How Swift Changes iOS Development

@jonfriskics

Code School
LEARN BY DOING

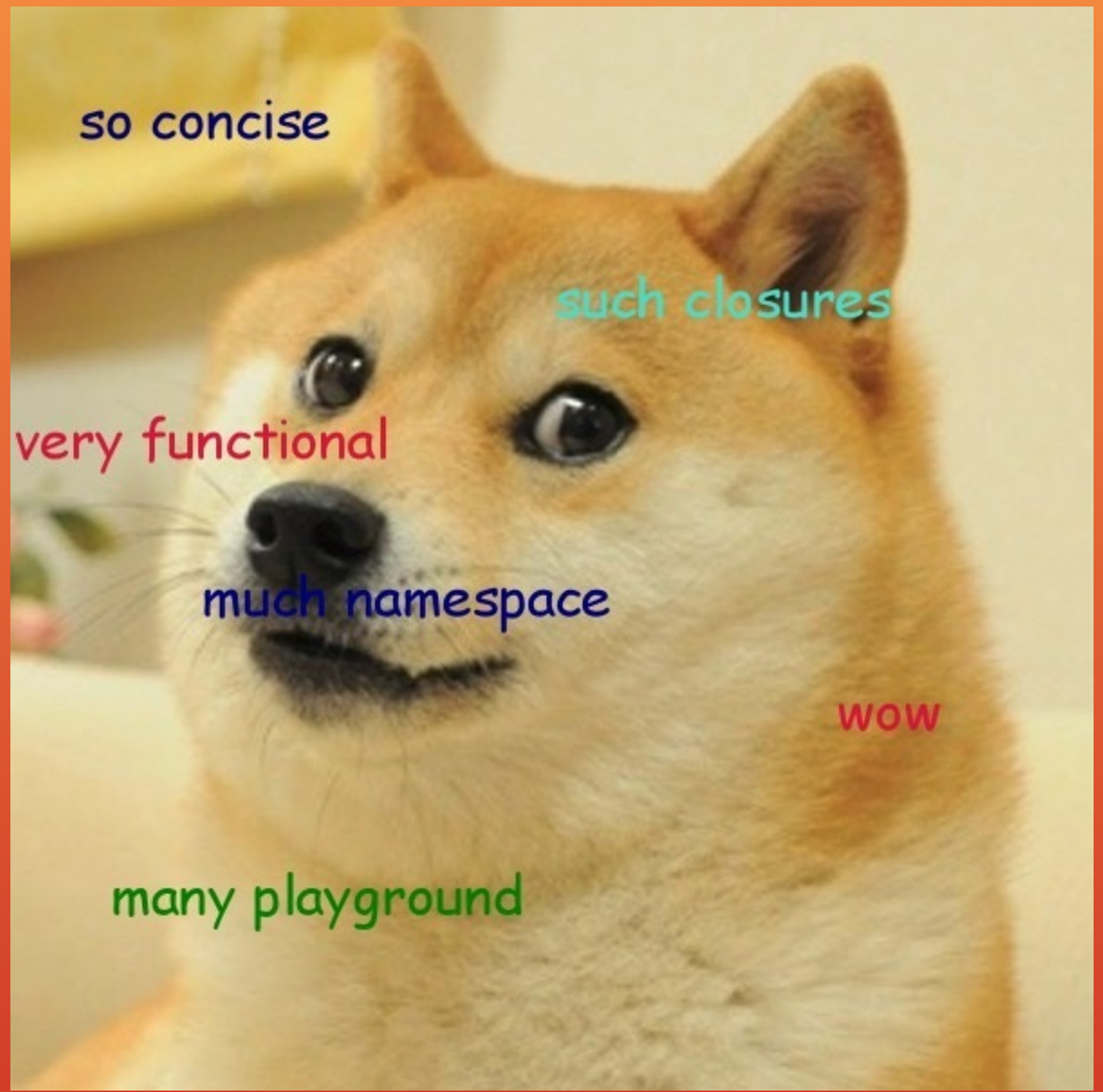
Who has heard of Swift?

Who has started writing an app in Swift?

