How to Get Started with Swift in 2020

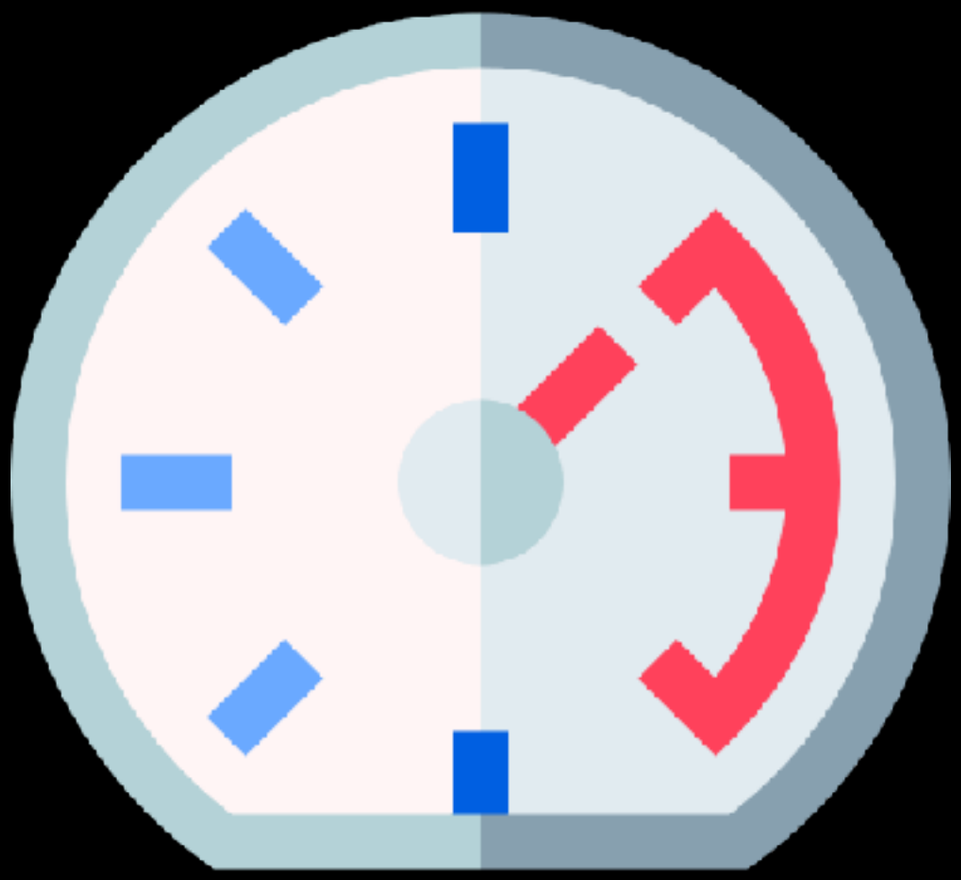
Leo Dion
@leogdion
BrightDigit
leo@brightdigit.com

