Swift 101

Axel Rivera

AXEL RIVERA

Twitter

@arivera

Github

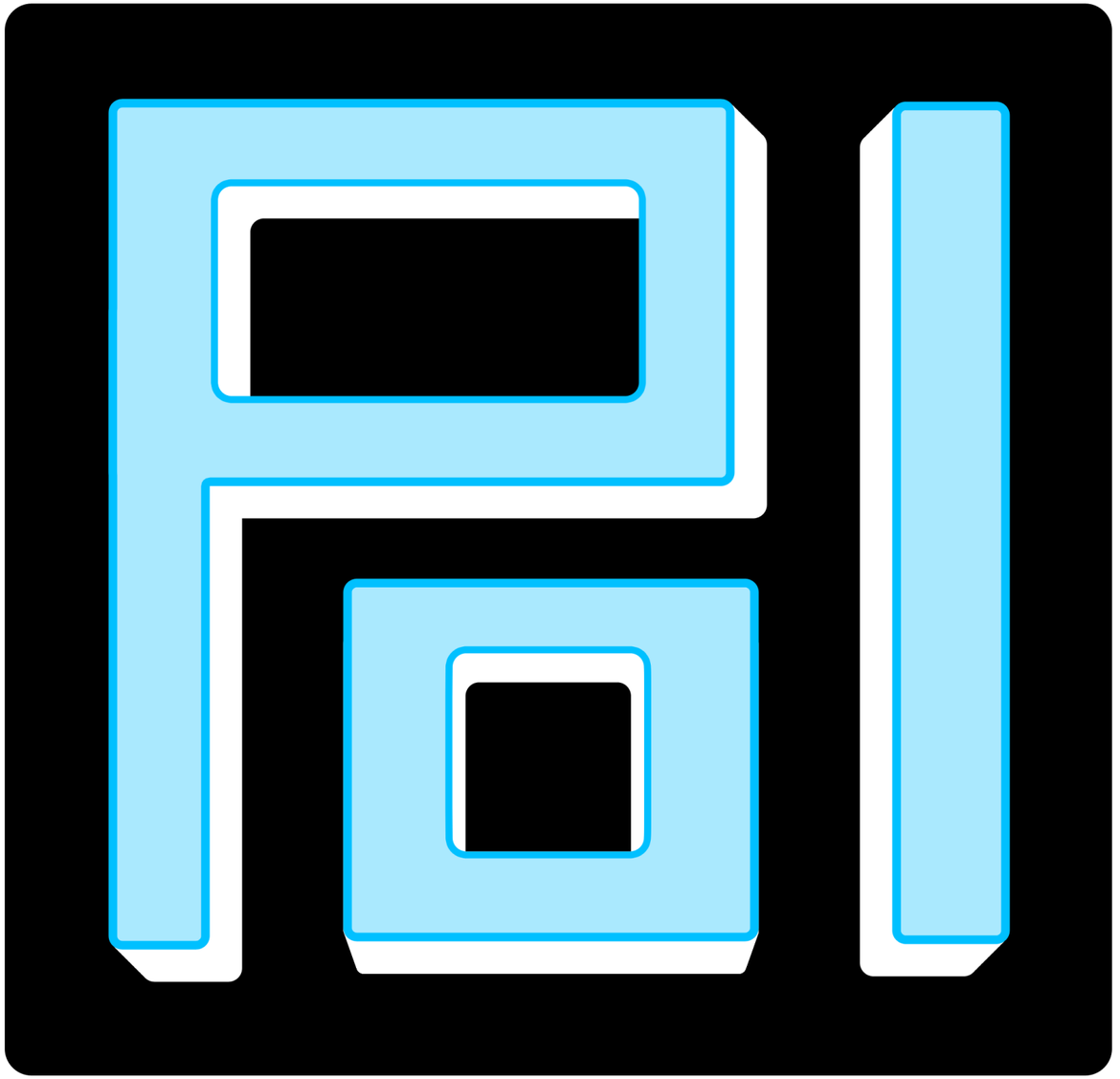
https://github.com/riveralabs/swift_workshop

FULL-TIME
IOS DEVELOPER (ISH)

