CSC 581: Mobile App Development

Spring 2019

Unit 1: Getting Started with App Development

- Xcode
  - installing XCode, creating a project, MVC pattern
  - interface builder, storyboards, object library
  - outlets vs. actions, attributes inspector
  - UIView, UILabel, UISwitch, UIButton, …
  - auto layout, constraints, alignment
- Swift
  - language history, playgrounds vs. projects
  - let vs. var, types, type inference
  - operators, if-else, while, for

1

Installing Xcode

the latest version is Xcode 10 (specifically, 10.1), with Swift 4.2

- requires macOS 10.13.4 (High Sierra) or above

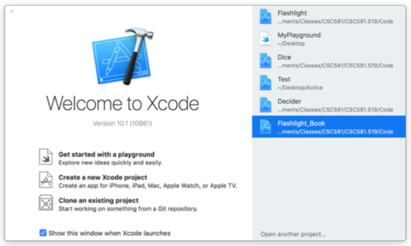
to download and install:

1. launch the App Store from Applications
2. search for and select Xcode
3. click on the download icon

4. once downloaded, launch from Applications

App Store.app

Xcode.app



Welcome to Xcode

Version 10.1 (10B61)

Get started with a playground
Explore new ideas quickly and easily.

Create a new Xcode project
Create an app for iPhone, iPad, Mac, Apple Watch, or Apple TV.

Clone an existing project
Start working on something from a Git repository.

☑ Show this window when Xcode launches

Flashlight
…ments/Classes/CSC581/CSC581.S19/Code

MyPlayground
~/Desktop

Dice
…ments/Classes/CSC581/CSC581.S19/Code

Test
~/Desktop/Active

Decider
…ments/Classes/CSC581/CSC581.S19/Code

Flashlight_Book
…ments/Classes/CSC581/CSC581.S19/Code

Open another project…

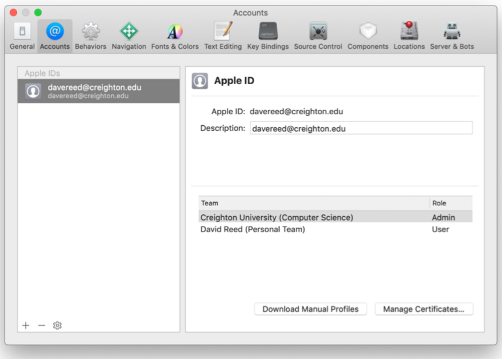
2

Customizing Xcode

add your Apple ID to Xcode Preferences

this step is necessary if you want to download apps to your own device

1. go to the Xcode menu
2. *select Preferences*
3. select *Accounts*
4. click on the + at the bottom left and enter your Apple ID



davereed@creighton.edu
davereed@creighton.edu

Apple ID

Apple ID: davereed@creighton.edu
Description: davereed@creighton.edu

Team	Role
Creighton University (Computer Science)	Admin
David Reed (Personal Team)	User

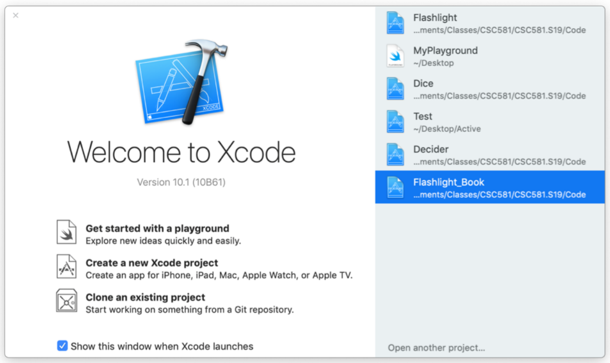
3

Xcode

XCode can be used to create/open a playground or a project

- a Playground is an interactive environment for testing Swift code
- similar to the IDLE interpreter for Python
- very useful tool, but not always stable (e.g., locks up in my installation)

- a Project is an app, which can be run in a simulator or an actual device



Welcome to Xcode

Version 10.1 (10B61)

Get started with a playground
Explore new ideas quickly and easily.

Create a new Xcode project
Create an app for iPhone, iPad, Mac, Apple Watch, or Apple TV.

Clone an existing project
Start working on something from a Git repository.

✓ Show this window when Xcode launches

Flashlight
...ments/Classes/CSC581/CSC581.S19/Code
MyPlayground
~/Desktop
Dice
...ments/Classes/CSC581/CSC581.S19/Code
Test
~/Desktop/Active
Decider
...ments/Classes/CSC581/CSC581.S19/Code
Flashlight_Book
...ments/Classes/CSC581/CSC581.S19/Code

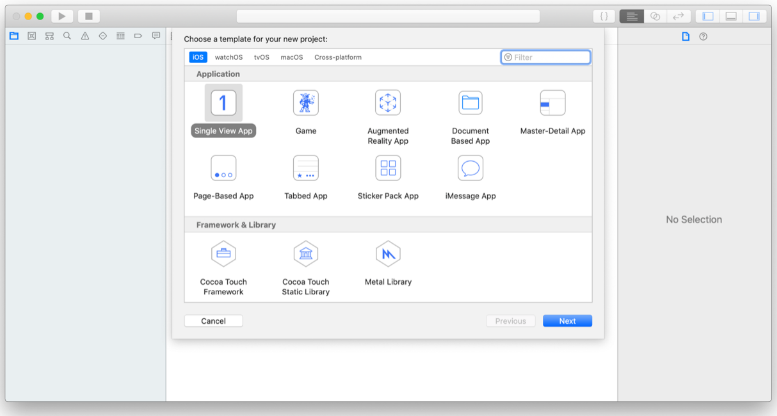
Open another project...

4

Creating a project in Xcode

select *Create a new Xcode project* from the opening window

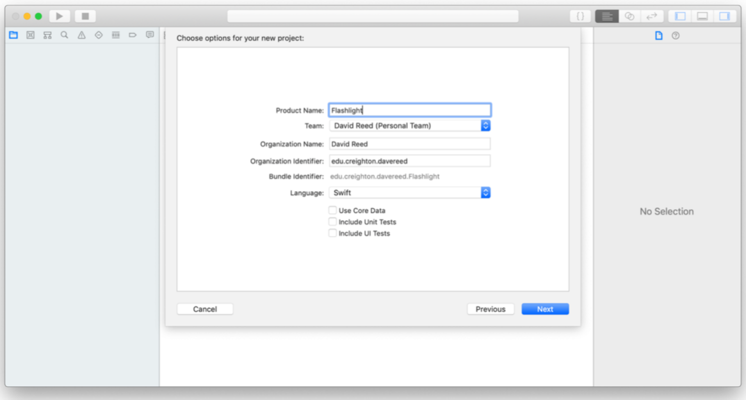
- you are then shown a collection of templates to choose from
- we will use the generic *Single View App*



Creating a project in Xcode (cont.)