About Leo

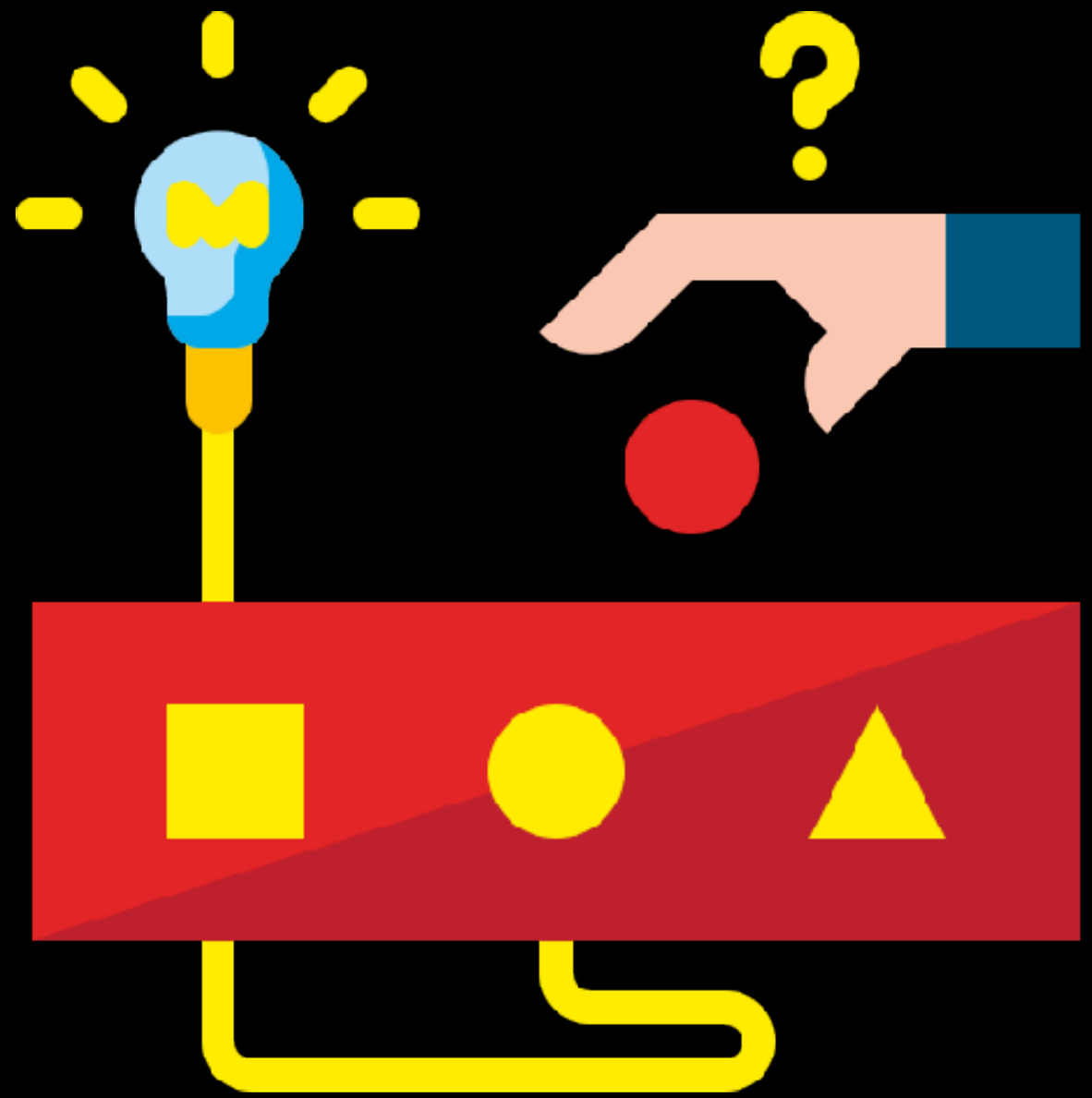
- iOS since 2011
- Swift since 2014
- Building Apps For:
  watchOS, macOS, Vapor
- Run
  brightdigit.com
- Podcast
  okproductive.com
  empowerapps.show
- Learning Swift
  learningswift.brightdigit.com

