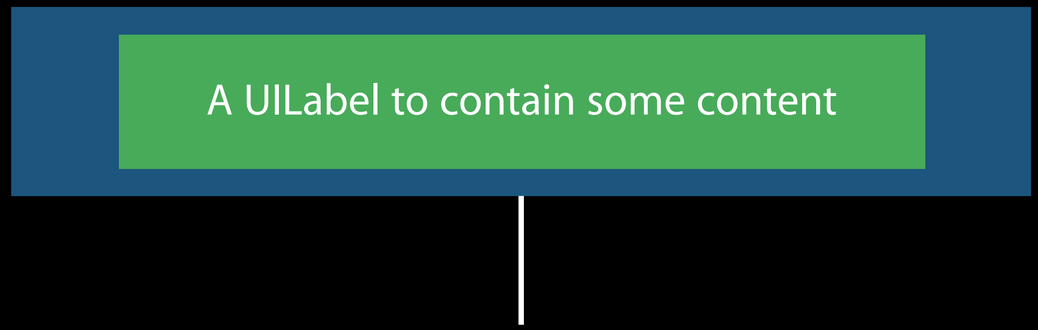
Either or both dimensions may scale



Layout to Proportions

Avoid hard-coded layout values

Either or both dimensions may scale



Size Classes

Size Classes

Some size thresholds trigger major change

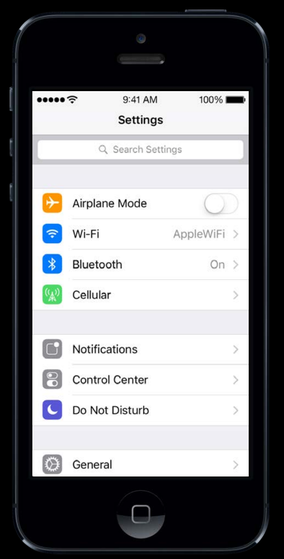
Size Classes

Some size thresholds trigger major change

iPhone 4S

Size Classes

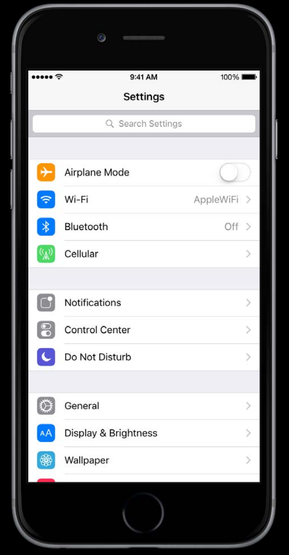
Some size thresholds trigger major change



iPhone 5

Size Classes

Some size thresholds trigger major change



iPhone 6

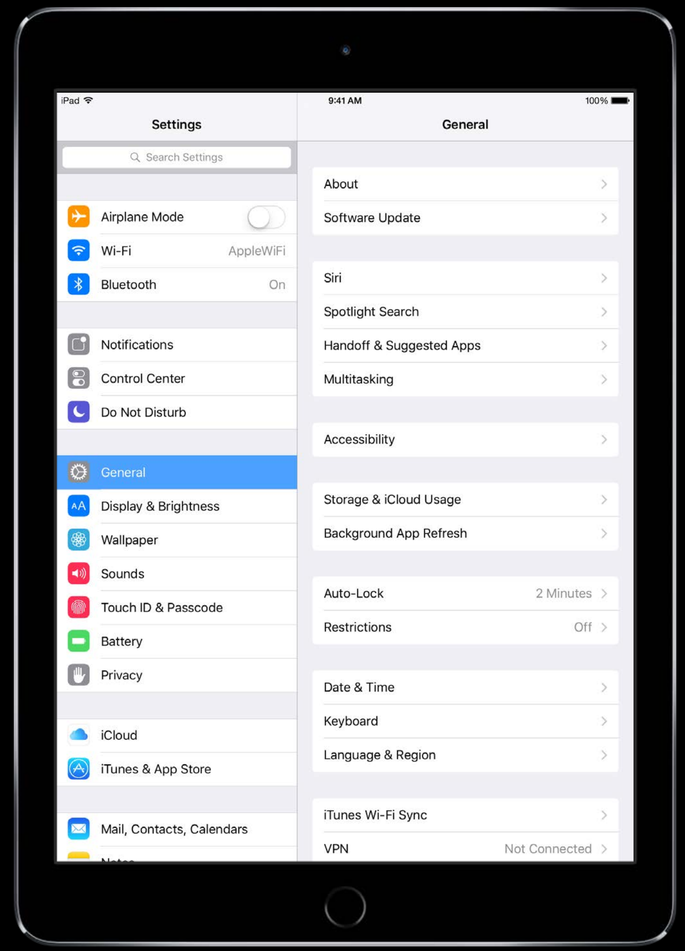
Size Classes

Some size thresholds trigger major change

iPhone 6 Plus

Size Classes

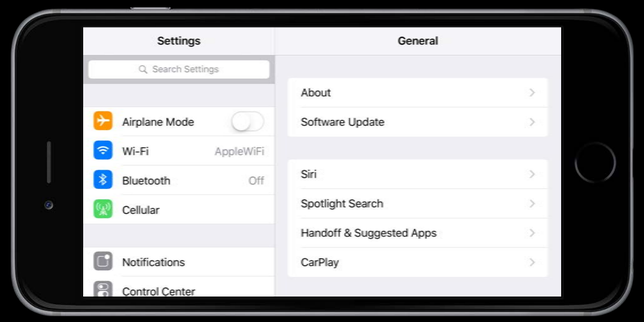
Some size thresholds trigger major change



iPad

Size Classes

Some size thresholds trigger major change

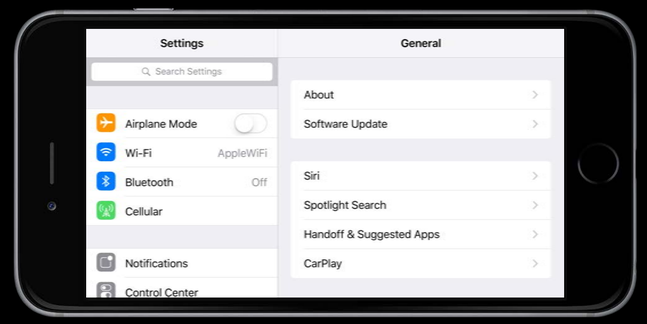


iPhone 6 Plus

Size Classes

Some size thresholds trigger major change

Packaged in UITraitCollection



iPhone 6 Plus

Properties, Not Tags!

Properties, Not Tags!

Avoid `setTag:` and `viewWithTag:`

Properties, Not Tags!

Avoid `setTag:` and `viewWithTag:`

- Possible collisions with other code

Properties, Not Tags!

Avoid `setTag:` and `viewWithTag:`

- Possible collisions with other code
- No compiler warnings

Properties, Not Tags!

Avoid `setTag:` and `viewWithTag:`

- Possible collisions with other code
- No compiler warnings
- No runtime errors

Properties, Not Tags!

Avoid `setTag:` and `viewWithTag:`

- Possible collisions with other code
- No compiler warnings
- No runtime errors

Instance variables and properties provide better alternative

Properties, Not Tags!

```
let ImageViewTag = 1000

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        let imageView = UIView(frame: CGRect(x: 0, y: 0, width: 50, height: 50))
        imageView.tag = ImageViewTag
        view.addSubview(imageView)
    }
}
```

Properties, Not Tags!

```
let ImageViewTag = 1000

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        let imageView = UIView(frame: CGRect(x: 0, y: 0, width: 50, height: 50))
        imageView.tag = ImageViewTag
        view.addSubview(imageView)
    }
}
```

Properties, Not Tags!

```
class ViewController: UIViewController {
    var imageView : UIImageView

    override func viewDidLoad() {
        super.viewDidLoad()
        imageView = UIView(frame: CGRect(x: 0, y: 0, width: 50, height: 50))
        view.addSubview(imageView)
    }
}
```

Make Timing Deterministic

Make Timing Deterministic

Leverage `UIViewControllerTransitionCoordinator`

Make Timing Deterministic

Leverage `UIViewControllerTransitionCoordinator`

- Animate alongside a transition

Make Timing Deterministic

Leverage `UIViewControllerTransitionCoordinator`

- Animate alongside a transition
- Get accurate completion timing

Make Timing Deterministic

Leverage `UIViewControllerTransitionCoordinator`

- Animate alongside a transition
- Get accurate completion timing
- Support interactive and cancelable animations

Auto Layout

Layout ninjas unite

Modify Constraints Efficiently

Modify Constraints Efficiently

Identify constraints that get changed, added, or removed

Modify Constraints Efficiently

Identify constraints that get changed, added, or removed

Unchanged constraints are optimized

Modify Constraints Efficiently

Identify constraints that get changed, added, or removed

Unchanged constraints are optimized

Avoid removing all constraints

Modify Constraints Efficiently

Identify constraints that get changed, added, or removed

Unchanged constraints are optimized

Avoid removing all constraints

Use explicit constraint references

Modify Constraints Efficiently

```
override func updateViewConstraints() {
    super.updateViewConstraints()

    view.removeConstraints(view.constraints())
    view.addConstraints(self.generateConstraints())
}
```

Modify Constraints Efficiently

```
override func updateViewConstraints() {
    super.updateViewConstraints()

    view.removeConstraints(view.constraints())
    view.addConstraints(self.generateConstraints())
}
```

Modify Constraints Efficiently

```
override func updateViewConstraints() {
    super.updateViewConstraints()

    view.removeConstraint(imageViewHorizontalConstraint)
    imageViewHorizontalConstraint = self.generateImageViewHorizontalConstraint()
    view.addConstraint(imageViewHorizontalConstraint)
}
```

Modify Constraints Efficiently

```
override func updateViewConstraints() {
    super.updateViewConstraints()

    view.removeConstraint(imageViewHorizontalConstraint)
    imageViewHorizontalConstraint = self.generateImageViewHorizontalConstraint()
    view.addConstraint(imageViewHorizontalConstraint)
}
```

Modify Constraints Efficiently

```
override func updateViewConstraints() {
    super.updateViewConstraints()

    view.removeConstraint(imageViewHorizontalConstraint)
    imageViewHorizontalConstraint = self.generateImageViewHorizontalConstraint()
    view.addConstraint(imageViewHorizontalConstraint)
}
```

Modify Constraints Efficiently

```
override func updateViewConstraints() {
    super.updateViewConstraints()

    view.removeConstraint(imageViewHorizontalConstraint)
    imageViewHorizontalConstraint = self.generateImageViewHorizontalConstraint()
    view.addConstraint(imageViewHorizontalConstraint)
}
```

