Lecture 3: Create application using Pan Gesture

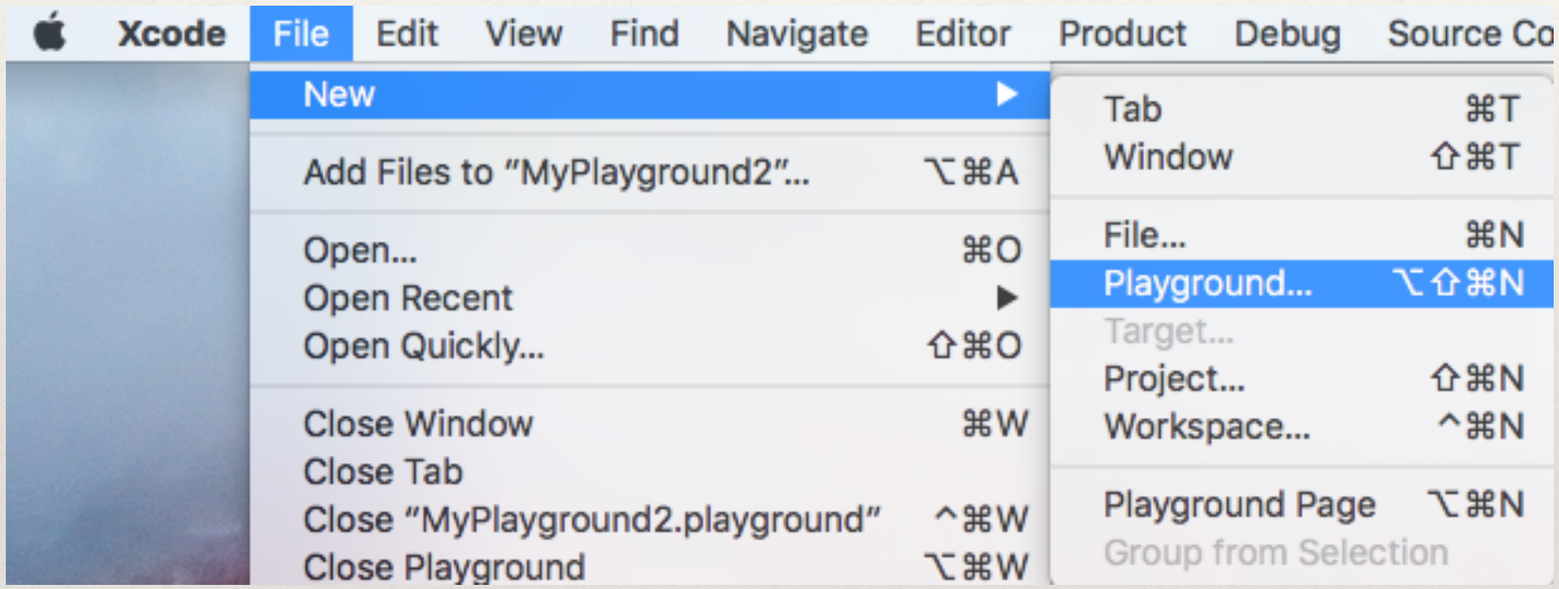
ADVANCED MOBILE DEVELOPMENT (iOS)