@leogdion

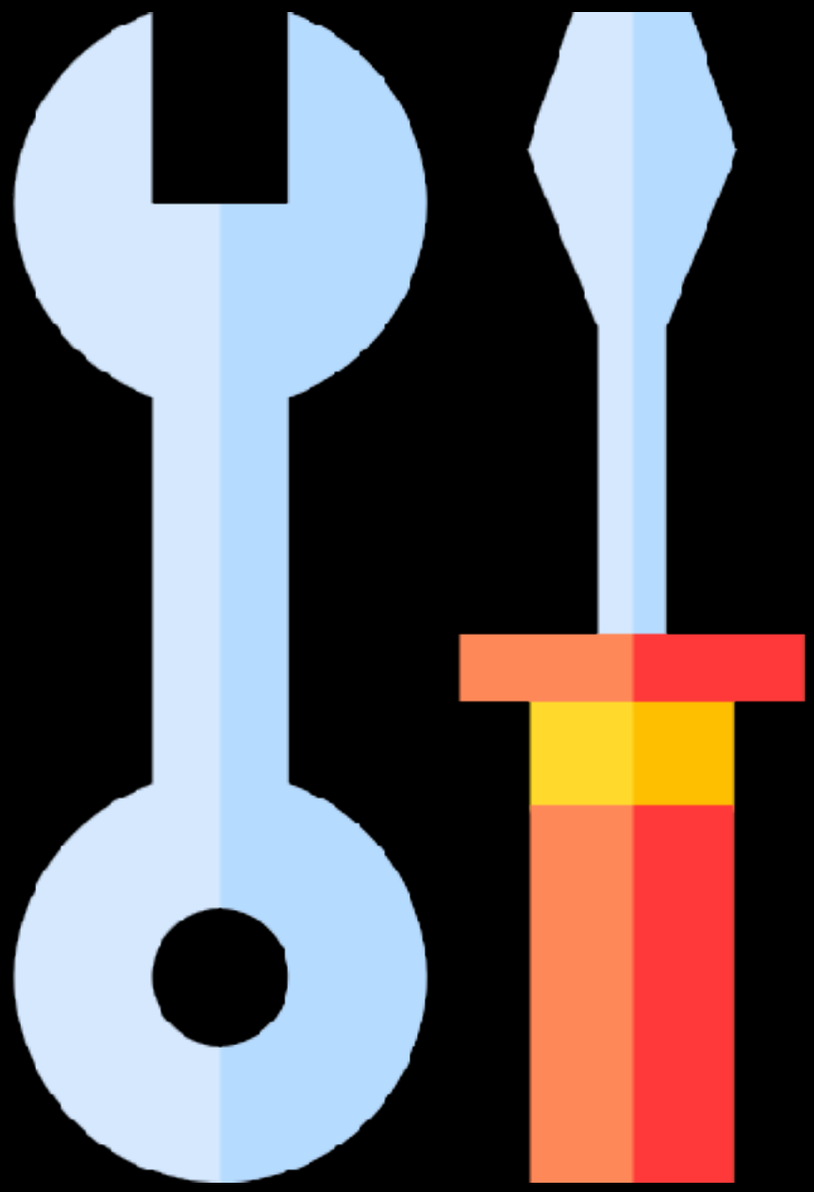
Why Swift?



Performance



User Experience



Maintainability

LET'S GET STARTED.

Hardware


iPad

Mac Mini

iMac

MacBook Pro

Developer Account

Software

Xcode

Choose a template for your new project:

