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
  - ¾ 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 Dalyrumple
Wh@t Is 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)

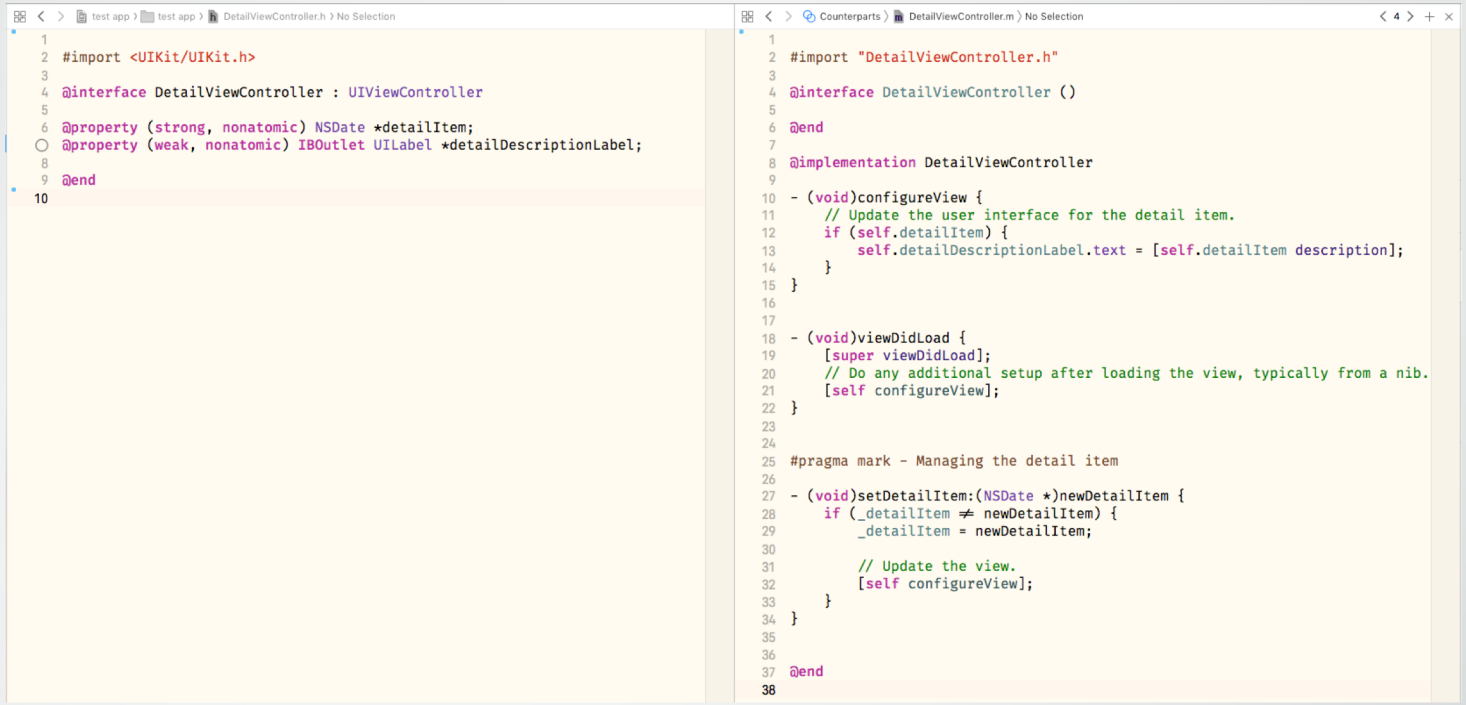
Tom Doron (Apple Swift NIO)
Implementing JSON-RPC with SwiftNIO

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



```
#import <UIKit/UIKit.h>

@interface DetailViewController : UIViewController

@property (strong, nonatomic) NSDate *detailItem;
@property (weak, nonatomic) IBOutlet UILabel *detailDescriptionLabel;

@end
```

```
#import "DetailViewController.h"

@interface DetailViewController ()

@end

@implementation DetailViewController

- (void)configureView {
    // Update the user interface for the detail item.
    if (self.detailItem) {
        self.detailDescriptionLabel.text = [self.detailItem description];
    }
}


- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    [self configureView];
}


#pragma mark - Managing the detail item

- (void)setDetailItem:(NSDate *)newDetailItem {
    if (_detailItem != newDetailItem) {
        _detailItem = newDetailItem;

        // Update the view.
        [self configureView];
    }
}


@end
```

- 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`.