Lecturer: Christine Jiang

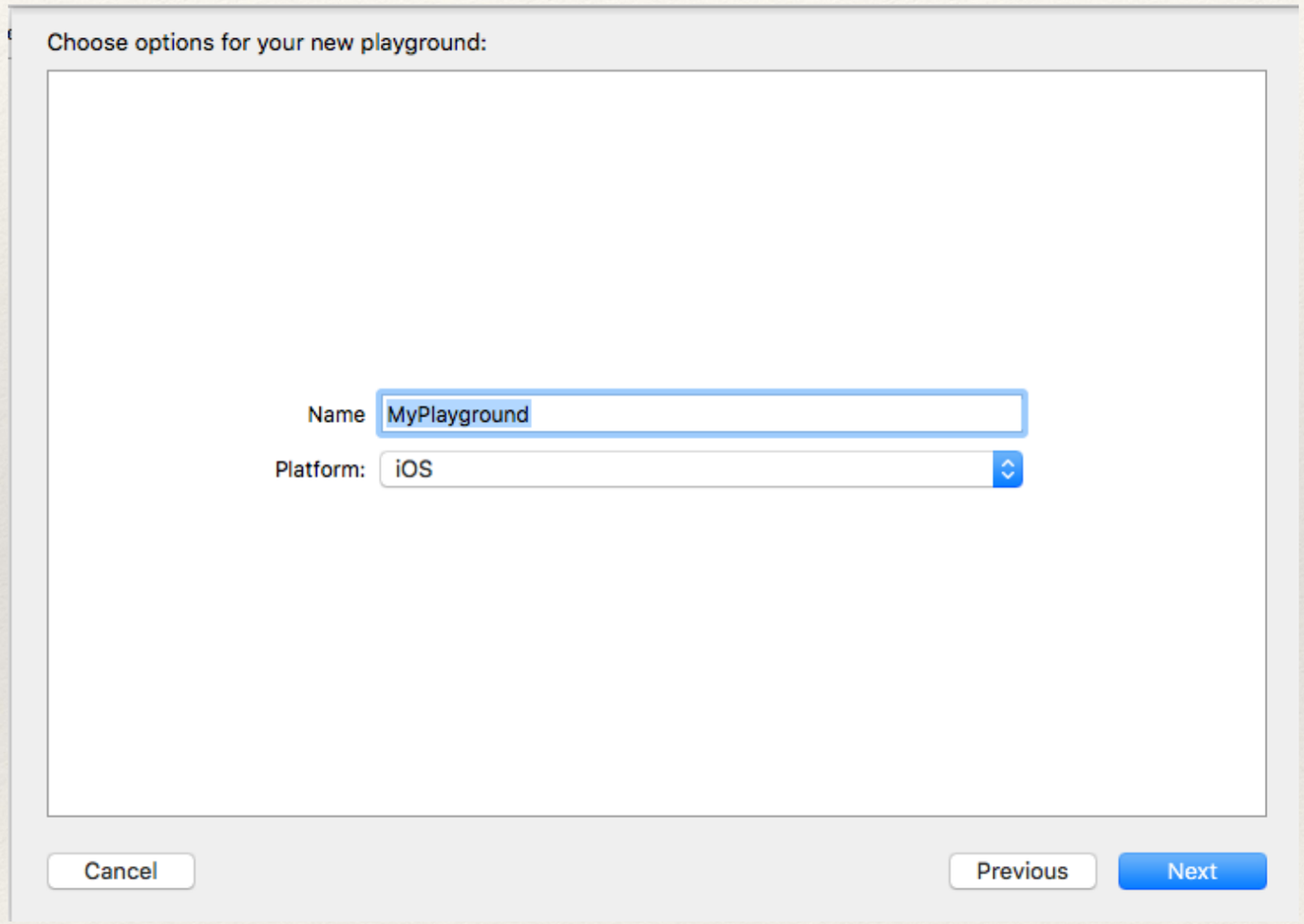
Senior Mobile Application Developer at IBM

Swift Playground (1)

- Create a Swift playground



Swift Playground (2)



Swift Syntax

- Array
- Set
- Dictionary
- String append
- Prefix
- Suffix
- if
- Switch Case
- Check API availability

String Array

```
let names = ["Anna", "Alice", "Nina", "Charles"]

let count = names.count

for i in 0..<count
{
   print("Person \(i + 1) is called \(names[i])")
}
```

Character Array

```
let catCharacters: [Character] = ["C", "a", "t", "!", "🐱"]

for character in "Dog!🐶".characters
{
  print(character)
}
```

String Appending

```
var welcome : String = "Hello world”

let exclamationMark: Character = “!"

welcome.append(exclamationMark)
```

Double()

```
let multiplier = 5

let message = "(multiplier) times 1.5 is \(Double(multiplier) * 1.5)”

print(message)
```

