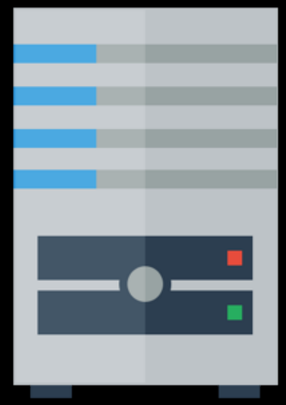
iOS and You

Web and mobile together!



RoR, Node.JS,
Python, .Net

back end
server

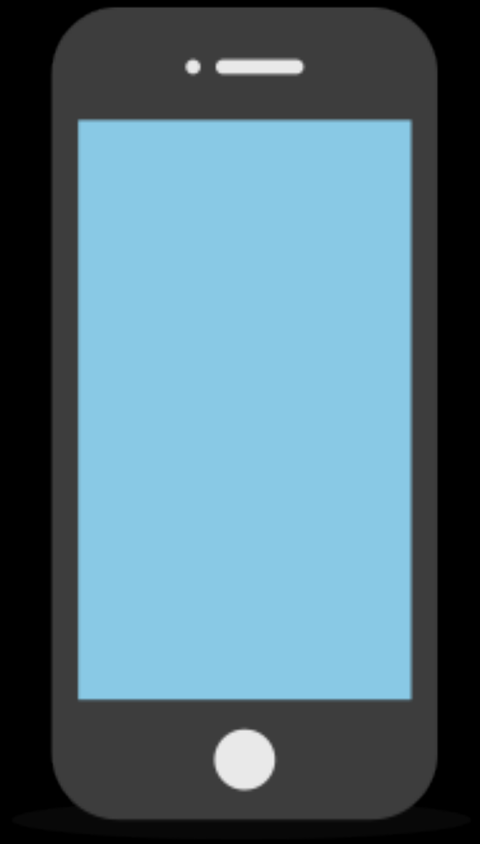
HTML+CSS
+JS

web front end

Swift(New!), Obj-C,
Java(Android)

mobile app

iOS development pros

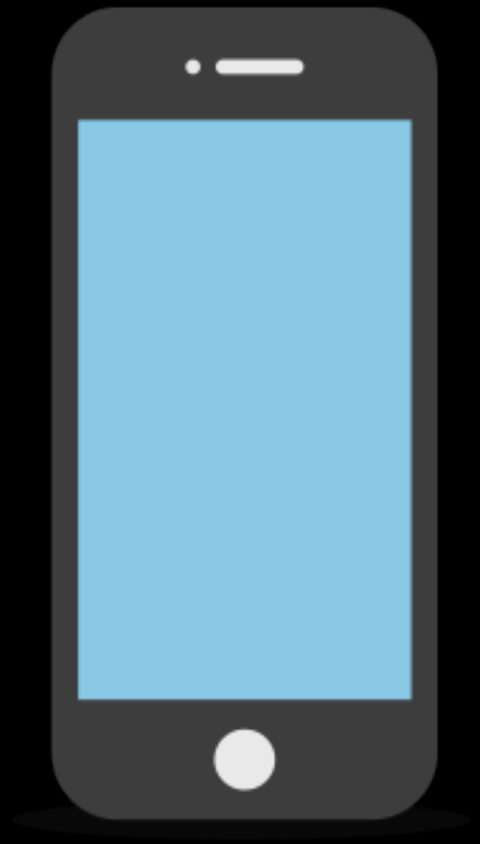


- Developing a native app gives you:
  - Full access to the device's hardware, like its GPU (for super fast and responsive controls), motion censors (detecting steps, rotations, etc), and local disk.
  - Standardized full feature SDK and IDE from Apple.
  - Your entire mobile app is in one programming language
  - App deployment and monetization through the app store.

iOS development cons:

NONE, YA HATERZ

iOS development cons

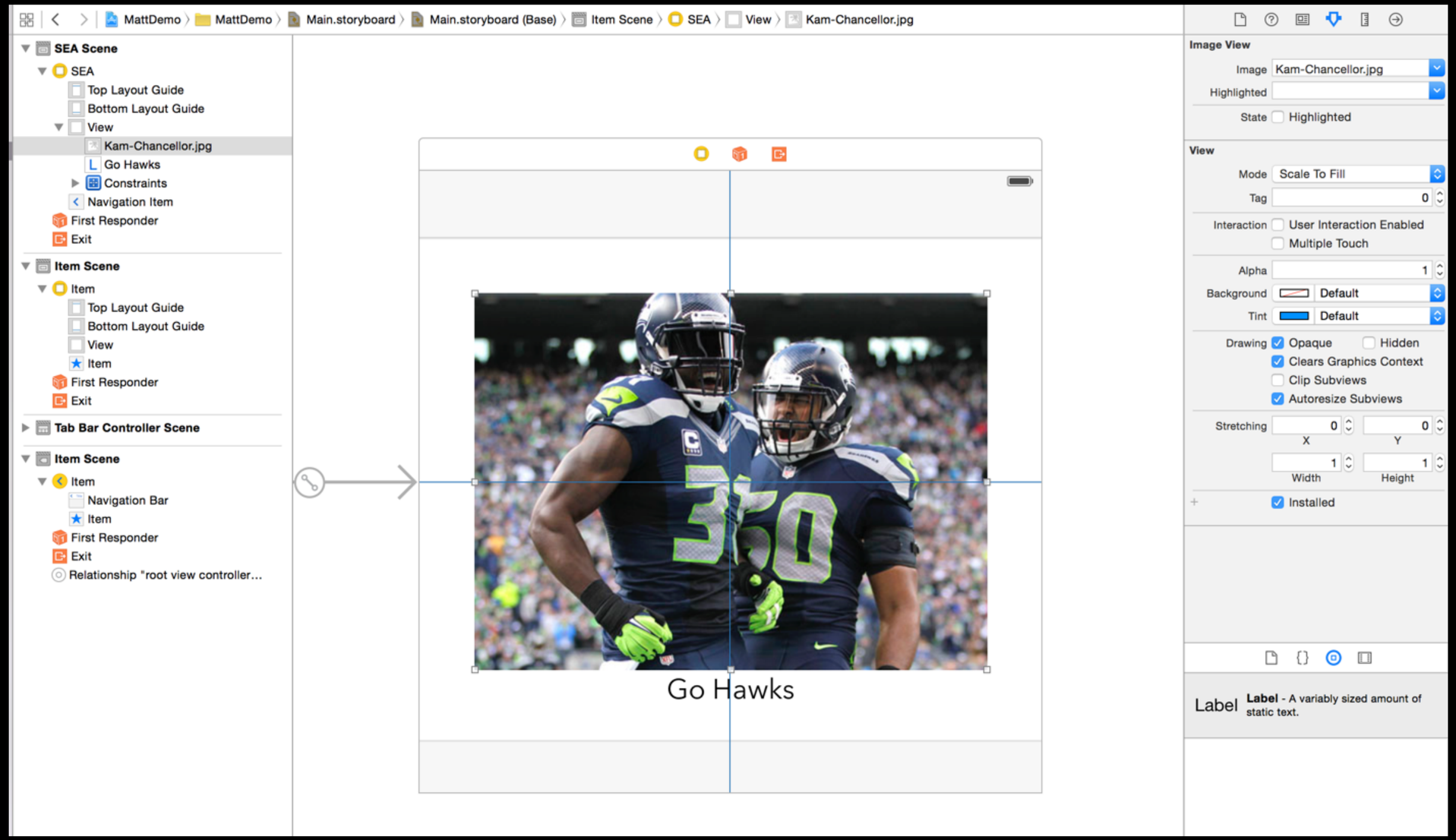


- Your app can only be used by people with iOS devices
- Apple takes a 30% cut of all sales
- Making it as an indie dev is incredibly hard

So how do we make iOS apps?