ENTERPRISE

GOVERNMENT

OWN PRODUCTS

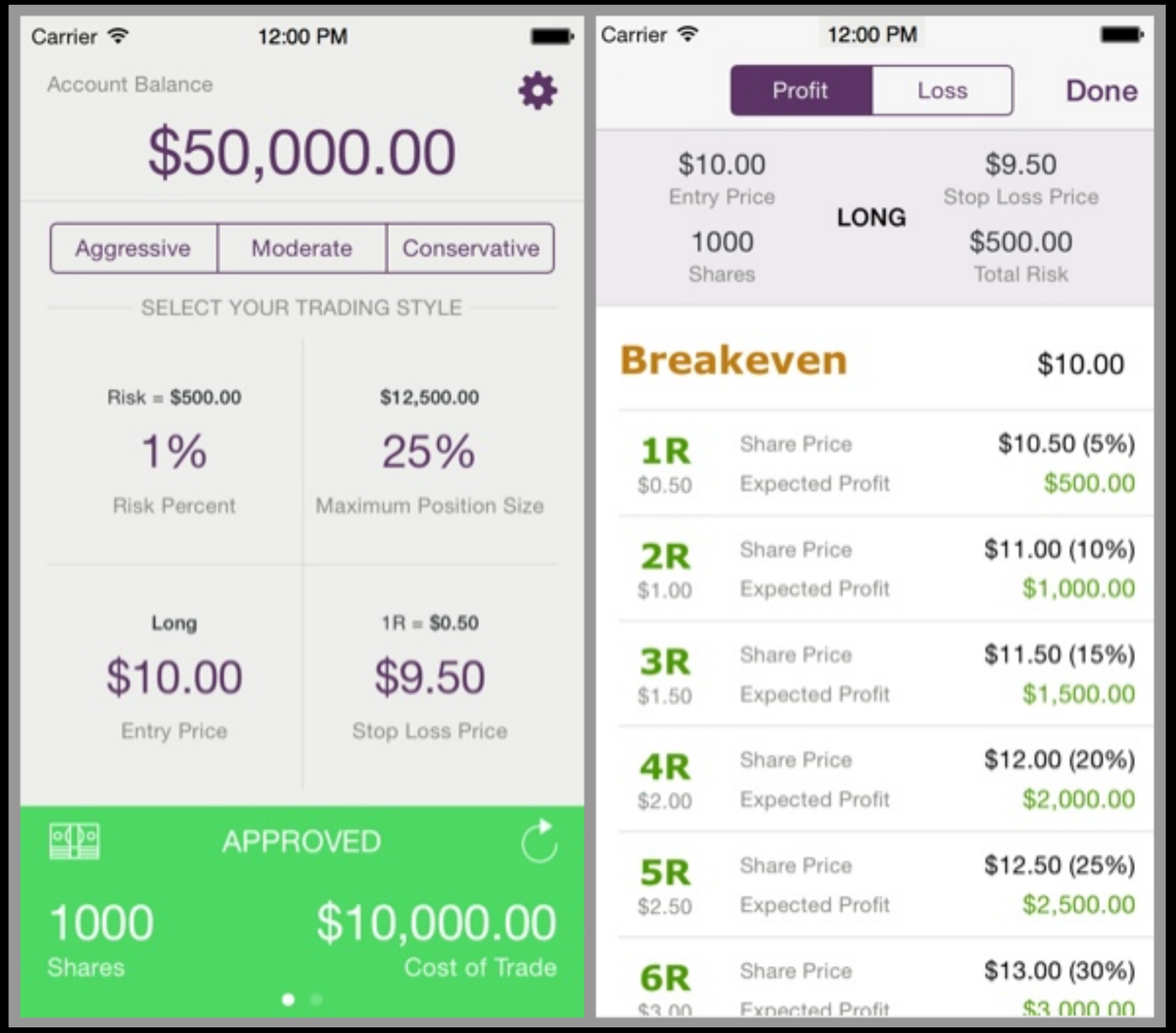


POLSENSE

INDIE IOS DEVELOPER

My First Swift App

Trade Size


Carrier
12:00 PM
Account Balance
$50,000.00
Aggressive
Moderate
Conservative
SELECT YOUR TRADING STYLE
Risk = $500.00
1%
Risk Percent
$12,500.00
25%
Maximum Position Size
Long
$10.00
Entry Price
1R = $0.50
$9.50
Stop Loss Price
APPROVED
1000
Shares
$10,000.00
Cost of Trade
Carrier
12:00 PM
Profit
Loss
Done
$10.00
Entry Price
LONG
$9.50
Stop Loss Price
1000
Shares
$500.00
Total Risk
Breakeven
$10.00
1R
$0.50
Share Price
Expected Profit
$10.50 (5%)
$500.00
2R
$1.00
Share Price
Expected Profit
$11.00 (10%)
$1,000.00
3R
$1.50
Share Price
Expected Profit
$11.50 (15%)
$1,500.00
4R
$2.00
Share Price
Expected Profit
$12.00 (20%)
$2,000.00
5R
$2.50
Share Price
Expected Profit
$12.50 (25%)
$2,500.00
6R
Share Price
$13.00 (30%)