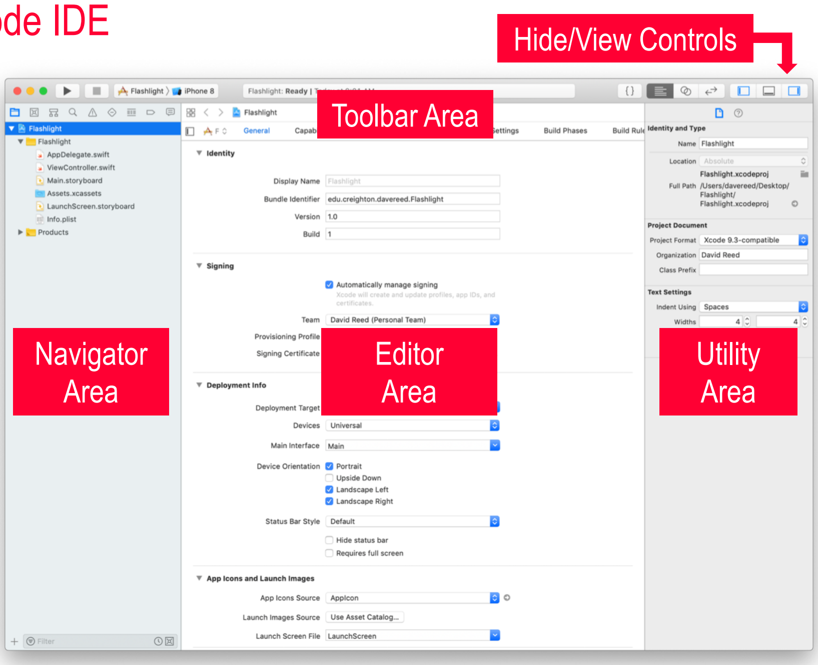
- enter the name of the project
- also personal information
  the organization identifier should be unique, use `edu.creighon.NETID`

- you will also be prompted for the location where the project will be stored


Choose options for your new project:

Product Name: Flashlight
Team: David Reed (Personal Team)
Organization Name: David Reed
Organization Identifier: edu.creighton.davereed
Bundle Identifier: edu.creighton.davereed.Flashlight
Language: Swift
☐ Use Core Data
☐ Include Unit Tests
☐ Include UI Tests

Xcode IDE



Identity
Display Name: Flashlight
Bundle Identifier: edu.creighton.davereed.Flashlight
Version: 1.0
Build: 1

Signing
✓ Automatically manage signing
Team: David Reed (Personal Team)

Deployment Info
Devices: Universal
Main Interface: Main
Device Orientation: ✓ Portrait, Upside Down, ✓ Landscape Left, ✓ Landscape Right
Status Bar Style: Default
Hide status bar
Requires full screen

App Icons and Launch Images
App Icons Source: AppIcon
Launch Images Source: Use Asset Catalog...
Launch Screen File: LaunchScreen

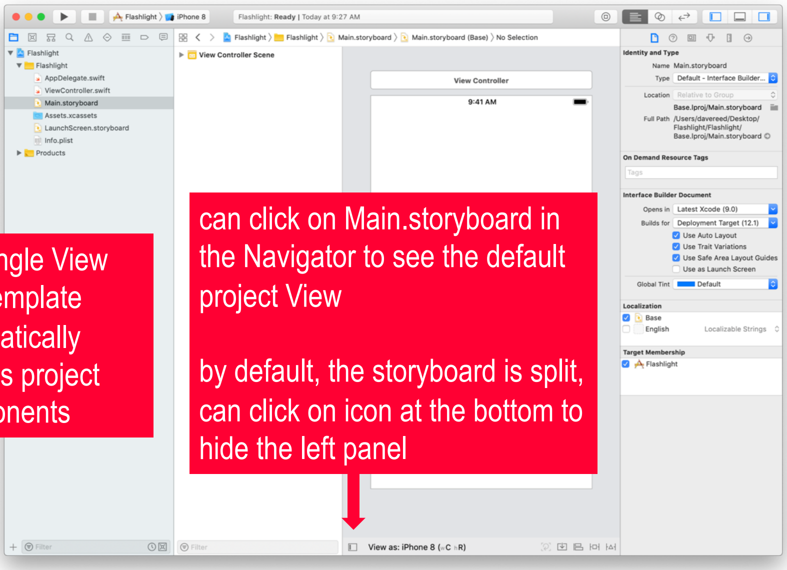
Identity and Type
Name: Flashlight
Location: Absolute
Full Path: /Users/davereed/Desktop/Flashlight/Flashlight.xcodeproj

Project Document
Project Format: Xcode 9.3-compatible
Organization: David Reed

Text Settings
Indent Using: Spaces
Widths: 4 4

7

Storyboard



Identity and Type
Name: Main.storyboard
Type: Default - Interface Builder...
Location: Relative to Group
Full Path: /Users/davereed/Desktop/Flashlight/Flashlight/Base.lproj/Main.storyboard

Interface Builder Document
Opens in: Latest Xcode (9.0)
Builds for: Deployment Target (12.1)
✓ Use Auto Layout
✓ Use Trait Variations
✓ Use Safe Area Layout Guides
Use as Launch Screen
Global Tint: Default

Localization
✓ Base
English

Target Membership
✓ Flashlight

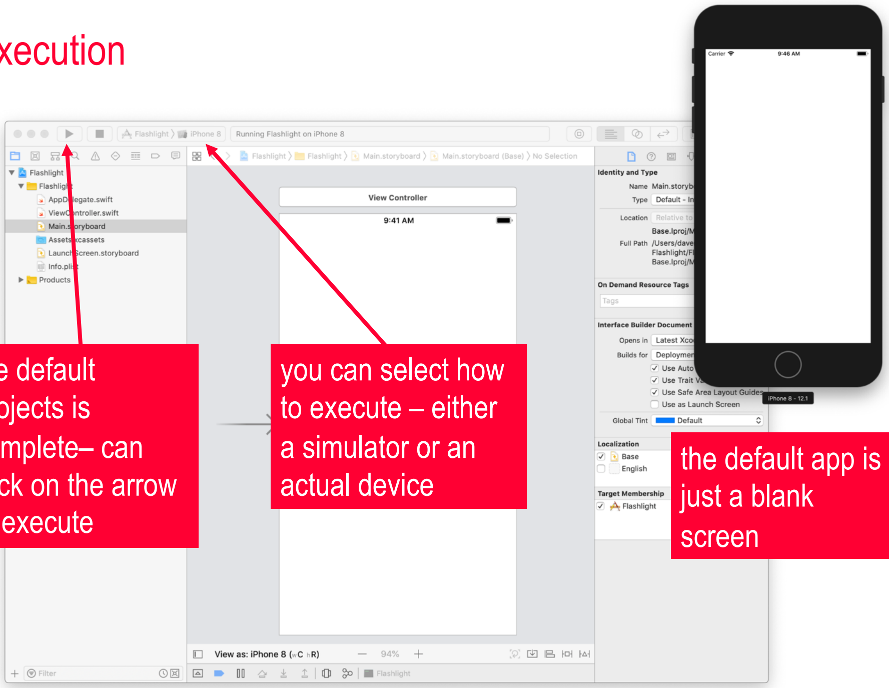
the Single View App template automatically creates project components