iOS watchOS tvOS macOS Cross-platform

Filter

Application

Single View App

Game

Augmented Reality App

Document Based App

Master-Detail App

Tabbed App

Sticker Pack App

iMessage App

Projects

Framework & Library

Framework

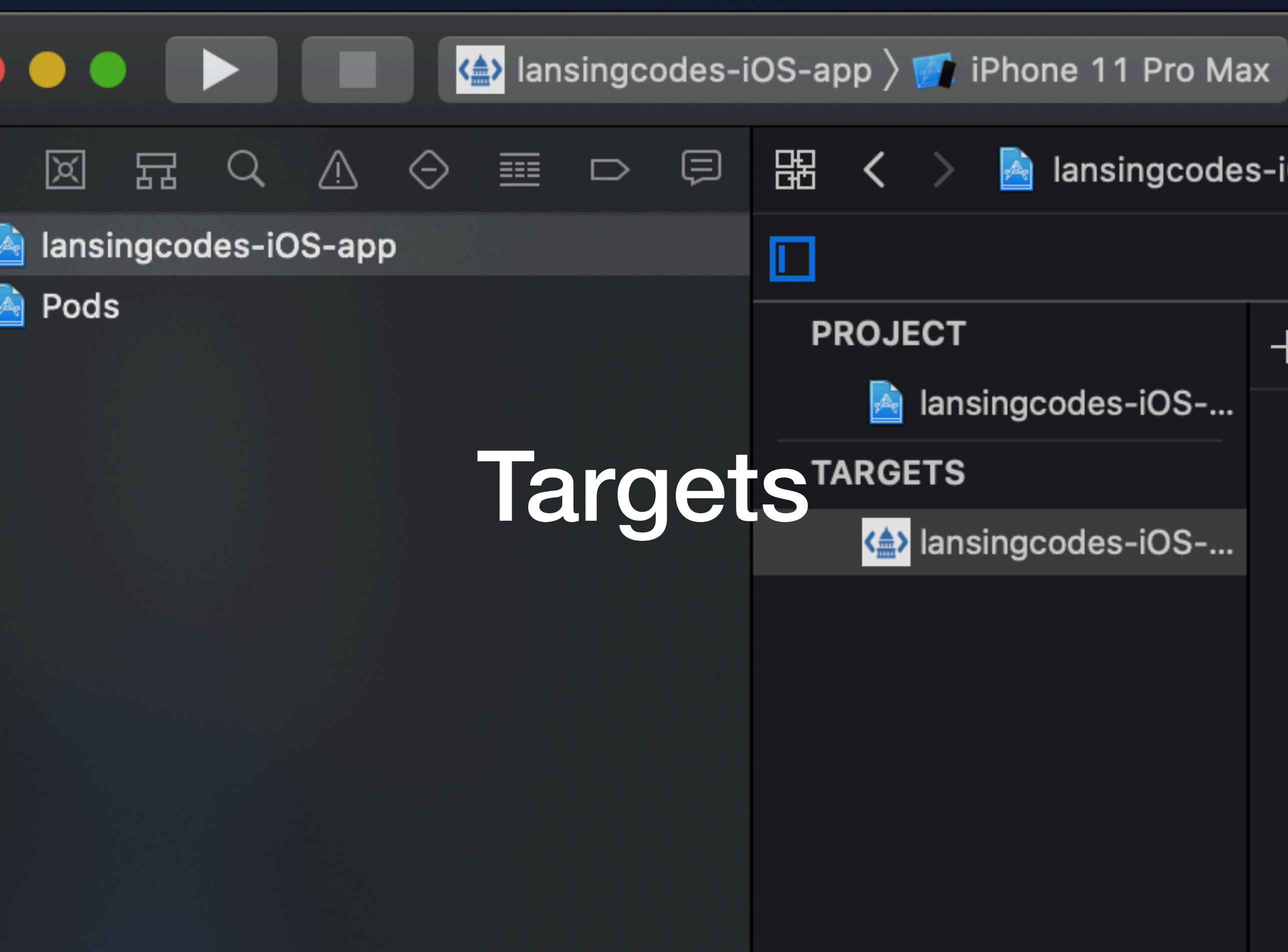
