Short Note on Optional, Forced Unwrap, Optional Binding, Optional Chaining and Nil-Coalescing Operator.

Optionals:

In Swift, the each and every variable that we initialize must comes with some value. We cannot be declared it as a empty variable.

For example,

```
let a : String          //We can't declare like this, it will throw a error
```

In order to fix this we error we have to add a Question Mark (?) Symbol. This Symbol is known as **Optional.**

```
let a : String?         //Here the variable a is declared as Optional.
```

i) Example Program:

```
import UIKit

class ViewController: UIViewController {

    var a : String?

    Override func viewDidLoad() {

        print(a)

    super.viewDidLoad()

}

}
```

Output:

```
Nil
```

ii) Example Program:

```
import UIKit
class ViewController: UIViewController {
    var a : String?
    Override func viewDidLoad() {
        a = “abc”
        print(a)
    super.viewDidLoad()
}
}
```

Output:

Optional(abc)

Forced Unwrapping:

Forced unwrap is used to get a original value from the optionally declared variable.

The exclamation symbol (!) is used to denote forced unwrap operation.

i) Example Program:

```
import UIKit
class ViewController: UIViewController {
    var a : String?
    Override func viewDidLoad() {
        a = “abc”
        print(a!)
```

```
        super.viewDidLoad()
    }}
```

Output:

```
    abc
```

Optional Binding:

The “If Let” function is known as Optional Binding.

Optional binding is one of the Error Handling Method.

i) Example Program:

```
    import UIKit
    class ViewController: UIViewController {
        var a : String?
        Override func viewDidLoad() {
            if let b = a {
            print(b)
            }
            a = “abc”
            print(a!)
        super.viewDidLoad()
    }
    }
```

Output:

```
    abc
```

- The If Let part will be executed only if the optionally initialized variable “a” has some value. Otherwise it won’t executed as well as doesn’t throw any errors
- The Application will not get crash because of this optional binding method.

Optional Chaining:

More than one optional binding condition involves.

i) Example Program:

```
import UIKit

class ViewController: UIViewController {

    var username : String?

    var password : String?

    Override func viewDidLoad() {

        username = “admin”

        if let uservalue = username, let passvalue = password {

        print(uservalue)

        print(passvalue)

        }

        print(username!)

    super.viewDidLoad()

}

}
```

Output:

```
admin
```

Nil-Coalescing Operator: (??)

It is just a shorthand for saying !=nil. First it checks if the return value is nil, if it is indeed nil, then the left value is presented, and if it is nil then the right value is presented.

i) Example Program:

```
import UIKit

class ViewController: UIViewController {

Override func viewDidLoad() {

        var name : String?

        name = "2"

        print(name ?? 3)

    super.viewDidLoad()

}

}
```

Output:

2

ii) Example Program:

```
import UIKit
class ViewController: UIViewController {
Override func viewDidLoad() {
        var name : String?
        print(name ?? 3)
    super.viewDidLoad()
}
}
```

Output:

3