



# dotSwift 2019

Adam Gask & James Froggatt

# Conference Information

<https://www.dotswift.io>

# Conference Information

- The European Swift Conference
- 28 January 2019
- Paris
- Théâtre de Paris





# Théâtre de Paris

# Conference Information

- Hosted by [DotConferences](#) (5 Years Running)
  - dotGo (The European Go Conference)
  - dotCSS (The largest CSS Conference in Europe)
  - dotJS (The largest JavaScript Conference in Europe)
  - dotSwift (The European Swift Conference)
  - dotAI (The Artificial Intelligence & Machine Learning Conference for Developers)
  - dotScale (The European Tech Conference on Scalability)
  - dotSecurity (The Security Conference for Developers)

# Conference Information

- Format
  - 8 Talks
  - 5 Lightning Talks
  - $\frac{3}{4}$  Day event
  - Two optional days of courses/workshops





Lots of Coffee and Cake

# Key Speakers

# Key Speakers



Daniel Jalkut  
[Just Enough Objective-C](#)  
(Combining Obj-C & Swift)



Saroush Khanlou  
[From Problem To Solution](#)  
(Code Reuse)



Jeff Biggus  
[Scientific Swift](#)  
(Custom Operators &  
Unicode)



Mark Dalrymple  
[Wh@t ls Th@t \(Attributes\)](#)

# Key Speakers



Johannes Weiss (Apple Swift NIO)  
[High-Performance Systems In Swift](#) (Structs vs Classes)



Lea Marolt Sonnenschein  
[Making Table Views Great Again!](#) (Reusable patterns)



James Dempsey (former Apple)  
Scripting In Swift (It's a tradeoff)



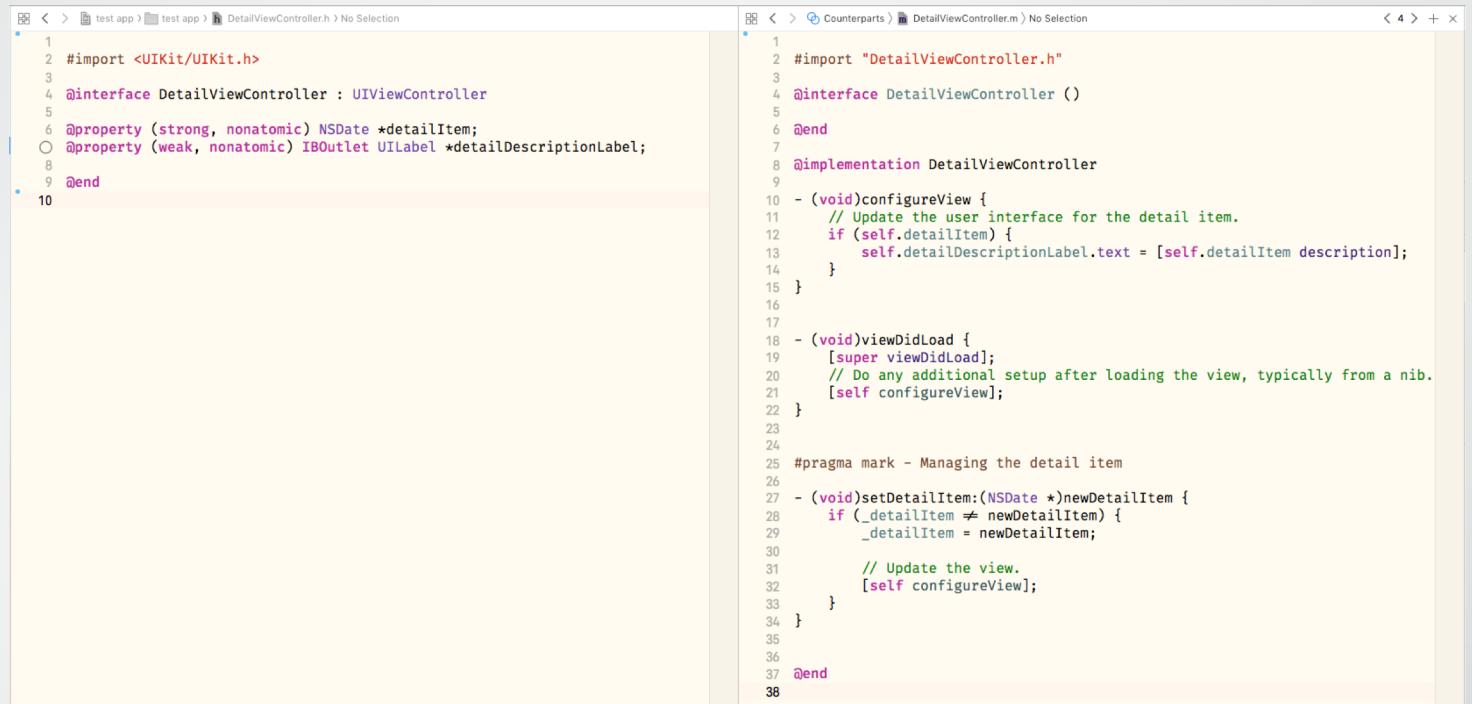
Tom Doron (Apple Swift NIO)  
[Implementing JSON-RPC with SwiftNIO](#)