Static Library

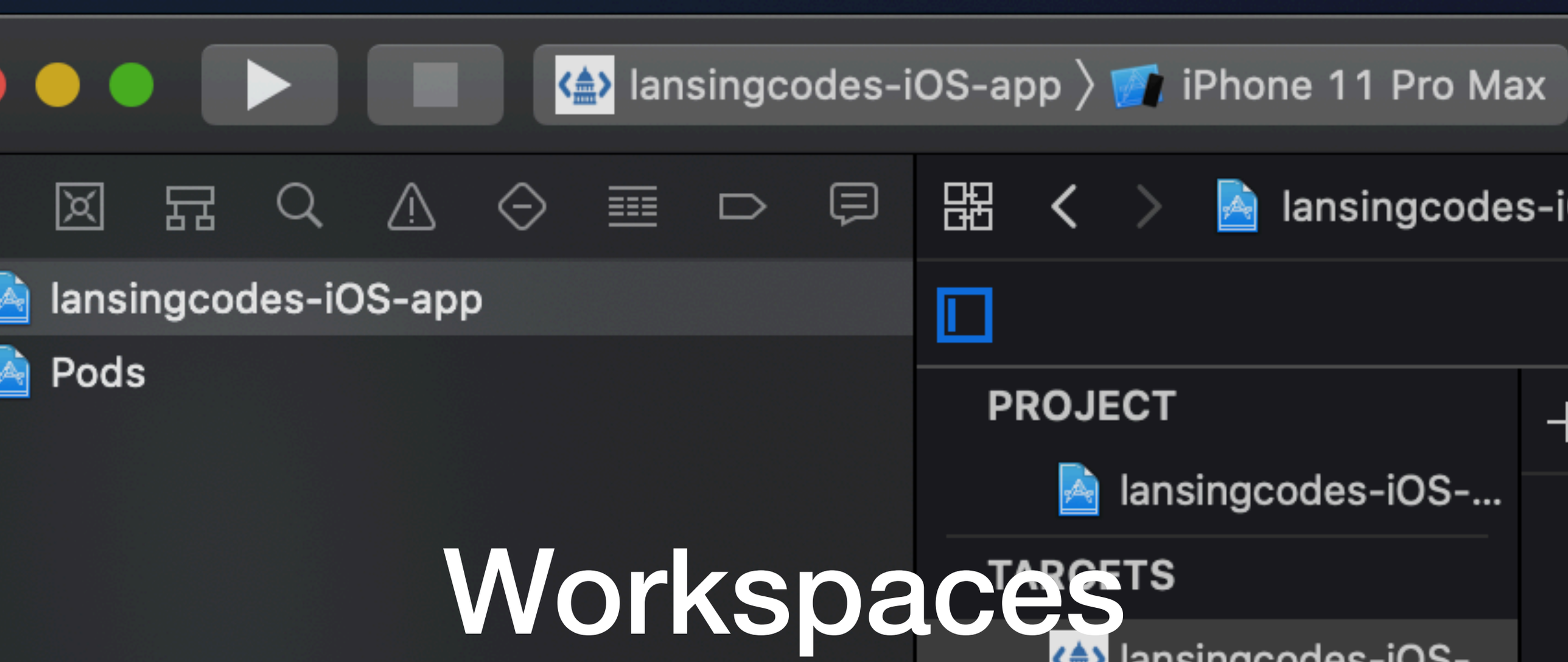
Metal Library

Cancel

Previous

Next


lansingcodes-iOS-app
iPhone 11 Pro Max
lansingcodes-iOS-app
Pods
lansingcodes-i
PROJECT
lansingcodes-iOS-...
TARGETS
lansingcodes-iOS-...
Targets