TWO DAYS

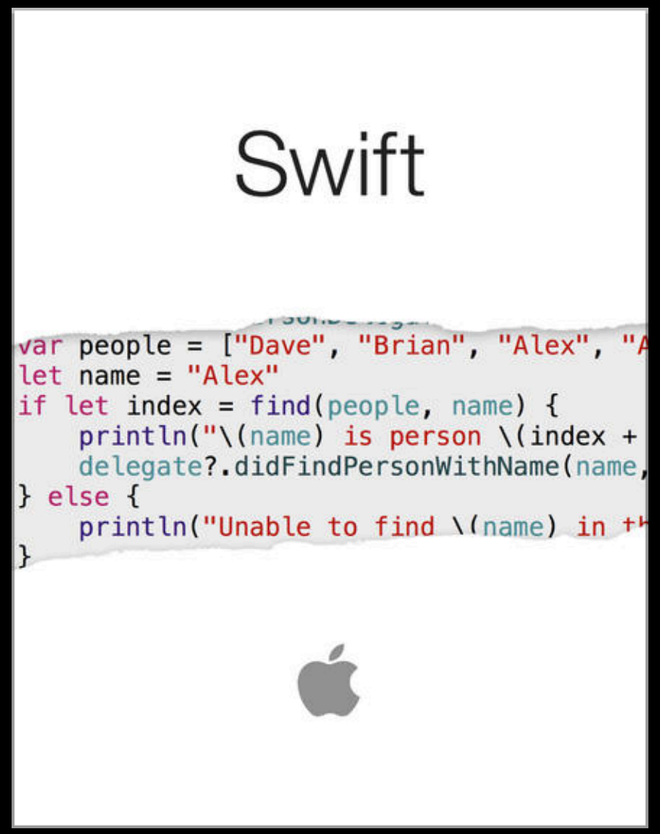
DAY 1

BASIC INTRODUCTION PLAYGROUNDS

DAY 2

SAMPLE STOCKS APP

The Swift Programming Language



Free on iBooks

```
println("Hello World!")
```

VARIABLES

```
var deviceName: String = "iPhone 6"
var systemVersion: Double = 8.0
var introduced: Int = 2014
var isAwesome: Bool = true
```

VARIABLES AND CONSTANTS

```
let deviceName: String = "iPhone 6"
var systemVersion: Double = 8.0
let introduced: Int = 2014
let isAwesome: Bool = true
```

TYPE INFERENCE

```
let deviceName = "iPhone 6"      // inferred as String
var systemVersion = 8.0          // inferred as Double
let introduced = 2014            // inferred as Int
let isAwesome = true             // inferred as Bool
```

STRING

```
let myString = "This is a string"
// inferred type String

urlRequest.HTTPMethod = "GET"

let components = "~/Documents/SomeUser".pathComponents
// ["~", "Documents", "SomeUser"]
```

STRING INTERPOLATION

```
let x = 2, y = 10

let result = "\(x) times \(y) is \(x * y)"
// 2 times 10 is 20
```

STRING MUTABILITY

```
var variableString = "For every action"
variableString += " there is a reaction"
// "For every action there is a reaction"

let constantString = "This will result"
constantString += " in an errror!"
// error - constantString cannot be changed
```

ARRAY AND DICTIONARY LITERALS

```
var stocks = ["AAPL", "GOOG", "FB", "TWTR"]

var numberOfWheels = ["car": 4, "bike": 2, "boat": 0]
```

ARRAY AND DICTIONARY WITH TYPES

```
var stocks: [String] = ["AAPL", "GOOG", "FB", "TWTR"]

var numberOfWheels: [String: Int] = ["car": 4, "bike": 2, "boat": 0]
```

MODIFYING ARRAYS

```
var stocks = ["AAPL", "GOOG", "FB"]
println(stocks[0])
// AAPL

stocks += "TWTR"
// ["AAPL", "GOOG", "FB", "TWTR"]

stocks += ["MSFT", "YHOO"]
// ["AAPL", "GOOG", "FB", "TWTR", "MSFT", "YHOO"]
```

MODIFYING DICTIONARIES

```
var prices = ["AAPL": 100.11, "GOOG": 573.36, "FB": 79.00]
println(prices["AAPL"])
// 100.11

prices["AAPL"] = 120.00
// ["AAPL": 120.00, "GOOG": 573.36, "FB": 79.00]
```

LOOPS

```
while awake {
    code()
}

for var index = 0; index < 10; index++ {
    println("The current index is \(index)")
}
```

FOR-IN

```
for character in "Hello" {
    println(character)
}

let numbersArray = ["One", "Two", "Three"]

for string in numbersArray {
    println("The current number is \(string)")
}

let numbersDictionary ["one": 1, "two": 2, "three": 3]

for (key, value) in numbersDictionary {
    println("The current key is \(key) and value \(value)")
}
```