# Just Enough Objective-C

Using the Objective-C runtime from Swift, private APIs

Daniel Jalkut, <https://www.dotconferences.com/2019/01/daniel-jalkut-just-enough-objective-c>

# Objective-C Runtime



```
test app > DetailViewController.h No Selection
1 #import <UIKit/UIKit.h>
2
3 @interface DetailViewController : UIViewController
4
5 @property (strong, nonatomic) NSDate *detailItem;
6 @property (weak, nonatomic) IBOutlet UILabel *detailDescriptionLabel;
7
8 @end
9
10

Counterparts > DetailViewController.m No Selection
1 #import "DetailViewController.h"
2
3 @interface DetailViewController () {
4
5 @end
6
7
8 @implementation DetailViewController
9
10 - (void)configureView {
11     // Update the user interface for the detail item.
12     if (self.detailItem) {
13         self.detailDescriptionLabel.text = [self.detailItem description];
14     }
15 }
16
17 - (void)viewDidLoad {
18     [super viewDidLoad];
19     // Do any additional setup after loading the view, typically from a nib.
20     [self configureView];
21 }
22
23
24 #pragma mark - Managing the detail item
25
26 - (void)setDetailItem:(NSDate *)newDetailItem {
27     if (_detailItem != newDetailItem) {
28         _detailItem = newDetailItem;
29
30         // Update the view.
31         [self configureView];
32     }
33 }
34
35
36
37 @end
38
```

- Objective-C defines types as a set of properties and methods.
- Methods are declared with headers
  - Undeclared methods can still be accessed publicly – no API is truly ‘private’.
- Internally, methods calls are string messages passed between objects.

# objc\_msgSend

```
[self configureView];
```

↓ Compiles to...

## Summary

Sends a message with a simple return value to an instance of a class.

## Declaration

```
id objc_msgSend(id self, SEL op, ...);
```

## Discussion

When it encounters a method call, the compiler generates a call to one of the functions `objc_msgSend`, `objc_msgSend_stret`, `objc_msgSendSuper`, or `objc_msgSendSuper_stret`. Messages sent to an object's superclass (using the `super` keyword) are sent using `objc_msgSendSuper`; other messages are sent using `objc_msgSend`. Methods that have data structures as return values are sent using `objc_msgSendSuper_stret` and `objc_msgSend_stret`.

# Key-Value Observation

## Objective-C

```
[object addObserver:self  
    forKeyPath:@"propertyName"  
    options:(NSKeyValueObservingOptionInitial & NSKeyValueObservingOptionNew)  
    context:NULL];
```

```
- (void)observeValueForKeyPath:(NSString *)keyPath  
    ofObject:(id)object  
    change:(NSDictionary<NSKeyValueChangeKey,id> *)change  
    context:(void *)context {  
    if ([keyPath isEqualToString:@"propertyName"]) {  
        // Do something  
    }  
}
```

## Swift

```
object.addObserver(self,  
    forKeyPath: #keyPath(NSObject.propertyName),  
    options: [.initial, .new],  
    context: nil)
```

Allows safe syntax for Obj-C  
keypath strings

```
override func observeValue(forKeyPath keyPath: String?,  
    of object: Any?,  
    change: [NSKeyValueChangeKey : Any]?,  
    context: UnsafeMutableRawPointer?) {  
    if keyPath == "propertyName" {  
        // Do something  
    }  
}
```

