QUICK TIPS : iOS

# WHAT ARE INITIALIZERS IN SWIFT?

Rohit Saini
iOS Developer

## INIT:-

- Initialization is the process of preparing an instance of a class, structure, or enumeration for use. This process involves setting an initial value for each stored property on that instance and performing any other setup or initialization that is required before the new instance is ready for use.

## TYPES OF SWIFT INITIALIZERS:-

- Designated Initializers
- Convenience Initializers
- Required Initializers
- Failable Initializer

QUICK TIPS : iOS

## Designated Initializers:-

- Primary initializer of the class.
- A class can have more than one designated initializer.
- Every class must have at least one designated initializer

Example:-

```
class Student{
    var name : String
    var degree : String
    //Designated Initializer
    init(name : String, degree: String) {
        self.name = name
        self.degree = degree
    }
}
//Student Instance created using Designated Initializer
let student = Student(name: "Rohit", degree: "B.Tech")
```

## Convenience Initializers:-

- These are secondary, supporting initializers for a class.
- They must call a designated initializer of the same class.
- These are optional and can be used for a custom setup.

QUICK TIPS : iOS

Example:-

```
class Student{
    var name : String
    var degree : String
    //Designated Initializer
    init(name : String, degree: String) {
        self.name = name
        self.degree = degree
    }
    //Convenience Initializer
    convenience init()
    {
        //Custom Setup by applying uppercased() on name and degree string
        //Calling Designated Initializer using self.init
        self.init(name: "Convenience".uppercased(), degree: "iOS".uppercased())
    }
}
//Student Instance created using Convenience Initializer.
var student = Student()
student.degree // "IOS"
student.name // “CONVENIENCE”
```

## Required Initializers:-

- Each subclass must implement that initializer.
- You do not have to provide an explicit implementation of a required initializer if you can satisfy the requirement with an inherited initializer.
- Use only when your class requires a certain setup when subclassed.

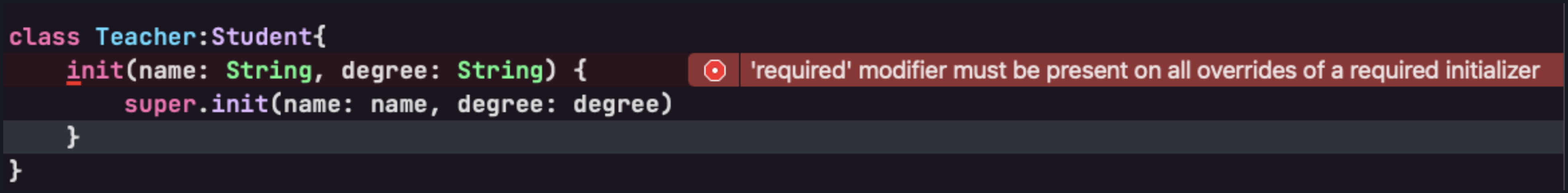
QUICK TIPS : iOS

Example:-

```
class Student{
    var name : String
    var degree : String
    //Required Initializer
    required init(name : String, degree: String) {
        self.name = name
        self.degree = degree
    }
}

class Teacher:Student{
    init(name: String, degree: String) {
        super.init(name: name, degree: degree)
    }
}
```

- When we subclass the Student class for Teacher class and add a default init inside teacher class we will receive an error.
- Which says 'required' modifier must be present on all overrides of a required initializer.
- We can resolve this by adding required keyword before our teacher class init method.



```
class Teacher:Student{
    init(name: String, degree: String) {        'required' modifier must be present on all overrides of a required initializer
        super.init(name: name, degree: degree)
    }
}
```

QUICK TIPS : iOS

## Solution:-

```
class Teacher:Student{
    required init(name: String, degree: String) {
        super.init(name: name, degree: degree)
    }
}
```

## Failable Initializer:-

- A Failable initializer creates an optional value of the type it initializes.
- Failable Initialization can Fail for various reasons: Invalid parameter values, absence of an external source etc.
- A Failable Initializer creates an optional value of the type it initializes.

## QUICK TIPS : iOS

```swift
class Student {
    let name: String
    //Failable Initializer
    //This will return nil if string is empty
    init?(name: String) {
        if name.isEmpty { return nil }
        self.name = name
    }
}
//Creating Instance of Failable Initializer.
var name  = Student(name: "")
if name ≠ nil {
    print("init success")
}
else{
    print("init failed")
}
```

LIKE, SHARE, ENJOY!

CONNECT.IOS

Scan to Connect

Rohit Saini
iOS Developer