can click on Main.storyboard in the Navigator to see the default project View

by default, the storyboard is split, can click on icon at the bottom to hide the left panel

8

Execution



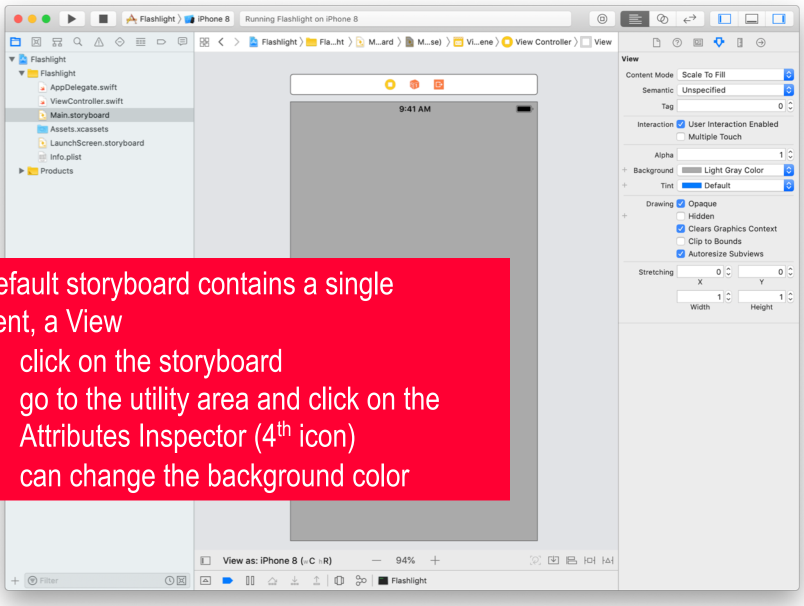
the default projects is complete– can click on the arrow to execute

you can select how to execute – either a simulator or an actual device

the default app is just a blank screen

9

Changing view attributes

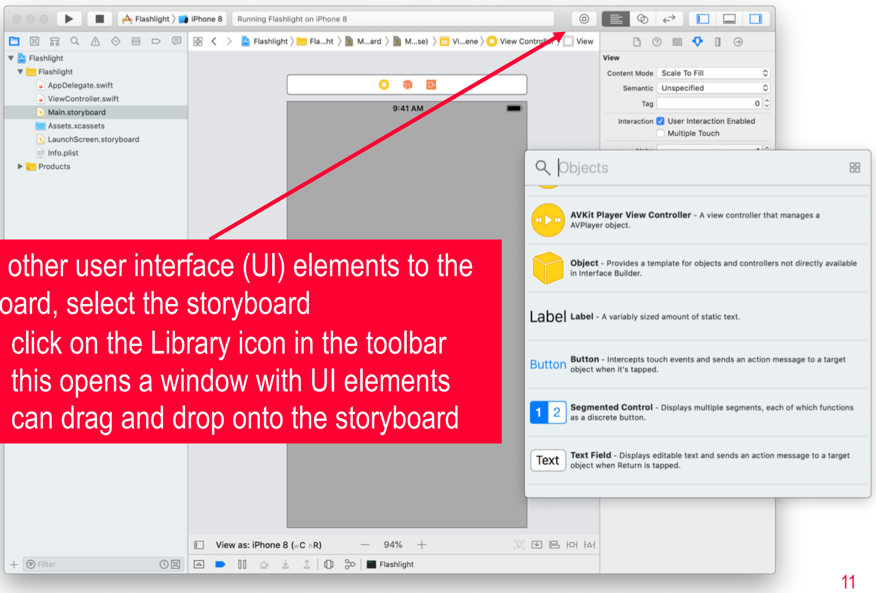


the default storyboard contains a single element, a View

- click on the storyboard
- go to the utility area and click on the Attributes Inspector (4th icon)
- can change the background color

10

UI Library



View

Content Mode: Scale To Fill
Semantic: Unspecified
Tag: 0
Interaction: [✓ User Interaction Enabled] Multiple Touch

AVKit Player View Controller - A view controller that manages a AVPlayer object.

Object - Provides a template for objects and controllers not directly available in Interface Builder.

Label - A variably sized amount of static text.

Button - Intercepts touch events and sends an action message to a target object when it's tapped.

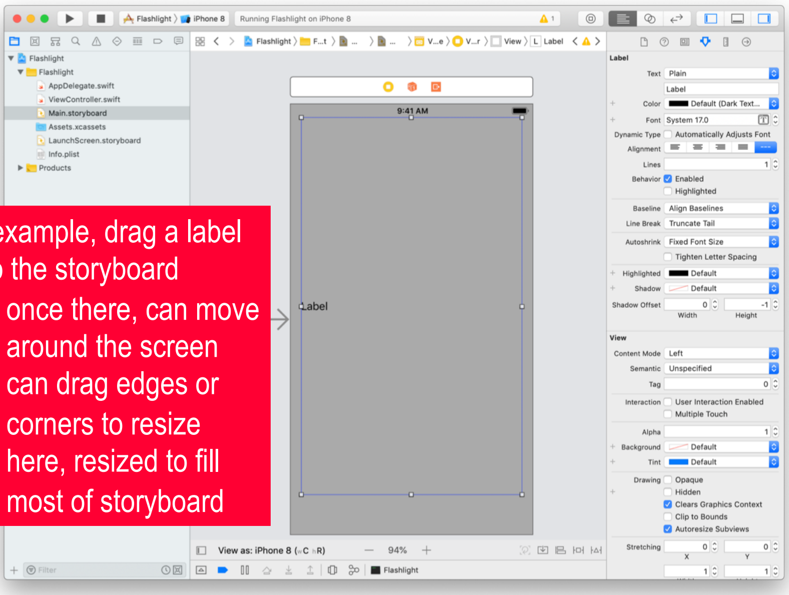
Segmented Control - Displays multiple segments, each of which functions as a discrete button.