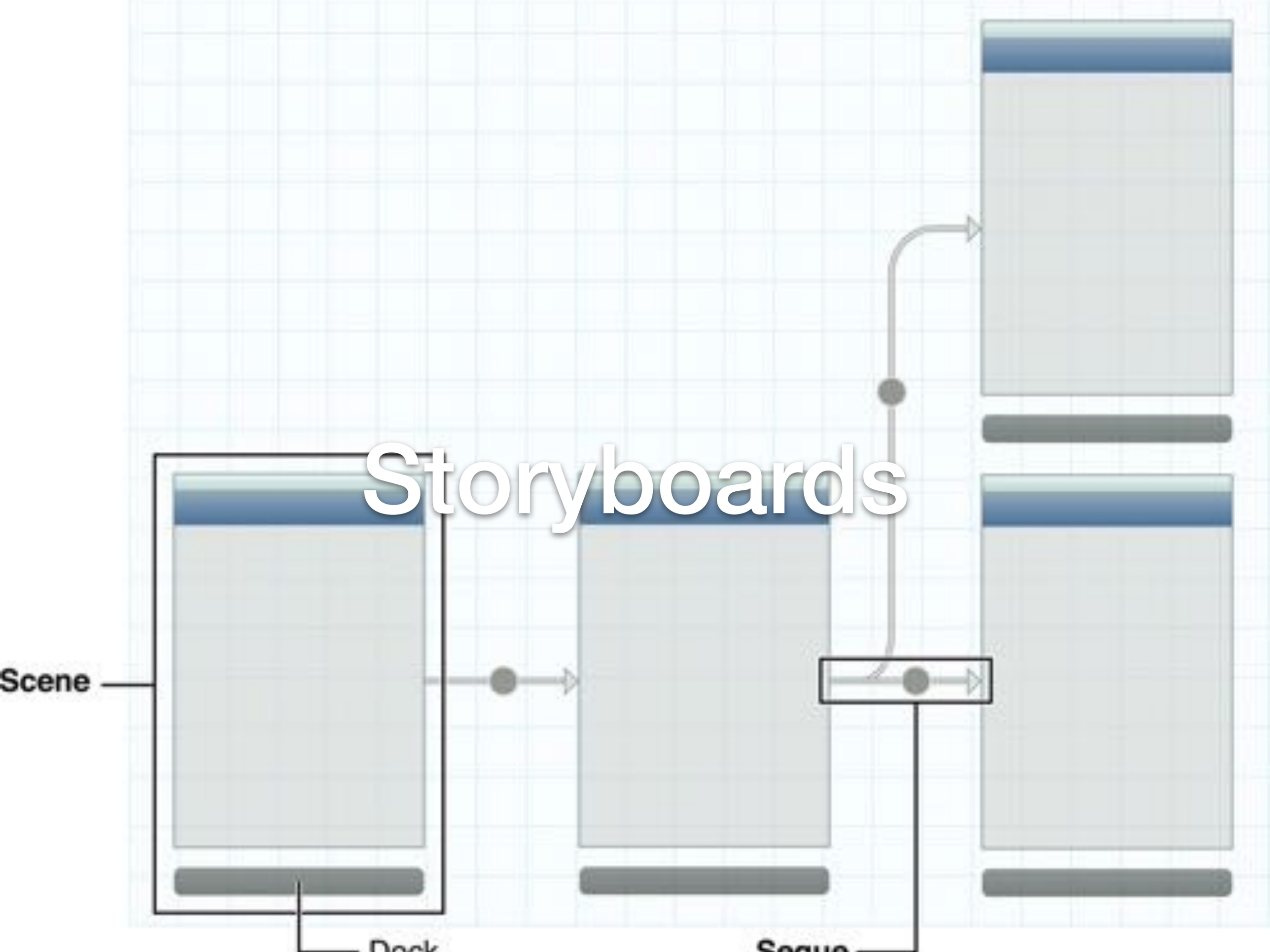

lansingcodes-iOS-app
iPhone 11 Pro Max
lansingcodes-iOS-app
Pods
lansingcodes-i
PROJECT
lansingcodes-iOS-...
TARGETS
lansingcodes-iOS-...

Workspaces

	Type	Value
rmation Property List	Dictionary	(16 items)
Localization native development re...	String	$(DEVELOPMENT_LANGUAGE)
Document types	Array	(0 items)
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER
nfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	$(PRODUCT_BUNDLE_PACKAGE_T
Bundle versions string, short	String	1.0
Bundle version	String	$(CURRENT_PROJECT_VERSION)
Application requires iPhone enviro...	Boolean	YES
Supports opening documents in pl...	Boolean	YES
Application Scene Manifest	Dictionary	(2 items)
Launch screen interface file base...	String	LaunchScreen
Required device capabilities	Array	(1 item)
Supported interface orientations	Array	(1 item)
Supported interface orientations (i...	Array	(4 items)

Plists

Storyboards



2x

3x

iPhone Notification
iOS 7-13
20pt

2x

3x

iPhone
Settings - iOS 7-13
29pt

2x

3x

iPhone Spotlight
iOS 7-13
40pt

Asset Libraries

2x

3x

iPhone App
iOS 7-13
60pt

1x

2x

1x

2x

1x

2x

Base32Crockford 〉 Test

Passed Failed Mixed Total durati

	Duration
e32CrockfordTests › Base32CrockfordTests	7 passed (100%) in 0.594s
testExample()	0.154s
testGenerateArray()	0.419s
testGenerateArrayLessThanZero()	0.00233s
testIdentifierDataTypeCodable()	0.00352s
testMinimumUniqueCount()	0.00173s
testMinimumUniqueCountLessThanZ...	0.00101s
testUUID()	0.0133s