Constraint Specificity

De-duplicating constraints

Constraint Specificity

De-duplicating constraints

Duplicates are implied by existing constraints

Constraint Specificity

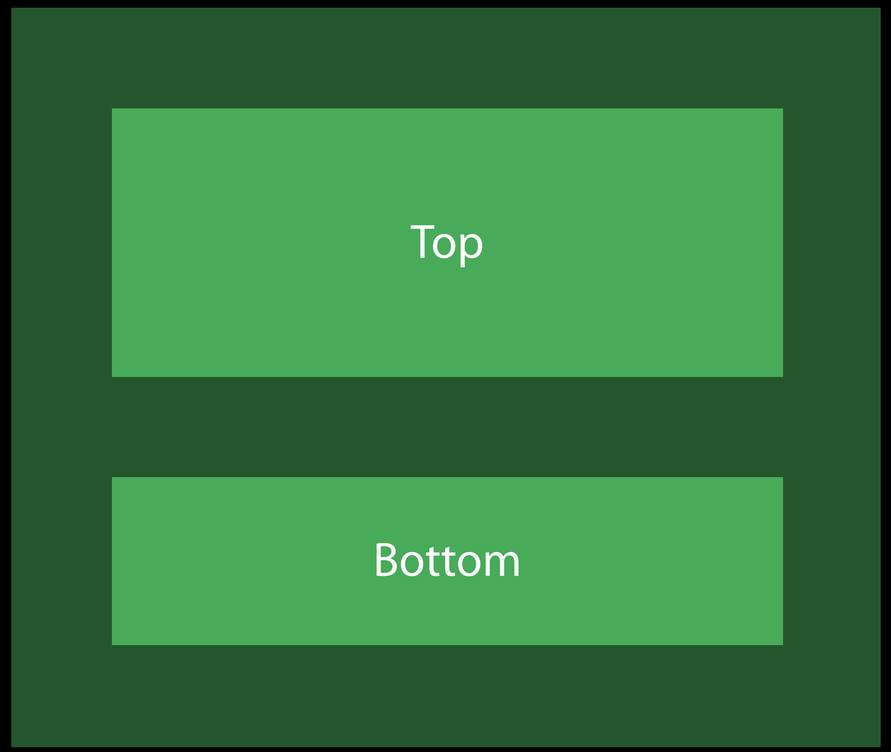
De-duplicating constraints

Duplicates are implied by existing constraints

Duplicates cause excess work in layout engine

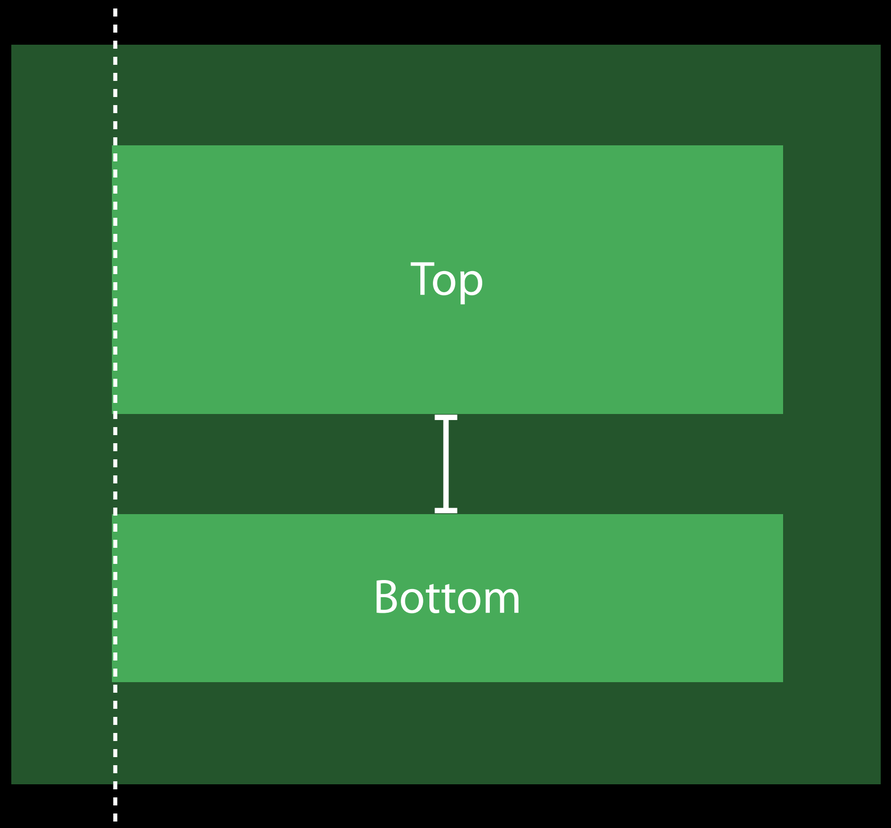
Constraint Specificity

De-duplicating constraints



Constraint Specificity

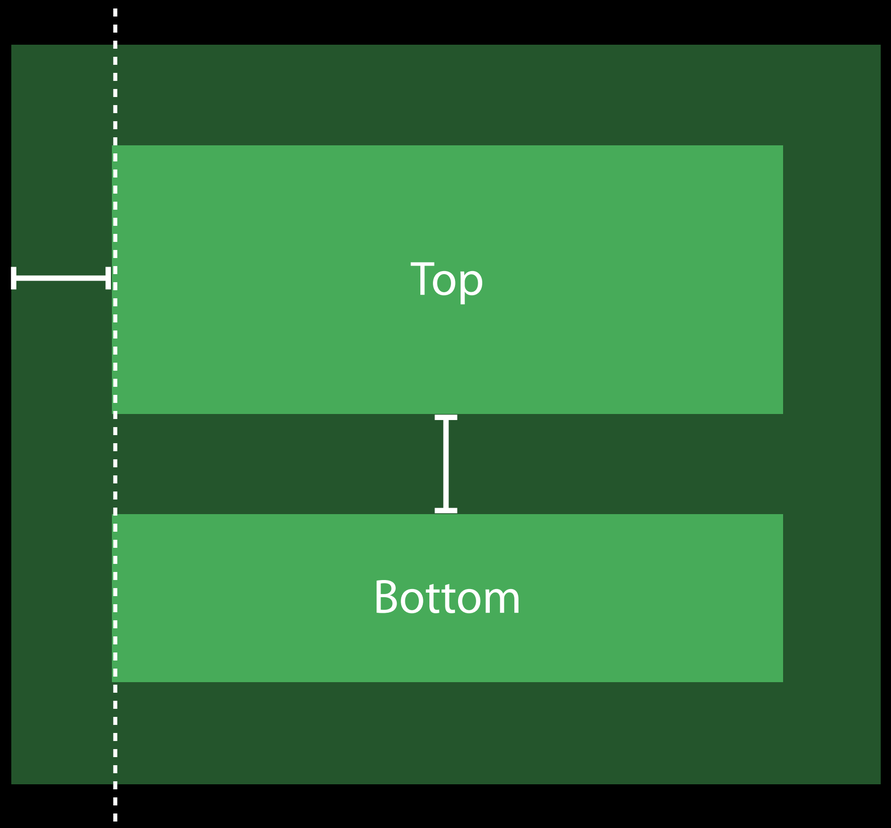
De-duplicating constraints



"V:[Top]-[Bottom]-|" + NSLayoutFormatAlignAllLeft

Constraint Specificity

De-duplicating constraints

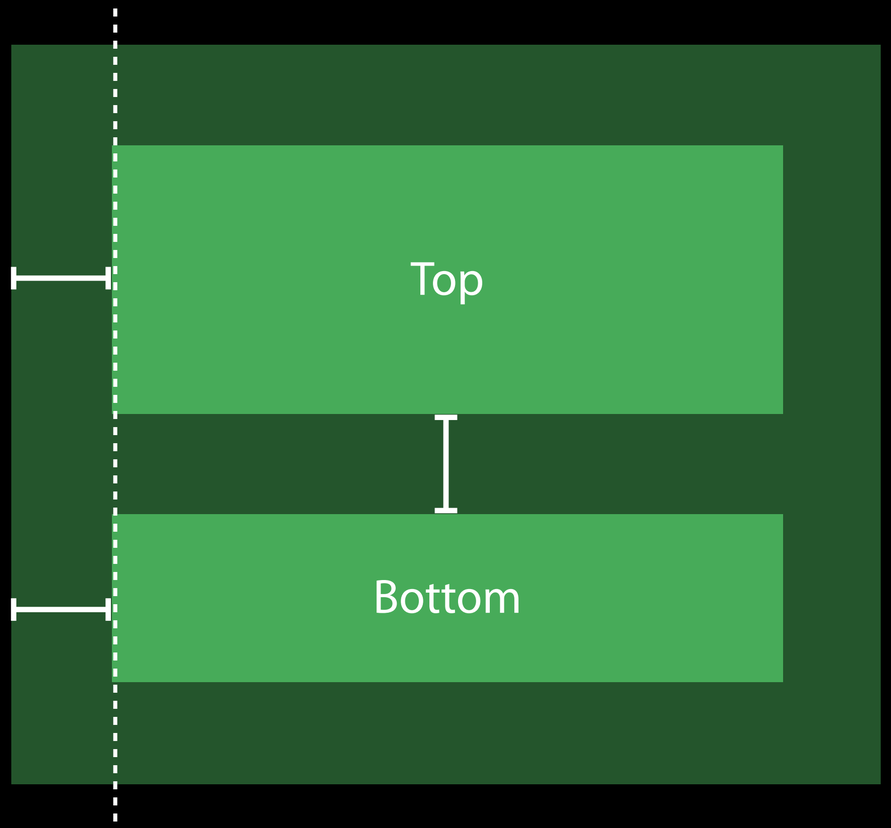


"V:[Top]-[Bottom]-|" + NSLayoutFormatAlignAllLeft

"H:|-[Top]-|"

Constraint Specificity

De-duplicating constraints



"V:[Top]-[Bottom]-|" + NSLayoutFormatAlignAllLeft

"H:|-[Top]-|"

"H:|-[Bottom]-|"