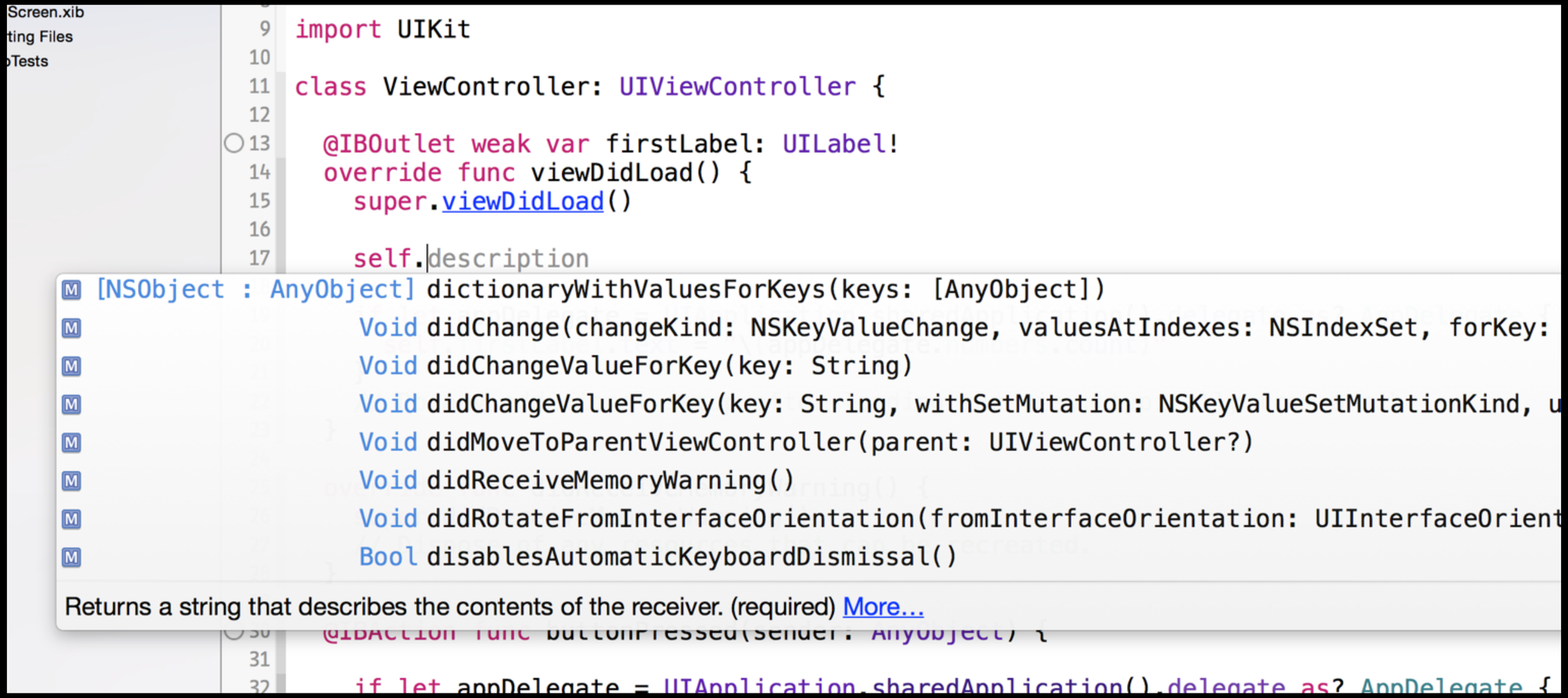
- Well, with Xcode!
- Available for free in the mac app store and also online at apple's developer portal
- First released in 2003
- It's pretty amazing
- Some of its best features are…

Storyboards



No more yucky HTML

Autocomplete



Compile time code checking

```
16
17  override func viewDidLoad() {
18    super.viewDidLoad()
19
20    doSomethingWithString(32)        ! Cannot invoke 'doSomethingWithString' with an argument list of type '(Int)'
21  }
22
23  func doSomethingWithString(input : String) {
24    println(input)
25  }
```

Xcode Tour

Swift vs Objective-C

- Introduced in June of 2014 at WWDC
- Strong static typing (the opposite of JavaScript)
- Encourages Functional programming

- Introduced in 1983
- Weak static typing
- Since its just an OOP layer built on top of C, you can write C and C++ code side by side with your Objective-C

