for Haters

A new programming language

Talk at CocoaHeads Aachen
Werner Lonsing

5/24/2018

for Haters

Looking back: Objctive-C

Is Objective-C a language at all?

Technically: A preprocessor as thin layer above C-Compiler.
Classes have 2 files, interface and implementation,
top level keywords (@end), implicit type defaults to ‘id’
Syntax for method-calls: [obj with: val], declaration ‘-‘ or ‘+’,’
Usually instance-methods returned ‘self’ for chaining.
The rest is strict C

Developed in the early 1980s by Brad Cox, as ‘soldering gun’ for ‘Software-ICs’, then used and later acquired by NeXT in 1987.

There is almost no documentation, and it is not needed.

The values of the language are in its runtime and the related frameworks.

```
@interface MyClass : NXObject {
 // instance variables
id theObject;
}

+ classMethod;

- instanceMethod;
- instanceMethodWithParameter:myObject;
- otherMethod: (float)value;

@end


@implementation MyClass
+ classMethod
  {…
  }

- instanceMethod
  {
    [theObject doSomething];
    return self;
  }

@end
```

chaining

```
[[[[myObject do:now]andThen]later]finally];
```

for Haters

Objective-C: Going South

Getters and setters

‘retain’ and ‘release’

ARC and properties

GCD and Blocks

Literals

Dot-syntax

Nullability

Punching holes

Distributed ownership

Clogged interface: `@property(getter=theDisplay, readonly, retain) NSView *theView;`

```
[who knows:^void (blocks) {anybody}];
```

Not that bad: `@“string”, @[a,b,c]; @{ k1: o1, k2: o2};`

Same thing twice: `[myView window];` vs. `myView.window;`

Greetings from Swift: `@property (readonly, nonatomic) NSView * _Nonnull aView;`

… we were doomed …

for Haters

Looking @ Swift

A new language to the rescue.

Safety first: Empty object-pointers and type mismatching is made (almost) impossible, because such code never compiles.

Swift is a real computer language, not a bloated preprocessor, with a full range of features, namely dot-syntax with round and curled brackets on a single file.
It has concurrent types: ‘String’ vs. ‘NSString’, ‘Array’ vs. ‘NSArray’ etc.
It offers a lot of rude expressions nobody immediately understands:

```
[_] [String:Any] [NSString]
((value?)!)
```

empty array of type optionals

```
func swapTwoValues<T>(_ a: inout T, _ b: inout T)
```

generic typing

for Haters

Looking @ Swift

Mixing types: ***What?***

Float, CGFloat, Double

Why is there a difference?
Who has, as a developer in need of a ‘soldering gun’, intentions to make a difference like:

```
let a:CGFloat = 1.2
let b = Float(a)
```

Why does this not compile?

```
let a = 1.2 as! CGFloat
let b = 0.1 as! Float

let c = a * b
```

instead of explicit typing

```
let c = Float(a) * b
```

for Haters

Looking @ Xcode

May Xcode and its automated correctional facilities be always with you!

There is one question:
If the machine, the software, knows what is wrong and how to do it right, why can’t the machine do it on its own?

Maybe some more questions:

Why the torture?
Why is that the stuff not under the hood?

for Haters

Looking @ Updates

Who likes compatibility?

2014-09-09	Swift 1.0
2014-10-22	Swift 1.1
2015-04-08	Swift 1.2
2015-09-21	Swift 2.0
2016-09-13	Swift 3.0
2017-09-19	Swift 4.0
2018-03-29	Swift 4.1

Demo

The End

Thank you.