Apple platform for Javascripters

Jan Dědeček
Tomáš Novella

SALSITA SOFTWARE

	JS Frontend	xOS
Language	"Javascript"	(Objective-C) Swift
Toolchain	Transpiler(babel), Bundler(webpack)	Compiler, Linker (xcodebuild, part of Xcode)
Libraries	React/Angular, npm modules	Cocoa, (Cocoa)pods
User Interface	HTML/CSS	Xcode Interface Builder (outputs XML blob)
Distribution	Hosted Website	AppStore

1. Language

Objective-C

- Designed around the same time as C++
- Superset of C, Smalltalk-like message passing
- Dynamic type system with optional type annotations (think flow/typescript)

Objective-C drawbacks

- Bad performance (dynamic dispatch, compiler can’t optimize method dispatch)
- Old-fashioned, weird syntax
- Lack of modern features (namespaces, pattern matching)
- Limited (type) safety

```objc
@implementation ProtocolHandlerJSBridge

// mapping of meta-protocol resources to "hostnames"
// which will be encoded in the request
static NSDictionary *_bridgeRequestHosts;

+(BOOL)isBridgeRequestURL:(NSURL *)url {
  return [[Settings bridgeScheme] isEqualToString:url.scheme];
}

+(BOOL)isVirtualResourceBridgeRequestURL:(NSURL *)url {
  return [self isBridgeRequestURL:url] &&
    ([[_bridgeRequestHosts allKeysForObject:url.host] count] != 0);
}

+(NSURL *)URLWithBridgeResource:(JSBridgeResource)resource path:(NSString *)
  path
{
  // make absolute path
  if(!path) {
    path = @"/";
  } else if(![path hasPrefix:@"/"]) {
    path = [NSString stringWithFormat:@"/%@", path];
  }
  return [[NSURL alloc] initWithScheme:[Settings bridgeScheme]
                                  host:_bridgeRequestHosts[@(resource)]
                                  path:path];
}
```

Swift

- Is "Objective-C without the C"
- Interoperable with Objective-C via bridging
- More robust
- The Swift programming Guide

[Xcode Playground Jamming]

***Alternatively: https://iswift.org/playground**

Memory 101

```
> function repeat(str, count) {
    let ret = ""
    for (let i = 0; i < count; ++i) {
      ret += str
    }
    return ret
  }
<· undefined
> repeat("hello", 3)
<· "hellohellohello"
```

- Objects can be saved on stack or heap (disregard closures and escape analysis)
- Stack variables deallocated after end of scope
- What about heap variables ?

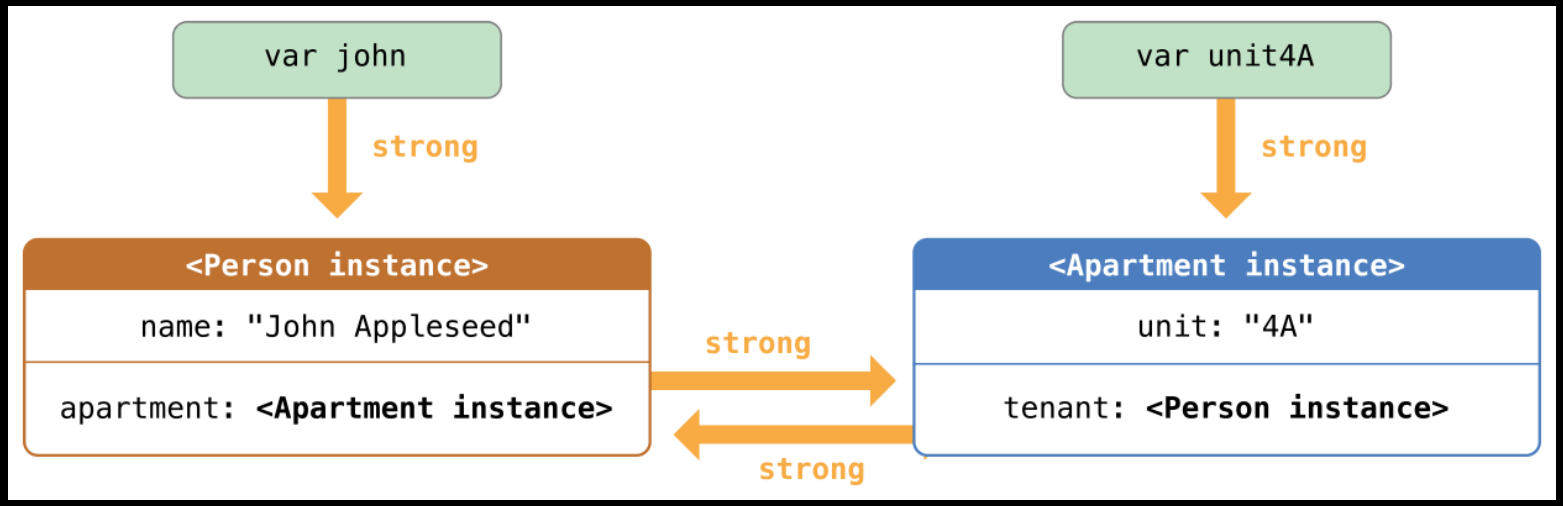
Heap Management

- C/C++ manually - "imperative" management
- Swift - ref. counting - "declarative" management
- Javascript/Java automatically - no management

Reference Counting

- Counts active references to every object
- Operations:
  - Retain - increases reference counter on variable assignment
  - Release - decreases reference counter at the end of the variable lifetime

Reference Cycle



Reference Counting - Properties

+ Smaller overhead

+ More predictable behavior

+ Does not stop the world

- Danger of strong reference cycles

- Atomic reference updates essential

[Reference Counting Demo]