Classic unsafe raw strings work too

# Private APIs

```
let selector = Selector("recursiveDescription")
let text = view.perform(selector)?.takeRetainedValue() as? NSString
```

```
W+H; layer = <CALayer: 0x129e48720>
| <_UIBackdropView: 0x129e86f60; frame = (0 0; 1112 834); opaque = NO;
userInteractionEnabled = NO; layer = <_UIBackdropViewLayer: 0x129e2bcc0>>
| | <_UIBackdropEffectView: 0x129e53e20; frame = (0 0; 1112 834); opaque = NO;
autoresizing = W+H; userInteractionEnabled = NO; layer = <CABackdropLayer:
0x129e54010>>
| | | <UIView: 0x129e64610; frame = (0 0; 1112 834); hidden = YES; opaque = NO;
autoresizing = W+H; userInteractionEnabled = NO; layer = <CALayer: 0x129e3a870>>
| | | <_UIActionSlider: 0x129e70600; frame = (0 60; 1112 75); alpha = 0; opaque = NO;
tintColor = UIExtendedSRGBColorSpace 1 0 0 1; gestureRecognizers = <NSArray:
0x129e8ba40>; layer = <CALayer: 0x129e26760>>
| | | | <UIView: 0x129e676a0; frame = (0 0; 1112 75); layer = <CALayer:
0x129e70900>>
| | | | | <UIView: 0x129e3aa30; frame = (423 0; 266 75); layer = <CALayer:
0x129e43130>>
| | | | | <_UIActionSliderTrackComponentView: 0x129e872f0; frame = (0 0; 266
75); layer = <CALayer: 0x129e3ad10>>
| | | | | <_UIActionSliderKnob: 0x129e8abd0; frame = (426.5 5; 65 65); opaque = NO;
layer = <CALayer: 0x129e8ab00>>
| | | | | | <UIImageView: 0x129e8b320; frame = (21.5 20; 21.5 23.5); opaque = NO;
userInteractionEnabled = NO; layer = <CALayer: 0x129e8b550>>
| | | | | <_UIGlintyStringView: 0x129d8ce30; frame = (509 20.5; 156 32); text = 'slide to
power off'; opaque = NO; layer = <CALayer: 0x129d8ae00>>
| | | | | <UIView: 0x129e93840; frame = (0 0; 156 32); alpha = 0; opaque = NO;
animations = { opacity=<CABasicAnimation: 0x129e91c20> }; layer = <CALayer:
0x129e93d60>>
| | | | | <_UIGlintyShapeView: 0x129e97490; frame = (0 0; 156 32); transform = [1,
0, 0, -1, 0, 0]; opaque = NO; layer = <CAShapeLayer: 0x129e91510>>
| | | | | <UIView: 0x129e97b10; frame = (0 0; 156 32); clipsToBounds = YES;
opaque = NO; layer = <CALayer: 0x129e97990>>
| | | | | <UIImageView: 0x129e97cf0; frame = (-3 -1.86; 646 38); alpha = 0.9;
opaque = NO; userInteractionEnabled = NO; layer = <CALayer: 0x129e97a30>>
| | | | | <_UIGlintyGradientView: 0x129e97f20; frame = (-3 -1.86; 646 38);
alpha = 0.35; layer = <CAGradientLayer: 0x129e98100>>
| | | | | <UIImageView: 0x129e982d0; frame = (-3 -1.86; 646 38); alpha = 0.75;
opaque = NO; userInteractionEnabled = NO; layer = <CALayer: 0x129e98500>>
| | | | | <UIView: 0x129e98760; frame = (0 0; 156 32); layer = <CALayer:
0x129e982a0>>
| | | | | <UIImageView: 0x129e99500; frame = (0 0; 156 32); opaque = NO;
userInteractionEnabled = NO; layer = <CALayer: 0x129e99730>>
| | <UIButton: 0x129d8c1a0; frame = (520 702; 72 72); alpha = 0; opaque = NO; layer =
<CALayer: 0x129d8b870>>
| | | <UIImageView: 0x129d998b0; frame = (0 0; 72 72); clipsToBounds = YES;
opaque = NO; userInteractionEnabled = NO; layer = <CALayer: 0x129d99380>>
| | | <UILabel: 0x129e8dd80; frame = (535 781.5; 42 16); text = 'Cancel'; alpha = 0;
userInteractionEnabled = NO; layer = <_UILabelLayer: 0x129e8e330>>
| | | <SBUIShapeView: 0x129d84090; frame = (0 0; 1112 834); alpha = 0; layer =
<CAShapeLayer: 0x129d842f0>>
| | | <UIView: 0x129d69780; frame = (0 0; 1112 834); alpha = 0; layer = <CALayer:
0x129d83740>>
```

# Wh@t Is Th@T (Attributes)

Swift open-source, going behind the scenes, find-ing

Mark Dalrymple, <https://www.dotconferences.com/2019/01/mark-dalrymple-what-is-that>

# Swift Open-Source

- The compiler:
  - Written in C++, hosted on Github
  - Hosted at <https://github.com/apple/swift>

The screenshot shows the GitHub repository page for `apple / swift`. The page includes the following details:

- Repository name: `apple / swift`
- Watchers: 2,577
- Stars: 47,270
- Forks: 7,485
- Pull requests: 525
- Insights
- Code tab (highlighted)
- The Swift Programming Language <https://swift.org>
- Statistics: 85,065 commits, 188 branches, 1,119 releases, 679 contributors, Apache-2.0 license
- Branch: master
- New pull request button
- Create new file, Upload files, Find File, Clone or download buttons
- A list of recent pull requests and commits, including:

  - xedin Merge pull request #23436 from xedin/keypath-dynamic-lookup ... Latest commit 16b6501 9 hours ago
  - .github Reduce boilerplate in the GitHub PR template 3 years ago
  - apinotes [oslog] [stdlib-private] Add a prototype of the new os\_log swift APIs 22 days ago
  - benchmark [ownership] Enable ownership verification by default. 11 days ago
  - bindings/xml [Markup] Print Tags in documentation comment XML 2 years ago
  - cmake [android] Allow BlocksRuntimeStub to compile for Android 3 days ago
  - docs Update WindowsBuild.md 5 days ago
  - include Merge pull request #23436 from xedin/keypath-dynamic-lookup 9 hours ago
  - lib Merge pull request #23436 from xedin/keypath-dynamic-lookup 9 hours ago
  - stdlib Revert SE-0246 (#23800) 16 hours ago
  - test Merge pull request #23436 from xedin/keypath-dynamic-lookup 9 hours ago
  - tools Merge pull request #23633 from alexshap/make\_reflection\_work\_on\_linux... 2 days ago
  - unitests build: use clang/clang-cl to build SwiftSyntaxParser 9 days ago

# Find-ing things

- `find . -type f -exec grep SomethingInteresting {} \; -print`
- With a local clone of Swift compiler, this command lets you search the file contents from the command line, to find points of interest in the Swift compiler.
- One such example is the file identifying each Swift attribute as a keyword:
  - <https://github.com/apple/swift/blob/master/include/swift/AST/Attr.def>

Access control keywords:

```
DECL_ATTR(private, AccessControl,
          OnFunc | OnAccessor | OnExtension | OnGenericType | OnVar | OnSubscript |
          OnConstructor |
          DeclModifier |
          NotSerialized, 46)
DECL_ATTR_ALIAS(fileprivate, AccessControl)
DECL_ATTR_ALIAS(internal, AccessControl)
DECL_ATTR_ALIAS(public, AccessControl)
CONTEXTUAL_DECL_ATTR_ALIAS(open, AccessControl)
```

# Swift Community

- Swift.org:
  - Central location for learning about the Swift Project
  - Home of the Swift Blog, API Design Guidelines, and many useful links
  - Hosted at <https://swift.org/>
- Swift Forums:
  - For evolution, development, community projects, and user questions
  - Gathering place for the Swift community
    - Not just Apple developers
  - Hosted at <https://forums.swift.org>

The screenshot shows a forum interface with a dark header. The header includes navigation links: 'all categories', 'all tags', 'Categories', 'Latest' (which is highlighted in orange), 'New (2)', 'Unread (2)', and 'Top'. Below the header, there's a section titled 'Topic' with two entries. Each entry has a small thumbnail, the topic name, a 'Discussion' tag, and a timestamp. The first entry is 'Variadic generics discussion' with 7 replies, 1.6k views, and posted 5 hours ago. The second entry is 'Vector manifesto' with 10 replies, 1.1k views, and posted 12 hours ago.

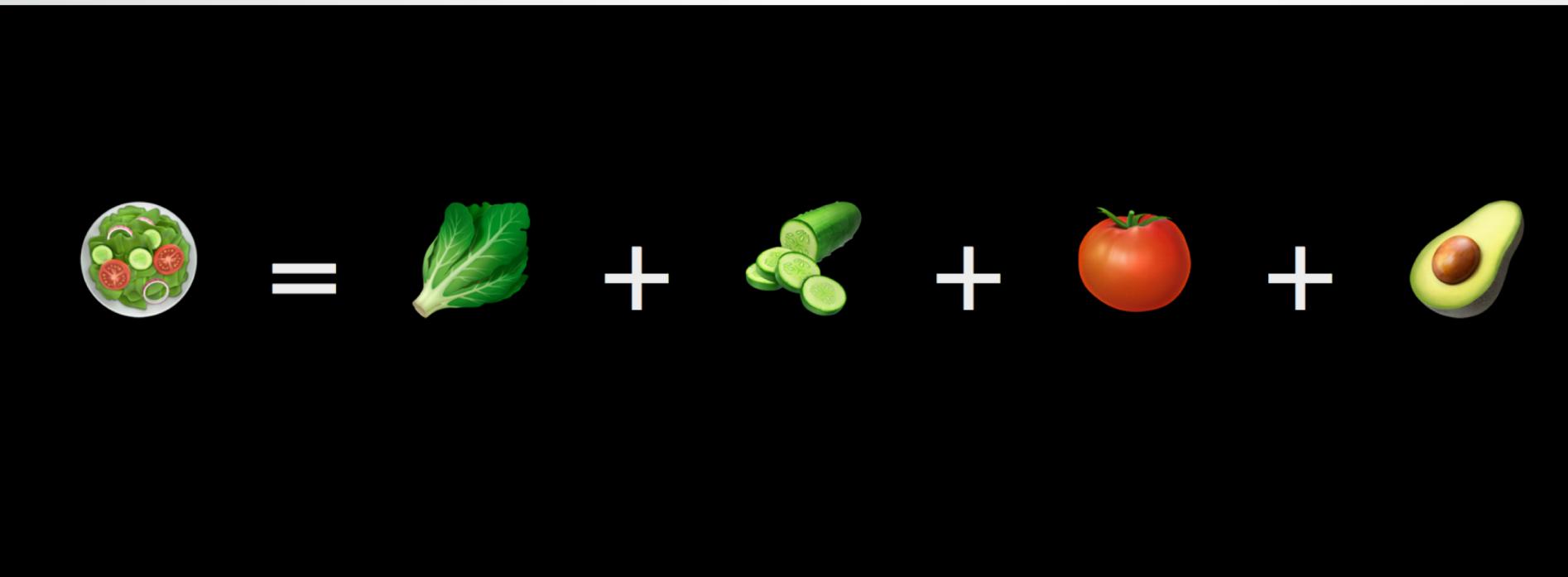
The screenshot shows a GitHub repository page for 'apple/swift-evolution'. The top bar shows the repository name, a 'Watch' button (1,300), a 'Star' button (9,626), a 'Fork' button (1,537), and a 'Code' tab. Below the top bar, there's a navigation bar with 'Branch: master', 'swift-evolution / proposals /', and buttons for 'Create new file', 'Upload files', 'Find file', and 'History'. The main area displays a list of pull requests. The first pull request is by 'rjmccall' titled 'Accept SE-0241', with a note 'Latest commit c8e3dc4 3 days ago'. Below it is a list of commits, each with a file name, a brief description, and a timestamp. The commits are: '0001-keywords-as-argument-lab...', '0002-remove-currying.md', '0003-remove-var-parameters.md', '0004-remove-pre-post-inc-decre...', '0005-objective-c-name-translati...', '0006-apply-api-guidelines-to-the...', and '0007-remove-c-style-for-loops.md'. The timestamps for these commits range from '5 months ago' to '3 years ago'.