Swift & JavaScript

- Swift actually looks similar to JavaScript
- Declaring a variable:

```
var myName = “Brad”
```

- Calling a function:

```
someFunction()
```

- Even creating an object (which we call a dictionary in iOS):

```
var teams = {“football” : “Seahawks”, “soccer” : “Sounders”, “baseball” : “Not the mariners because they frickin suck”}
```

Type System

- The big difference between JavaScript and Swift (and also Objective-C), is their type systems.
- Swift is statically and strongly typed.
- This means every variable and parameter must be given a type when it is declared, so the compiler can check it before run time. This helps avoid many bugs.
- For example, Xcode will yell at you if you try to pass in a string to a function that has a parameter of type Int.
- Lets take a look at Swift in an Xcode playground

Swift Playground Tour

Cocoa & Cocoa Touch

- Cocoa & Cocoa Touch are a suite of frameworks you use to create both iOS and OS X apps.
- They all come installed with Xcode ready to use out of the box
- Some of my favorites are:
  - UIKit - Contains everything you need to make amazing interfaces
  - MapKit - provides an interface for embedding and working with maps in your app
  - CoreData - data persistence and object graph management. uses SQLite under the hood by default
  - SpriteKit - Apple's very own 2D game development framework
- Every year in June Apple updates and releases new frameworks at WWDC, it's pretty much Christmas for us Cocoa developers.

App Tour

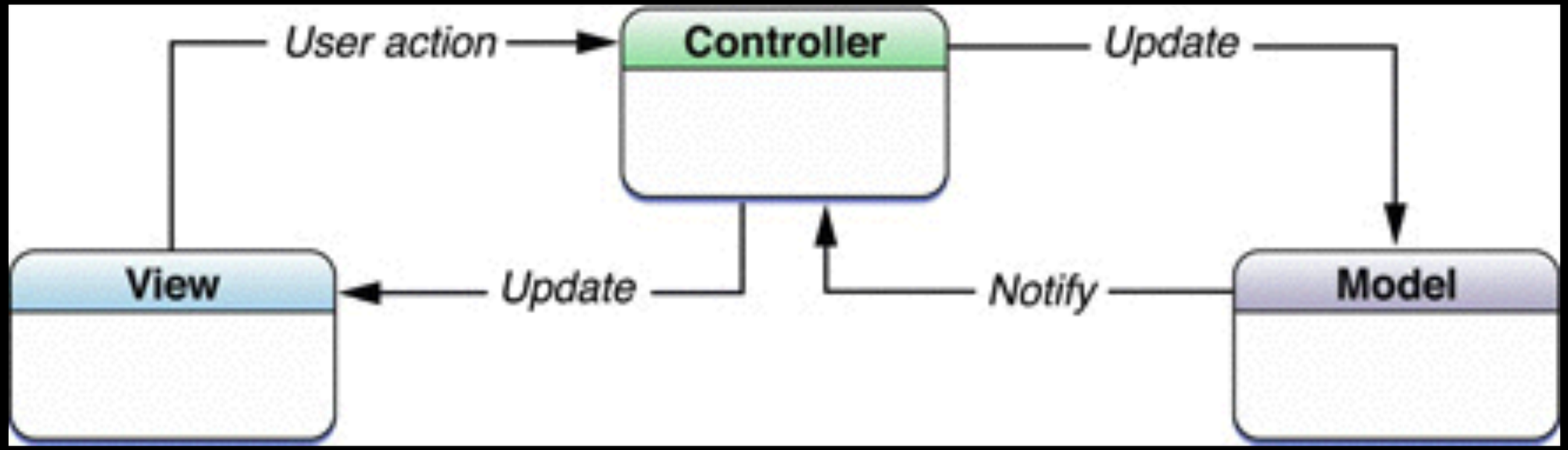
MVC

- Model View Controller, or MVC for short, is a design pattern that is championed by Apple.
- But MVC is not just mobile, web apps use MVC as well!
- JS has many 3rd party frameworks that use MVC or MV*: angular, ember, backbone, and many more.
- So what the heck is MVC?