See also, the Objective C Runtime framework: https://developer.apple.com/documentation/objectivec/objective-c_runtime?language=objc

Key-Value Observation

Objective-C

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

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

Swift

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

Allows safe syntax for Obj-C keypath strings

```swift
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;
autoresize = W+H; userInteractionEnabled = NO; layer = <CABackdropLayer:
0x129e54010>>
   |    | <UIView: 0x129e64610; frame = (0 0; 1112 834); hidden = YES; opaque = NO;
autoresize = 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: 0x129d8aeb0>>
   |    |    | <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>>
```

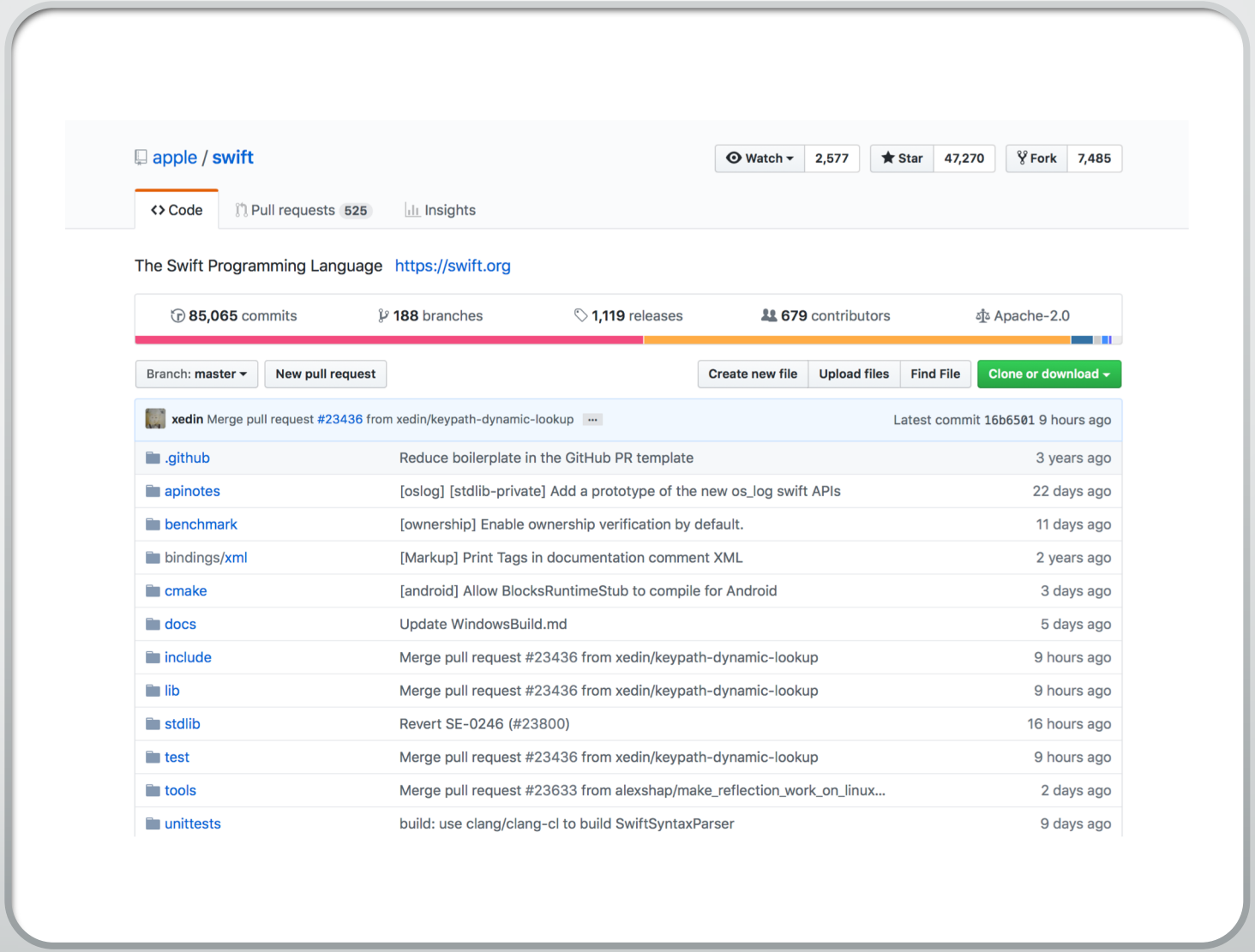
Wh@t Is Th@T (Attributes)