# Unicode support & defining operators

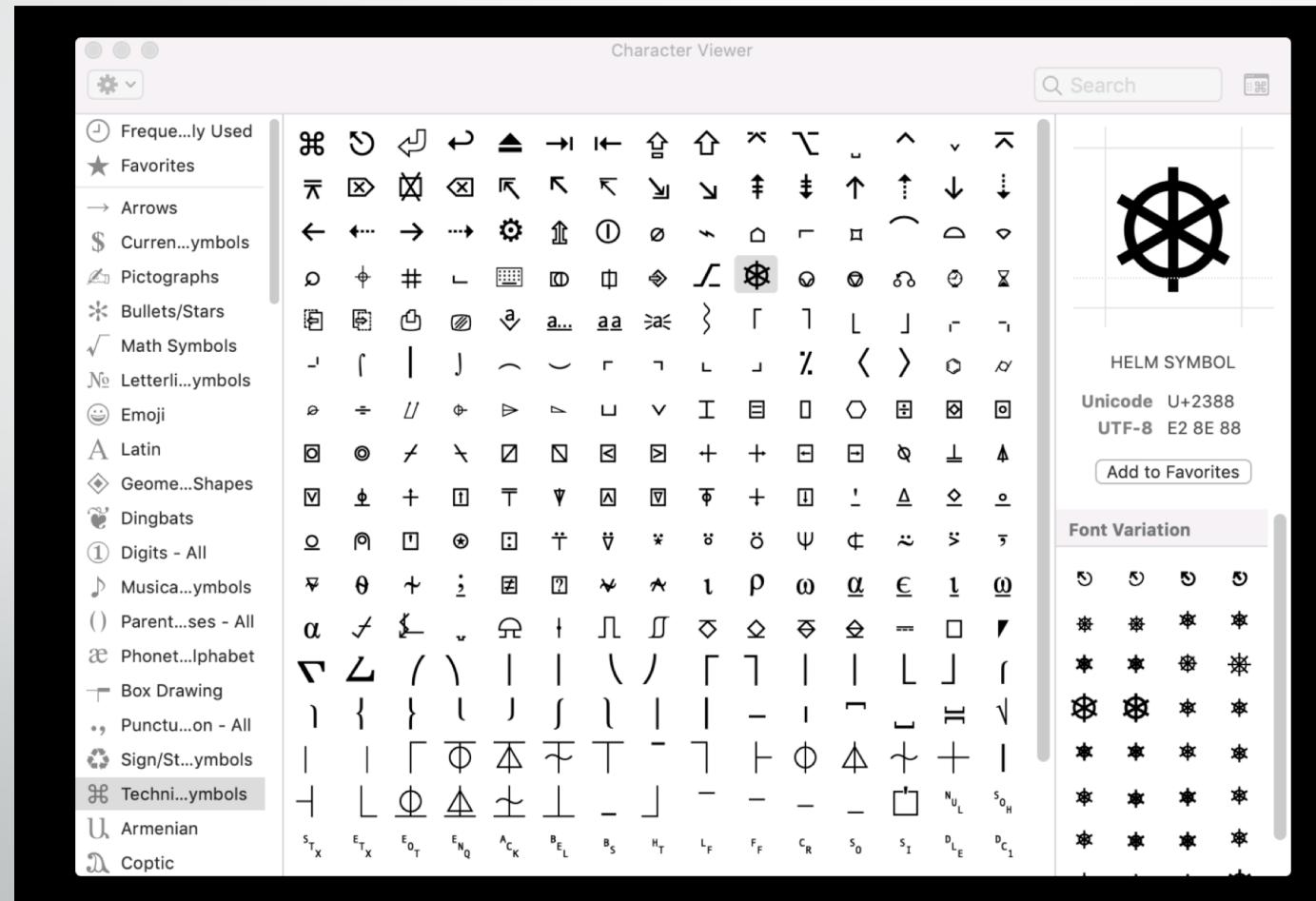
How Scientific Swift can be improved using unicode support