FOR-IN: RANGES

```
for number in 1...5 {
    println("\(number) x 2 = \(number * 2)")
}
```

1 x 2 = 2
2 x 2 = 4
3 x 2 = 6
4 x 2 = 8
5 x 2 = 10

FOR-IN: RANGES

```
for number in 1..<5 {
    println("\(number) x 2 = \(number * 2)")
}
```

1 x 2 = 2
2 x 2 = 4
3 x 2 = 6
4 x 2 = 8

OPTIONALS

```
let stocks = ["AAPL": 100.11, "GOOG": 573.36, "FB": 79.00]
let possiblePrice: Double? = stocks["ORCL"]

if possiblePrice == nil {
    println("Possible Price for Oracle wasn't found")
} else {
    let price = possiblePrice!
    println("Oracle's Price is \(price)")
}

// there's a better way

if let price = possiblePrice {
    println("Oracle's Price is \(price)")
}
```

IF STATEMENTS

```
if numberOfLegs == 0 {
    println("It slides")
} else if numberOfLegs = 1 {
    println("It hops")
} else {
    println("It walks")
}
```

SWITCH

```
switch numberOfLegs {
    case 0:
        println("It slides")

    case 1:
        println("It hops")

    default:

        println("It walks")
}
```

FUNCTIONS

```
// Basic Function
func sayHello() {
    println("Hello World!")
}

// With Parameter
func sayHello(name: String) {
    println("Hello \(name)!")
}
```

FUNCTIONS

```
// With Default Value
func sayHello(name: String = "World") {
    println("Hello \(name)!")
}

// With Return Value
func greeting(name: String = "World") -> String {
    return "Hello \(name)!"
}
```

FUNCTIONS ARE FIRST CLASS CITIZENS

CLOSURES

```
let helloPrinter = {
    println("Hello World!")
}

// Long Version

let helloPrinter: () -> () = {
    println("Hello World!")
}
```

CLOSURES AND PARAMETERS

```
func repeat(count: Int, task:() -> ()) {
    for i in 0...count {
        task()
    }
}

repeat(2, {
    println("Hello World!")
})
```

Hello World!
Hello World!

TRAILING CLOSURES

```
func repeat(count: Int, task:() -> ()) {
    for i in 0..count {
        task()
    }
}

repeat(2) {
    println("Hello World!")
}
```

Hello World!
Hello World!

CLASSES

```
class Shape {
    // properties
    // methods
    // initializers
}
```

CLASS INHERITANCE

```
class Shape {

}

class Square: Shape {

}
```

CLASS PROPERTIES

```
class Shape {
    // Stored Property
    var numberOfSides = 0

    // Computed Property
    var description: String {
        return "number of sides \(numberOfSides)"
    }
}
```

INITIALIZER SYNTAX

```
class Shape {
    var numberOfSides = 0

    var description: String {
        return "number of sides \(numberOfSides)"
    }
}

let myShape = Shape()

// Dot Syntax

myShape.numberOfSides = 4

println(myShape)
// number of sides 4
```

CLASS INITIALIZATION

```
class Square: Shape {
    init() {
        super.init()
        numberOfSides = 4
    }
}

let mySquare = Square()
println(mySquare.description)
```

OVERRIDING A PROPERTY

```
class Square: Shape {

    init() {
        super.init()
        numberOfSides = 4
    }

    override var description: String {
        return "a square has \(numberOfSides) sides"
    }
}

let mySquare = Triangle()
println(mySquare.description)
```

METHODS

```
class Square: Shape {
    var sideLength = 0.0

    init() {
        super.init()
        numberOfSides = 4
    }

    override var description: String {
        return "a square has \(numberOfSides) sides"
    }

    func area() -> Double {
        return sideLength * sideLength
    }
}
```

STRUCTURES

```
struct Rect {
    var origin: Point
    var size: Size

    var area: Double {
        return size.width * size.height
    }
}
```

CLASS VS STRUCT
WHAT'S THE DIFFERENCE?

CLASSES ARE PASSED BY REFERENCE

STRUCTURES ARE PASSED BY VALUE

ENUMERATIONS

```
enum SecurityType {
    case Stock, MutualFund, ETF, Bond
}

var asset = SecurityType.Stock
// asset is inferred to be SecurityType
asset = .MutualFund
```

