Improving your Existing Apps with Swift

Getting Swifty with It
Session 403

Woody L. 🐳 in the Sea of Swift
Demo
Set the Time Machine to 2012

The Elements has been restored from a vault
Agenda

The “before” app
Modernizing UI
Mix and match
Swift Structs
Availability
Live filtering
Modernizing UI

Tile Sizes

Small Tile

Detail Tile

Amercium

Atomic Weight: 243.00
State: Artificial
Period: 7
Group: 3
Discovered: 1944 A.D.
Argon

Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
Argon

Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
Pre-Rendered Backgrounds

37px

256px
Pre-Rendered Backgrounds

37px

256px
256px = 256pt @ 1x

Does Not Exist

512px = 256pt @ 2x

Does Not Exist

768px = 256pt @ 3x
Argon
Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1894 A.D.

View on Wikipedia
Argon

Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Argon

Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
Tile Appearance

Argon

Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
Tile Appearance

Argon

Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
Tile Appearance

18

Argon

Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
## Benefits of Custom Drawing Code

<table>
<thead>
<tr>
<th>Goals</th>
<th>Custom Drawing Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modernize the Look</td>
<td>✓</td>
</tr>
<tr>
<td>Crisp, Sharp Corners</td>
<td>✓</td>
</tr>
<tr>
<td>Resolution Independence</td>
<td>✓</td>
</tr>
</tbody>
</table>
Mix and Match
Extending Objective-C Classes with Swift

An interoperability overture in the key of Swift.
Class Design

Base Class

Class Definition
Class Design
Base Class + Category

Class Definition
Class Design
Base Class + Categories

Class Definition
Class Design
Base Class + Categories

Class Definition
Class Design
Base Class + Categories + Swift Extensions

Class Definition
Class Design
Mixed-Language Classes

Base

Generated Header

Bridging Header

Extension

SWIFT
Class Design

Mixed-Language Classes

Bridging Header

- Created by Xcode
- Maintained by you
- Contains `#import "MyClass.h"`
- Exists in File Navigator

Generated Header

- Created by Compiler
- Maintained by Compiler
- `#import "Product-Swift.h"` into MyClass.m
- Exists in DerivedData
Demo
Modernizing The Elements UI

Boasting 120% daily intake of vitamin Swift.
Partially Rounded Corners
Partially Rounded Corners
Swift Structs
With Playground prototyping

Global utility functions: shake ‘em off.
Argon

Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
Argon

Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
Draw Stroke
Stroke’s midpoint is the Bezier Path

Argon
Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
Argon
Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
Argon

Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
Argon

Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
Solution

Inset the Bezier Path

Argon

Atomic Weight: 39.95
State: Gas
Period: 3
Group: 18
Discovered: 1849 A.D.
Utility Functions
Core Graphics API

Struct

CGRect

Global Utility Functions

CGRectZero()
CGRectMake()
CGRectGetMaxY()
CGRectGetWidth()
CGRectUnion()
CGRectInset()
Utility Functions
Core Graphics API

CGRect
- CGRectZero()
- CGRectMake()
- CGRectGetMaxY()
- CGRectGetWidth()
- CGRectUnion()
- CGRectInset()

Struct
Methods
Utility Functions
Core Graphics API

CGRect
- zeroRect (x:, y:, w:, h:)
- maxY
- width
- union(r)
- rectByInsetting(dx:, dy:)

Struct Initializers + Methods
Utility Functions
Core Graphics API

CGRect

- .zeroRect (x:, y:, w:, h:)
- .maxY
- .width
- .union(r)
- .rectByInsetting(dx:, dy:)

Struct Initializers + Methods

Consistent
Initializer Syntax
Encapsulation
Code Completion
API Discovery
Playgrounds for Prototyping

Tweak-build loop

- Commit
- Check Work
- Navigate to UI
- Tweak Code
- Build / Compile
- Copy into Simulator
Playgrounds for Prototyping

Tweak-build loop

- Commit
- Tweak Code
- Build / Compile
- Copy into Simulator
- Navigate to UI
- Check Work

Risk of Destabilizing Known-Good Code

Slow
Playgrounds for Prototyping

The plan

Reduce roundtrip time
Iterative + experimental development

- Tweak Code
- Check Work
Demo

Struct Methods

With Playground Prototyping
Availability: the Next Generation

Put down your if respondsToSelector() and step away from Xcode.
Runtime Interrogation
The classic way

if ([UIImagePickerController instancesRespondToSelector:
    @selector(availableCaptureModesForCameraDevice:)]) {
    // Method is available for use.
} else {
    // Method is not available.
}
Availability Based on OS Version