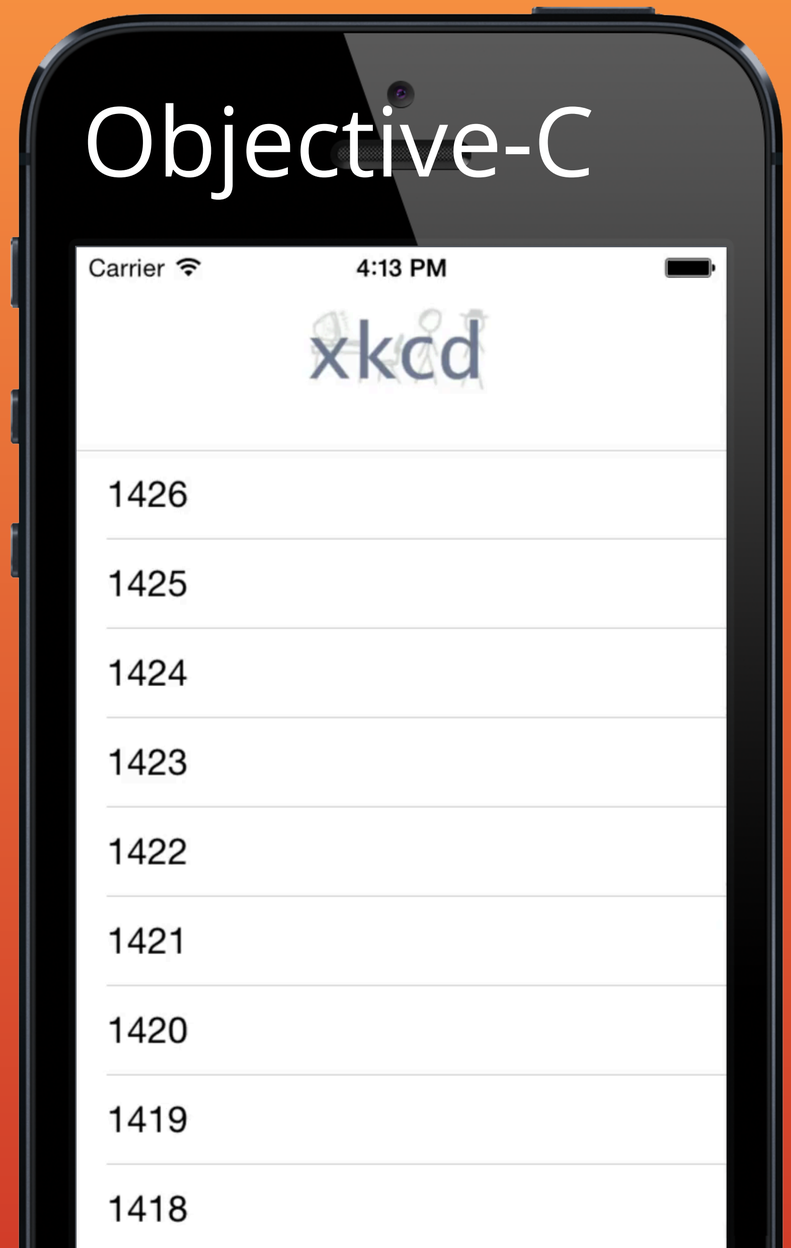
So why Swift?