ENUMERATIONS ARE ADVANCED

Properties

Associated Values

Initializers

Read More in Swift Book

EXTENSIONS

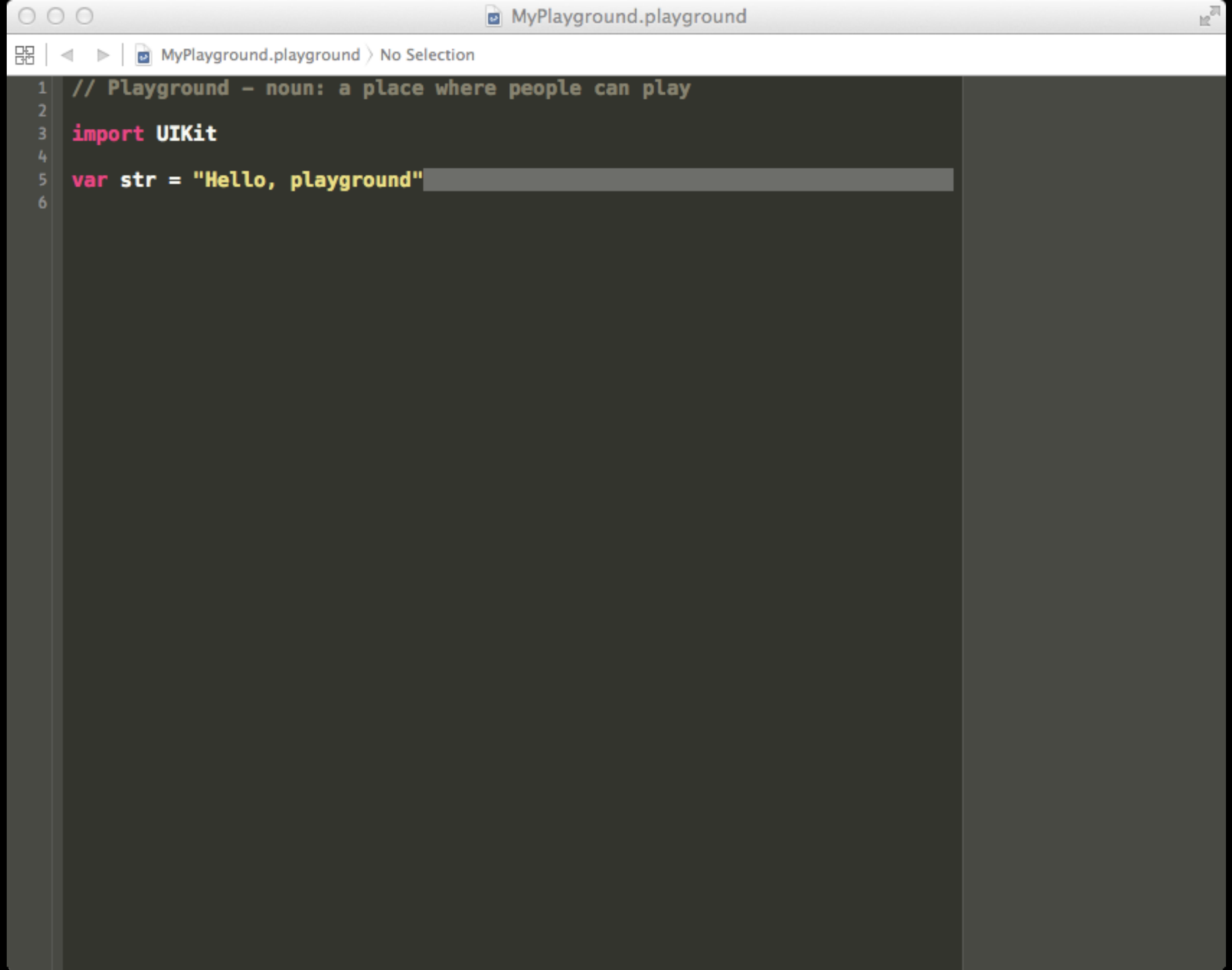
```
extension Int {
    func times(task: () -> ()) {
        for i in 1..self {
            task()
        }
    }
}

100.times {
    println("Hello World!")
}
```

ADVANCED TOPICS

GENERICS

PROTOCOLS

PLAYGROUNDS


MyPlayground.playground
MyPlayground.playground
No Selection
1 // Playground - noun: a place where people can play
2
3 import UIKit
4
5 var str = "Hello, playground"
6