Text Field - Displays editable text and sends an action message to a target object when Return is tapped.

to add other user interface (UI) elements to the storyboard, select the storyboard
- click on the Library icon in the toolbar
- this opens a window with UI elements
- can drag and drop onto the storyboard

11

Adding UI elements to the storyboard



Label

Text: Plain
Label
Color: Default (Dark Text...
Font: System 17.0
Dynamic Type: Automatically Adjusts Font
Lines: 1
Behavior: [✓ Enabled] Highlighted
Baseline: Align Baselines
Line Break: Truncate Tail
Autoshrink: Fixed Font Size
Tighten Letter Spacing
Highlighted: Default
Shadow: Default
Shadow Offset: 0 (Width) -1 (Height)

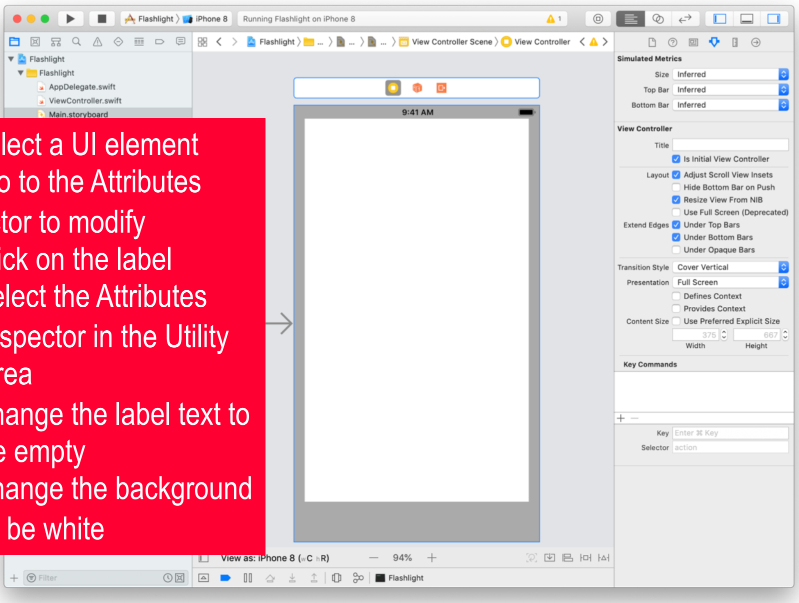
View

Content Mode: Left
Semantic: Unspecified
Tag: 0
Interaction: User Interaction Enabled, Multiple Touch
Alpha: 1
Background: Default
Tint: Default
Drawing: Opaque, Hidden, [✓ Clears Graphics Context], Clip to Bounds, [✓ Autoresize Subviews]
Stretching: 0 (X) 0 (Y) 1 1

for example, drag a label onto the storyboard
- once there, can move around the screen
- can drag edges or corners to resize
- here, resized to fill most of storyboard

12

Changing UI elements attributes



Simulated Metrics
Size: Inferred
Top Bar: Inferred
Bottom Bar: Inferred

View Controller
Title:
☑ Is Initial View Controller
Layout: ☑ Adjust Scroll View Insets
☐ Hide Bottom Bar on Push
☑ Resize View From NIB
☐ Use Full Screen (Deprecated)
Extend Edges: ☑ Under Top Bars
☑ Under Bottom Bars
☐ Under Opaque Bars
Transition Style: Cover Vertical
Presentation: Full Screen
☐ Defines Context
☐ Provides Context
Content Size: ☐ Use Preferred Explicit Size
Width: 375 Height: 667

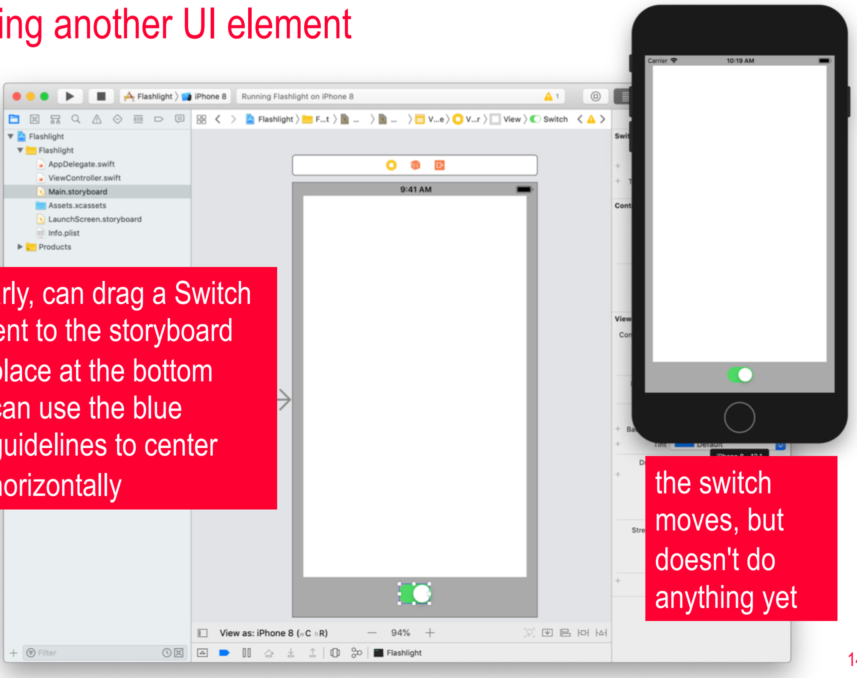
Key Commands
Key: Enter ⌘ Key
Selector: action

can select a UI element then go to the Attributes Inspector to modify

- click on the label
- select the Attributes Inspector in the Utility Area
- change the label text to be empty
- change the background to be white

13

Adding another UI element



similarly, can drag a Switch element to the storyboard

- place at the bottom
- can use the blue guidelines to center horizontally

the switch moves, but doesn't do anything yet

14

Model-View-Controller

well designed apps utilize the *Model-View-Controller* software pattern:

- the **Model** defines the logic underlying the app
  e.g., if you are building a poker app, you would define classes to represent and model cards, decks of cards, poker hands, …
- the **View** defines the look of the app
  e.g., images of the cards on a tabletop, buttons or other controls for selecting or displaying cards, …
- the **Controller** defines actions of the app - it connects the user interface (View) with the app logic (Model)
  e.g., when the user clicks on a button to deal a card, it calls the appropriate method on a deck of cards object, then displays the resulting image