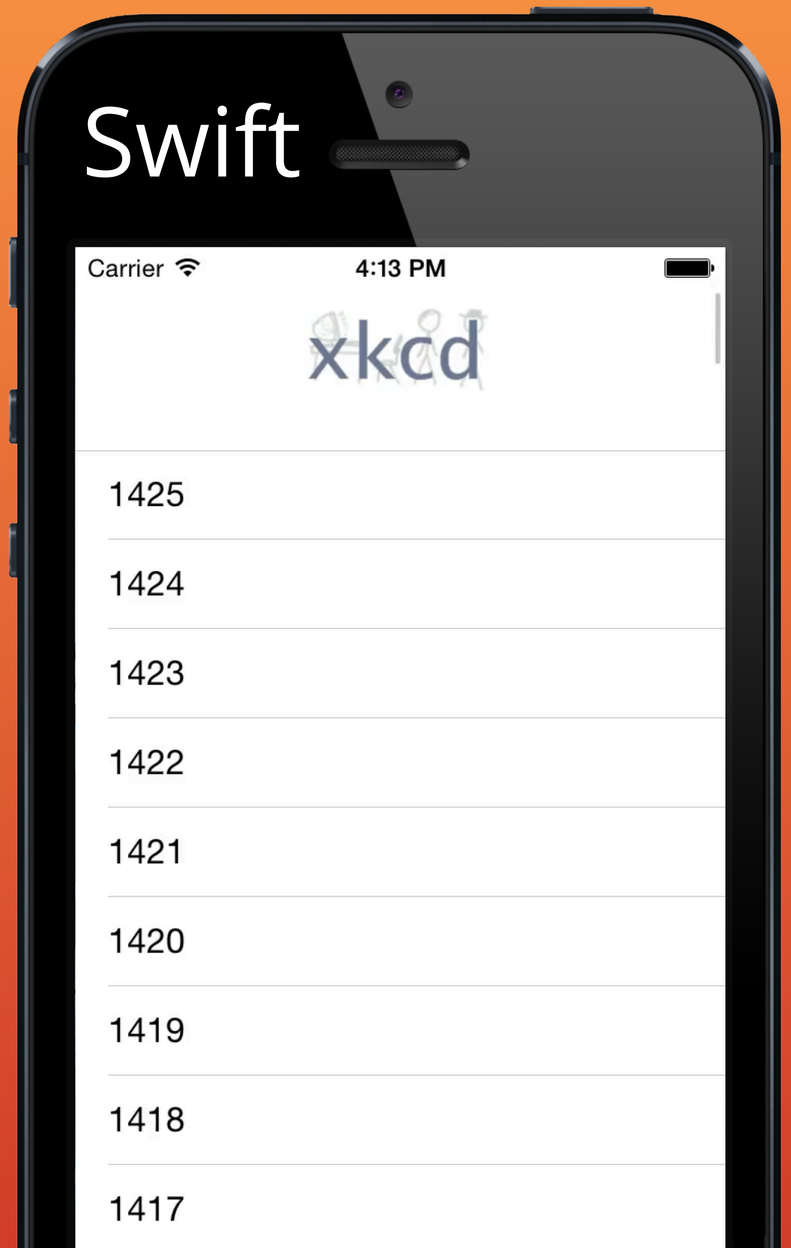

Functional patterns
Protocols and extensions on structs
Pattern matching
Concise syntax
Closures
Generics
Fast iteration
Native collections
Optional types
Operator overloading
Object orientation
Namespaces
Tuples
Type inference
Clear mutability syntax
Read-Eval-Print-Loop (REPL)
Interactive playground
Multiple return types
Compile to native code

So why Swift?

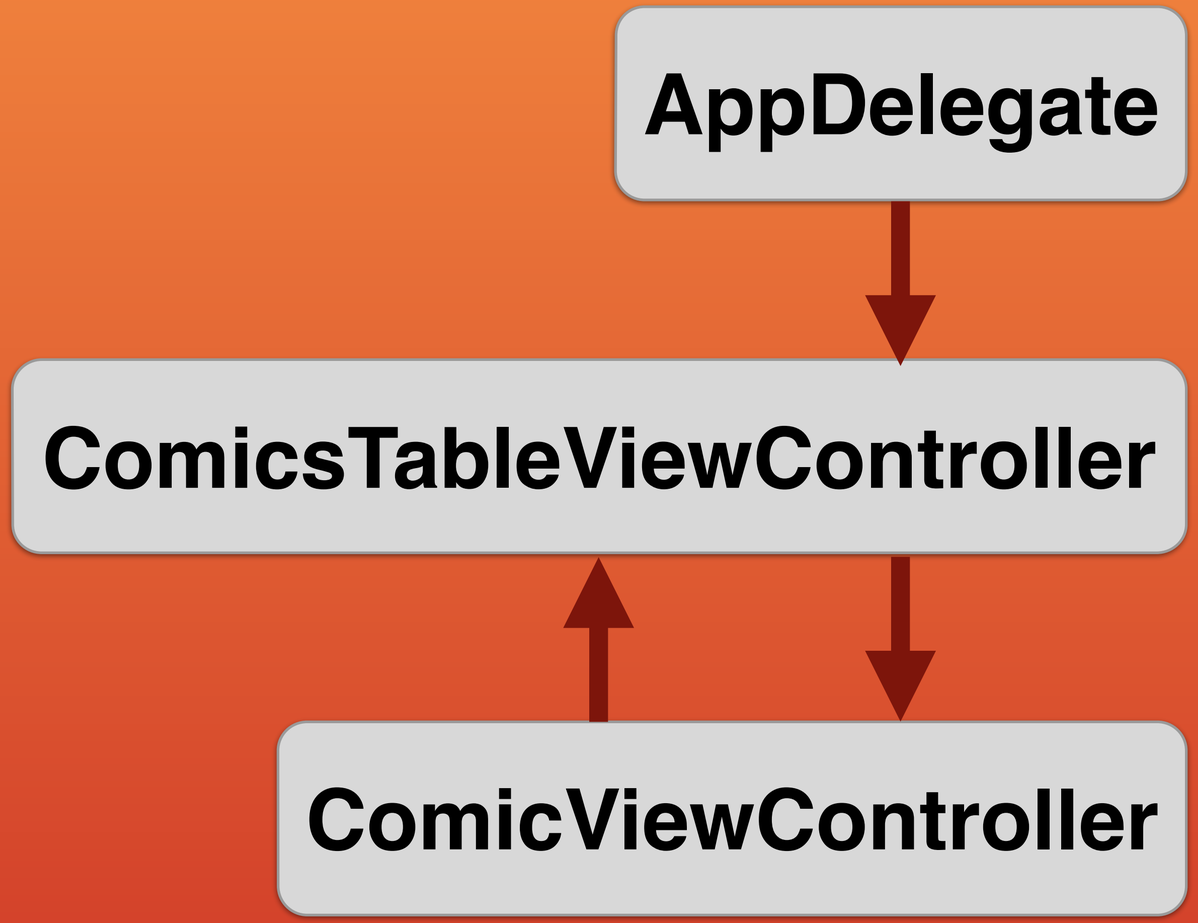

so concise
such closures
very functional
much namespace
wow
many playground

Simple XKCD viewer





XKCD Viewer Structure



Runs when the app starts and sets up the first screen

Shows a list of available comics in a table, one comic per cell

Shows the actual comic when a cell is tapped

AppDelegate Differences

Objective-C

```
ComicsTableViewController *comicsTVC =
    [[ComicsTableViewController alloc]
          initWithStyle:UITableViewStyleGrouped];
```