The officially endorsed, modern way

```swift
if #available(iOS 8.3, *) {
    viewController.modalPresentationStyle = .Popover
    if let popoverPC = viewController.popoverPresentationController {
        let cell = tableView.cellForRowAtIndexPath(indexPath)
        popoverPC.sourceView = cell
        popoverPC.delegate = self
    }
    presentViewController(viewController, animated: true, completion: nil)
} else {
    // Earlier than iOS 8.3 APIs
    navigationController?.pushViewController(viewController, animated: true)
}
```
viewController.modalPresentationStyle = .Popover

if let popoverPC = viewController.popoverPresentationController {
    let cell = tableView.cellForRowAtIndexPath(indexPath)
    popoverPC.sourceView = cell
    popoverPC.delegate = self
}

```swift
viewController.modalPresentationStyle = .Popover
if let popoverPC = viewController.popoverPresentationController {
    let cell = tableView.cellForRowAtIndexPath(indexPath)
    popoverPC.sourceView = cell
    popoverPC.delegate = self
}
```
viewController.modalPresentationStyle = .Popover
if let popoverPC = viewController.popoverPresentationController {
  let cell = tableView.cellForRowAtIndexPath(indexPath)
  popoverPC.sourceView = cell
  popoverPC.delegate = self
}

error: ‘popoverPresentationController’ is only available on iOS 8.0 or newer
Demo

Availability

APIs must be this tall to ride.
Availability

New in Swift 2
Compilation-time checking of API Availability
Runtime checks inserted automatically
Live Search

Filtering elements since antiquity.
TableView

View Controller
<UITableViewDataSource, UISearchBarDelegate>

Content
Fe
Na
Cs
Ni
Cu
IRON
SODIUM
CESIUM
NICKEL
COPPER
Filter

searchBar:textDidChange:

func searchBar(searchBar: UISearchBar, textDidChange searchText: String) {
    if searchText.isEmpty {
        content = atomicElements
    } else {
        content = atomicElements.filter{$0.name.hasPrefix(searchText)}
    }
    tableView?.reload()
}
func searchBar(searchBar: UISearchBar, textDidChange searchText: String) {
    if searchText.isEmpty {
        content = atomicElements
    } else {
        content = atomicElements.filter{!$0.name.hasPrefix(searchText)}
    }
    tableView?.reload()
func searchBar(searchBar: UISearchBar, textDidChange searchText: String) {
    if searchText.isEmpty == true {
        content = atomicElements
    } else {
        content = atomicElements.filter{$0.name.hasPrefix(searchText)}
    } tableView?.reload()
}
Filter

searchBar::textDidChange:

```swift
func searchBar(searchBar: UISearchBar, textDidChange searchText: String) {
    if searchText.isEmpty == true {
        content = atomicElements
    } else {
        content = atomicElements.filter{$0.name.hasPrefix(searchText)}
    }
    tableView?.reload()
}
```

Original Array: Fe IRON, Au GOLD, Cs CESIUM, Ni NICKEL, Ne NEON, Cu COPPER

Closure: `{ _ .name.hasPrefix("N") }`

Filtered Array:
Demo
Filtering

Only the worthy may pass.
Sums of Atomic Weights

Introducing Map & Reduce, in harmony like Ebony & Ivory.
Sum of Atomic Weights
### Sum of Atomic Weights

Two or more rows selected

<table>
<thead>
<tr>
<th>Element</th>
<th>Atomic Weight</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co</td>
<td>58.9334</td>
<td>Co</td>
</tr>
<tr>
<td>Cr</td>
<td>52.0050</td>
<td>Cr</td>
</tr>
<tr>
<td>Cs</td>
<td>132.9055</td>
<td>Cs</td>
</tr>
<tr>
<td>Cu</td>
<td>63.5460</td>
<td>Cu</td>
</tr>
<tr>
<td>Db</td>
<td>243.0505</td>
<td>Db</td>
</tr>
<tr>
<td>Dy</td>
<td>162.9310</td>
<td>Dy</td>
</tr>
<tr>
<td>Er</td>
<td>167.2590</td>
<td>Er</td>
</tr>
<tr>
<td>Es</td>
<td>253.0400</td>
<td>Es</td>
</tr>
<tr>
<td>Eu</td>
<td>151.9640</td>
<td>Eu</td>
</tr>
<tr>
<td>F</td>
<td>19.0030</td>
<td>F</td>
</tr>
<tr>
<td>Fe</td>
<td>55.8470</td>
<td>Fe</td>
</tr>
</tbody>
</table>