15

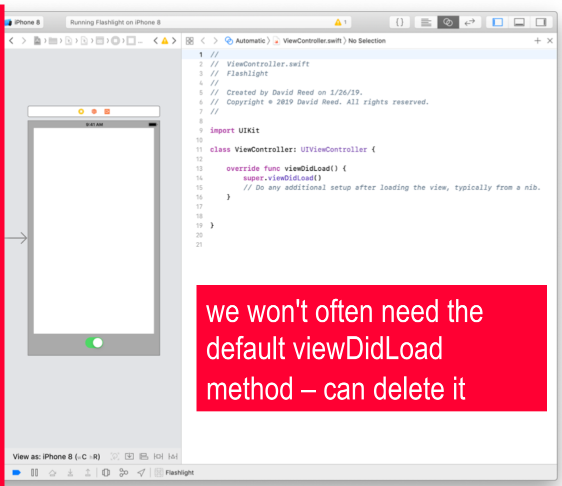
Flashlight MVC

Model: since the Flashlight app is so simple, its model will be implicit

View: the storyboard defines the look of the app (i.e., its View)

Controller: by default, each storyboard view has predefined controller (instance of the UIViewController class)

- click on linked circles icon in toolbar to open side-by-side Assistant editor



```
//
//  ViewController.swift
//  Flashlight
//
//  Created by David Reed on 1/26/19.
//  Copyright © 2019 David Reed. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }


}
```

we won't often need the default viewDidLoad method – can delete it

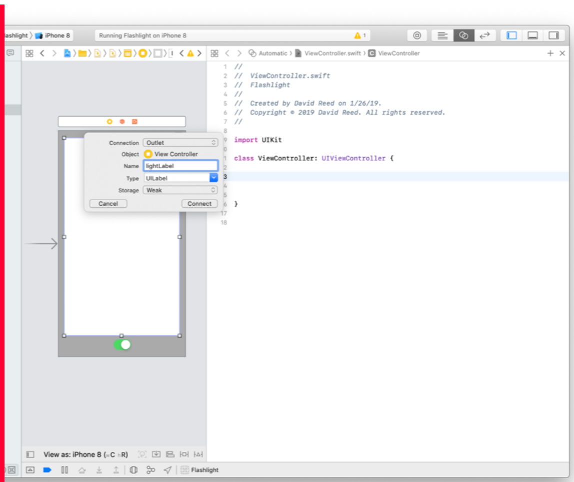
16

Connecting the View & Controller

to connect View & Controller,

- Control-drag (or right-click-drag) from elements in the View to code in the Controller

- e.g., control-drag from the flashlight label to inside the class definition
- a line will show the connection
- once released, a window will appear with choices
- select *Outlet*, enter a name, then hit *Connect*



Connection: Outlet
Object: View Controller
Name: lightLabel
Type: UILabel
Storage: Weak

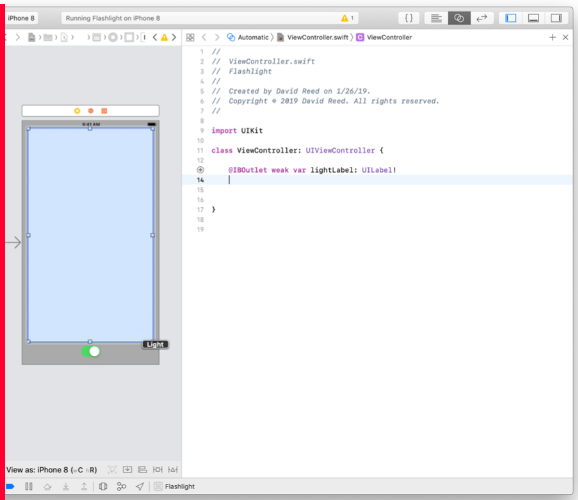
```
//
//  ViewController.swift
//  Flashlight
//
//  Created by David Reed on 1/26/19.
//  Copyright © 2019 David Reed. All rights reserved.
//

import UIKit

class ViewController: UIViewController {



}
```

17

Outlets/Fields

- as a result, a field is added to the ViewController class with the specified name
- essentially, that UI element is now accessible to the controller code

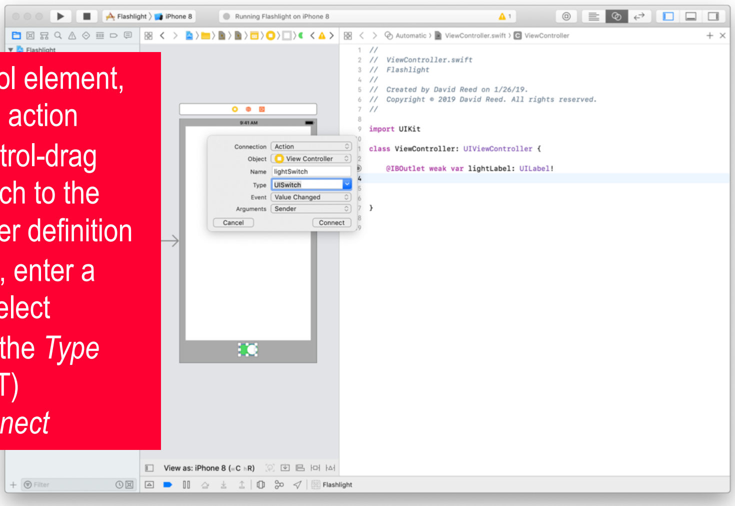
note: if you move the mouse of the circle to the left of the field, the corresponding UI field, the corresponding UI element in the storyboard is highlighted



```
//
//  ViewController.swift
//  Flashlight
//
//  Created by David Reed on 1/26/19.
//  Copyright © 2019 David Reed. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var lightLabel: UILabel!
    


}
```

18

Defining actions via control elements



Connection	Action
Object	View Controller
Name	lightSwitch
Type	UISwitch
Event	Value Changed
Arguments	Sender