Explicit vs. implied type

Full vs. short enum names

Swift

```
let comicsTVC =
    ComicsTableViewController(style: .Grouped)
```

alloc/init is implied in Swift initializer

ComicsTableVC Initializer Differences

Objective-C

```
- (id)initWithStyle:(UITableViewStyle)style {
    self = [super initWithStyle:style];
    if(self) {
        [self setupDataSource];
    }
    return self;
}
```

No need to call self in a class

Less checking

Swift

```
override init(style: UITableViewStyle) {
    super.init(style: style)
    setupDataSource()
}
```

Response error handling in Objective-C

Objective-C

in network response callback

```
NSError *jsonParsingError;
NSDictionary *jsonResponse = [NSJSONSerialization ...

if(jsonResponse == nil) {
    NSLog(@"error reading json: %@“,
          jsonParsingError.localizedDescription);
} else {
    // do stuff with the network response
}
```

check if no response

do something if response exists

IF YOU DON'T CHECK YOUR RESPONSE OBJECT BEFORE YOU TRY TO USE IT

YOU'RE GONNA HAVE A BAD TIME

Response error handling in Swift

Swift

in network response callback

```
var jsonParsingError:NSError?
let jsonResponse = NSJSONSerialization ... as NSDictionary?

if let jsonResp = jsonResponse {
    // the response exists, do something with it
} else {
    println("Error reading json
            \(jsonParsingError!.localizedDescription)")
}
```

do something else if no response exists

Wait, this seems like the same thing?

Check if a response comes back

- Report the error if something is wrong
- Carry on if everything is good

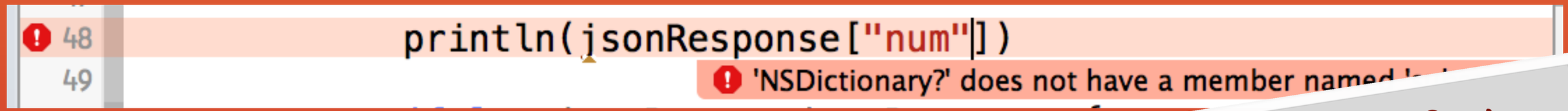
let's take a closer look

Question Mark == Optionals

Swift

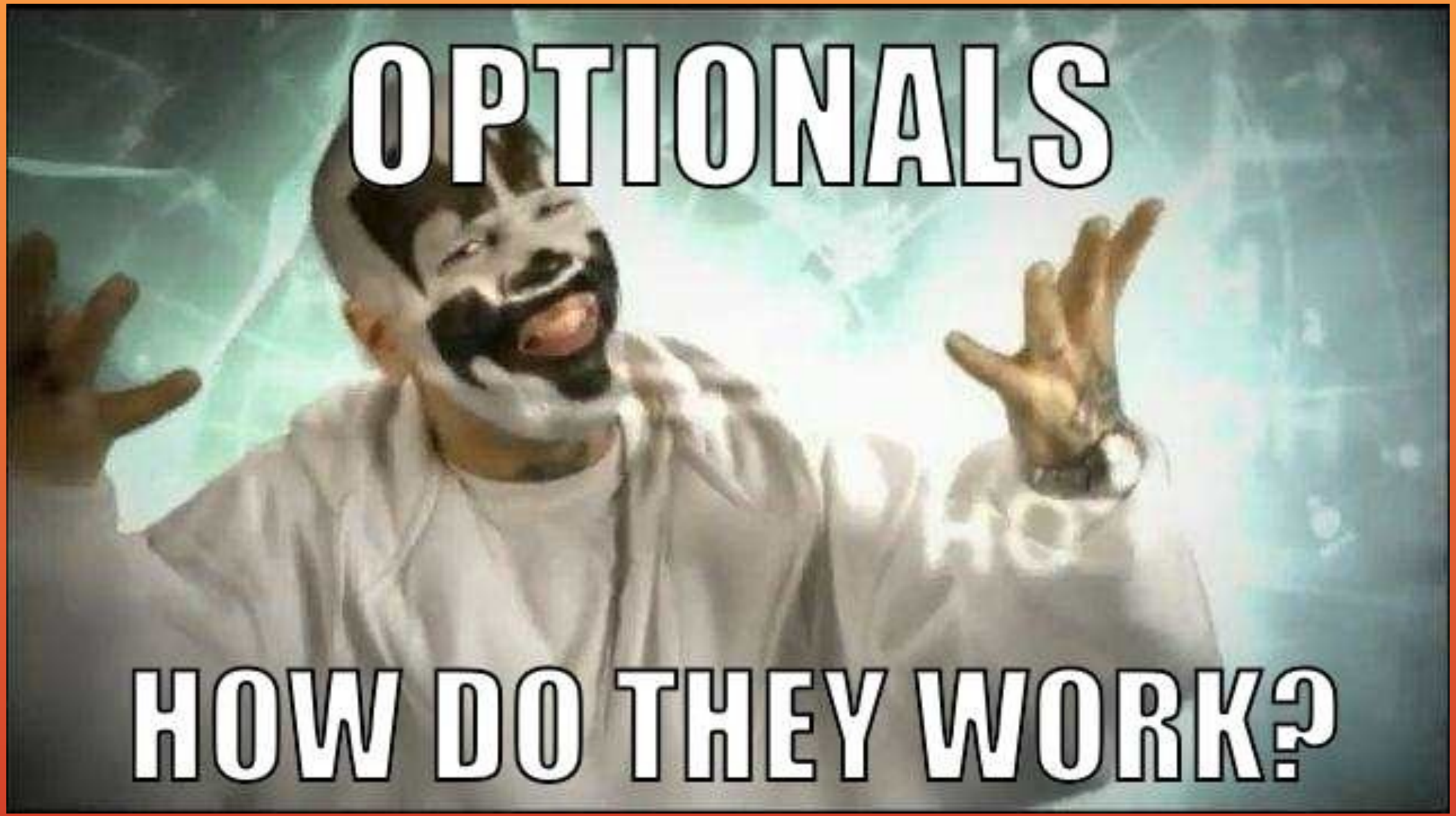
```
var jsonParsingError:NSError?
let jsonResponse = NSJSONSerialization ... as NSDictionary?
```