Prefix

```
let romeoAndJuliet = [
  "Act 1 Scene 1: Verona, A public place",
  "Act 1 Scene 2: Capulet's mansion",
  "Act 1 Scene 3: A room in Capulet's mansion",
  "Act 1 Scene 4: A street outside Capulet's mansion",
  "Act 1 Scene 5: The Great Hall in Capulet's mansion",
  "Act 2 Scene 1: Outside Capulet's mansion",
  "Act 2 Scene 2: Capulet's orchard",
  "Act 2 Scene 3: Outside Friar Lawrence's cell",
  "Act 2 Scene 4: A street in Verona",
  "Act 2 Scene 5: Capulet's mansion",
  "Act 2 Scene 6: Friar Lawrence's cell"
]

var act1SceneCount = 0
for scene in romeoAndJuliet {
  if scene.hasPrefix("Act 1 ") {
    act1SceneCount += 1
  }
}
print("There are \(act1SceneCount) scenes in Act 1")
```

Suffix

```
var mansionCount = 0
var cellCount = 0
for scene in romeoAndJuliet {
  if scene.hasSuffix("Capulet's mansion") {
    mansionCount += 1
  } else if scene.hasSuffix("Friar Lawrence's cell") {
    cellCount += 1
  }
}
```

Collection Types

- Swift provides three primary collection types, known as arrays, sets, and dictionaries, for storing collections of values.
- Arrays are ordered collections of values.
- Sets are unordered collections of unique values.
- Dictionaries are unordered collections of key-value associations.

