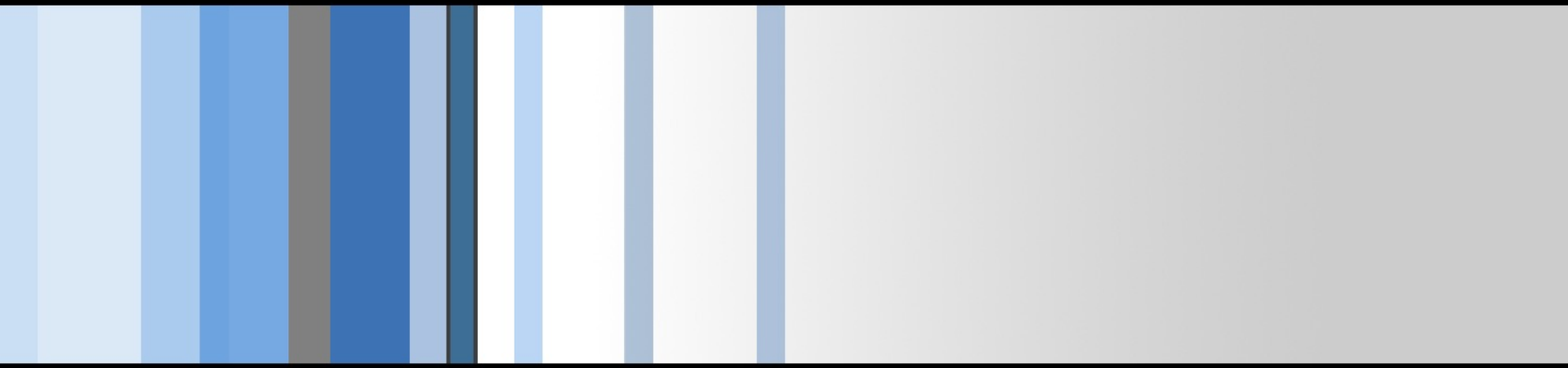
Class 4

CSC 496: iOS App Development

Swift Fundamentals: Arrays, Loops, Strings, Optional Type

Si Chen (schen@wcupa.edu)



Creating and initializing an array

Basic syntax for declaring and initializing arrays:

```
// Syntax
var arrayName: [Type] = [value1, value2, value3]
```

```
var fruits = ["Apple", "Banana", "Cherry"]
```

or:

```
var fruits: [String] = ["Apple", "Banana", "Cherry"]
```

Accessing and Modifying Arrays

```swift
var fruits: [String] = ["Apple", "Banana", "Cherry"]

// Accessing
let firstFruit = fruits[0]

// Modifying
fruits.append("Orange")

fruits.remove(at: 1)
```

Iterating Over Arrays

```
for fruit in fruits {
    print("I have a \(fruit).")
}
```

Array Methods and Properties

```swift
// Count
let numberOfFruits = fruits.count

// Empty Check
let isEmpty = fruits.isEmpty
```

Array Equality

- == can be used to check the equality of two arrays.

```
// array equality
var array1 = ["AAA", "BB", "C"]
var array2 = ["AAA", "BB", "C"]
print(array1 == array2)
```

For-in Loop

Basic syntax of a for-in loop:

```
5  var myFirstInt: Int = 0
6  for i in 1...5{
7      myFirstInt += 1
8      print(myFirstInt)
9  }
```

For-in Loop

```
5  var myFirstInt: Int = 0
6  for _ in 1...5{
7      myFirstInt += 1
8      print(myFirstInt)
9  }
```

Use the explicit iterator **i** if you want to refer to the iterator within your loop's code block or the wildcard **_** if you do not.

A for-in loop with a where clause.

```
for i in 1...100 where i % 3 == 0{
    print(i)
}
```

While Loop

```
var i = 1
while i < 6
{
    print(i)
    i += 1
}
```

repeat-while Loops

```
var j = 0
repeat{
      print(j)
      j += 1
} while j < 6
```

Exercise: Array Manipulation with Loops and Built-in Methods in Swift

- **Objective:**
  Learn how to manipulate arrays using loops and explore Swift's built-in Array methods for more convenient operations.
- **Task 1: Reverse Array Using Loop**
  1. Create an array named **toDoList** with the following elements: *"Take out the trash", "Pay bills", "Cross off finished items".*
  2. Write a for-in loop to reverse the order of the elements in toDoList.
  3. Log the reversed array to the console.
- **Task 2: Reverse Array Using Built-in Method**
  1. Examine the Swift Array documentation to find a more convenient built-in method for reversing an array.
  2. Use this method to reverse toDoList again and log the result to the console.
- **Task 3: Shuffle Array**
  1. Refer to the Array documentation to find a built-in method for shuffling the elements in an array.
  2. Use this method to rearrange the items in toDoList into a random order.
  3. Log the shuffled array to the console.