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

Mark Dalyrumple, 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



Find-ing things

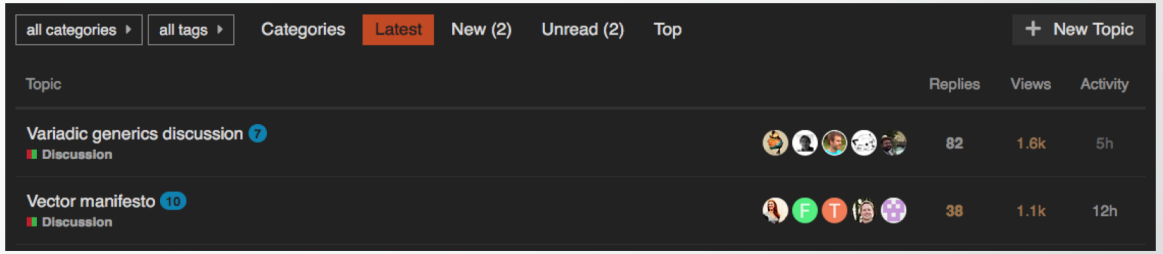
- `find . -type f -exec grep SomethingInteresting {} \; -print`
- With a local clone of Swift compiler, this command lets you search the file contents from the command line, to find points of interest in the Swift compiler.
- One such example is the file identifying each Swift attribute as a keyword: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

