Swift Optional

In Swift, an **Optional** is a type that can hold either a value or **nil** to indicate the absence of a value. Optionals are a powerful feature in Swift that help you handle the possibility of **nil** values in a type-safe manner.

Declaring Optionals

You declare an optional by appending a **?** to the type of the value that might be **nil**.

```
var optionalString: String?
```

Unwrapping Optionals

To use the value inside an optional, you need to "unwrap" it. There are several ways to unwrap an optional:

Forced Unwrapping:
Use **!** to force unwrap an optional. This should be used only when you are certain the optional contains a non-**nil** value, as it will crash your program if the optional is **nil**.

```
var optionalString: String? = "Hello"
if optionalString != nil {
    print(optionalString!) // prints "Hello"
}
```

Optional Binding:
Use **if let** or **guard let** to safely unwrap an optional.

```
// Using if let
if let unwrappedString = optionalString {
    print(unwrappedString) // prints "Hello"
}

// Using guard let
func printString(optionalString: String?) {
    guard let unwrappedString = optionalString else {
        print("optionalString is nil")
        return
    }
    print(unwrappedString) // prints the value if it's not nil
}
```

Optional Chaining:
Use optional chaining to call properties, methods, and subscripts on optional that might currently be **nil**.

```
var optionalString: String? = "Hello"
let uppercaseString = optionalString?.uppercased()
print(uppercaseString) // prints "Optional("HELLO")"
```

Nil Coalescing Operator:
Use **??** to provide a default value in case the optional is **nil**.