Unit Testing

Dependency Management

Cocoapods

Swift Package Manager

Swift Language

Hello World

```swift
import Foundation
let str = "Hello, playground"

print(str)
```

Function

```
import Foundation

func hello(name: String) {
    print("hello", name)
}

hello(name: "playground")
```

```
import Foundation

func hello(name: String) {
    print(“hello \(name)”)
}

hello(name: "playground")
```

Class vs Struct

```
class PersonObject {
  var name : String

  init(name: String) {
    self.name = name
  }
}
```

```
struct PersonValue {
  var name : String

  init(name: String) {
    self.name = name
  }
}
```

```
func hello(person value:
  PersonValue) {
  print("hello", value.name)
}
```

```
func hello(person object:
  PersonObject) {
  print("hello", object.name)
}
```

```swift
let personObject =
  PersonObject(name: "object")

let personValue =
  PersonValue(name: “value")

hello(person: personObject) // hello object
hello(person: personValue) // hello value
```

```
personValue.name = “value" // !!DOES NOT COMPILE!!
personObject.name = “object" // does compile
```

```swift
var newPersonValue = personValue
var newPersonObject = personObject
newPersonValue.name = "new Value"
newPersonObject.name = "new Object"

hello(person: newPersonObject) // hello new Object
hello(person: newPersonValue) // hello new Value

hello(person: personObject) // hello new Object
hello(person: personValue) //hello value
```

Optionals

```
class Employee : CustomStringConvertible {
  let profile : Profile
  var manager : Employee?

  init (profile : Profile, manager : Employee? = nil)
{
    self.profile = profile
    self.manager = manager
  }

  var description: String {
    return
       "name:\(self.profile.name) with manager:
 \(self.manager?.profile.name)”
  }

  func hire (employee: Employee) {
    employee.manager = self
  }
}
```

```
let names = ["Byron Fisher","Garrett
Doyle","Angel Hammond","Brianna
Bradley","Evelyn Ellis"]

var manager : Employee? = nil
var employees = [Employee]()
for name in names {
  let newEmployee = Employee(profile:
Profile(name: name), manager: manager)
  employees.append(newEmployee)
  manager = newEmployee
}
print(employees)
```

```
[
name:Byron Fisher with manager:nil,

name:Garrett Doyle with
manager:Optional("Byron Fisher"),

name:Angel Hammond with
manager:Optional("Garrett Doyle"),

name:Brianna Bradley with
manager:Optional("Angel Hammond"),

name:Evelyn Ellis with
manager:Optional("Brianna Bradley”)
]
```

```
var foundCeo : Employee?

for employee in employees {
   if foundCeo != nil {
      break
   }
   if employee.manager == nil {
      foundCeo = employee
   }
}
```

```
if let ceo = foundCeo {
   for employee in employees {
      if employee !== ceo {
         ceo.hire(employee: employee)
      }
   }
}

print(employees)
```

```
guard let ceo = foundCeo else {
    …
}
```

```
[
name:Byron Fisher with manager:nil,

name:Garrett Doyle with
manager:Optional("Byron Fisher"),

name:Angel Hammond with
manager:Optional("Byron Fisher"),

name:Brianna Bradley with
manager:Optional("Byron Fisher"),

name:Evelyn Ellis with
manager:Optional("Byron Fisher”)
]
```

Enums

```
enum Transform {
   case scale(Double)
   case move(CGPoint)
}
switch transform {
case .scale(let factor):
   …
case .move(let point):
   …
}
```