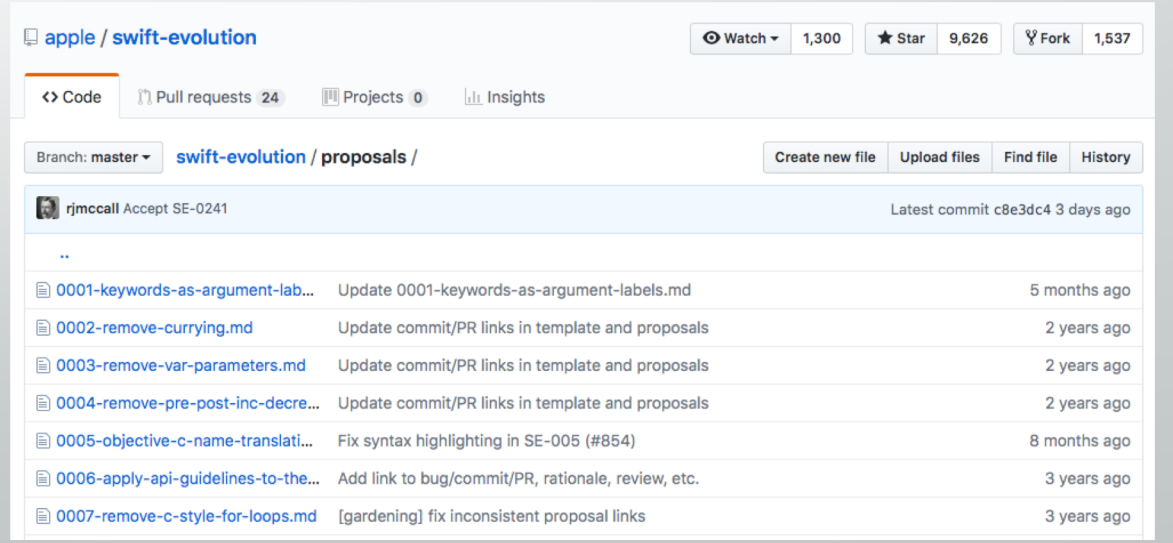
Access control keywords:

```
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



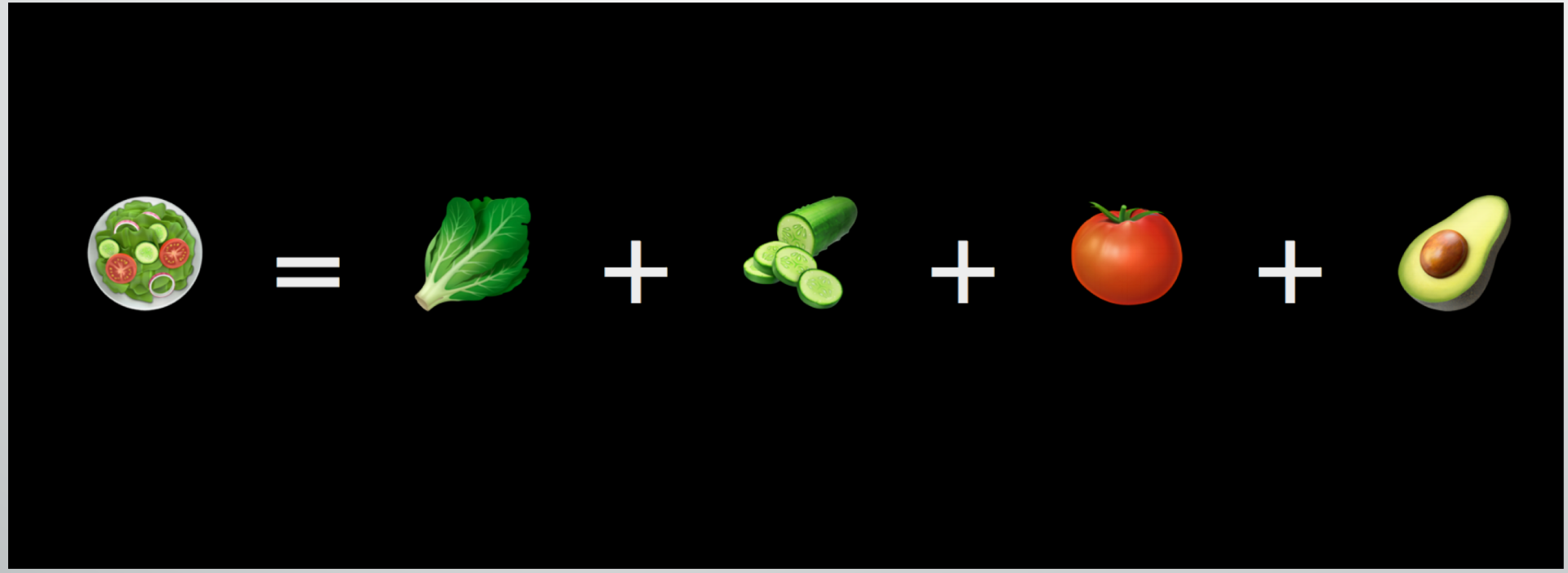


Unicode support & defining operators