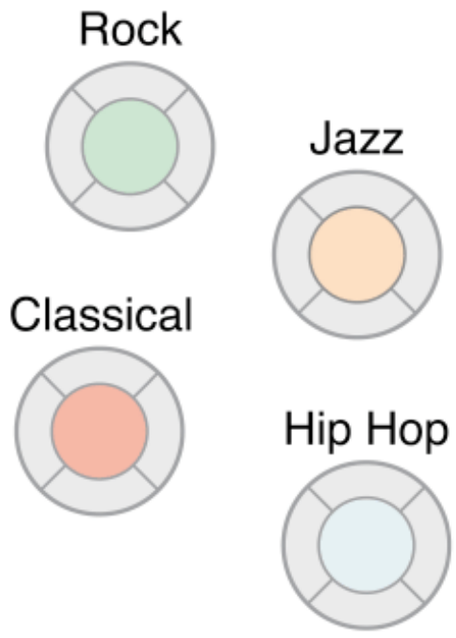
Collection Types

Array

Indexes	Values
0	Six Eggs
1	Milk
2	Flour
3	Baking Powder
4	Bananas

Set

Values



Dictionary

Keys	Values
YYZ	Toronto Pearson
DUB	London Heathrow
LHR	Dublin Airport

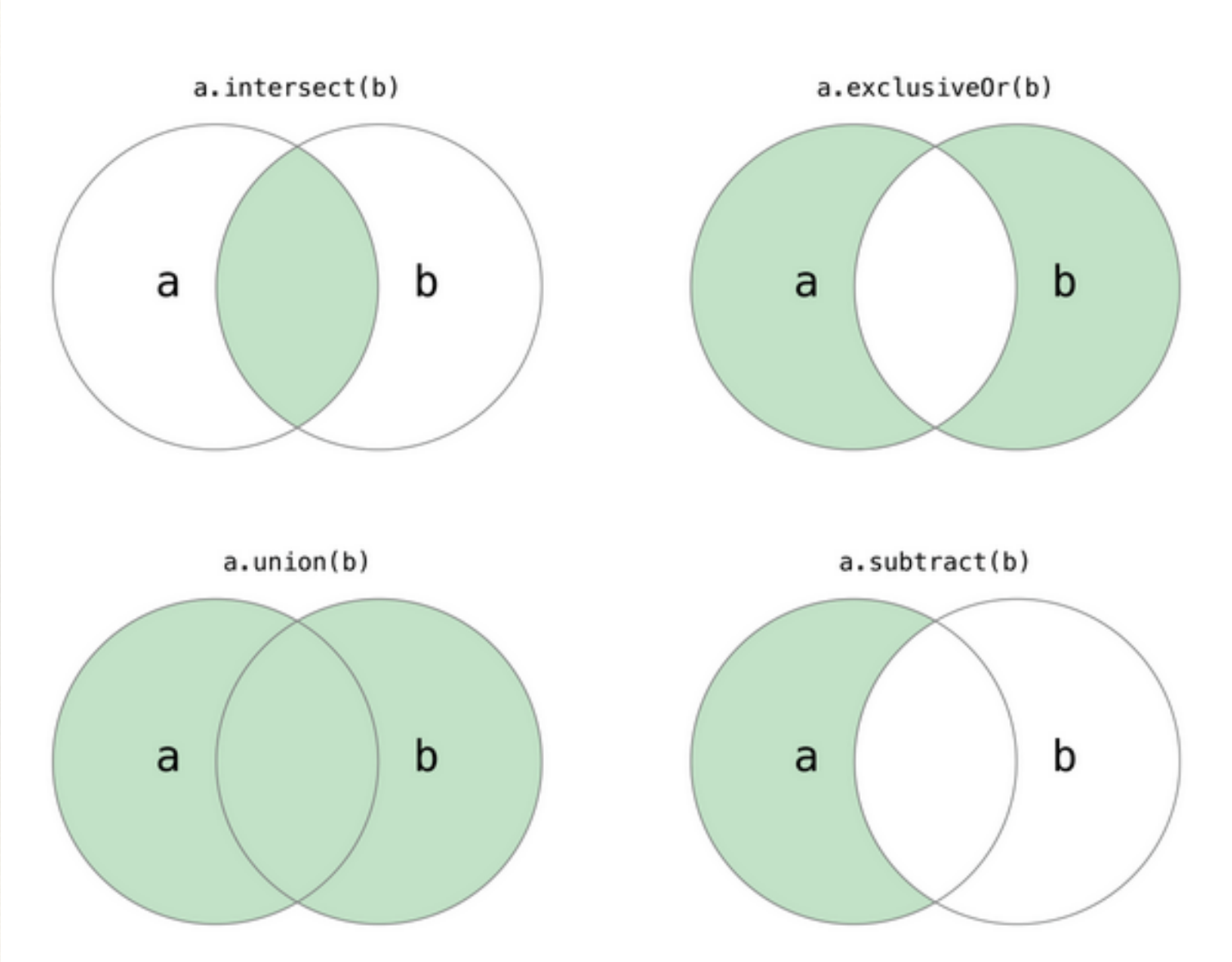
Creating an Empty Array

```
var someInts = [Int]()
print("someInts is of type [Int] with \(someInts.count) items.")
// Prints "someInts is of type [Int] with 0 items.”

someInts.append(3)
// someInts now contains 1 value of type Int

someInts = []
// someInts is now an empty array, but is still of type [Int]
```

Creating an Array with a Default Value

```
var threeDoubles = [Double](count: 3, repeatedValue: 3.0)
// threeDoubles is of type [Double], and equals [3.0, 3.0, 3.0]
```