Patterns and Practices

Protocol Oriented Programming

Protocols

```
protocol Automobile {
  func horn() -> String
}

struct Car : Automobile {
  func horn() -> String {
    return "beep"
  }
}

struct Truck : Automobile {
  func horn() -> String {
    return "honk"
  }
}

let vehicles : [Automobile] = [Car(), Truck()]

for vehicle in vehicles {
  print(vehicle.horn())
}
```

beep

honk

Extensions

```
extension String : Automobile {
   func horn() -> String {
      return "\(self) on wheels"
   }
}
```

```
let vehicles : [Automobile] =
[Car(), Truck(), "Cow"]

for vehicle in vehicles {
   print(vehicle.horn())
}
```

beep
honk

Cow on wheels

Delegation Pattern

UITableView - UIKit

```
class UITableView : UIView {
  var delegate: UITableViewDelegate
  var dataSource: UITableViewDataSource
}
```

```
class UITableViewController {
  init (…){
    self.tableView.delegate = self
    self.tableView.dataSource = self
  }
}
```

```
class TableViewController : UITableViewController {

  override func viewDidLoad() {
    super.viewDidLoad()
    tableView.register(UITableViewCell.self,
forCellReuseIdentifier: "default")
  }

  override func tableView(_ tableView: UITableView,
numberOfRowsInSection section: Int) -> Int {
    return names.count
  }

  override func tableView(_ tableView: UITableView,
cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell =
tableView.dequeueReusableCell(withIdentifier: "default",
for: indexPath)
    cell.textLabel?.text = names[indexPath.row]
    return cell
  }
}
```

Functional Programming

```swift
let names = ["Byron Fisher","Garrett Doyle","Angel
Hammond","Brianna Bradley","Evelyn Ellis"]

struct Person : Identifiable {
  let id : UUID
  let name : String

  init(id : UUID? = nil, name: String) {
    self.id = id ?? UUID()
    self.name = name
  }
}

let people = names.map{
  Person(id: nil, name: $0)
}.sorted { (lhs, rhs) -> Bool in
  lhs.name.compare(rhs.name) == .orderedAscending
}
```

Subscribers and Publishers

List - SwiftUI

```
class ApplicationData : ObservableObject {
  @Published var people : [Person]

  init (people : [Person]) {
    self.people = people
  }
}


struct ContentView : View {
  @EnvironmentObject var data : ApplicationData
  var body : some View {
    List(self.data.people) { (person) in
      Text(person.name)
    }
  }
}

let liveView =
  ContentView().environmentObject(
    ApplicationData(people: people))

PlaygroundPage.current.setLiveView(liveView)
```

Angel Hammond

Brianna Bradley

Byron Fisher

Evelyn Ellis

Garrett Doyle

List w/ URL Request - SwiftUI

```
let url = URL(string: "https://jsonplaceholder.typicode.com/
users")!
let decoder = JSONDecoder()
let publisher =  URLSession.shared.dataTaskPublisher(for:
url)
        .map(\.data)
        .decode(type: [Person].self, decoder: JSONDecoder())
        .assertNoFailure().eraseToAnyPublisher()
let data = ApplicationData(people: [Person]())
let cancellable = publisher.receive(on:
DispatchQueue.main).assign(to: \.people, on: data)
let liveView = ContentView().environmentObject(data)
PlaygroundPage.current.setLiveView(liveView)
```

Leanne Graham
Ervin Howell
Clementine Bauch
Patricia Lebsack
Chelsey Dietrich
Mrs. Dennis Schulist
Kurtis Weissnat
Nicholas Runolfsdottir V
Glenna Reichert
Clementina DuBuque

Multiple Platforms

iPad vs iPhone

macOS

watchOS

tvOS

Linux

Server-Side

What's Next?

What's Next?

- Start with Swift Playgrounds
- Basics of Storyboards and UIKit
  - UITableView
- Basics of SwiftUI and Combine
- Specialize in an API:
  - WatchKit, AVFoundation, Notifications, CloudKit, Vapor

https://brightdigit.typeform.com/to/W9aFhs