So what is MVC?

- MVC is the separation of **M**odel, **V**iew, and **C**ontroller.
- It is a separation of concerns for your code. Being able to separate out these components makes your code easier to read, write, re-use, test, think about, and discuss.
- In addition to **separation**, we will see MVC is also all about **communication**.
- Every entity in your app is assigned to one of three roles: model, view, or controller.
- The **Model layer** is the data of your app, the **View layer** is anything the user sees and interacts with, and the **Controller layer** mediates between the two.

MVC in action



Model Layer

- Model objects encapsulate data and logic that are going to be used by your application.
- The Twitter App has a Tweet model class, a User model class, a Favorite model class, etc.

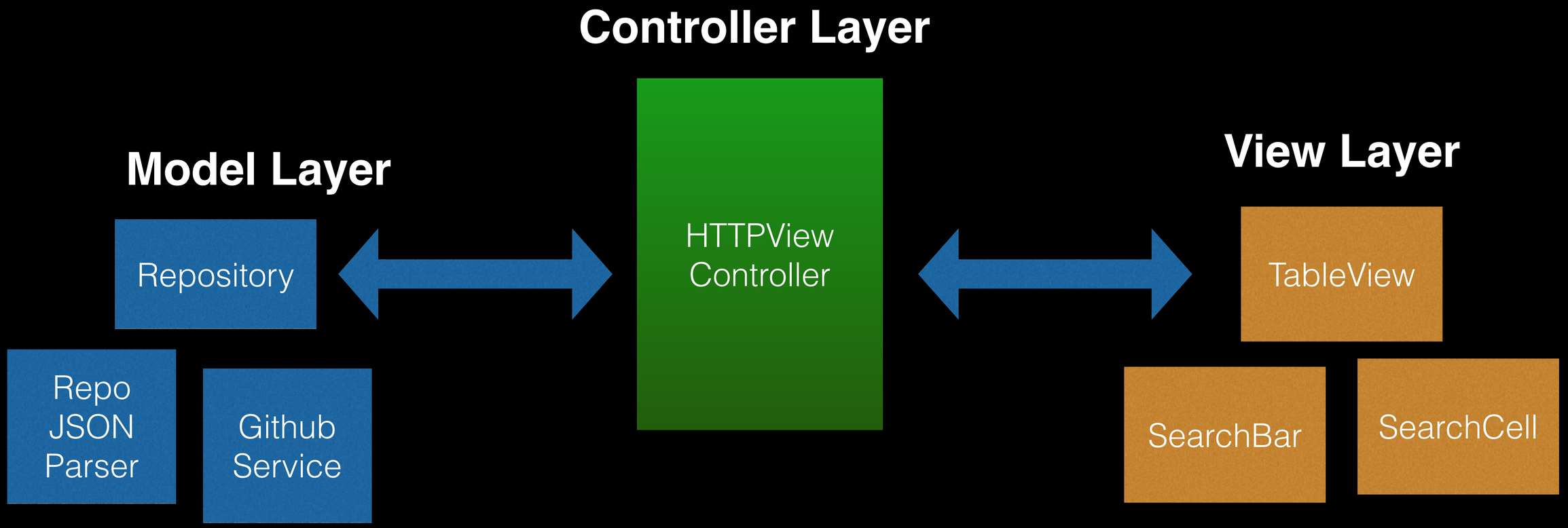
View Layer

- A View object is an object the user can see and possibly interact with.
- Enables displaying and editing of the app's data.
- Communication between the View and Model layers is made possible by…..

Controller Layer

- Act as the intermediary between the model layer and view layer.
- The most common form of a controller in iOS is a view controller.
- At first your view controllers will have a lot of code. Eventually you should strive to make them lighter so its easier to understand what they are doing at a glance.

MVC and our HTTP Demo



MVC Demo

Where to get started?

- Take my Foundations II iOS Course (one starts in September)
- Check out http://www.raywenderlich.com for great tutorials and books (I'm a staff member there!)
- Stanford iOS course on iTunes U (pretty hard)
- Code School/ Code Academy/ Tree house/ Udemy