The question mark means that this is now an **optional** value



Swift won't let you use an **optional** without first checking if it exists

forced safety


OPTIONALS
HOW DO THEY WORK?

Marking a value as optional

Swift

```
let jsonResponse = NSJSONSerialization ... as NSDictionary?
```

this just might be an NSDictionary

Optionals force you to ask the question **does this thing exist?** before you try to use it

First way to answer: !!!!!!!!!!!!!

Swift

```
let jsonResponse = NSJSONSerialization ... as NSDictionary?
```

1. Add an exclamation point to the end of the object name

Swift

```
println("response is \(jsonResponse!)")
```

This says trust me, the thing exists

crashes if it doesn't exist

Second way to answer: Optional Binding

Swift

```
let jsonResponse = NSJSONSerialization ... as NSDictionary?
```

2 Bind the optional to a non-optional value with **if let**

Swift

```
if let resp = jsonResponse {
    println("response is \(resp)")
}
```

only true if jsonResponse is not nil

Optionals in use

don’t try to set the navigation bar style
if the navigation bar doesn’t exist

Swift

```
override func viewWillAppear(animated: Bool) {
    super.viewWillAppear(animated)

    navigationController?.setNavigationBarHidden(true, animated: true)
}
```

Shortcut for if let

Optionals in use

Swift

this sets the label text if the label exists

```
var comicTitle:UILabel?


...


func displayComic() {
    self.comicTitle?.text = self.comicInfo!["safe_title"] as? String
    self.comicTitle?.sizeToFit()
}
```

Optionals in use

Swift

```
var window: UIWindow?

func application(application: UIApplication!,
    didFinishLaunchingWithOptions launchOptions: NSDictionary!) -> Bool {
    ...
    window = UIWindow(frame: UIScreen.mainScreen().bounds)
    window!.rootViewController = navController
    window!.makeKeyAndVisible()
}
```

window is an optional now

if you're pretty sure the object exists,
use the exclamation point to unwrap it

Moving on… View Creation

Objective-C

```
@property (strong, nonatomic) UILabel *comicTitle;
```

create the idea of a label called comicTitle

```
...

- (void)viewDidLoad {
    [super viewDidLoad];

    self.comicTitle = [[UILabel alloc] init];
```

create the actual label

```
    self.comicTitle.textAlignment = NSTextAlignmentCenter;
```

set some properties

```
    [self.imageScrollView addSubview:self.comicTitle];
```

add it to the screen

```
}
```

this works, but things can get out of hand with lots of views

View Creation in Swift

Swift

```
var comicTitle:UILabel?
...
override func viewDidLoad() {
    super.viewDidLoad()

    comicTitle = UILabel()

    if let label = comicTitle {
       label.textAlignment = .Center
    }

    view.addSubview(comicTitle!)
}
```

create the idea of a label called comicTitle

create the actual label

set some properties

add it to the screen

A little bit less code, but still too mixed in with everything

Swift lazy instantiation

Swift

```
lazy var comicTitle:UILabel? = {
    let label = UILabel()
    label.textAlignment = .Center
    return label
}()

override func viewDidLoad() {
    super.viewDidLoad()

    view.addSubview(comicTitle!)
}
```

Put all of the label creation code in one spot

Closures

this starts the closure

Swift

```
lazy var comicTitle:UILabel? = {
    let label = UILabel()
    label.textAlignment = .Center
    return label
}()
```

lazy means don't run this code until the variable is first accessed

Since closures are functions, they can return stuff

Closures in Swift == Blocks in Objective-C, but not ugly

http://goshdarnblocksyntax.com/

What else? Named parameters

Objective-C

```
- (void)viewWillLayoutSubviews
{
   [super viewWillLayoutSubviews];

   self.scrollView.frame = CGRectMake(0, self.topLayoutGuide.length, 320, 400);
}
```

I usually write it like this so I won't lose track of the order

```
self.scrollView.frame = CGRectMake(0,
                                   self.topLayoutGuide.length,
                                   320,
                                   400
                                  );
```

What else? Named parameters

Swift

```
override func viewWillLayoutSubviews() {
    super.viewWillLayoutSubviews()

    imageScrollView!.frame = CGRect(x: 0, y: topLayoutGuide.length,
                                    width: 320, height: 400)
}
```

Parameters are named so you don't have to just remember them all the time

Is Swift better than Objective-C?

