GENERAL ASSEMBLY

DATA PERSISTENCE

Zeke Abuhoff

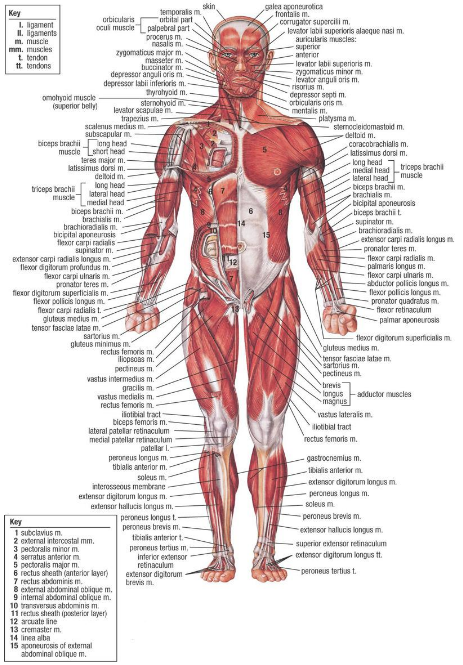
Lead iOS Instructor, General Assembly

LEARNING OBJECTIVES

+ Explain the utility of data serialization
+ Implement the NSCoding protocol
+ Save data to UserDefaults

DATA PERSISTENCE

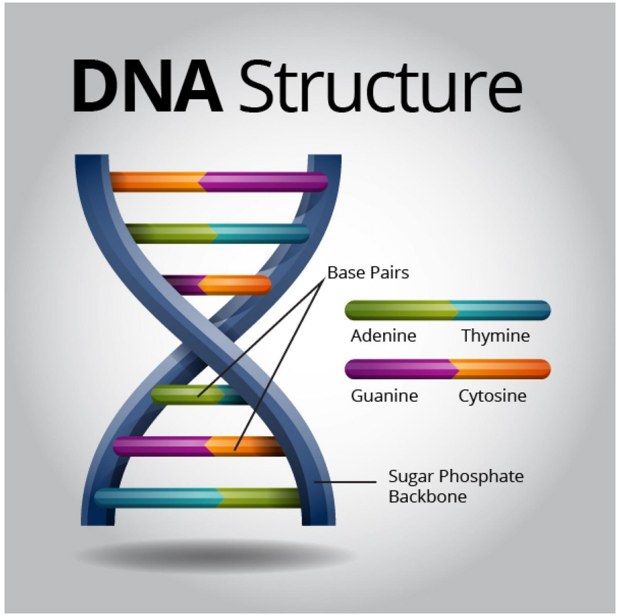
DATA SERIALIZATION



The human body is a hugely complex set of interwoven systems.

How are those systems recorded as information?

DATA SERIALIZATION



DNA is the body's way of recording information about how its systems are composed. Instead of being arranged in diagrams, its information in linear series.

Why is that?

DATA SERIALIZATION

0010110101101100101001101001011100

DNA stores information in series because it’s efficient.

The same dynamic holds true in programming - storing information in a serial format is efficient. We call the process of turning miscellaneous data into series **data serialization**.

0010110101101100101001101001011100

NSCODING

NSCoding is the protocol we use to serialize data in iOS.

By making a class conform to NSCoding, you establish the way by which the operating system will serialize instances of your class.

```
class Widget: NSObject, NSCoding {
    required init?(coder aDecoder: NSCoder) { }
    func encode(with aCoder: NSCoder) { }
}
```

NSCODING

Practice:

1) Create a new class called Bicycle.
2) Add properties to your Bicycle class that describe the qualities of a bicycle.
3) Make the class conform to NSCoding, encoding all its properties.

```
class Widget: NSObject, NSCoding {
    required init?(coder aDecoder: NSCoder) { }
    func encode(with aCoder: NSCoder) { }
}
```

SAVING TO DOCUMENTS

Your app has a file system. Use the FileManager class to get URLs that make use of this system.

```
    let documentsDirectoryURL: URL =
FileManager().urls(for: .documentDirectory, in:
.userDomainMask).first!

    let archiveURL: URL =
documentsDirectoryURL.appendingPathComponent("widget")
```

SAVING TO DOCUMENTS

Once you have the URL where you'll be storing your data, you can use NSKeyedArchiver to write to that storage.

```
    let saveSuccessful: Bool =
NSKeyedArchiver.archiveRootObject(widget, toFile:
Widget.archiveURL.path)
```

Be aware that the archiveRootObject method can fail. Check its return value.

LOADING FROM DOCUMENTS

When you want to load your stored data, use NSKeyedUnarchiver.

```
    let loadedWidget: Widget? =
NSKeyedUnarchiver.unarchiveObject(withFile:
Widget.archiveURL.path) as? Widget

    guard let successfullyLoadedWidget = loadedWidget
    else { return nil }
```

SAVING TO USERDEFAULTS

The documents directory isn't the only place to store your data.

Commonly, data that pertains to user preferences and settings is stored in **UserDefaults**.

```
    let savedData =
NSKeyedArchiver.archivedData(withRootObject: myWidget)
    let defaults = UserDefaults.standard
    defaults.set(savedData, forKey: "widget")
```

LOADING FROM USERDEFAULTS

To load data from UserDefaults, you reverse the process.

```
let widgetData = defaults.object(forKey: "widget")

if let presentWidgetData = widgetData as? Data {
    let myWidget =
NSKeyedUnarchiver.unarchiveObject(with: widgetData) as!
Widget
   return myWidget
}
```

USERDEFAULTS

Practice:

1) Create three instances of your Bicycle class.
2) Place the Bicycle instances in an array.
3) Store the array in UserDefaults.
4) Load the array from UserDefaults and print the properties of each bicycle instance.