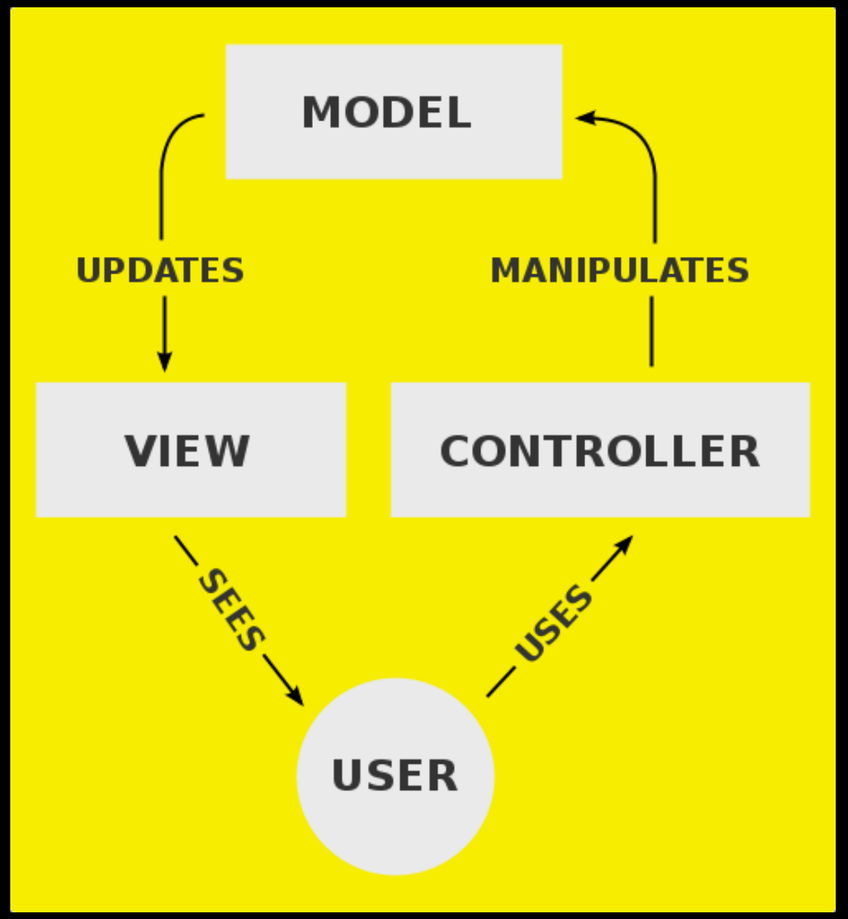
FOUNDATION

+

UIKIT

SWIFT USES OBJECTIVE-C LIBRARIES

MODEL-VIEW-CONTROLLER (MVC)