Constraint Specificity

Create flexible constraints

Constraint Specificity

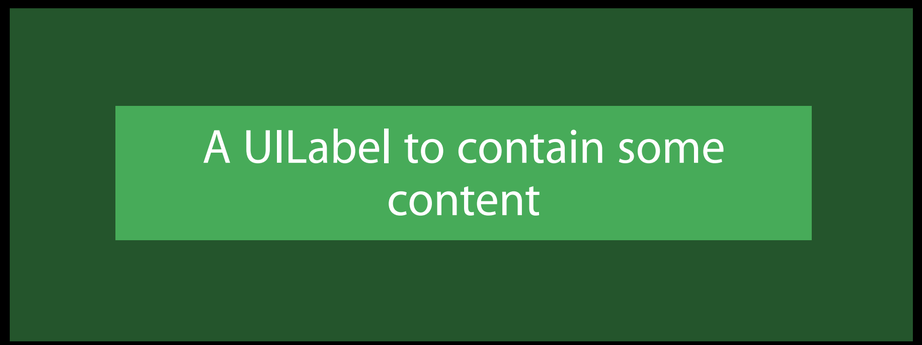
Create flexible constraints

Avoid hard-coded values

Constraint Specificity

Create flexible constraints

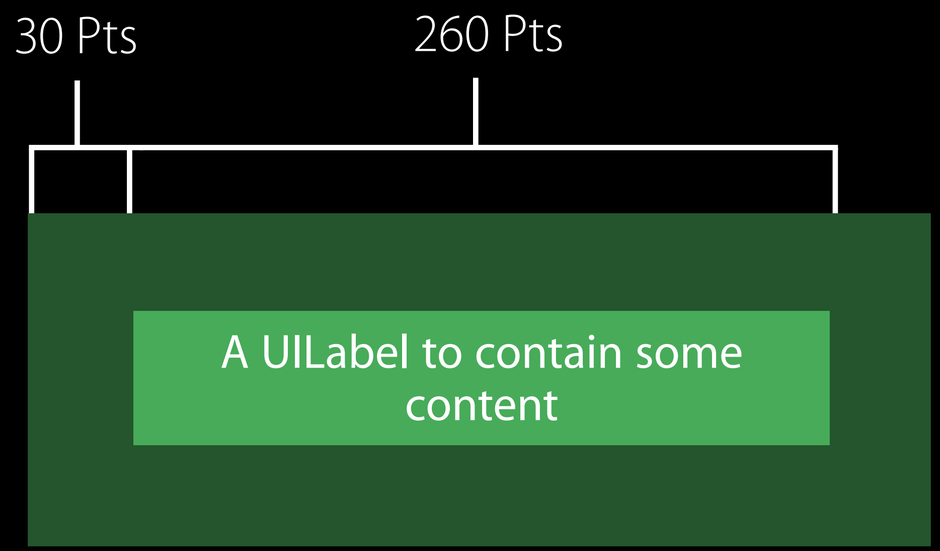
Avoid hard-coded values



Constraint Specificity

Create flexible constraints

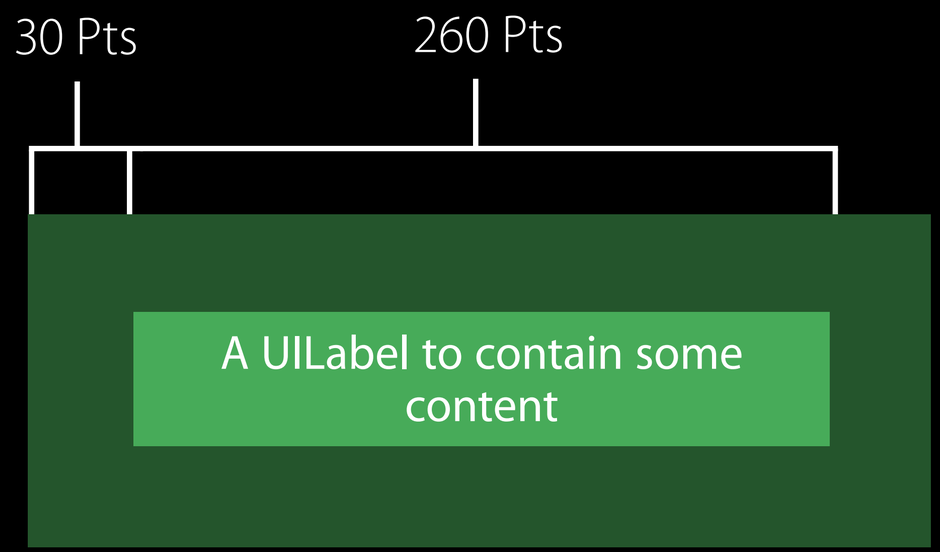
Avoid hard-coded values



Constraint Specificity

Create flexible constraints

Avoid hard-coded values



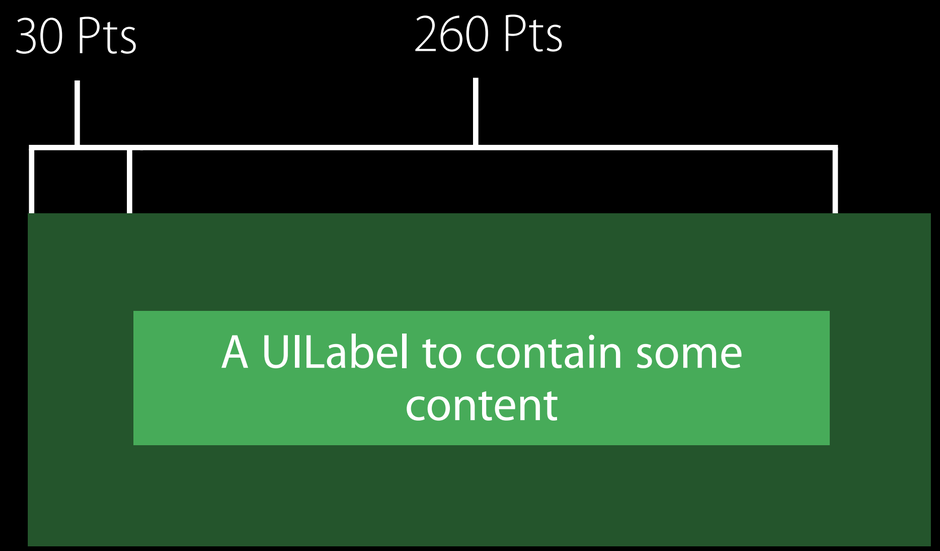
“V:-30-[Label(==260)]”

Constraint Specificity

Create flexible constraints

Avoid hard-coded values

Describe constraints using bounds

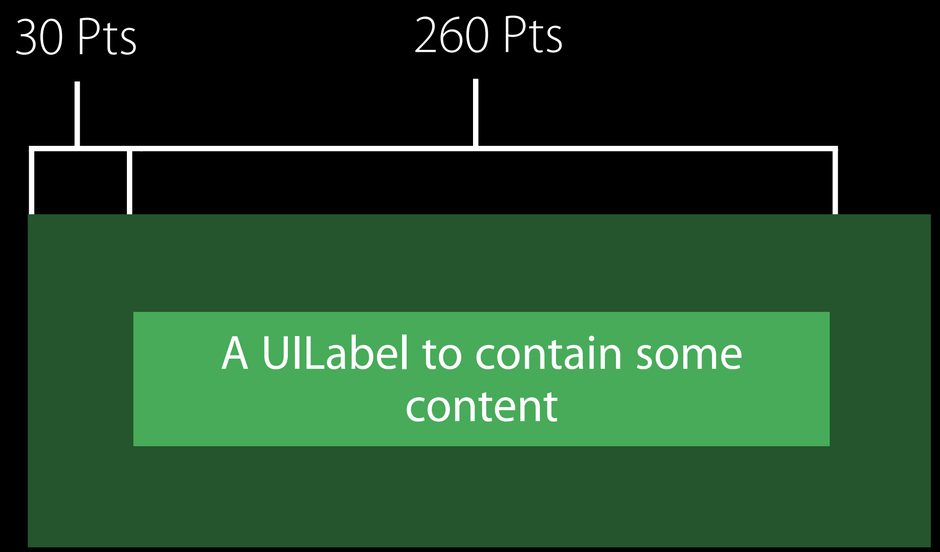


Constraint Specificity

Create flexible constraints

Avoid hard-coded values

Describe constraints using bounds



“V:[Label(<=superview - 60)]”

NSLayoutFormatAlignAllCenterX

Constraint Specificity

Fully specify constraints

Apple Expands Capital Return Program to Over $130 Billion and Announces Seven-for-One Stock Split

Constraint Specificity

Fully specify constraints

Underspecification generates ambiguity

Apple Expands Capital Return Program to Over $130 Billion and Announces Seven-for-One Stock Split