2. User Interface

Creating User Interfaces - Frameworks

- Cocoa for macOS (Foundation, AppKit, ...)
- Cocoa Touch for iOS (Foundation, UIKit, ...)
- Slightly divergent and incompatible UI frameworks for iOS and macOS
- *Check out GNUStep for sources

Creating User Interfaces - Cocoa vs Web

- Views instead of HTML elements
- No built-in support for declarative styling (no CSS)
- Completely different *layout* system
- All views have same layout semantic -> No equivalents for HTML table, div, span,... elements

Cocoa (Cocoa Touch)	HTML
NSView (UIView)	or
NSTextField	<input type="text" />
NSButton (UIButton)	<input type="button" />
NSTableView (UITableView)	<table />

Xcode Interface Builder

- UI designer for all xOS platforms
- Allows for:
  - Screens and transitions among them (segues)
  - Views and their properties (colors, fonts, sizes, positions)
  - Layout constraints

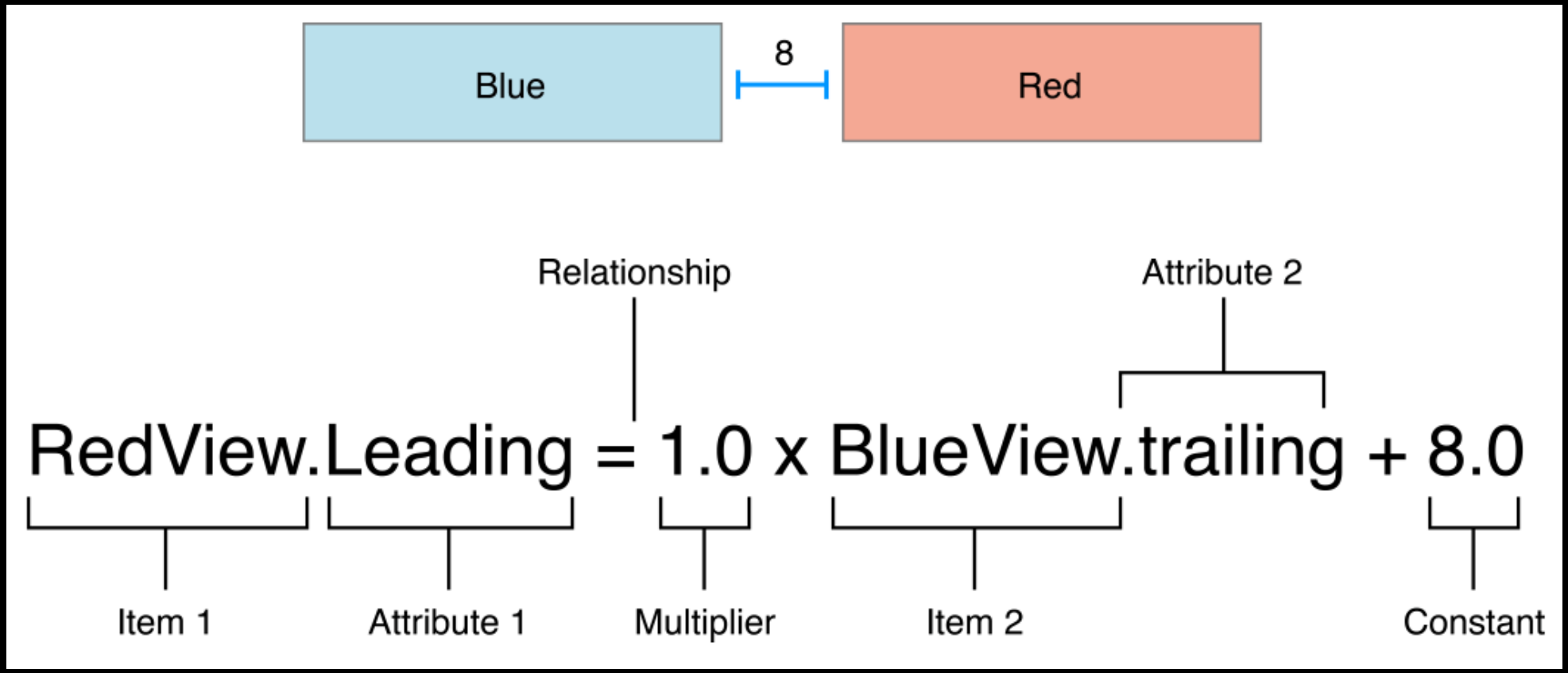
View positioning

- Manual
  - Programmer defines positions and dimensions of views
  - Does not scale well with different screens
- Auto Layout
  - = Constraint declaration
  - Programmer defines relations among view’s positions and dimensions
  - Requires more computation resources

Auto Layout

- Dynamically calculates the size and position of all views
- Internally solves system of linear constraints (Cassowary solver)
- View properties are bound by constraints (x, y, width, height)

Anatomy of a constraint



Auto Layout (cont'd)

- It is possible (at runtime) to:
  - Add/remove constraints
  - Alter constraint constants and multipliers
  - Alter constraints priorities
- Cocoa is to able animate constraint changes
- Cocoa is able to deal with ambiguous or/and infeasible system of constraints

Auto Layout (vs Browser document rendering)

+ Easy to create box-like user interfaces

- Unpredictable, especially with animation

- Unspeakably hard to create document-like layouts with floating elements

[UI Demo]

3. Real-life iOS App

Apple Dev policy

- Dev account 99 USD/1 Year
- App Store - one central place https://developer.apple.com/app-store/review/guidelines/
- Every new application must be reviewed by Apple
    - Might get rejected from App Store (private API, FB login)

DEMO App:
MovieHacker

Links

- Swift deveveloper's guide
- Swift source code on github
- GNU Step sources of Cocoa/Foundation
- Ray Wenderlich, objc.io