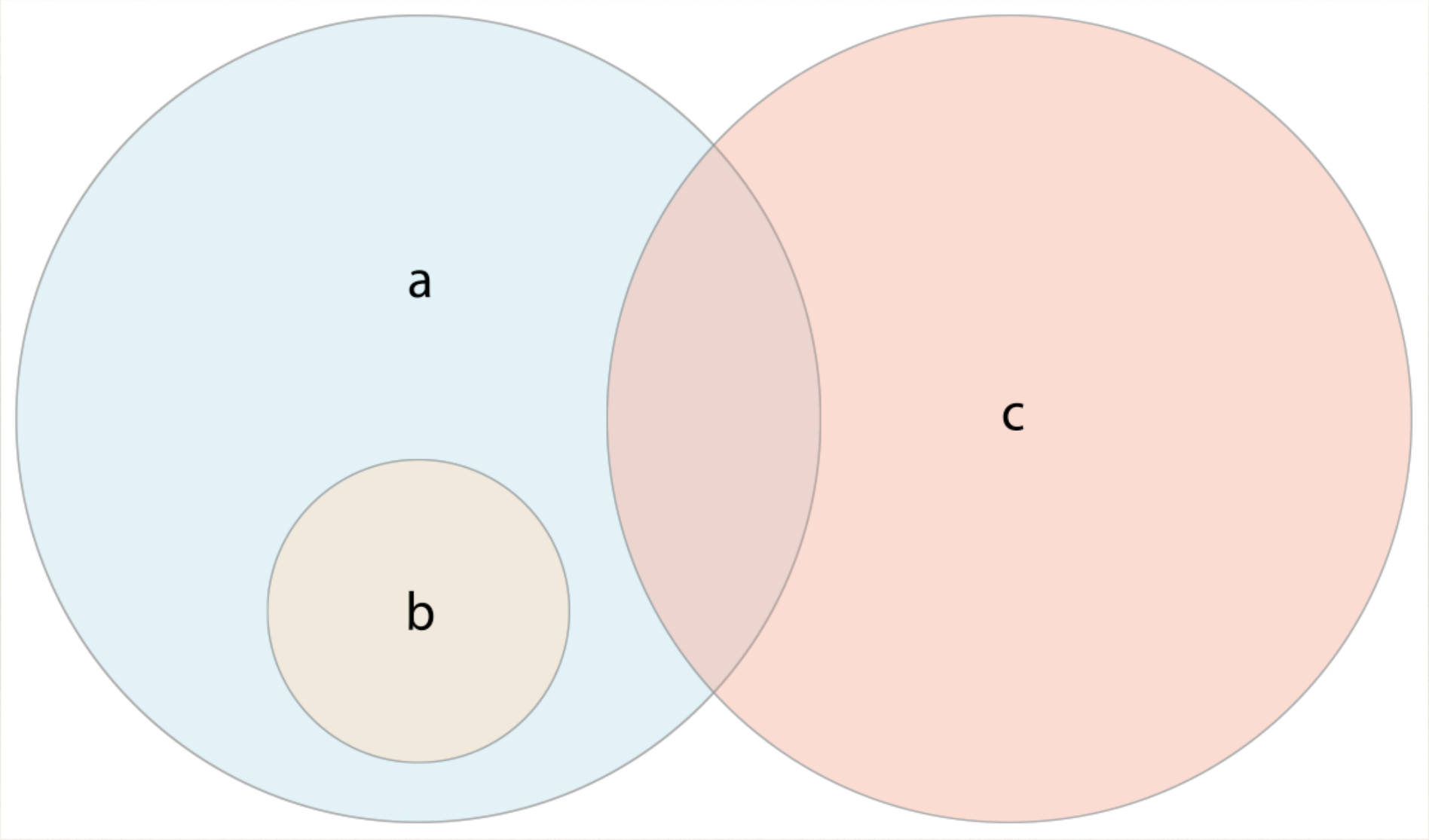
Fundamental Set Operations



Set Operation Examples

```
let oddDigits: Set = [1, 3, 5, 7, 9]
let evenDigits: Set = [0, 2, 4, 6, 8]
let singleDigitPrimeNumbers: Set = [2, 3, 5, 7]

oddDigits.union(evenDigits).sort()
// [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
oddDigits.intersect(evenDigits).sort()
// []
oddDigits.subtract(singleDigitPrimeNumbers).sort()
// [1, 9]
oddDigits.exclusiveOr(singleDigitPrimeNumbers).sort()
// [1, 2, 9]
```

Set Membership and Equality



Set Membership and Equality

```
let houseAnimals: Set = ["🐶", "🐱"]
let farmAnimals: Set = ["🐮", "🐔", "🐑", "🐶", "🐱"]
let cityAnimals: Set = ["🐦", "🐭"]

houseAnimals.isSubsetOf(farmAnimals)
// true
farmAnimals.isSupersetOf(houseAnimals)
// true
farmAnimals.isDisjointWith(cityAnimals)
// true
```

Creating an Empty Dictionary

```swift
var namesOfIntegers = [Int: String]()
// namesOfIntegers is an empty [Int: String] dictionary

namesOfIntegers[16] = "sixteen"
// namesOfIntegers now contains 1 key-value pair

namesOfIntegers = [:]
// namesOfIntegers is once again an empty dictionary of type [Int:
String]
```

Accessing & Modifying a Dictionary

```swift
var airports: [String: String] = ["YYZ": "Toronto Pearson", "DUB": "Dublin"]

print("The airports dictionary contains \(airports.count) items.")

if airports.isEmpty {
  print("The airports dictionary is empty.")
} else {
  print("The airports dictionary is not empty.")
}

airports["LHR"] = "London"
// the airports dictionary now contains 3 items

airports["LHR"] = "London Heathrow"
// the value for "LHR" has been changed to "London Heathrow”

if let oldValue = airports.updateValue("Dublin Airport", forKey: "DUB") {
  print("The old value for DUB was \(oldValue).")
}

if let removedValue = airports.removeValueForKey("DUB") {
  print("The removed airport's name is \(removedValue).")
} else {
  print("The airports dictionary does not contain a value for DUB.")
}
// Prints "The removed airport's name is Dublin Airport."
```