Any album recorded before the 90's was recorded on something like this...

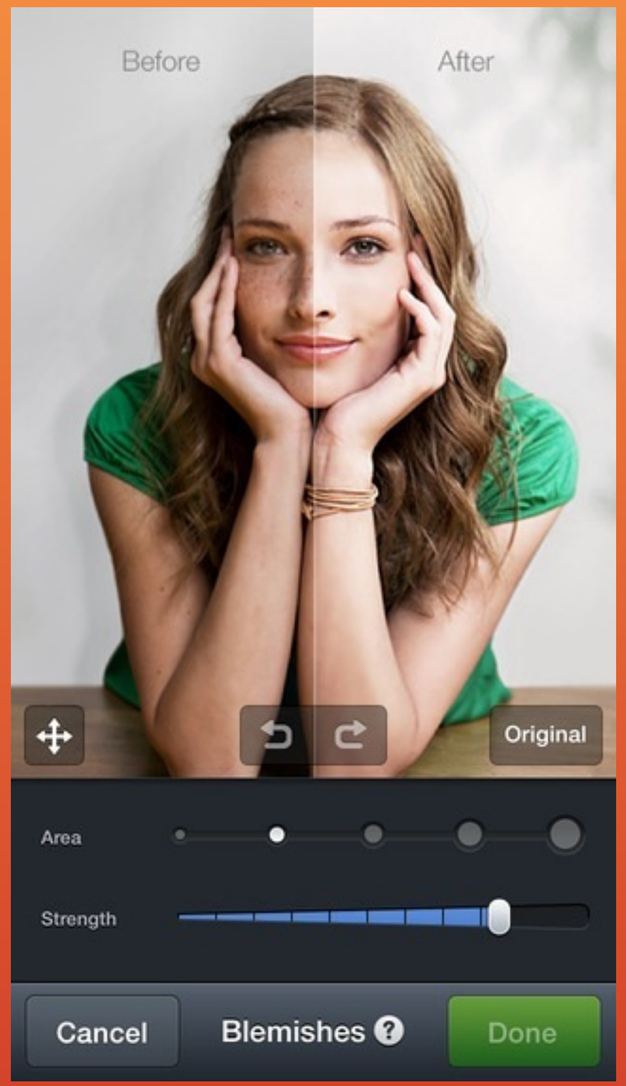
Is Swift better than Objective-C?

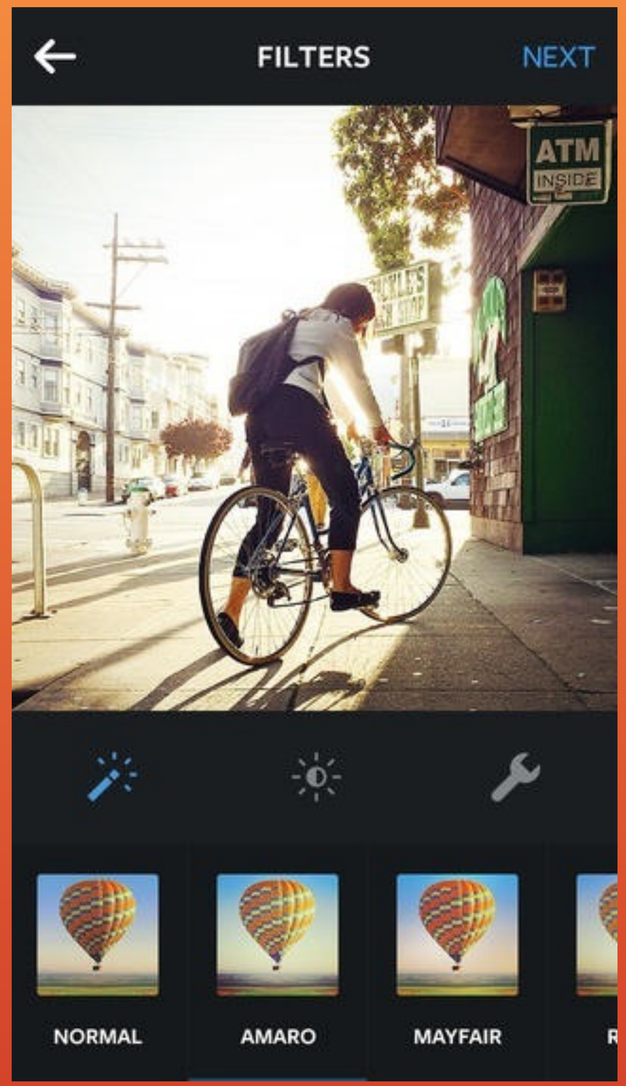
…and now albums are recorded on something like this