DEMO

DAY 2

SAMPLE STOCKS APP

STORYBOARDS

VS

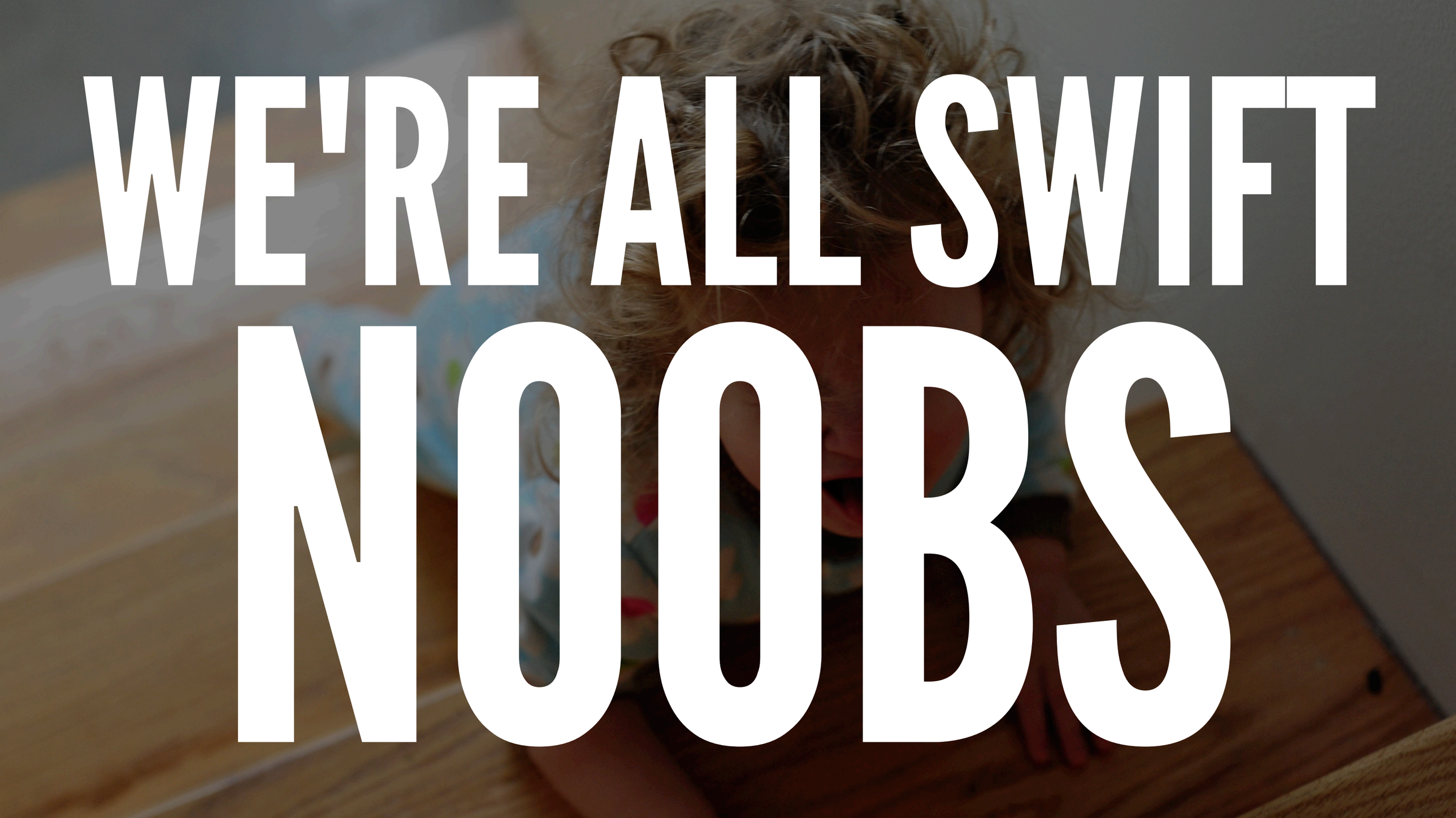
CODE

BOTH APPS ARE IDENTICAL (ISH)

BASED ON COMMON PRACTICES

THERE'S NO PERFECT WAY TO SWIFT

YET!!


WE'RE ALL SWIFT
NOOBS

NETWORKING

ALAMOFIRE

https://github.com/Alamofire/Alamofire

JSON

JSON-SWIFT

https://github.com/owensd/json-swift

AUTOLAYOUT

PURELAYOUT

https://github.com/smileyborg/PureLayout

WHICH IS BETTER STORYBOARDS OR CODE?

WHAT'S THE BEST CAR?

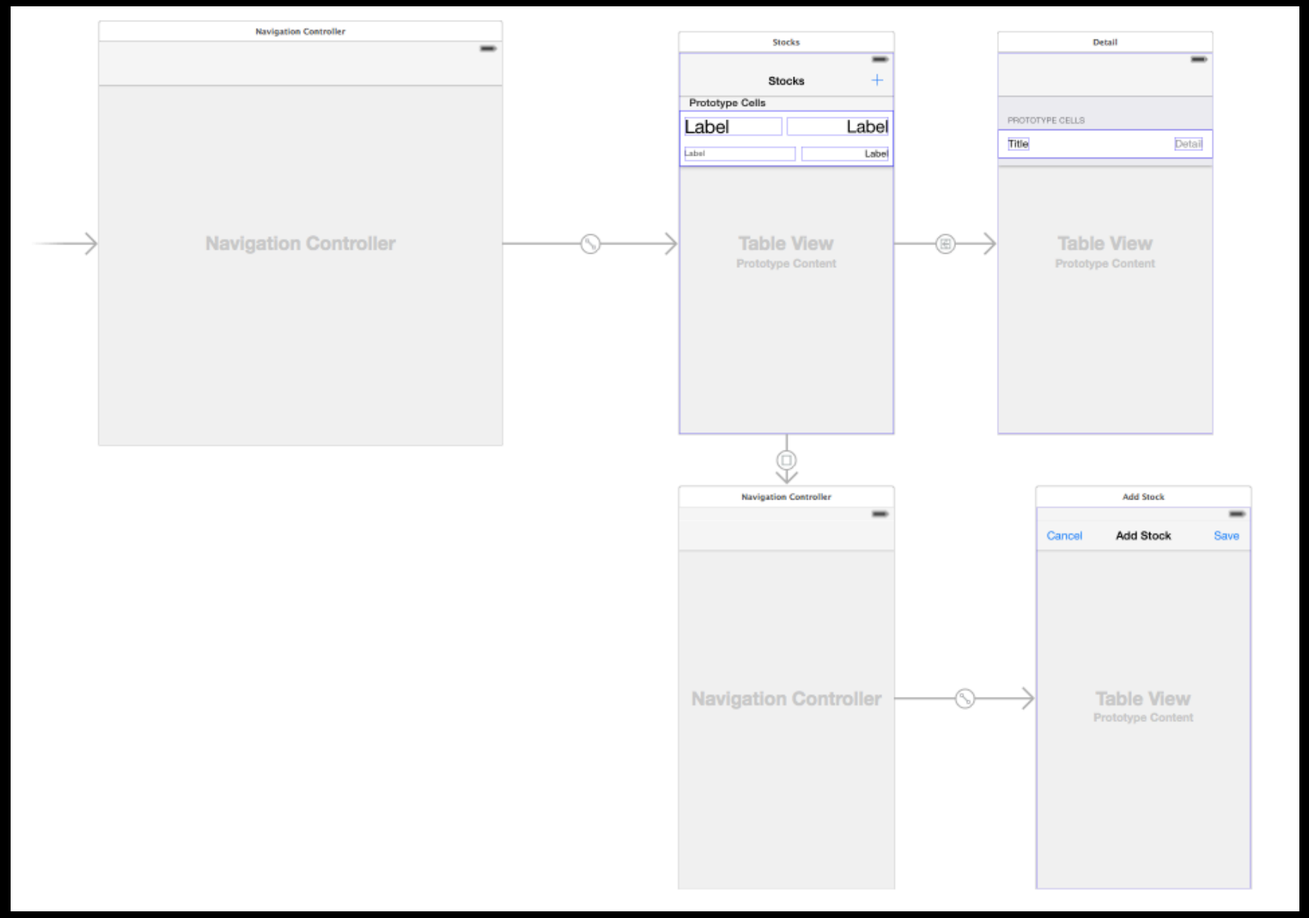
What's your budget?

