Override Init

Introduction

Welcome to Lesson 4 of The UIKIt Fundamentals Part 1 Object Oriented Programming. In this lesson, you are going to learn what it truly means by the world `super`. A lot of beginners often just type because of what they have seen on StackOver Flow or tutorials, today we are going to dive in what's really going on. There are a lot of examples, so let's get started with me, Bob the Developer.

Lecture Notes

Problem

Problem: Super... Super.init?

UIViewController

You might have seen something like this below. Many wonder what `super` and `override` stand for. Let us find out.

```
import UIKit

class MyViewController: UIViewController {
 override func viewDidLoad() {
   super.viewDidLoad()
   print("Hello")
 }
}
```

Create Vehicle Class

Create a class called `Vehicle`. It contains one gettable property and one method.

```
class Vehicle {

 var description: String {
   return "Hello, I'm moving at 30km/hr"
 }

 func warning() {
   print("Be careful, I'm going pretty fast")
 }

}
```

Subclass

You may customize inherited properties and methods by inserting `override`. You may access inherited properties and methods by calling `super`.

```
class ToyCar: Vehicle {

 let price = "$100"

 override var description: String {
   return "\(super.description), I'm \(price)"
 }

 override func warning() {
   super.warning() // "Be careful, I'm going pretty fast"
   print("Let me just tell you, I'm not dangerous much")
 }

}
```

Subclass Object

Create an object to check if the method and property have been overriden in the subclass, `ToyCar`.

```
let myCar = ToyCar()
myCar.description
myCar.warning()
```

Super Init

You may override init from super class. The number one rule is: you must put value to every property

Design Super Class

Create a class that contain one property called, `origin`.

```
class Human {
 var origin: String

 init(enterOrgin: String) {
   origin = enterOrgin
 }
}
```

Design Subclass

Create a subclass, called `Korean` that inherits from `Human`. The `Korean` class contains a property called, `name`. However, when you initialize, you must initialize the `origin` property from the `Human` class by calling `super.init`.

```
class Korean: Human {

 let name: String

 init(enterName: String) {
   name = enterName
   super.init(enterOrgin: "Bob the Developer")
 }

 init(enterName: String, myOrigin: String) {
   name = enterName
   super.init(enterOrgin: myOrigin)
 }


}
```

Create Object

There are two init methods in the `Korean` class. You may choose any since both initializes the `origin` property from the `Human` class.

```
let bob = Korean(enterName: "Bob the Dev")
let bobby = Korean(enterName: "Bob the Dev", myOrigin: "Korean")
```

Override Init

Similar to how you may override a method and property, you may include additional code when you call the overriden init method.

Design Base Class

Design a class called, `Tesla`. It contains a property called, `numberOfWheels`.

```
class Tesla {
  var numberOfWheels: Int

  init(enterWheelNumber: Int) {
    numberOfWheels = enterWheelNumber
  }
}
```

Override Init

You may override the init method within the `Tesla` class. Let us add a print statement when you call the init method.

```
class ModelS: Tesla {
  override init(enterWheelNumber: Int) {
    super.init(enterWheelNumber: 500)  // Calling the init method
    print("Yo, I've cerated this object bruh")
  }
}

let myFutureSexyCar = ModelS(enterWheelNumber: 500)
myFutureSexyCar.numberOfWheels
```

Source Code

1204_Override Init/Method

Conclusion

Now, you understand. What's really going on with super.init, super.viewDidLoad, override, super.property. By now, I hope you understand **why** we use super. Again, I like to use this analogy when it comes to iOS development. Building an app in this ecosystem is like operating a microwave. We definitely need to know how to read manuals which is analogous to being able to read the Swift programming language. To heat up your food, all you have to do is press a couple of buttons and then lights comes off from the side of the wall and the food starts heating up. We don't know exactly how the details work such as particle movements and electromagnetism, but we do know how to operate, In iOS programming, our job is to learn how to operate and understand why Apple engieers have implemented such features such as Error Handling and Computer Property. Remember this statement,

> Focus on why we use it over why it works. You understand their motives, not the principle behind it because you can't understand why it works. It just works. Microwave jsut works. When you tap on your phone, it just works. Why they did it because it's convenient and easy to use it. Why it works? It's too complicated. I don't know.

Stay Connected

If you'd like to be on my mailing list and receive personal updates on upcoming books and courses, feel free to send me an email at `bobleesj@gmail.com`

Personal Website | Facebook Like | YouTube Subscribe | Twitter Follow
Instagram Follow | LinkedIn Connect | Medium Read