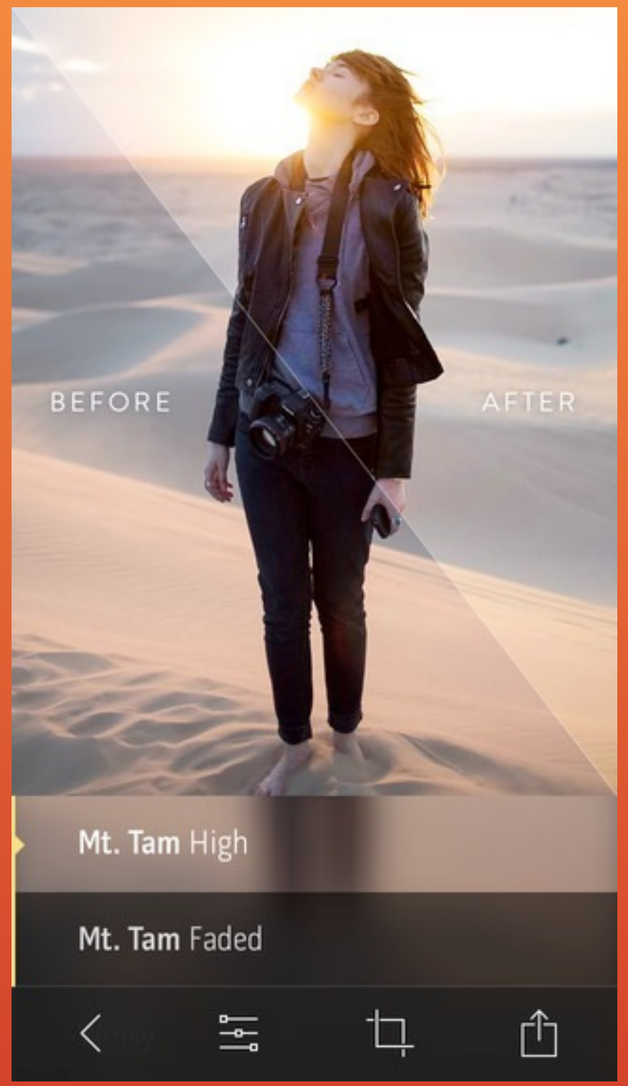
Is Swift better than Objective-C?

The first album recorded on this

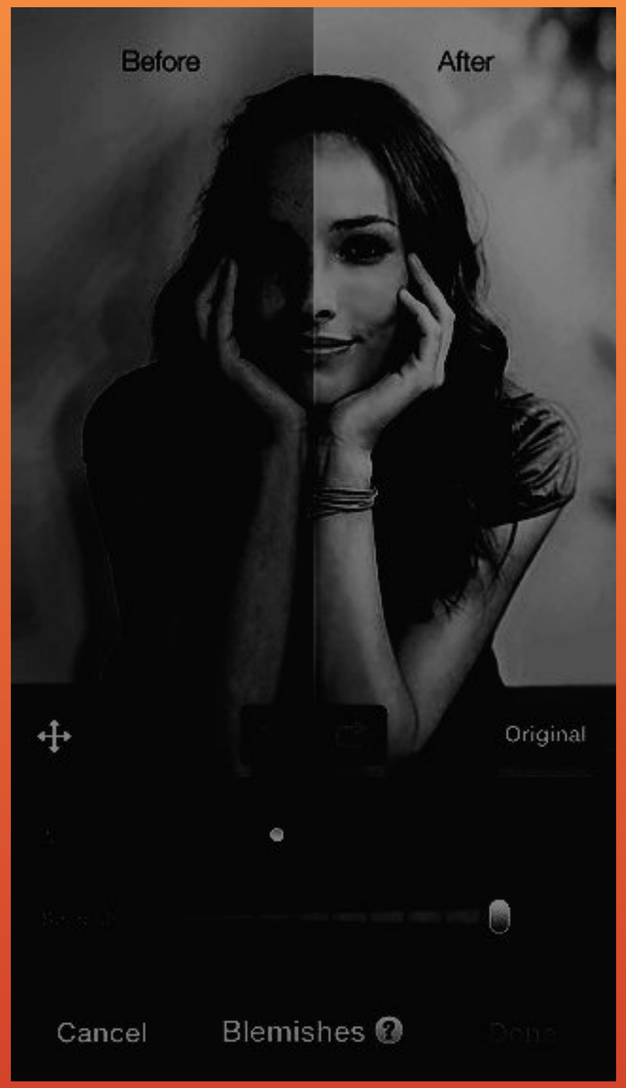
Which one is written in Swift?