Iterating Over a Dictionary

```
for airportCode in airports.keys {
  print("Airport code: \(airportCode)")
}
// Airport code: YYZ
// Airport code: LHR

for airportName in airports.values {
  print("Airport name: \(airportName)")
}
// Airport name: Toronto Pearson
// Airport name: London Heathrow
```

If

```
var temperatureInFahrenheit = 90
if temperatureInFahrenheit <= 32
{
  print("It's very cold. Consider wearing a scarf.")
} else if temperatureInFahrenheit >= 86
{
  print("It's really warm. Don't forget to wear sunscreen.")
} else
{
  print("It's not that cold. Wear a t-shirt.")
}
// Prints "It's really warm. Don't forget to wear sunscreen."
```

Switch Case

```
let somePoint = (1, 1)
switch somePoint {
case (0, 0):
  print("(0, 0) is at the origin")
case (_, 0):
  print("(\(somePoint.0), 0) is on the x-axis")
case (0, _):
  print("(0, \(somePoint.1)) is on the y-axis")
case (-2...2, -2...2):
  print("(\(somePoint.0), \(somePoint.1)) is inside the box")
default:
  print("(\(somePoint.0), \(somePoint.1)) is outside of the box")
}
```

Checking API Availability

```
if #available(iOS 9, OSX 10.10, *)
{
   // Use iOS 9 APIs on iOS, and use OS X v10.10 APIs on OS X
}
else
{
   // Fall back to earlier iOS and OS X APIs
}
```