Constraint Specificity

Fully specify constraints

Underspecification generates ambiguity

Constraint Specificity

Fully specify constraints

Underspecification generates ambiguity

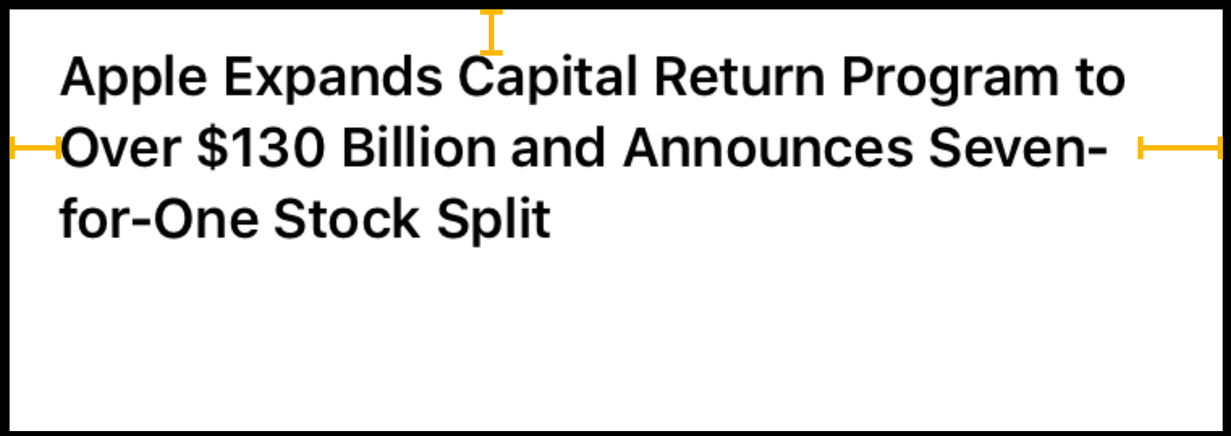
Ambiguity is undefined

Constraint Specificity

Fully specify constraints

Underspecification generates ambiguity

Ambiguity is undefined



Constraint Specificity

Testing and debugging

Constraint Specificity

Testing and debugging

```
-[UIView hasAmbiguousLayout]
```

Constraint Specificity

Testing and debugging

`-[UIView hasAmbiguousLayout]`

- When called on UIWindow, returns result for *entire* view tree

Constraint Specificity

Testing and debugging

`-[UIView hasAmbiguousLayout]`

- When called on UIWindow, returns result for *entire* view tree

`-[UIView _autolayoutTrace]`

Constraint Specificity

Testing and debugging

`-[UIView hasAmbiguousLayout]`

- When called on UIWindow, returns result for *entire* view tree

`-[UIView _autolayoutTrace]`

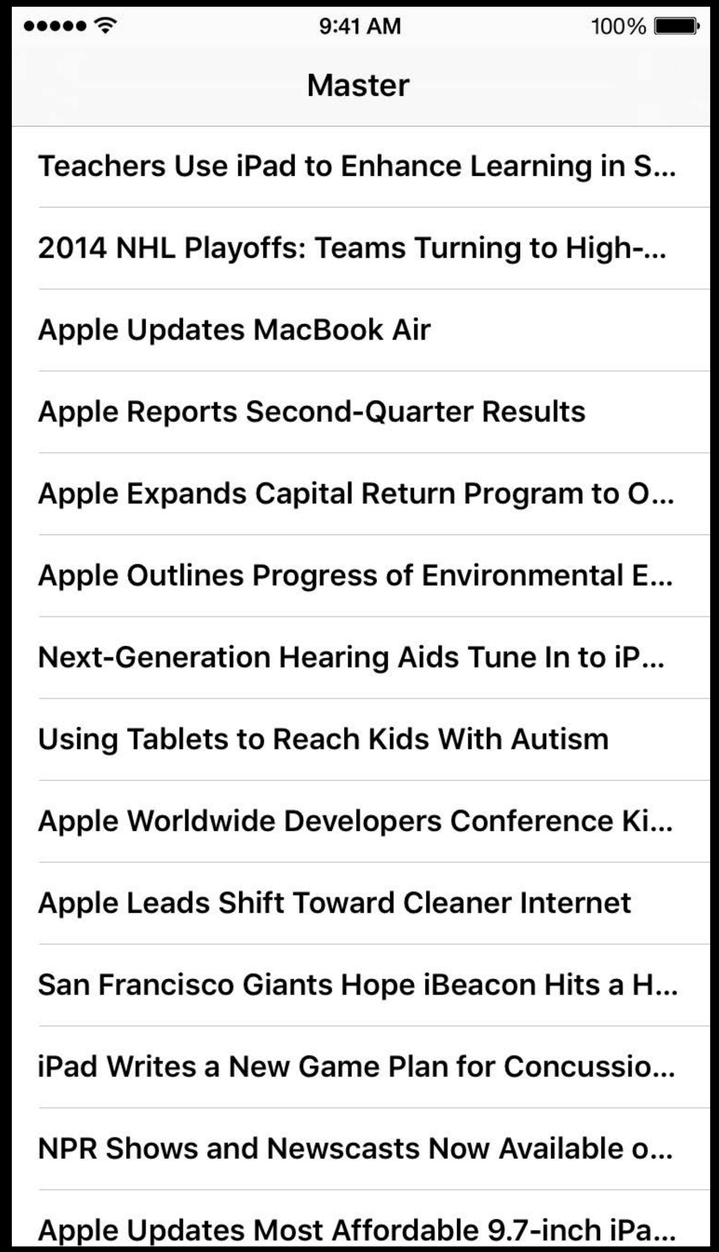
Both can be used for unit testing

Table and Collection Views

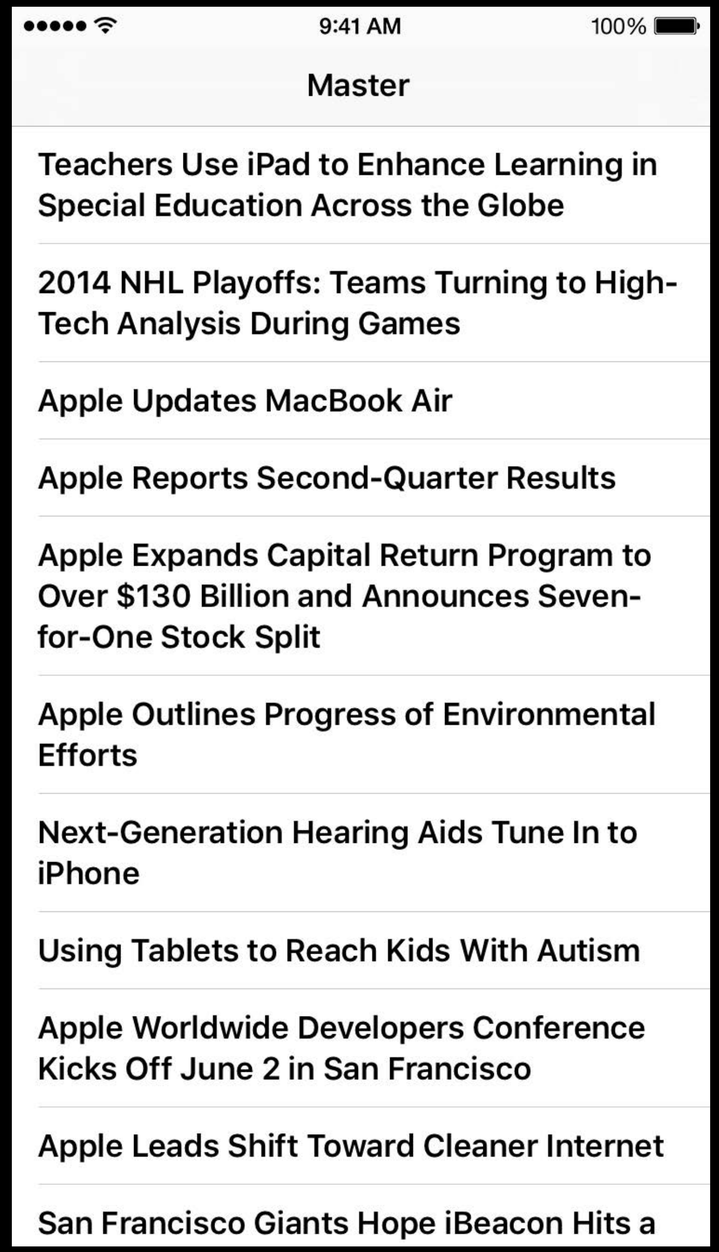
Impress your friends

Self-Sizing Cells

Self-Sizing Cells



Self-Sizing Cells



Self-Sizing Cells

Self-Sizing Cells

Fully specify constraints

Self-Sizing Cells

Fully specify constraints

Width = input; height = output

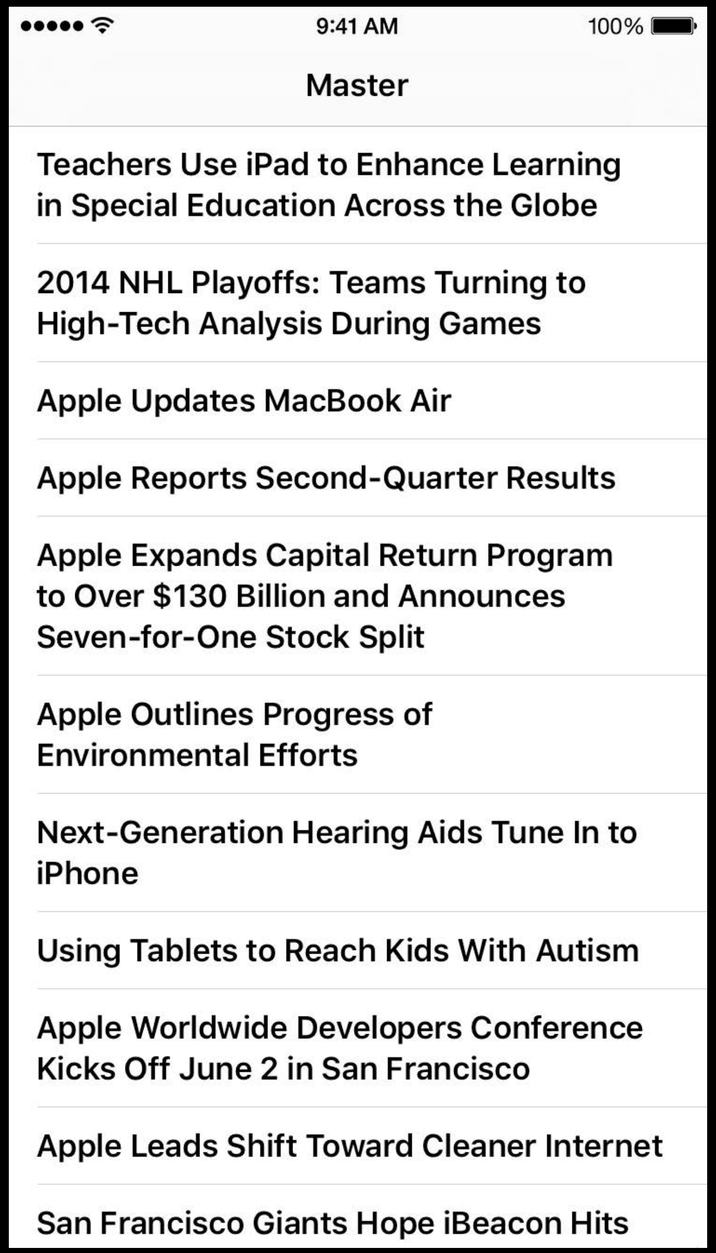
Self-Sizing Cells

Apple Expands Capital Return Program to Over $130 Billion and Announces Seven-for-One Stock Split