Jeff Biggus, <https://www.dotconferences.com/2019/01/jeff-biggus-scientific-swift>

# Unicode Support



# Unicode Support



# Unicode Support

```
let solutions = (-b ± √(b*b - 4*a*c)) / (2*a)
```

```
let a = Number(2)  
let b = Number(5)  
let c = Number(-3)
```

```
solution 1: -3.00  
solution 2: 0.50
```

```
let a = Number(14)  
let b = Number(9)  
let c = Number(10)
```

```
solution 1: -0.32 - 0.78i  
solution 2: -0.32 + 0.78i
```

# Unicode Support

$$\alpha = K * \lambda + 3 * m$$

$$\beta = \Sigma.\text{diagonal}$$

$$v = \beta - s \odot (L \setminus \alpha)$$

$$\mu = (L \setminus (s \odot (K * U + m)))$$

$$\tau = \alpha \oplus v^\dagger$$

$$\omega = \alpha - \mu \odot v^\dagger$$

# KeyPaths

Indices for the everyday object

Vincent Pradeilles, <https://www.dotconferences.com/2019/01/vincent-pradeilles-the-underestimated-power-of-keypaths>

# Overview

- Keypaths let you read & write to a property of an object
- They work similar to array indices or dictionary keys:
  - `object[keyPath: \.property] = newValue`

```
struct Person {  
    var name: String  
    let uniqueID: Int  
}  
  
var person = Person(name: "Fred", uniqueID: 42)
```

KeyPath (read-only)

```
let uniqueIDKeypath = \Person.uniqueID  
  
person[keyPath: uniqueIDKeypath] // 42  
  
type(of: uniqueIDKeypath) // KeyPath<Person, Int>  
  
// Compilation error:  
// person[keyPath: uniqueIDKeypath] = 99
```

WritableKeyPath

```
let nameKeypath = \Person.name  
  
person.name // "Fred"  
  
person[keyPath: nameKeypath] = "Jim"  
  
person.name // "Jim"  
  
type(of: nameKeypath) // WritableKeyPath<Person, String>
```

# Type Hierarchy

- Types are generic:
  - KeyPath<TypeWithProperty, PropertyType>
- Read & Write support is determined by the type:

Type	Access
KeyPath	Read-only
WritableKeyPath	Read-write

```
struct Person {  
    var name: String  
    let uniqueID: Int  
}  
  
var person = Person(name: "Fred", uniqueID: 42)
```

KeyPath (read-only)

```
let uniqueIDKeypath = \Person.uniqueID  
  
person[keyPath: uniqueIDKeypath] // 42  
  
type(of: uniqueIDKeypath) // KeyPath<Person, Int>  
  
// Compilation error:  
// person[keyPath: uniqueIDKeypath] = 99
```

WritableKeyPath

```
let nameKeypath = \Person.name  
  
person.name // "Fred"  
  
person[keyPath: nameKeypath] = "Jim"  
  
person.name // "Jim"  
  
type(of: nameKeypath) // WritableKeyPath<Person, String>
```

# Type Safety

- Attempting to write to a read-only KeyPath is a compilation error
- Literals resolve to the appropriate type:
  - `\.readOnlyProperty` → KeyPath
  - `\.readWriteProperty` → WritableKeyPath

```
struct Person {  
    var name: String  
    let uniqueID: Int  
}  
  
var person = Person(name: "Fred", uniqueID: 42)
```

KeyPath (read-only)

```
let uniqueIDKeypath = \Person.uniqueID  
  
person[keyPath: uniqueIDKeypath] // 42  
  
type(of: uniqueIDKeypath) // KeyPath<Person, Int>  
  
// Compilation error:  
// person[keyPath: uniqueIDKeypath] = 99
```

WritableKeyPath

```
let nameKeypath = \Person.name  
  
person.name // "Fred"  
  
person[keyPath: nameKeypath] = "Jim"  
  
person.name // "Jim"  
  
type(of: nameKeypath) // WritableKeyPath<Person, String>
```

# Usage

- We use KeyPaths in the Experian app for enhanced readability.
- Examples:

```
application
    .staticTexts["Daily Experian Credit Score"]
    .wait(for: \.exists)
```

```
struct Person {
    var name: String
    let uniqueID: Int
}

var person = Person(name: "Fred", uniqueID: 42)
```