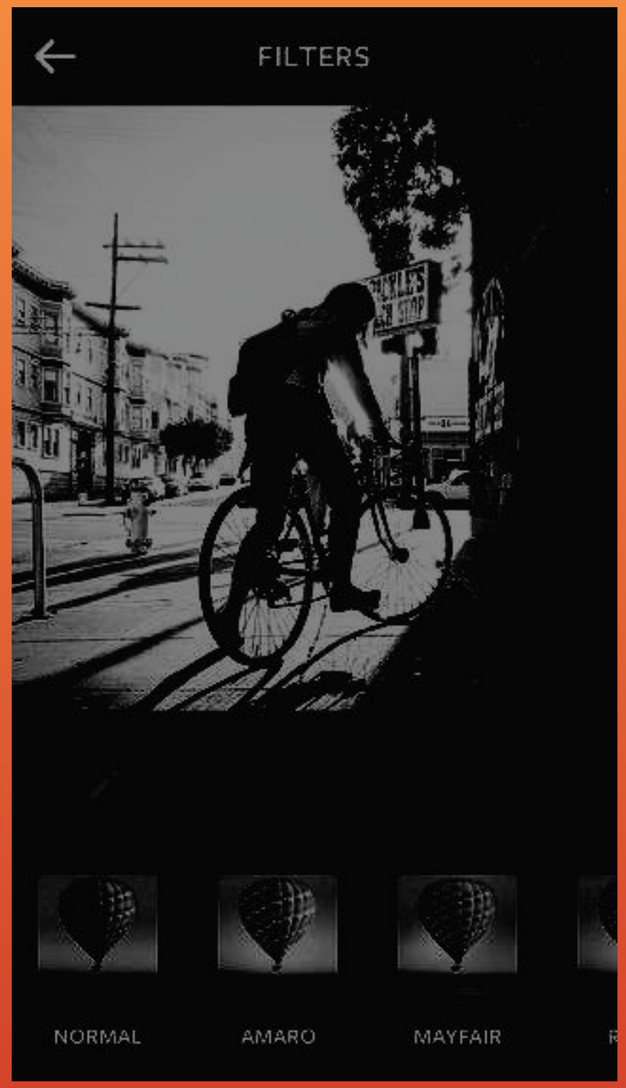


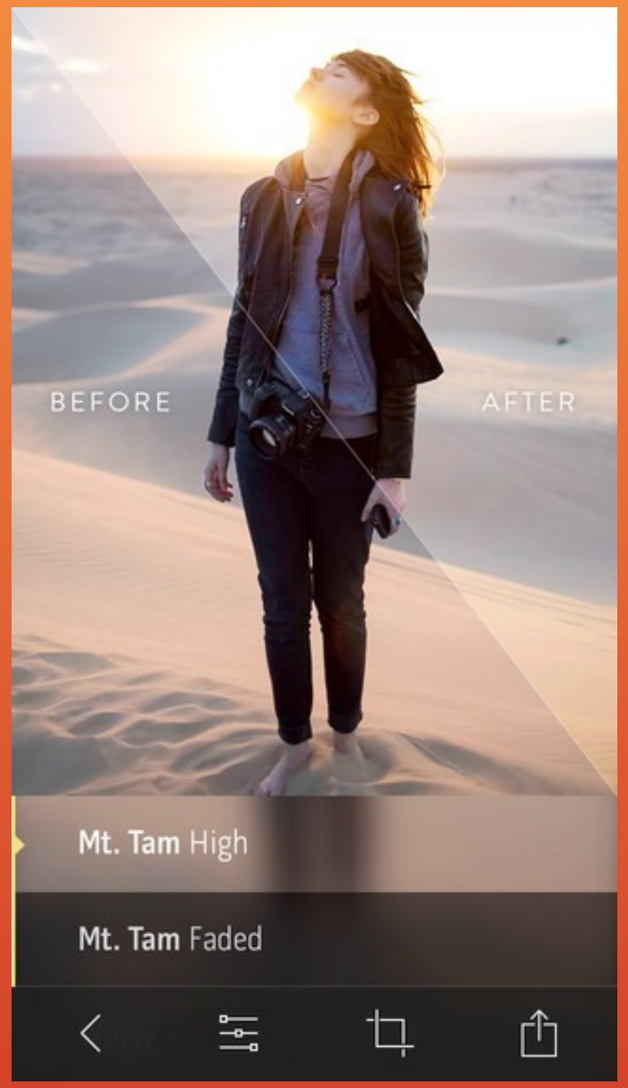




Litely by Sam Soffes







Swift was officially released September 17

Swift

```
let now:NSDate = NSDate().dateByAddingTimeInterval(0)

let swiftRelease:NSDate = {
    let releaseDate = NSDateComponents()
    releaseDate.month = 9
    releaseDate.day = 17
    releaseDate.year = 2014
    releaseDate.hour = 13

    return NSCalendar(
       calendarIdentifier: NSGregorianCalendar).dateFromComponents(releaseDate)!
}()

let daysFromRelease:NSDateComponents = {
    return NSCalendar(
       calendarIdentifier: NSGregorianCalendar).components(
       NSCalendarUnit.DayCalendarUnit, fromDate: swiftRelease,
       toDate: now, options: nil)
}()

println(daysFromRelease.day)
```

Make things

Have fun

Thanks!

@jonfriskics

Code School

LEARN BY DOING