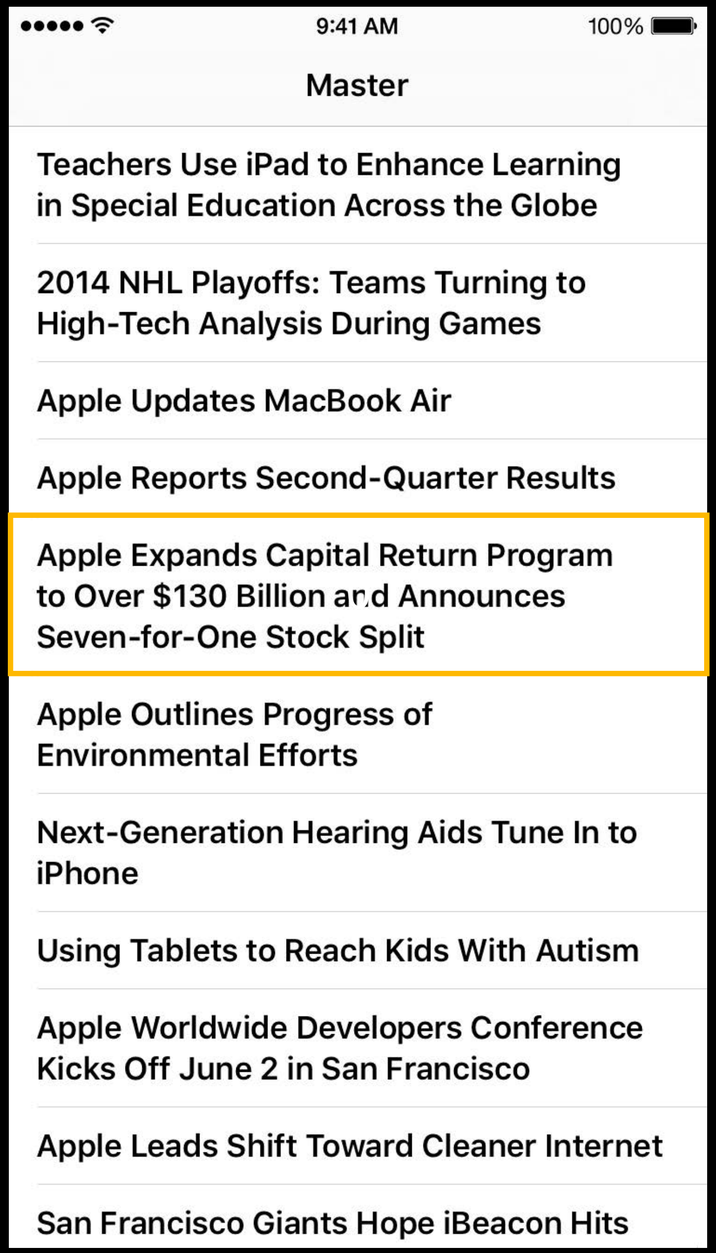
Self-Sizing Cells

Self-Sizing Cells

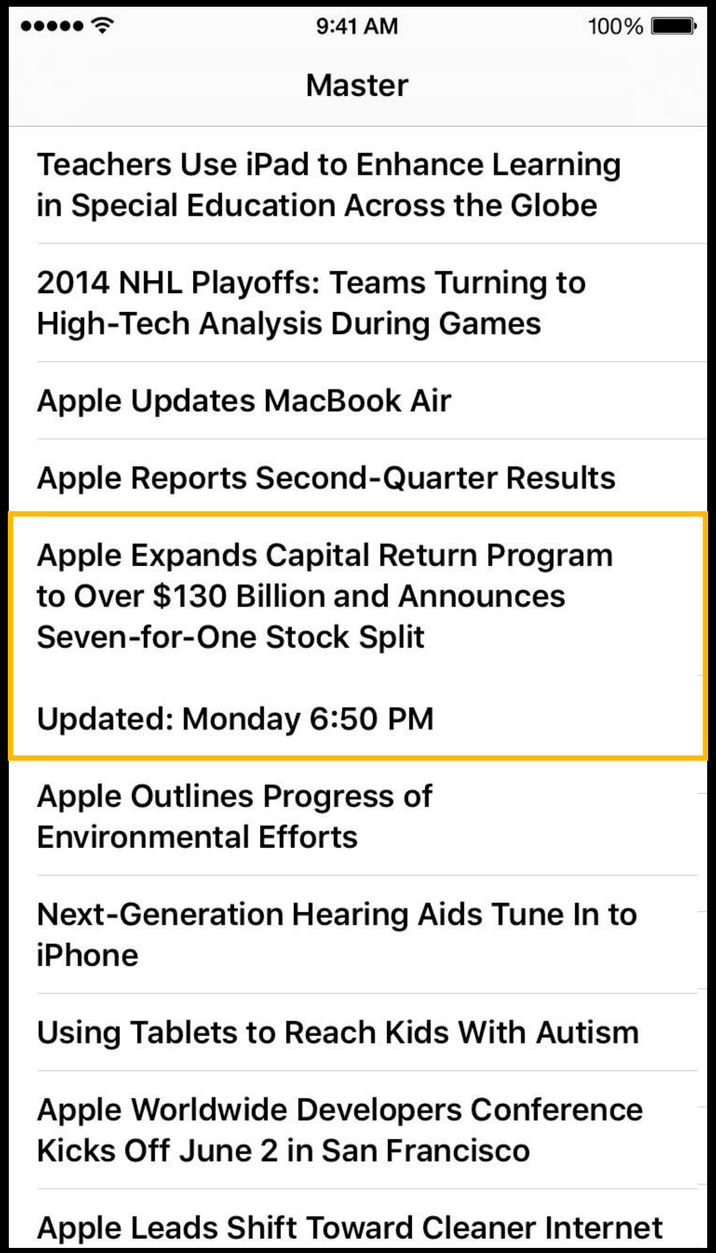
Animating Height Changes



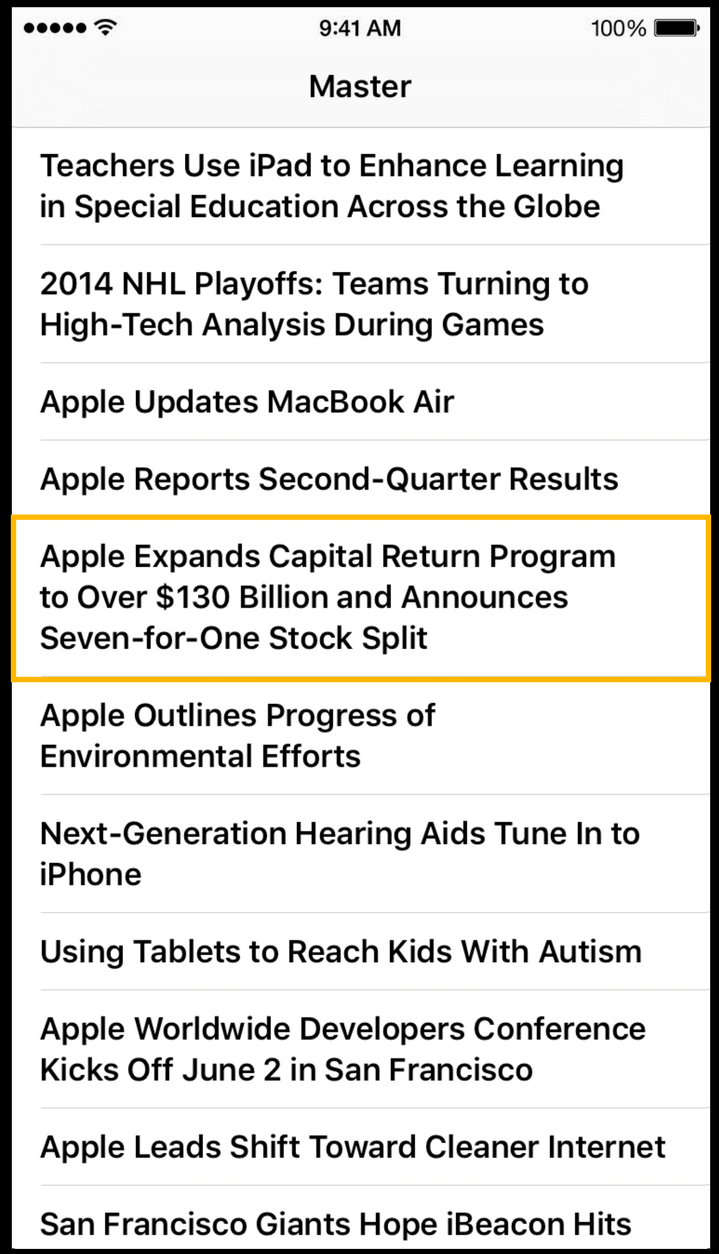
Animating Height Changes



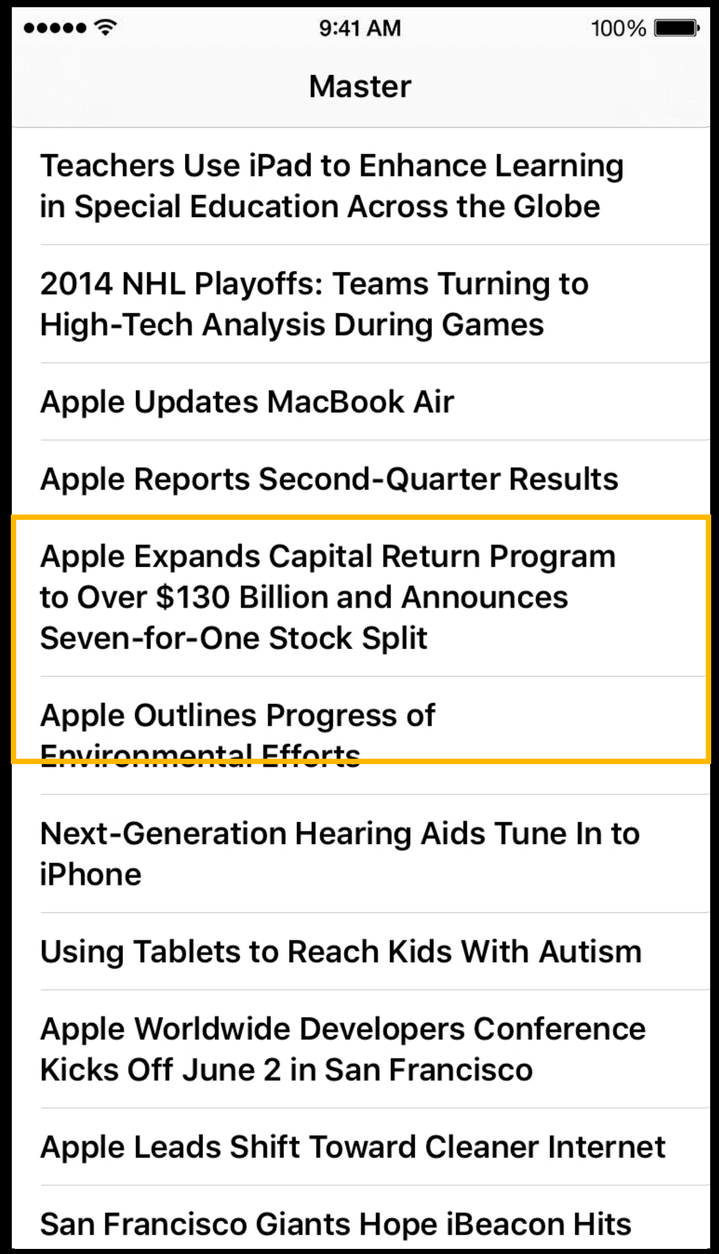
Animating Height Changes



Animating Height Changes

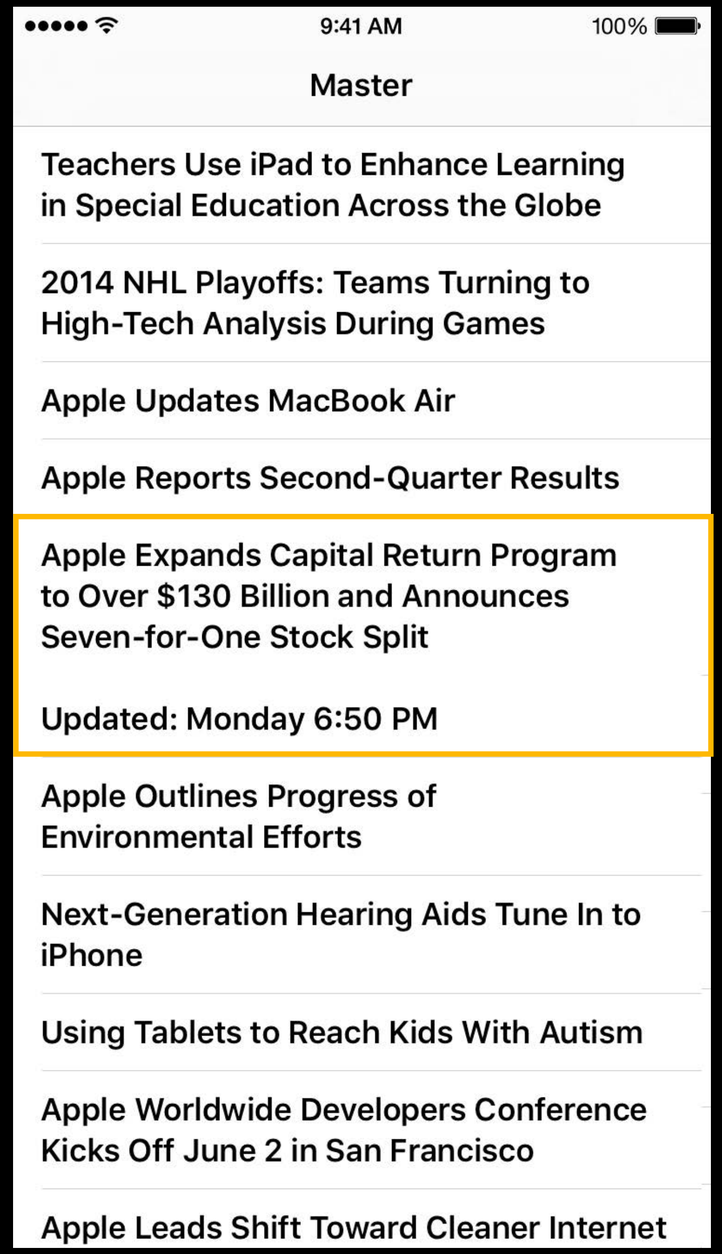


