More Swift

Vasilica Costescu

vasilica.costescu@gmail.com

@iOSStepByStep

What we'll learn today

- Optionals
- Guard
- Type casting
- Extensions
- Debugging

Optionals

A type in Swift with two possibilities:

- either you have a value
- you have no value

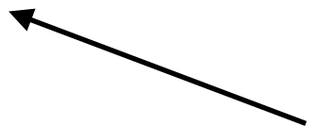
Declared like this:

```
var money: Int? // it's either a integer or no value
var money: Int // it's always an integer
```

Specifying the type of an optional

```
var name = "Vasi"
```

```
var name = nil
```



‘nil’ requires a contextual type

Always specify the optional type

```
var name: String? = nil
```

How to use optionals

```
var publicationYear: Int?
let unwrappedYear = publicationYear!
print(unwrappedYear)
```

force unwrapping

If you force unwrap an optional that doesn't have a value, the app will crash!

Preferred way

```
var publicationYear: Int?
if let unwrappedYear = publicationYear {
    print(unwrappedYear)
}
```

Optional chaining

```
class Person {
    var residence: Residence?
}

class Residence {
    var numberOfRooms = 1
}
```

```
let ana = Person()

let numberOfRooms = ana.residence!.numberOfRooms
```

Preferred way:

```
let numberOfRooms = ana.residence?.numberOfRooms
```

Implicitly unwrapped optionals

```
class ViewController: UIViewController {
    @IBOutlet var label: UILabel!
}
```

Type Casting

```
class CustomViewController: UIViewController {
    var customTitle: String?
}
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    let destinationViewController = segue.destination as!
CustomViewController
    destinationViewController.customTitle = "Hello World"
}
```

Preffered way:

```
if let destinationViewController = segue.destination as?
CustomViewController {
    destinationViewController.customTitle = "Hello World"
}
```

Guard statements

```
func divide(firstNumber: Int, secondNumber: Int) {
    if secondNumber != 0 {
        print("\(firstNumber/secondNumber)")
    }
}
```

Using guard:

```
func divide(firstNumber: Int, secondNumber: Int) {
    guard secondNumber != 0 else {
        return
    }
    print("\(firstNumber/secondNumber)")
}
```

Using guard with optionals

```
func greet(person: String?, location: String?) {
    if let person = person {
        if let location = location {
            print("Hello \(person) from \(location)")
        } else {
            print("Hello \(person)")
        }
    } else {
        print("Hello there")
    }
}
```

Using guard with optionals

```
func greet(person: String?, location: String?) {
    guard let person = person else {
        print("Hello there")
        return
    }
    guard let location = location else {
        print("Hello \(person)")
        return
    }

    print("Hello \(person) from \(location)")
}
```

Extensions

Adding a method:

```
extension String {
    func funkyString() -> String {
        return "🐱 \(self) 🐱"
    }
}
print("Vasi".funkyString())
```

Adding a computed property:

```
extension UIColor {
    static var favouriteColor: UIColor {
        return .red
    }
}
print("My favourite color \(UIColor.favouriteColor)")
```

Organise code with extensions

```
protocol Movable {
    func fly()
    func run()
}

class MyClass {
    var title: String?
}

extension MyClass: Movable {
    func fly() {}
    func run() {}
}
```

Fun Challenge!

Download project from

https://github.com/costescv/Challenge

1. Have a look at the project and try to understand what is going on. (You could start with the Main.storyboard)

2. Fix all the compile errors.

3. Try and run the app. What happens?

4. Look closely in the UsersViewController. What should it do and what does it do now?

Let’s talk about breakpoints!

```
44  }
45
46  extension UsersViewController: UITableViewDataSource {
47      func tableView(_ tableView: UITableView, numberOfRowsInSection secti
48          return users!.count
49      }
50
51      func tableView(_ tableView: UITableView, cellForRowAt indexPath: Ind
            UITableViewCell {
52          let cell = tableView.dequeueReusableCell(withIdentifier: "UserCe
                UserTableViewCell
53
54          let user = users![indexPath.row]
55          cell.nameLabel.text = "Vasi"
56          cell.photoImageView.image = UIImage(named: user.photo)
57
58          return cell
59      }
60  }
61
```

Back to our challenge

1. Now that cellForRow is called, we have another crash. Find the reason why and fix it.

2. Run the app. Can you see any rows on screen? If not, what's missing?

3. Could you change users variable from an optional to a non optional value? How would that work?

4. Make sure you're not using any force unwrapping in the extension for UITableViewDataSource conformance.

5. Run the app. Why do all the rows have the name "Vasi"? Fix it to use the correct name.

Fix UserDetailsViewController

1. When tapping on a cell, what happens? Does it go to another screen? If not, fix it.

2. Look at prepare(for segue:) in UsersViewController. Can you change it so it doesn't use as! for a "ShowUserDetails" screen?

3. Set a breakpoint and check if the method in the extension for UITableViewDelegate is called. If not, find out why and fix it.

4. Go to the Main.storyboard and look at the UserDetailsViewController. The photo of a user should be aligned to the left and the labels should stay as close as they are not. What constraints are you missing?

5. Add outlets for the image view and the two labels in UserDetailsViewController.

6. Use the user variable to set the new outlets and display information about a user. When you go to see the user details page, you should see the user photo, its name and the description. You can set values for the outlets in viewDidLoad method.

Fixing CreateUserViewController

1. What happens if you forget to enter something in a text field and you press “Save”? Use a guard to fix it.

2. The CreateUserViewController has a delegate variable. Is there a class that sets itself as the delegate for our view controller? If not, which class do you think it should do this? How would you start?

3. Make the delegate implement the save user method. It should add the user to the list of existing users. Check that the newly created user is shown on screen.

Is there anything else you would improve with this code?