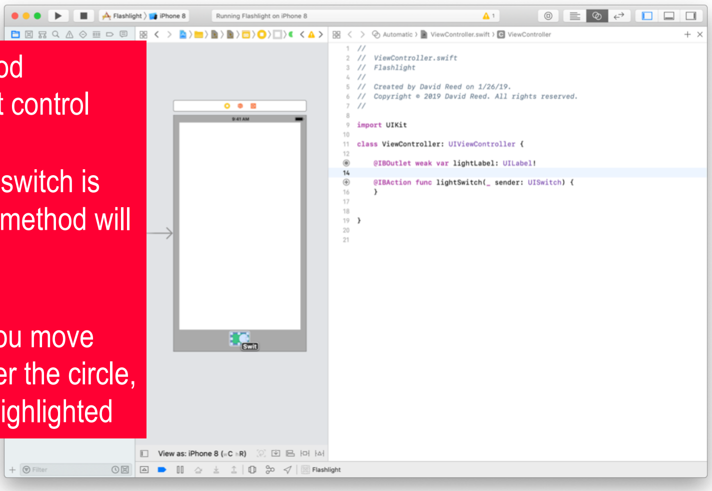
```swift
//
//  ViewController.swift
//  Flashlight
//
//  Created by David Reed on 1/26/19.
//  Copyright © 2019 David Reed. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var lightLabel: UILabel!



}
```

a switch is a control element, so it can define an action

- similarly, control-drag from the switch to the ViewController definition
- select *Action*, enter a name, and select *UISwitch* for the *Type* (IMPORTANT)
- then, hit *Connect*

19

Actions/methods



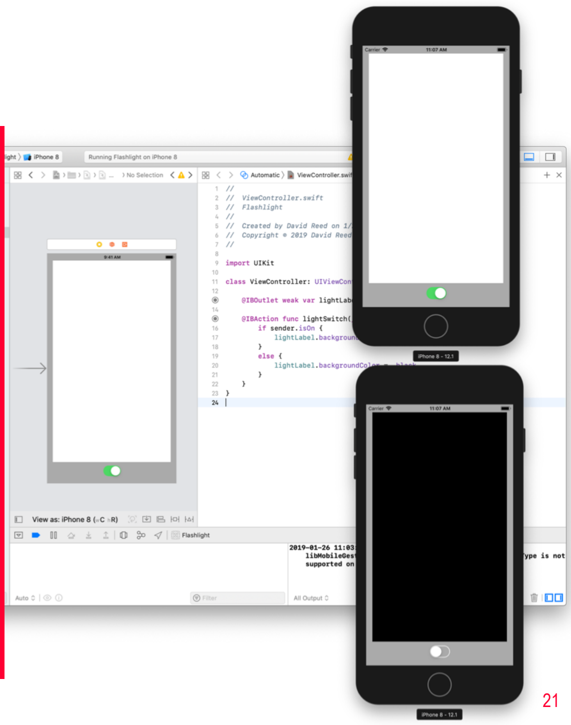
```swift
//
//  ViewController.swift
//  Flashlight
//
//  Created by David Reed on 1/26/19.
//  Copyright © 2019 David Reed. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var lightLabel: UILabel!

    @IBAction func lightSwitch(_ sender: UISwitch) {
    }


}
```

this creates a method associated with that control element

- each time the switch is changed, this method will be called
- as before, if you move the mouse over the circle, the switch is highlighted

20

Actions/methods



```
//
//  ViewController.swift
//  Flashlight
//
//  Created by David Reed on 1/
//  Copyright © 2019 David Reed
//

import UIKit

class ViewController: UIViewCon

    @IBOutlet weak var lightLab

    @IBAction func lightSwitch(
        if sender.isOn {
            lightLabel.backgroun
        }
        else {
            lightLabel.backgroundColo
        }
    }
}
```

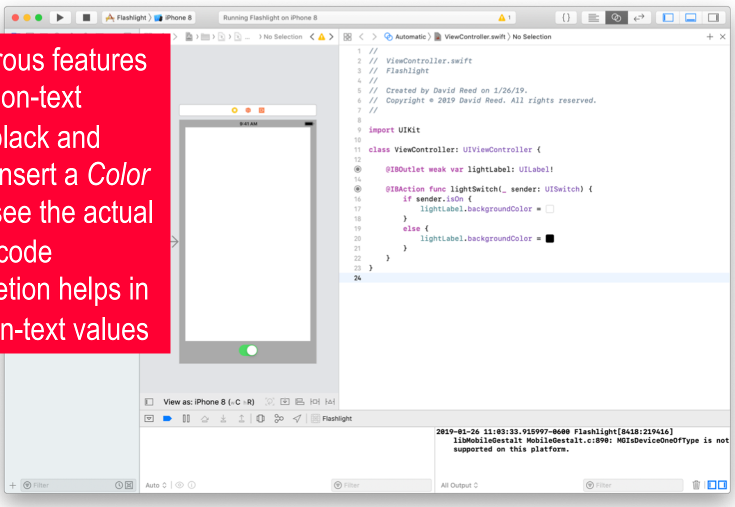
```
2019-01-26 11:03
    libMobileGes
    supported on
```

for more complex apps, we would now create a separate Model
e.g., define classes, have fields that are objects of those classes, call methods to implement actions

- here, the logic is so basic, the Model can be implicit in the Controller
- simply change the label color based on the switch position

21

Color literals



```
//
//  ViewController.swift
//  Flashlight
//
//  Created by David Reed on 1/26/19.
//  Copyright © 2019 David Reed. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var lightLabel: UILabel!

    @IBAction func lightSwitch(_ sender: UISwitch) {
        if sender.isOn {
            lightLabel.backgroundColor = ⬜
        }
        else {
            lightLabel.backgroundColor = ⬛
        }
    }
}
```

```
2019-01-26 11:03:33.915997-0600 Flashlight[8418:219416]
    libMobileGestalt MobileGestalt.c:890: MGIsDeviceOneOfType is not
    supported on this platform.
```

Xcode has numerous features that incorporate non-text

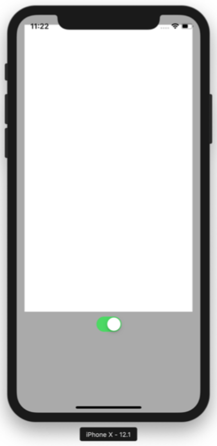
- instead of .black and .white, can insert a *Color Literal* and see the actual color in the code
- code completion helps in selecting non-text values

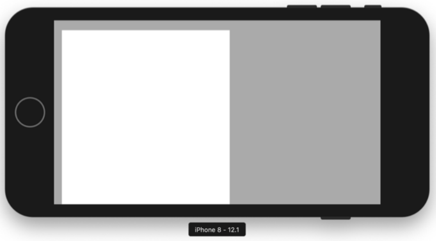
22

Different devices?

this app now works fine and looks good in the iPhone 8 simulator:

- what happens if we run it under a different simulator (e.g., iPhone X)?
- what happens if we rotate the phone (⌘→ or ⌘←)?