Animating Height Changes



Animating Height Changes

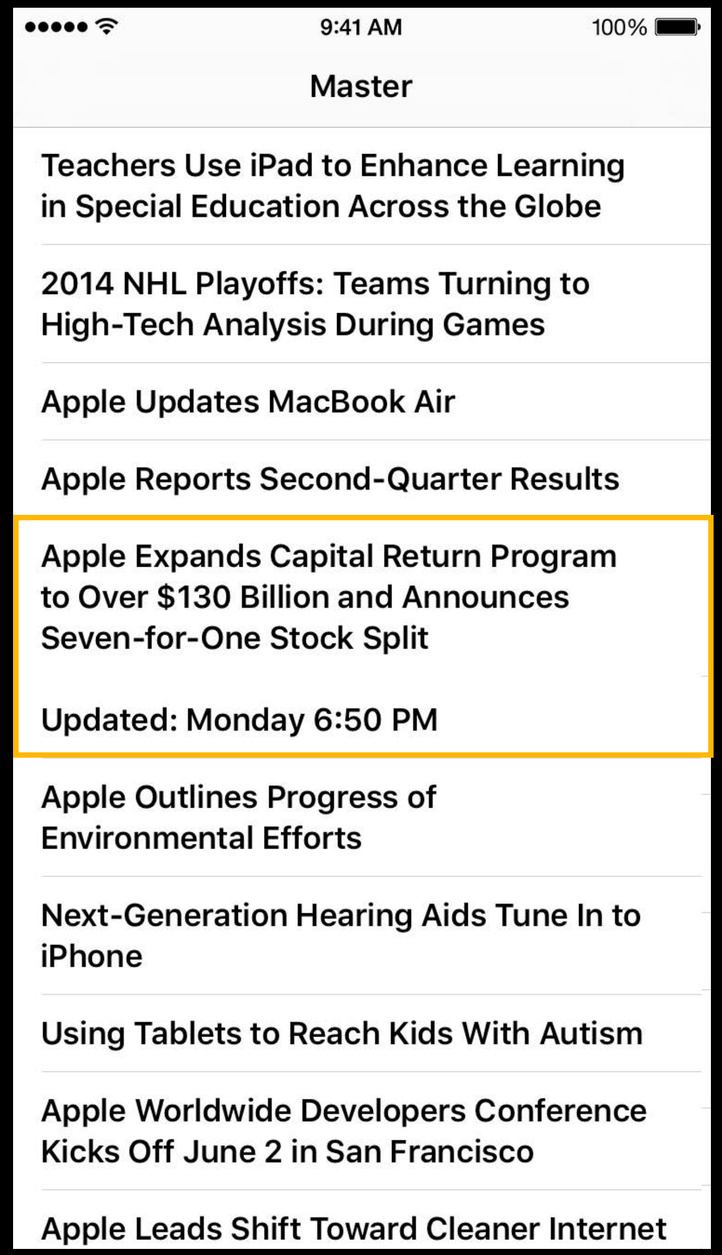
Steps



Animating Height Changes

Steps

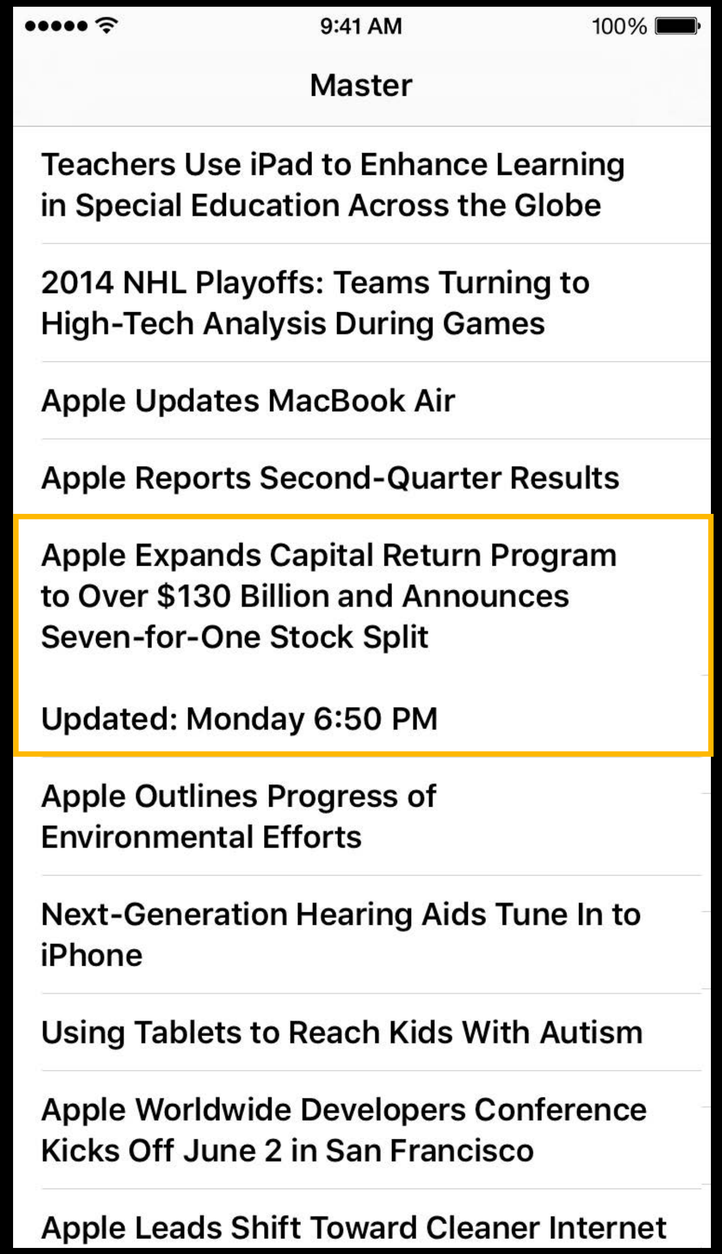
1. `tableView.beginUpdates`



Animating Height Changes

Steps

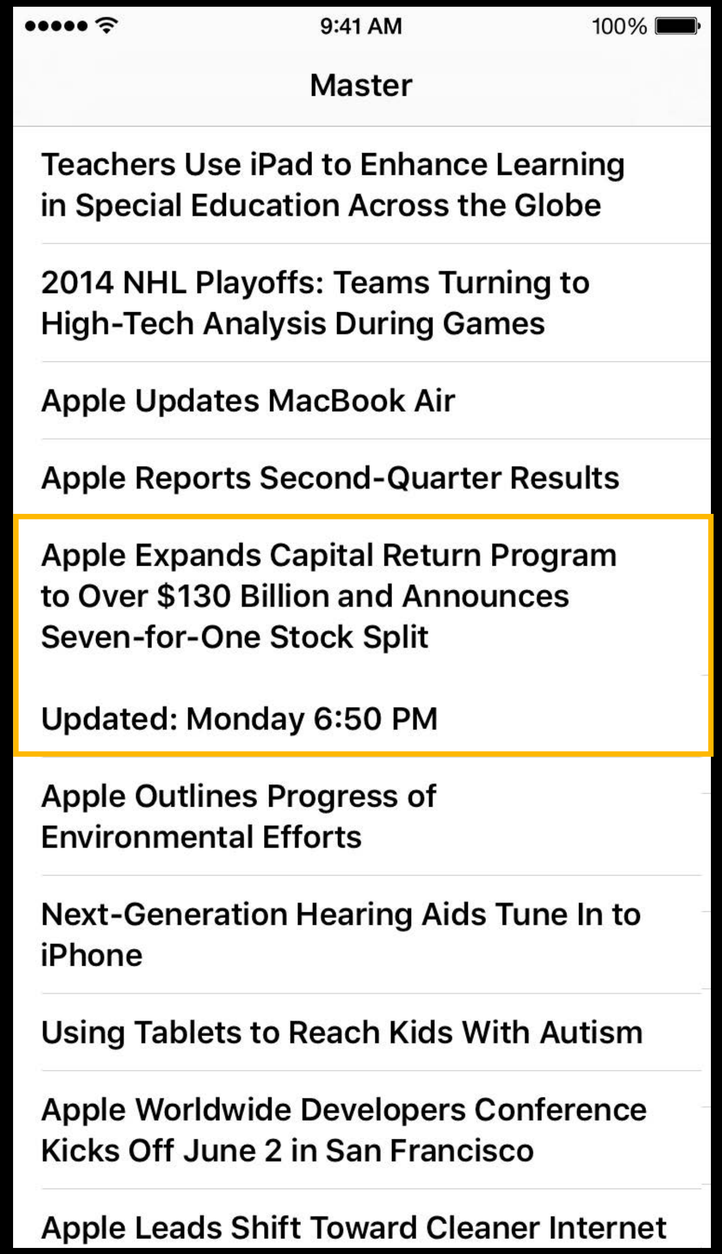
1. `tableView.beginUpdates`
2. Update model