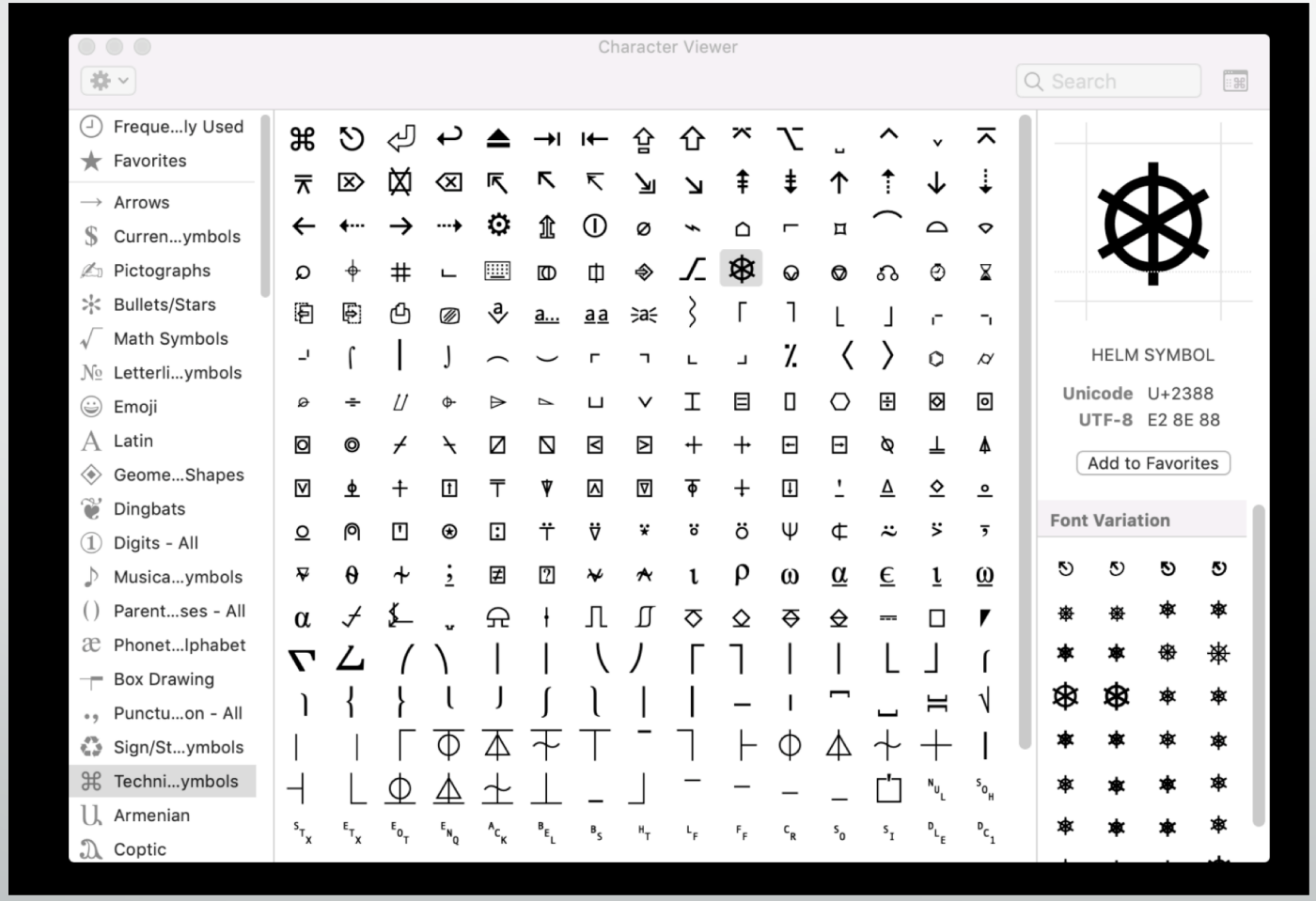
How Scientific Swift can be improved using unicode support

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 - 0.78¡
solution 2: -0.32 + 0.78¡
```