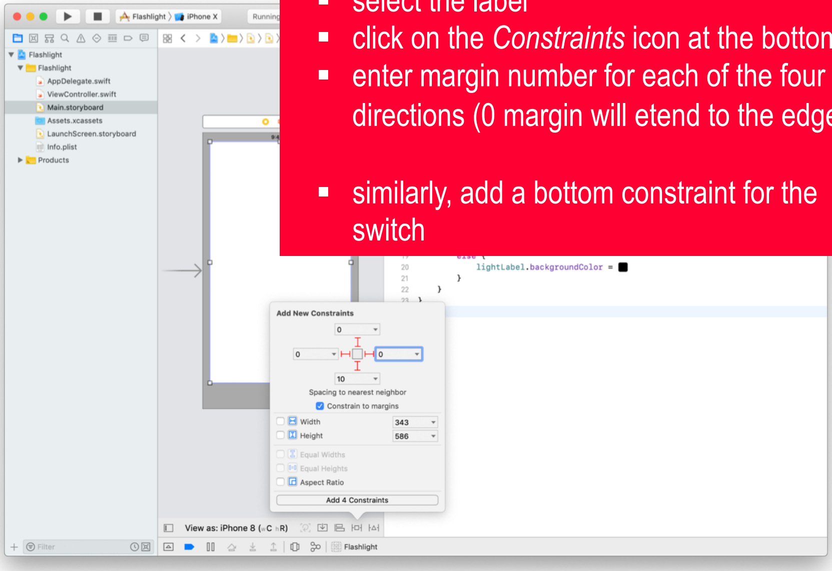




Auto layout

can use auto layout so that the layout of elements adjusts as the view size & orientation changes

- select the label
- click on the *Constraints* icon at the bottom
- enter margin number for each of the four directions (0 margin will etend to the edge)
- similarly, add a bottom constraint for the switch



```
            lightLabel.backgroundColor = ■
        }
    }
}
```

Add New Constraints

Top: 0; Left: 0; Right: 0; Bottom: 10

Spacing to nearest neighbor

☑ Constrain to margins

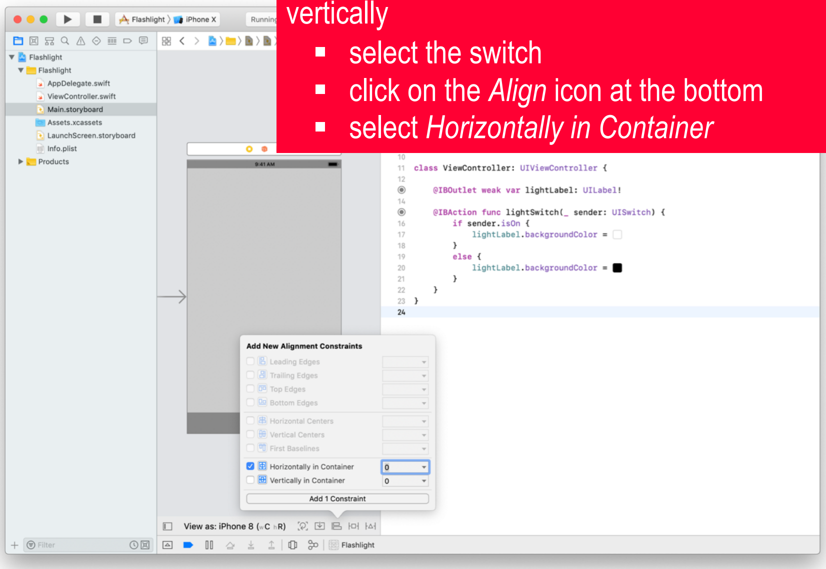
☐	Width	343
☐	Height	586
☐	Equal Widths	
☐	Equal Heights	
☐	Aspect Ratio	

Add 4 Constraints

Auto layout

similarly, can center elements horizontally and/or vertically

- select the switch
- click on the *Align* icon at the bottom
- select *Horizontally in Container*



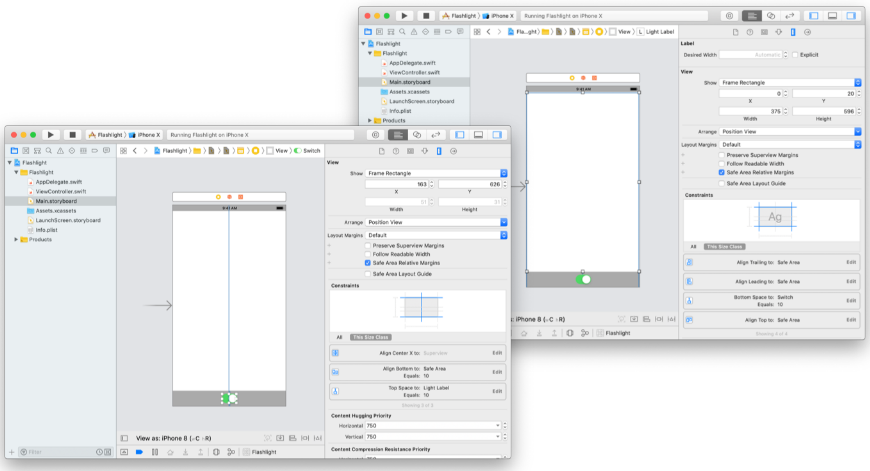
```
class ViewController: UIViewController {

    @IBOutlet weak var lightLabel: UILabel!

    @IBAction func lightSwitch(_ sender: UISwitch) {
        if sender.isOn {
            lightLabel.backgroundColor = ⬜
        }
        else {
            lightLabel.backgroundColor = ⬛
        }
    }
}
```

25

Viewing/editing constraints

if you select a UI element and go to the Size Inspector (top right)

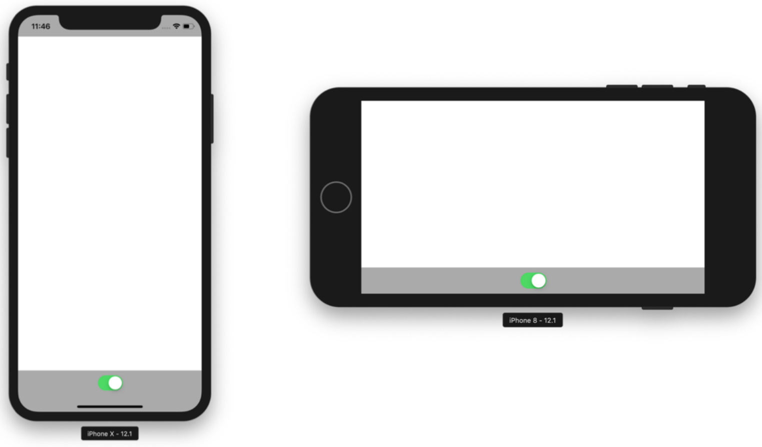
- can view the auto layout constraints assigned to that element
- you can edit and/or delete as well