Animating Height Changes

Steps

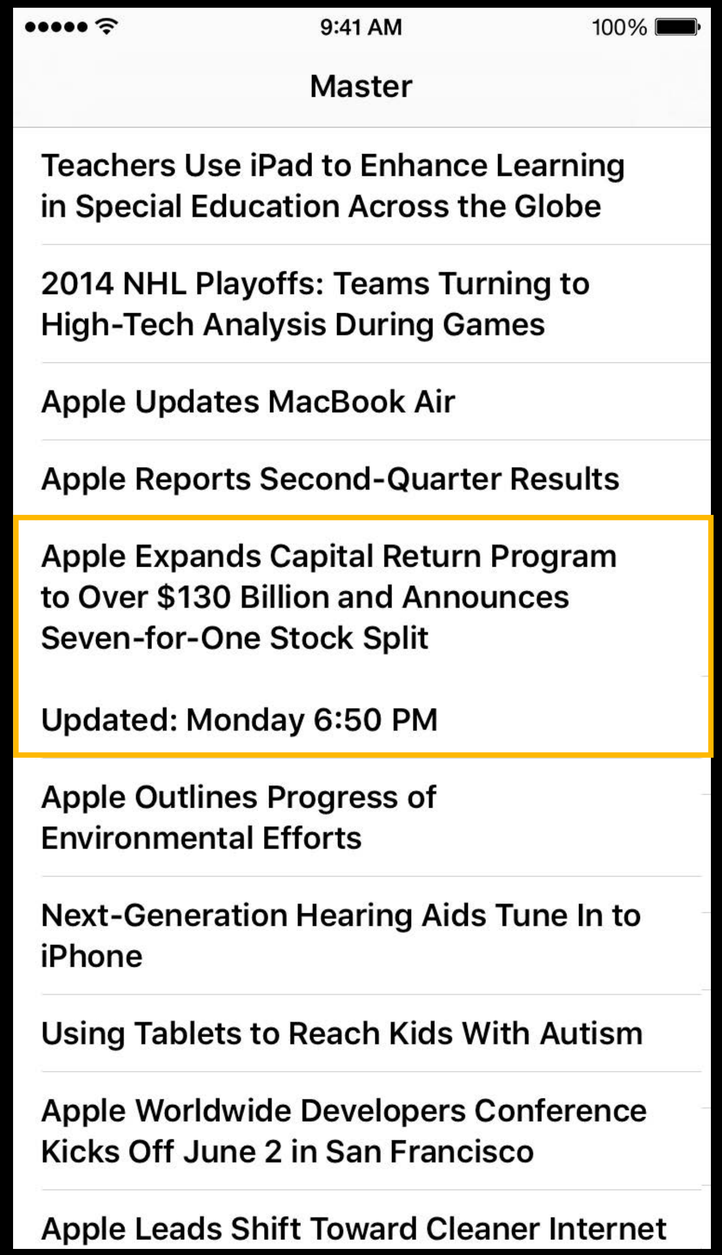
1. `tableView.beginUpdates`
2. Update model
3. Update cell contents



Animating Height Changes

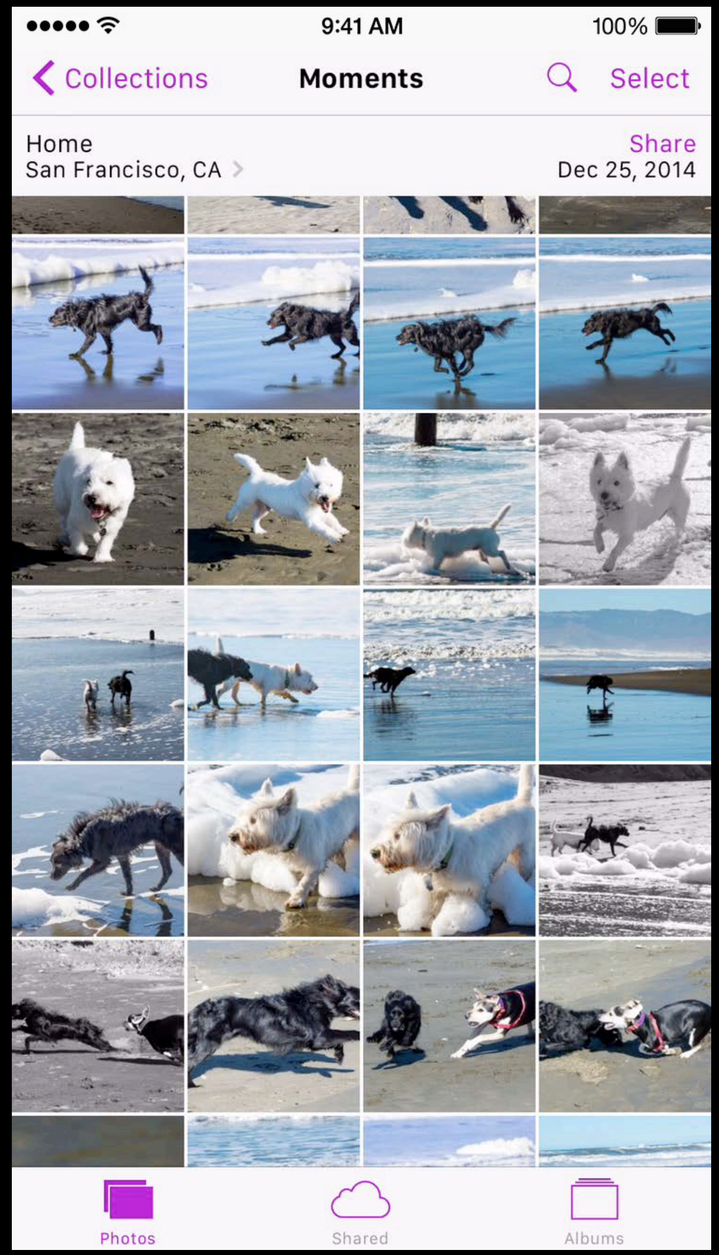
Steps

1. `tableView.beginUpdates`
2. Update model
3. Update cell contents
4. `tableView.endUpdates`