How many passengers?

Fuel consumption?

Like Sports Cars?

STORYBOARDS



STORYBOARDS

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document type="com.apple.InterfaceBuilder3.CocoaTouch.XIB" version="3.0" toolsVersion="6214">
    <dependencies>
        <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="6207"/>
        <capability name="Constraints with non-1.0 multipliers" minToolsVersion="5.1"/>
    </dependencies>
    <objects>
        <placeholder placeholderIdentifier="IBFilesOwner" id="-1" userLabel="File's Owner"/>
        <view contentMode="scaleToFill" id="iN0-l3-epB">
            <rect key="frame" x="0.0" y="0.0" width="480" height="480"/>
        </view>
    </objects>
</document>
```

STORYBOARDS

PRO: PERFORMANCE

STORYBOARDS

CON: MERGE CONFLICTS

STORYBOARDS

PRO: PROTOTYPES

STORYBOARDS

CON: REUSABILITY

STORYBOARDS

CON: DATA FLOW

```
init(ticker: Ticker) {
    super.init(nibName: nil, bundle: nil)
    self.title = ticker.symbol
    self.dataSource = ticker.toArray()
}
```

```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
    if segue.identifier == "detail" {
        var detailController = segue.destinationViewController as DetailViewController

        var indexPath = self.tableView.indexPathForSelectedRow()

        if let tmp = indexPath {
            var ticker = tickers[tmp.row]
            detailController.dataSource = ticker.toArray()
        }
    }
}
```

CODE

VIEWS

```
import UIKit

class MyView: UIView {

    var subview: UIView!

    override required init(frame: CGRect) {
        super.init(frame: frame)

        subview = UIView(frame: CGRectZero)
        subview.setTranslatesAutoresizingMaskIntoConstraints(false)
        self.addSubview(subview)
    }

    override func updateConstraints() {
        subview.autoPinEdgeToSuperviewEdges(UIEdgeInsetsZero)
        super.updateConstraints()
    }
}
```

CONTROLLERS

```
import UIKit

class MyViewController: UIViewController {

    required init(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override required init() {
        super.init(nibName: nil, bundle: nil)
    }

    override func loadView() {
        self.view = UIView(frame: UIScreen.mainScreen().bounds)
    }

}
```

CODE

PRO: UNDER THE HOOD

CODE

PRO: WHEN CODE IS THE ONLY OPTION

CODE

PRO: MERGE CONFLICTS

CODE

CON: PROTOTYPING

CODE

CON: REFACTORING

CODE
PRO: PERFORMANCE

CODE
PRO: REUSABILITY

RESOURCES

RAY WENDERLICH

http://www.raywenderlich.com/tutorials

More than 15 Swift tutorials FREE

Swift by Tutorials Bundle

Learn iOS 8 and Swift!

Three brand new books to get you up-to-speed with Swift, iOS 8 and Xcode 6 – for intermediate to advanced developers.

Swift by Tutorials [Available Now] gets you started with Swift the quick and easy way – via hands-on tutorials.

iOS 8 by Tutorials [Available Now] introduces you to the latest iOS 8 APIs and technologies such as Adaptive UI, App Extensions and Cloud Kit.

Core Data by Tutorials [Available 10/15] shows you how to take control of your app's data – with hands-on examples in Swift.

~~$162~~ **$149** ADD TO CART

http://www.raywenderlich.com/store/swift-tutorials-bundle

OFFICIAL SWIFT BLOG

https://developer.apple.com/swift/blog/

THE END

Twitter

@arivera

Github

https://github.com/riveralabs/swift_workshop