**Total:** 649.008µ
Intermediate Objective

Array of selected atomic elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Symbol</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co</td>
<td>Co</td>
<td>27</td>
</tr>
<tr>
<td>Cr</td>
<td>Cr</td>
<td>24</td>
</tr>
<tr>
<td>Cs</td>
<td>Cs</td>
<td>55</td>
</tr>
<tr>
<td>Cu</td>
<td>Cu</td>
<td>29</td>
</tr>
<tr>
<td>Db</td>
<td>Db</td>
<td>105</td>
</tr>
<tr>
<td>Dy</td>
<td>Dy</td>
<td>66</td>
</tr>
<tr>
<td>Er</td>
<td>Er</td>
<td>68</td>
</tr>
<tr>
<td>Es</td>
<td>Es</td>
<td>99</td>
</tr>
<tr>
<td>Eu</td>
<td>Eu</td>
<td>63</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>9</td>
</tr>
<tr>
<td>Fe</td>
<td>Fe</td>
<td>26</td>
</tr>
</tbody>
</table>

content

649.008μ

[Element Table]
Intermediate Objective

Array of selected atomic elements

content

selectedAtomicElements
Intermediate Objective

Array of selected atomic elements

content

selectedAtomicElements
Intermediate Objective

Array of selected atomic elements
var selectedElements = [AtomicElement]()
if let indexPaths = tv?.indexPathsForSelectedRows {
    for ip in indexPaths {
        let currentElement = content[ip.row]
        selectedElements.append(currentElement)
    }
}
Swiftier Way

Map

```swift
var selectedElements = [AtomicElement]()
if let indexPaths = tv?.indexPathsForSelectedRows {
    for ip in indexPaths {
        let currentElement = content[ip.row]
        selectedElements.append(currentElement)
    }
}

let selectedElements = tv?.indexPathsForSelectedRows?.map{content[$0.row]}
```
Adding Values of Items
Traditional way

selectedAtomicElements

Cu 29
Dy 66
Es 99
Eu 63
F 9

.atomicWeight
Adding Values of Items

Traditional way

Using a for-in loop

```swift
var d = 0.0
for element in selectedAtomicElements {
    d = d + element.atomicWeight
}
```
Sum of Atomic Element Weights

Swift’s reduce() method

Using a for-in loop
```swift
var d = 0.0
for element in selectedAtomicElements {
    d = d + element.atomicWeight
}
```

With Reduce()
```swift
reduce(0.0, combine: {$0 + $1.atomicWeight})
```
tableView?.indexPathsForSelectedRows?.map{self.content[$0.row]}.reduce(0.0, combine: { $0 + $1.atomicWeight })
Demo

Map & Reduce

Obtaining closure through closures.
Summary

You have existing Objective-C code? Keep it.
You're adding new code? Consider writing it in Swift.
More Information

Stefan Lesser
Swift Evangelist
slesser@apple.com

Swift Language Documentation
http://developer.apple.com/swift

Apple Developer Forums
developer.apple.com/forums
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date &amp; Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s new in Swift</td>
<td>Presidio</td>
<td>Tuesday 11:10AM</td>
</tr>
<tr>
<td>Swift and Objective-C Interoperability</td>
<td>Mission</td>
<td>Tuesday 1:30PM</td>
</tr>
<tr>
<td>Protocol Oriented Programming in Swift</td>
<td>Marina</td>
<td>Wednesday 2:30PM</td>
</tr>
<tr>
<td>Optimizing Performance in Swift</td>
<td>Presidio</td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td>Swift in Practice</td>
<td>Presidio</td>
<td>Thursday 2:30PM</td>
</tr>
<tr>
<td>Building Better UIKit Apps with Swift</td>
<td>Mission</td>
<td>Wednesday 2:00PM</td>
</tr>
</tbody>
</table>