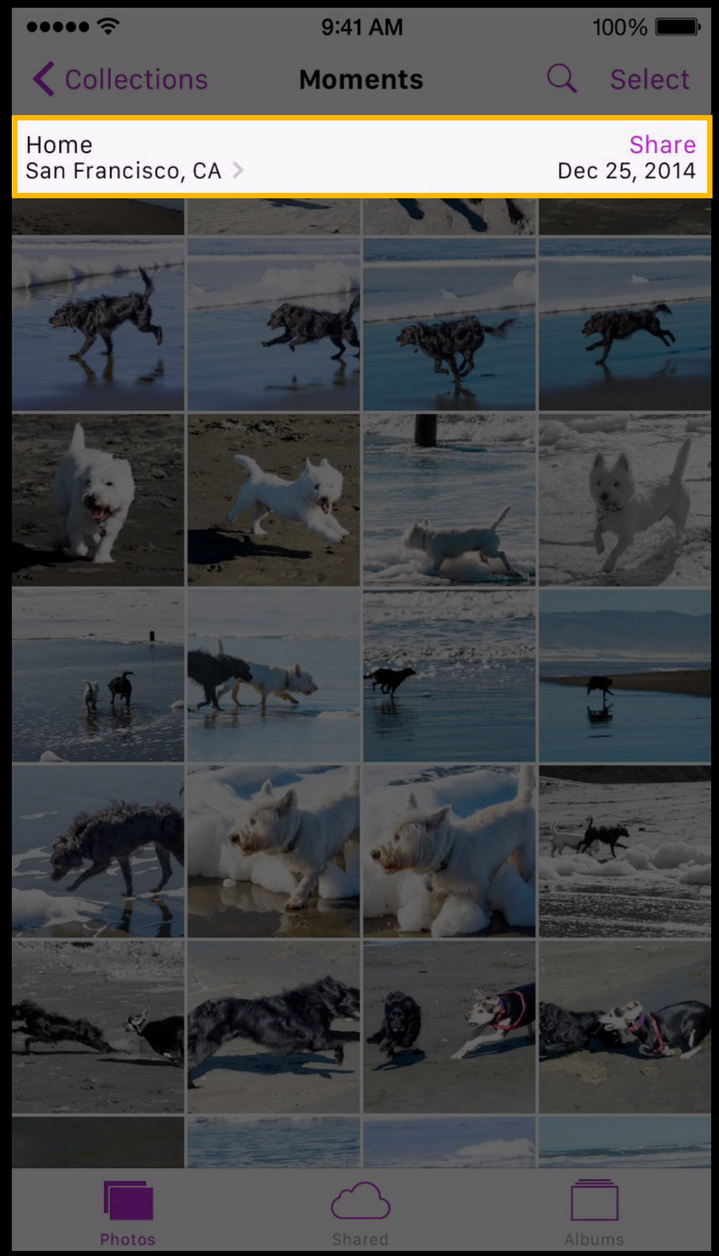


Fast Collection View Layout Invalidation

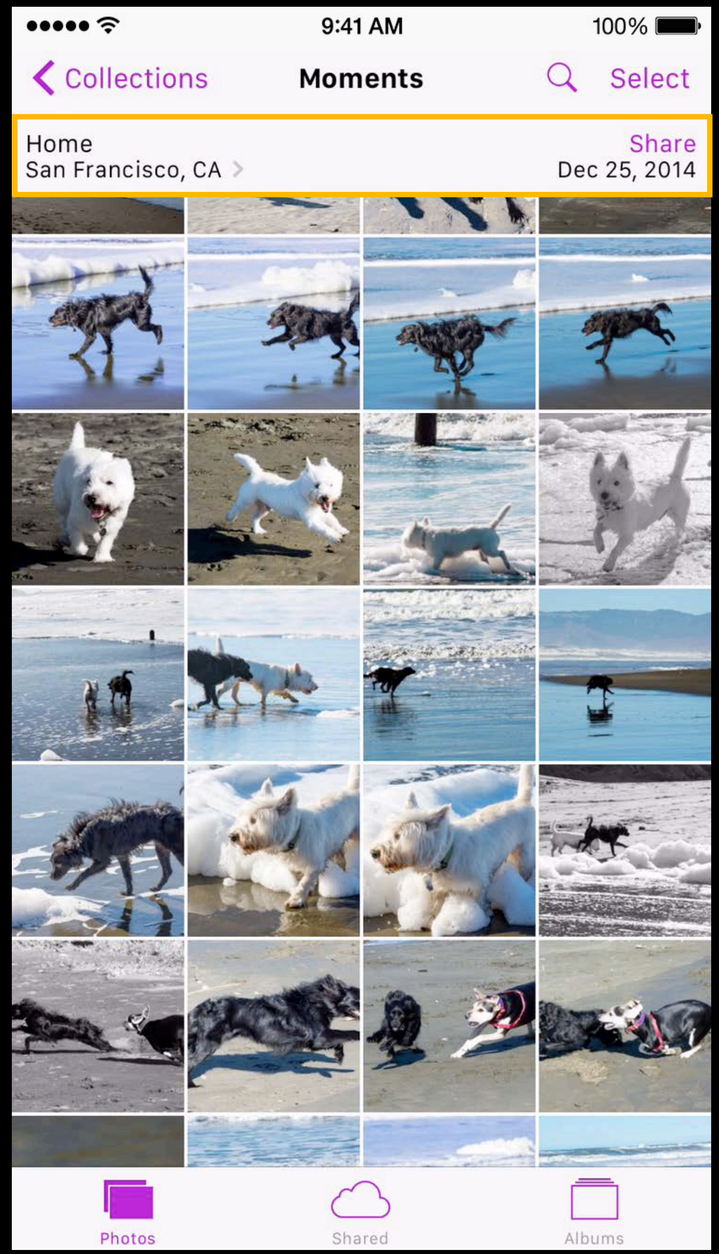
Fast Collection View Layout Invalidation



Fast Collection View Layout Invalidation

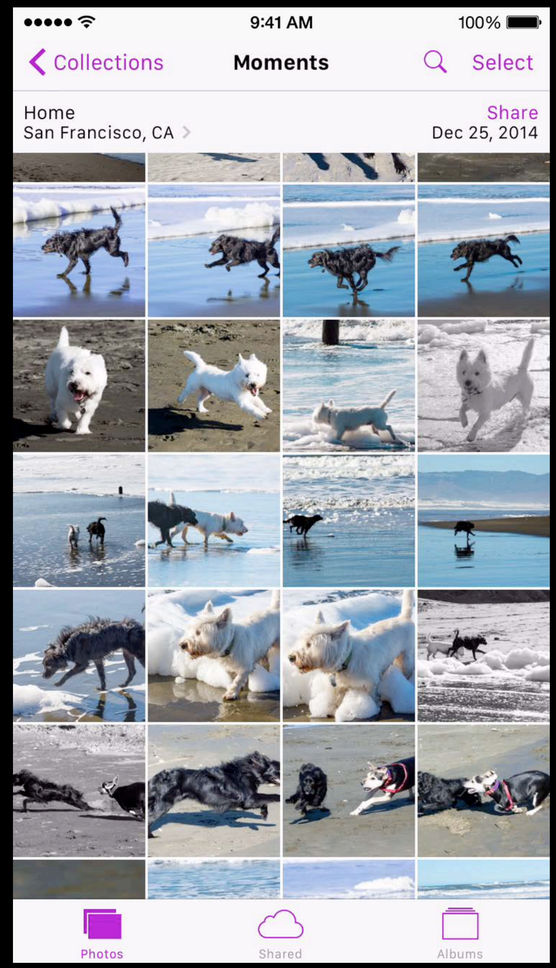


Fast Collection View Layout Invalidation



Fast Collection View Layout Invalidation

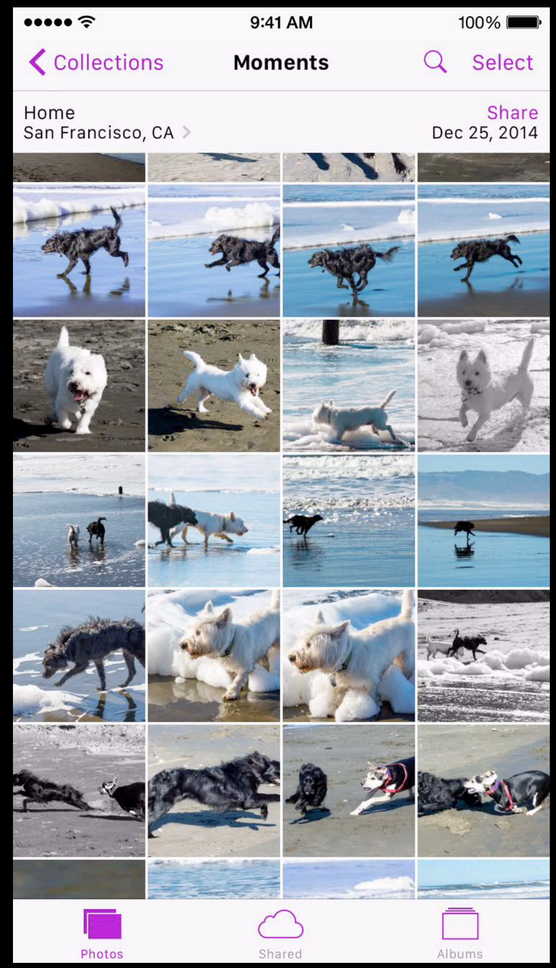
Modify only what is needed



Fast Collection View Layout Invalidation

Modify only what is needed

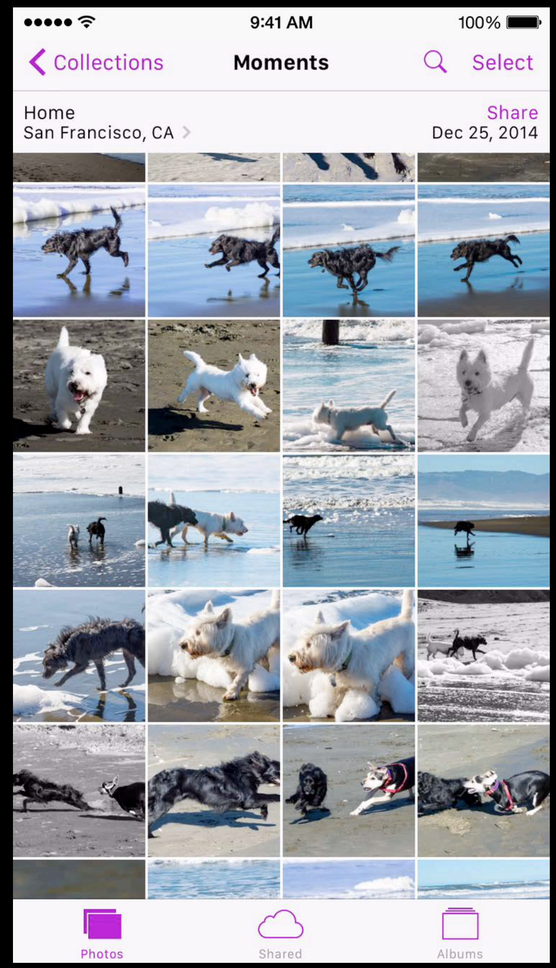
1. Invalidate on bounds change



Fast Collection View Layout Invalidation

Modify only what is needed

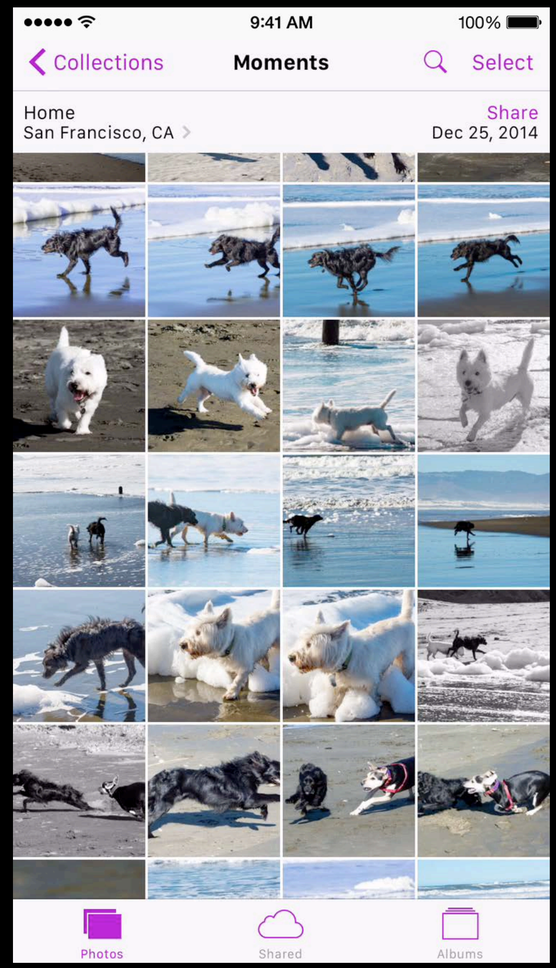
1. Invalidate on bounds change
2. Build targeted invalidation context



Fast Collection View Layout Invalidation

Modify only what is needed

1. Invalidate on bounds change
2. Build targeted invalidation context
3. Repeat as necessary



Summary

Summary

Performance

Summary

Performance

User experience

Summary

Performance

User experience

Future proofing

More Information

Documentation

What's New in iOS

Start Developing iOS Apps (Swift)

App Programming Guide for iOS

http://developer.apple.com/iOS

Technical Support

Apple Developer Forums

Developer Technical Support

Curt Rothert

App Frameworks Evangelist

rothert@apple.com

WWDC15