KeyPath (read-only)

```
let uniqueIDKeypath = \Person.uniqueID

person[keyPath: uniqueIDKeypath] // 42

type(of: uniqueIDKeypath) // KeyPath<Person, Int>

// Compilation error:
// person[keyPath: uniqueIDKeypath] = 99
```

WritableKeyPath

```
let nameKeypath = \Person.name

person.name // "Fred"

person[keyPath: nameKeypath] = "Jim"

person.name // "Jim"

type(of: nameKeypath) // WritableKeyPath<Person, String>
```

# Marzipan

overview

# Marzipan

- "Project Marzipan" announced last year (WWDC 2018)
- Aimed to be released this year (2019)
- Goal:
  - To build MacOS apps using UIKit
  - Handles system appearance automatically
    - Window Toolbar
    - Menu Bar
    - Touchbar Controller

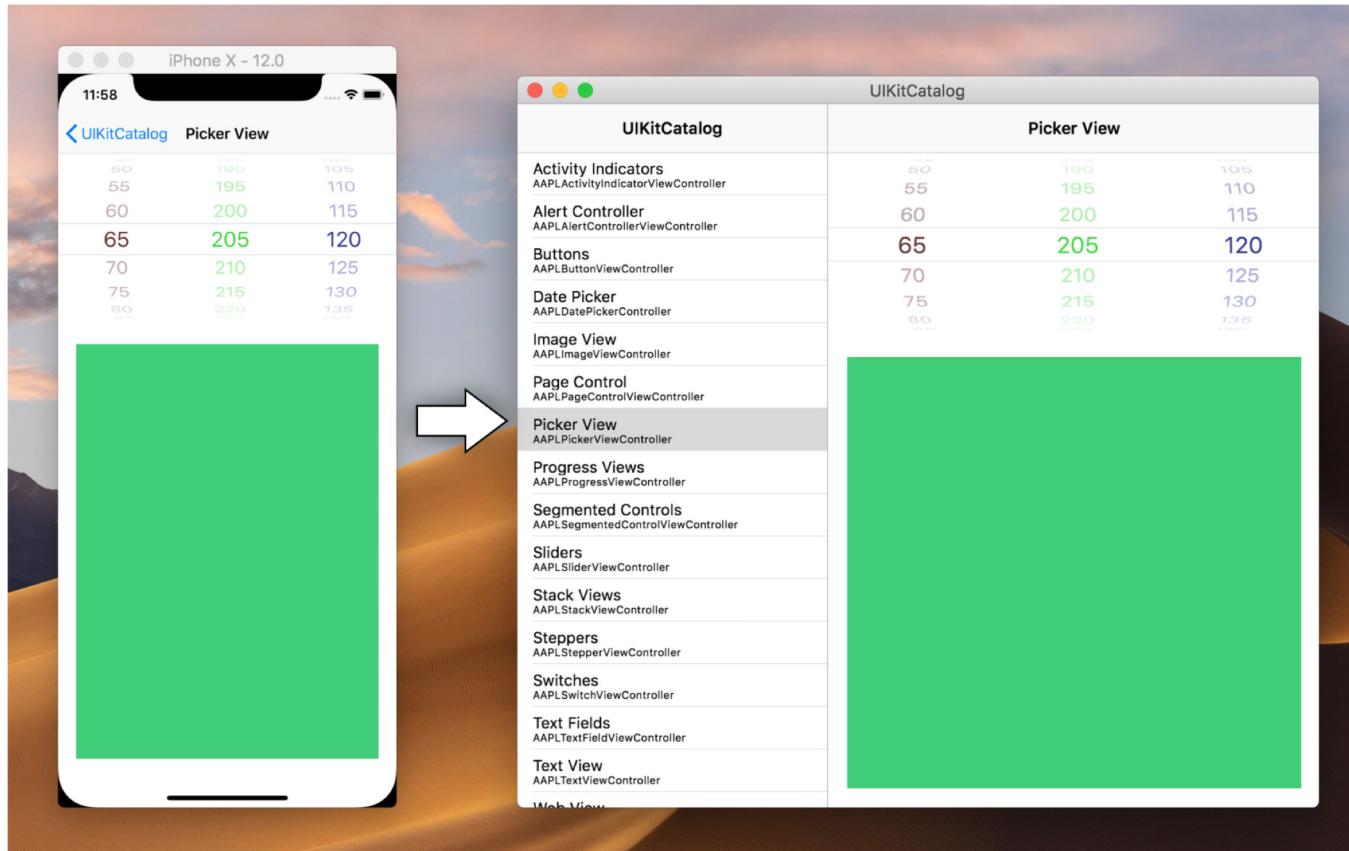
# Marzipanify

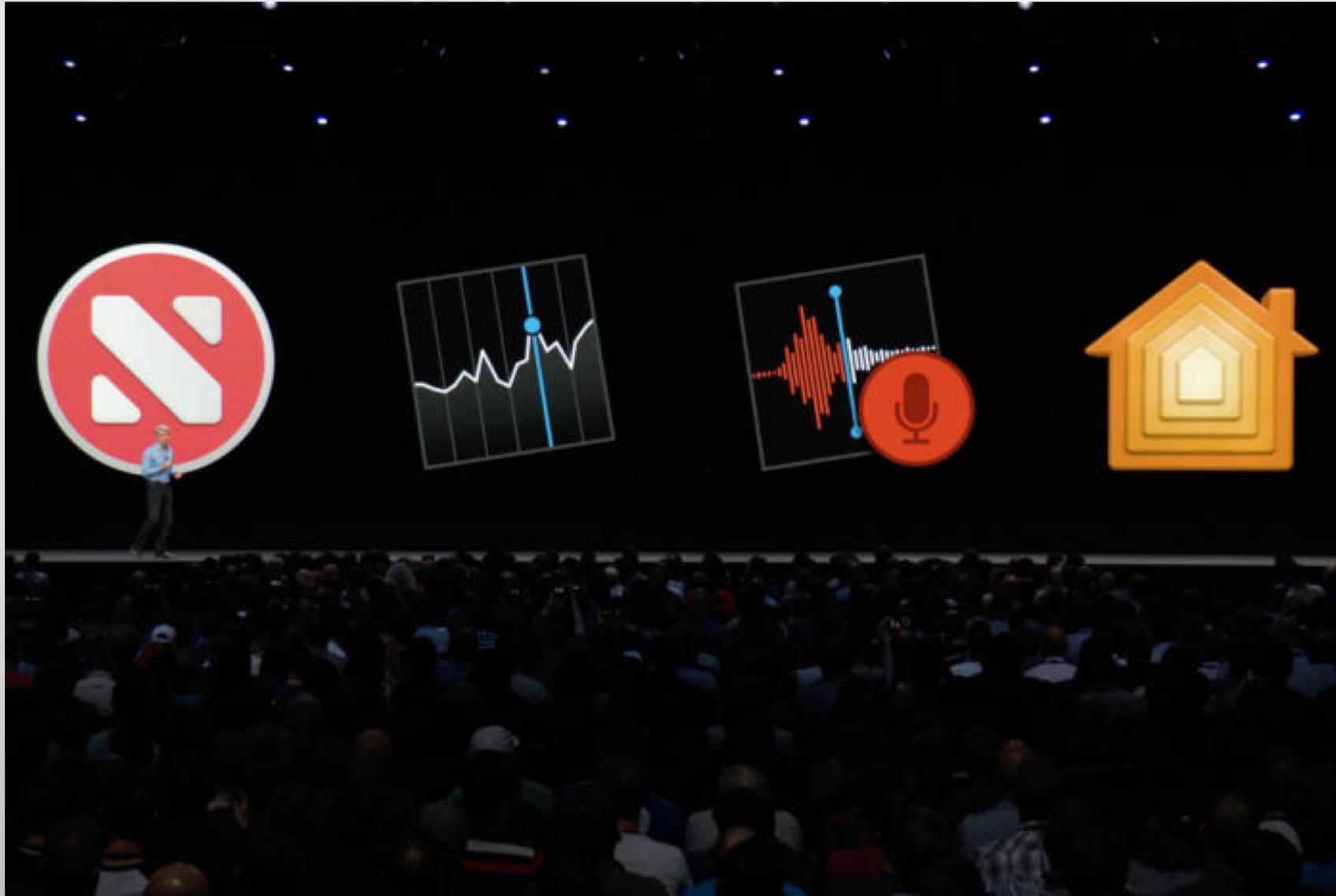
- <https://github.com/steventroughtonsmith/marzipanify>
- An unsupported commandline tool to convert iOS binaries to macOS
- You currently need to disable a load of security features for it to work
- Marzipan official should be supported/released in beta WWDC 2019

# Usage

```
marzipanify MyApp.app|MyFramework.framework|MyBinary
```

## Screenshot





# High Performance Systems In Swift

Structs, classes, and copy-on-write

Johannes Weiss, <https://www.dotconferences.com/2019/01/johannes-weiss-high-performance-systems-in-swift>

# Playground example

Classes, Structs, & Copy-On-Write

<https://git.gbl.experiancs.com/jfoggatt/ukmobile-dotSwift-2019>

# Wrap-up, thoughts, & experiences

# Wrap-up, thoughts, & experiences

- Many top speakers from the industry
- Advanced topics, highly relevant to a broad range of app developers
- Lots of cake 🍰
- Some very specialist lightning talks
- Venue was well suited for the event
- We would go back again 🇫🇷

A photograph of the Eiffel Tower in Paris, France, viewed from the Champ de Mars. The tower is shown in its entirety, from the base to the top. In the foreground, there is a large, open grassy area. In the background, there are some trees and buildings. The sky is clear and blue.

Questions?