Unicode Support

```
α = K * λ + 3 * m
β = Σ.diagonal

ν = β - s ⊙ (L \ α)
μ = (L \ (s ⊙ (K * u + m)))

τ = α ⊕ ν†
ω = α - μ ⊙ v†
```

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`

Try them with KeyPaths.playground
https://git.gbl.experiancs.com/jfroggatt/ukmobile-dotSwift-2019

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

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

KeyPath (read-only)

```swift
let uniqueIDKeypath = \Person.uniqueID

person[keyPath: uniqueIDKeypath] // 42

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

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

WritableKeyPath

```swift
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

Try them with KeyPaths.playground
https://git.gbl.experiancs.com/jfroggatt/ukmobile-dotSwift-2019

```
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`

Try them with KeyPaths.playground
https://git.gbl.experiancs.com/jfroggatt/ukmobile-dotSwift-2019

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

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

KeyPath (read-only)

```swift
let uniqueIDKeypath = \Person.uniqueID

person[keyPath: uniqueIDKeypath] // 42

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

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

WritableKeyPath

```swift
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)
```

Try them with KeyPaths.playground
https://git.gbl.experiancs.com/jfroggatt/ukmobile-dotSwift-2019

```
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

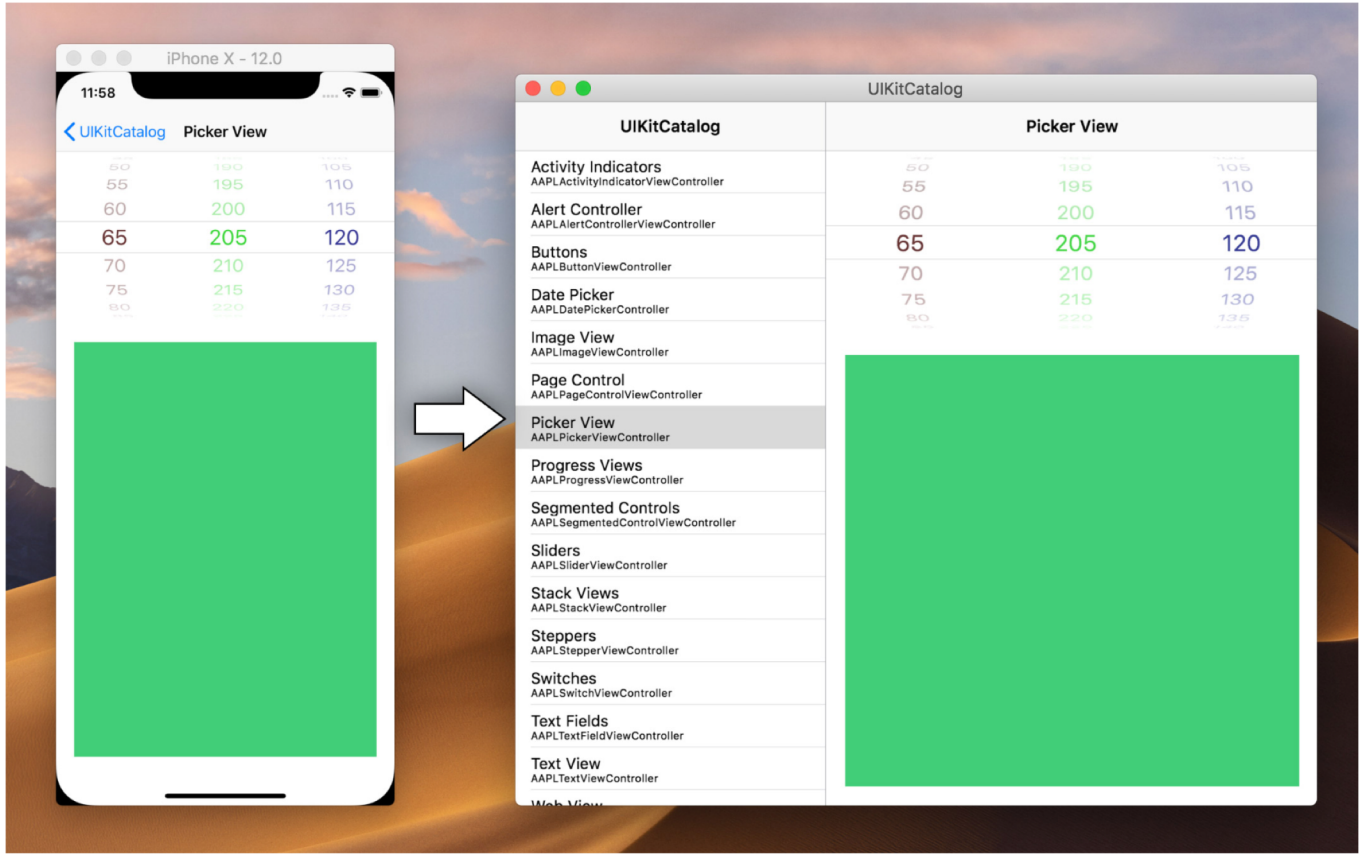
Marzipanify

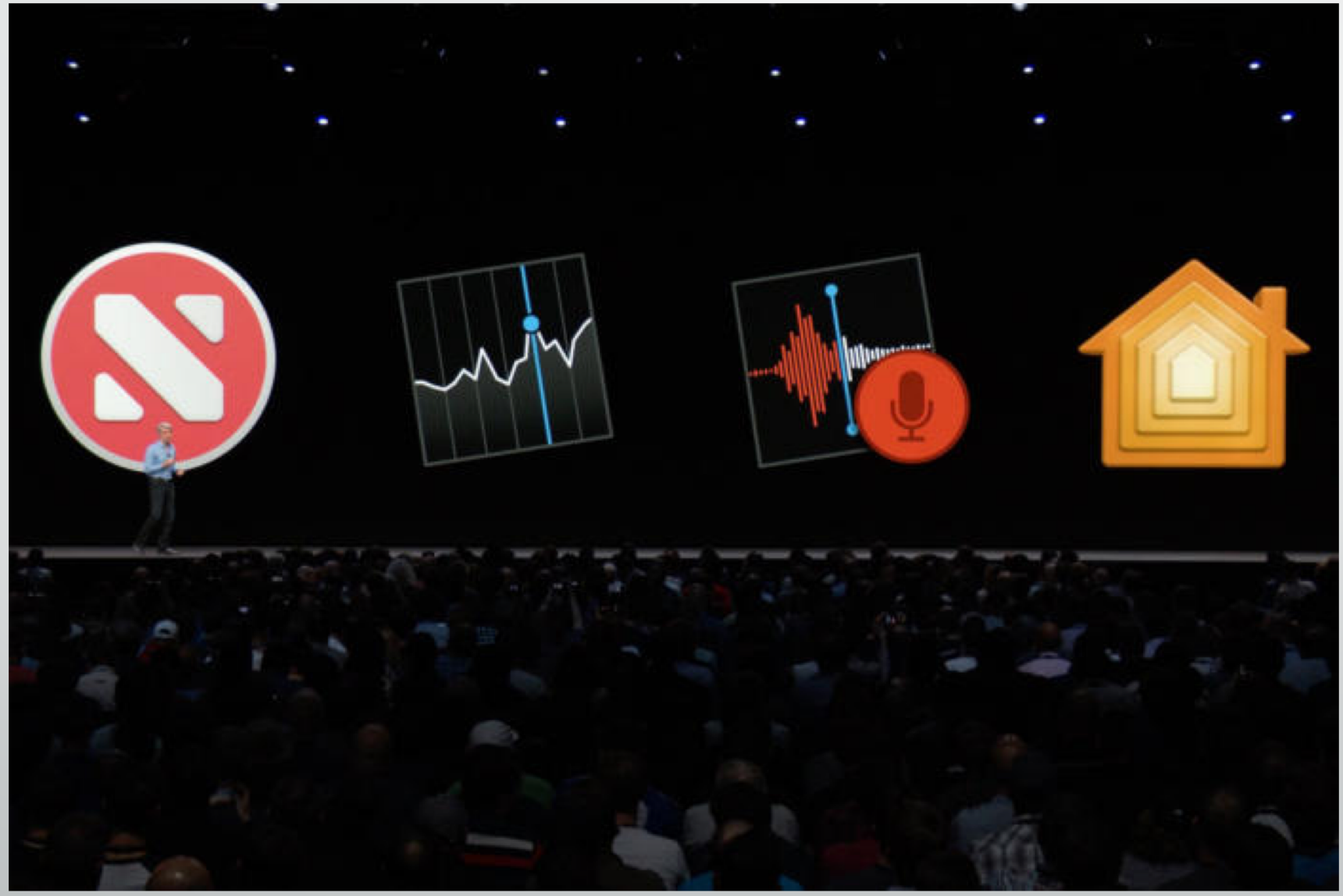
- https://github.com/steventroughtonsmith/marzipanify
- An unsupported commandline tool to convert iOS binaries to macOS
- You currently need to disable a load of security features for it to work
- Marzipan official should be supported/released in beta WWDC 2019

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/jfroggatt/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 🇫🇷

Questions?