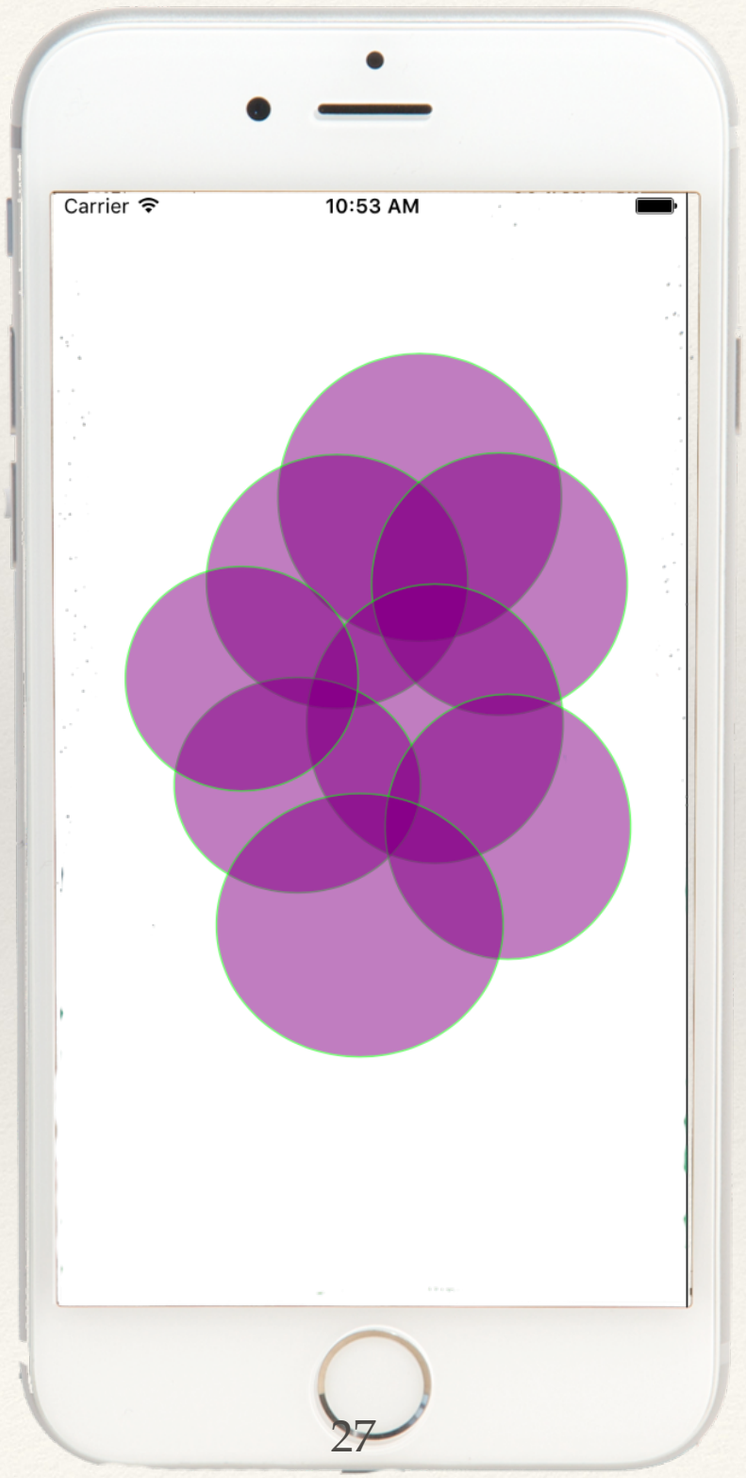
Gestures

Gesture	UIKit class
Tapping (any number of taps)	`UITapGestureRecognizer`
Pinching in and out (for zooming a view)	`UIPinchGestureRecognizer`
Panning or dragging	`UIPanGestureRecognizer`
Swiping (in any direction)	`UISwipeGestureRecognizer`
Rotating (fingers moving in opposite directions)	`UIRotationGestureRecognizer`
Long press (also known as "touch and hold")	`UILongPressGestureRecognizer`

Exercise 1

- Create an app to draw circles using Swift

Output



.locationInView

- Return the point computed as the location in a given view of the gesture represented by the receiver.

.translationInView

- The translation of the pan gesture in the coordinate system of the specified view.

Draw Circles

```
import UIKit

class ViewController: UIViewController
{
  var startPoint : CGPoint = CGPointZero
  var layer : CAShapeLayer?

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

  override func didReceiveMemoryWarning()
  {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
  }
  @IBAction func handlePan(sender: UIPanGestureRecognizer)
  {
    if sender.state == .Began
    {
      startPoint = sender .locationInView(sender.view)
      layer = CAShapeLayer()
      layer?.fillColor = UIColor .purpleColor().CGColor
      layer?.opacity = 0.5
      layer?.strokeColor = UIColor.greenColor().CGColor
      self.view.layer.addSublayer(layer!)
    }
    else if sender.state == .Changed
    {
      let translation = sender .translationInView(sender.view)
      layer?.path = (UIBezierPath(ovalInRect: CGRectMake(startPoint.x, startPoint.y, translation.x,
translation.y))).CGPath
    }
  }
}
```

Objective C

- Create a new application using Objective C
- Perform the same function as Swift application

Coordinates

- CGFloat

Just a floating point number
always use it for graphics

- CGSize

```
CGSize mySize = CGSizeMake(200.f, 200.f);

CGFloat height = mySize.height;
CGFloat width = mySize.width;
```

C struct with two CGFloats in it
width and height.

Coordinates

- CGPoint

```
CGPoint myPoint = CGPointMake(100.f, 100.f);

CGFloat posX =  myPoint.x;
CGFloat posY = myPoint.y;
```

C struct with two CGFloat in it
x and y.

- CGRect

```
CGRect rect = CGRectMake(20.f, 30.f, 100.f, 150.f);

CGFloat posX =  rect.origin.x;
CGFloat posY = rect.origin.y;
CGFloat width = rect.size.width;
CGFloat height = rect.size.height;
```

C struct with a CGPoint Origin and
a CGSize size.
x, y, width, height

Change Colors

```
shapeLayer.fillColor = [UIColor greenColor].CGColor;

shapeLayer.fillColor = [UIColor redColor].CGColor;

shapeLayer.fillColor = [UIColor yellowColor].CGColor;

shapeLayer.fillColor = [UIColor purpleColor].CGColor;

shapeLayer.fillColor = [UIColor darkGrayColor].CGColor;
```