26

Different devices?

with auto layout constraints, the app now looks good on different devices and different orientations

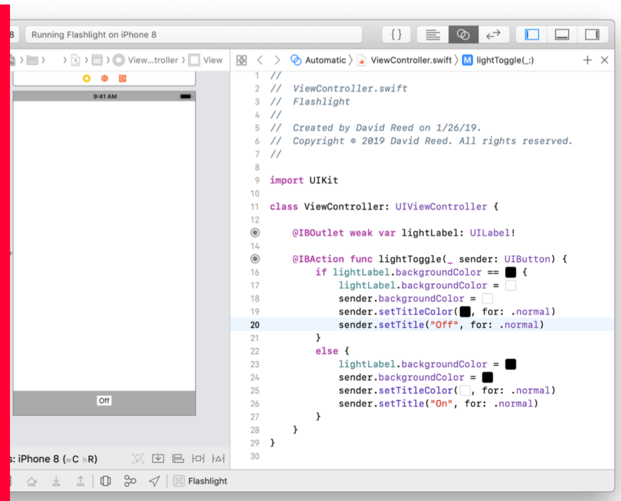


27

UIButton

alternatively, we could have used a button instead of a switch

- add a button labeled Off, with black text on a white background
- add an Action for handling button click
- when clicked, change the backgroundColor of the label, as well as the button text and background colors



```
//
//  ViewController.swift
//  Flashlight
//
//  Created by David Reed on 1/26/19.
//  Copyright © 2019 David Reed. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var lightLabel: UILabel!

    @IBAction func lightToggle(_ sender: UIButton) {
        if lightLabel.backgroundColor == ■ {
            lightLabel.backgroundColor = □
            sender.backgroundColor = □
            sender.setTitleColor(■, for: .normal)
            sender.setTitle("Off", for: .normal)
        }
        else {
            lightLabel.backgroundColor = ■
            sender.backgroundColor = ■
            sender.setTitleColor(□, for: .normal)
            sender.setTitle("On", for: .normal)
        }
    }
}
```

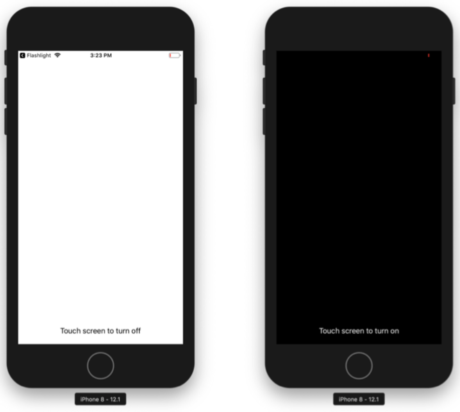
28

HW 1.1

the Lab at the end of Unit 1 leads you through a different Flashlight design

- end result is a single button that fills the screen
- when clicked, it switches between white and black backgrounds

- complete this tutorial, then add a label at the bottom of the screen with instructions



Swift

Apple released Swift in 2014, for iOS and macOS development

- easier to use & understand alternative to Objective-C
- now also used for watchOS and tvOS development
- Swift has been open-source since 2015, with a large community of supporters

Swift is fully object-oriented, but has the look of cleaner scripting languages

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var lightLabel: UILabel!

    @IBAction func lightToggle(_ sender: UIButton) {
        if lightLabel.backgroundColor == ■ {
            lightLabel.backgroundColor = □
            sender.backgroundColor = □
            sender.setTitleColor(■, for: .normal)
            sender.setTitle("Off", for: .normal)
        }
        else {
            lightLabel.backgroundColor = ■
            sender.backgroundColor = ■
            sender.setTitleColor(□, for: .normal)
            sender.setTitle("On", for: .normal)
        }
    }
}
```

ViewController is a Swift class

- with a field (lightLabel) and method (lightToggle)

- note: Swift statements do NOT end with ;

Swift variables

Swift variables must be declared (using `var`) before use

```
var num = 5
var word = "foobar"
var flag = true
```

Swift is type-safe, the type of every value must be known by the compiler

- you can declare the type along with a variable

  `var age: Int` (primitives are `Int`, `Double`, `String`, `Bool`)

- but this is rarely done, as Swift performs type inferencing on assignments

naming conventions for variables are more liberal than Java

- no spaces or math operators, can't start with a digit
- all legal: `maxValue` `salary$` `mañana` `π` 🎲

Swift constants

Swift distinguishes between variables and constants

- constants, whose values don't change, are declared using `let`

```
let maxNum = 10              // constant

var sum = 0                  // variable
for i in 1...maxNum {
    sum += i
}
```

- if you declare a variable with `var` but never reassign it, the compiler will warn you

Swift provides the same math operators as Java

`+, -, *, /, %, +=, -=, *=, /=, %=` (note: no `++` or `--`)

- unlike Java, operators must be applied to values of same type

```
var sum1 = 2 + 3.5            // illegal
var sum2 = Double(2) + 3.5    // but can cast values
```

Swift control statements

Swift if-else and while are similar to Java, but tests don't require parentheses

- Boolean operators are the same: `==, !=, <, >, <=, >=, &&, ||, !`

```
var letter: String                var num = 1024
let x = 92                        var count = 0
if x >= 70 {                      while num > 1 {
    letter = "P"                      num /= 2
}                                     count += 1
else {                            }
    letter = "F"
}
```

for loops can step through a range

`x..<y` is exclusive on the upper end; `x...y` is inclusive

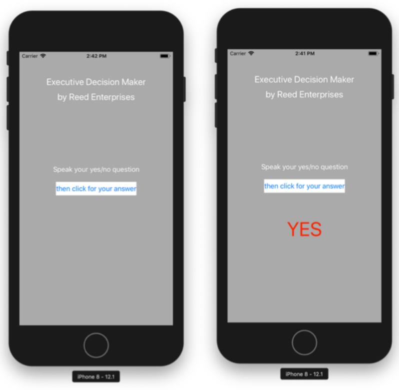
```
var sum1to9 = 0                   var sum1to10 = 0
for i in 1..<10 {                 for i in 1...10 {
    sum1to9 += i                      sum1to10 += i
}                                 }
```

33

HW 1.2

create an Executive Decision Maker app

- gray background & title with your name, centered at the top
- instructions and a button at the center
- an empty label centered, below the button
- when the button is clicked, display either YES, NO, or MAYBE in the label



```
let randomValue = Int.random(in: 1...3)
if randomValue == 1 {
    answerLabel.text = "YES"
}
else if randomValue == 2 {
    answerLabel.text = "NO"
}
else {
    answerLabel.text = "MAYBE"
}
```

34