Try ChatGPT

Strings

```
let playground = "Hello Playground"
print(playground)

var greeting = "Hello, playground"
greeting += "!"
print(greeting)
```

Characters

- When talking about text, the term “string” is short for “string of characters.
- In Swift, a **String** is a collection of instances of the **Character** type.

```
// Characters
for c in greeting{
    print("\(c)")
}
```

Counting characters & Indices and ranges

```
// counting characters
let b = greeting.count
print(b)
```

```
//Index and range
//Finding the fifth character
let start = greeting.startIndex
let end = greeting.index(start, offsetBy: 4)
let fifthCharacter = playground[end]
```

Swift uses a type called **String.Index** to keep track of indices in string instances. The Swift compiler **will not let** you access a specific character on a string via a **subscript index.**

Counting characters & Indices and ranges

```
//Index and range
//Finding the fifth character
let start = greeting.startIndex
let end = greeting.index(start, offsetBy: 4)
let fifthCharacter = playground[end]

// pulling out a range
let range = start...end
let firstFive = playground[range]
print(firstFive)
```

Substrings

```
//Index and range
//Finding the fifth character
let start = greeting.startIndex
let end = greeting.index(start, offsetBy: 4)
let fifthCharacter = playground[end]

// pulling out a range
let range = start...end
let firstFive = playground[range]
print(firstFive)
```

- What is the type of firstFive?

firstFive is a String.SubSequence

Exercise: Working with Empty Strings in Swift

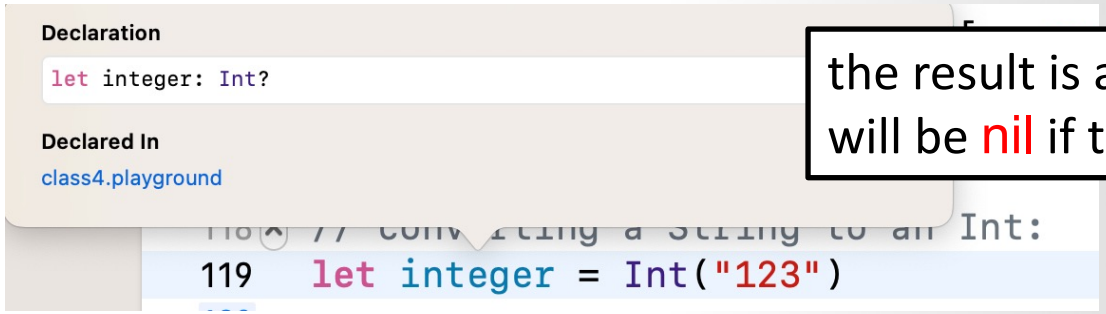
- Objective:
  - Learn how to initialize an empty string and utilize its **startIndex** and **endIndex** properties to check for emptiness. This skill is particularly useful when designing input forms to prevent users from submitting blank entries.
- Task:
  1. Initialize a new String variable named empty and assign it an empty string value: let empty = "".
  2. Use the **startIndex** and **endIndex** properties of the empty string to ascertain whether the string is genuinely empty.

Optional type

- A type that represents either a **wrapped value** or the **absence of a value**.
- You use the **Optional type** whenever you use **optional values**.
- Swift's type system usually shows the wrapped type's name with a trailing question mark (?) instead of showing the full type name.

What does that mean?

```
// converting a String to an Int:
let integer = Int("123")
```



the result is an optional Int (Int?), which will be nil if the conversion fails

Optional type

```
let validString = "42"
if let validInteger = Int(validString) {
    print("Successfully converted to \(validInteger)")
} else {
    print("Conversion failed")
}
```

Optional type

```
let validString = "42"
if let validInteger = Int(validString) {
    print("Successfully converted to \(validInteger)")
} else {
    print("Conversion failed")
}
```

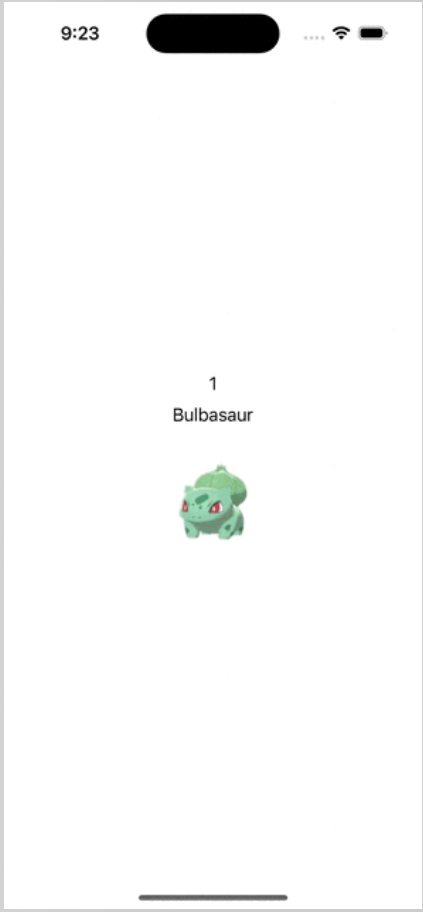
nil-coalescing operator -- ??

```
let invalidString = "abc"
let convertedInteger = Int(invalidString) ?? 0
print(convertedInteger)
```

1.Create a string that contains invalid characters for an integer.
2.Use the nil-coalescing operator (??) to provide a **default value if the conversion fails**.

Pokédex version 2

Objective:
To create an iOS app that displays the name and profile picture of a Pokémon based on the user-inputted Pokémon ID.



Instructions:

- **Download and Setup Project:**

Download the **pokedex_ver_2.zip** file from our class website. Unzip the file and open the project in Xcode.

- **Examine the Pokemon.Swift File:**

Open the **Pokemon.Swift** file in the project. Locate the array **firstGenPokemonNames** which contains the names of the first-generation Pokémon.

- **Implement User Input and Display in ContentView.swift:**

In **ContentView.swift**, write code to accomplish the following:

- Use the firstGenPokemonNames array to find the name of the Pokémon corresponding to the entered ID.
- Display the Pokémon's name and its corresponding profile picture based on the entered ID.

For example, if the user inputs Pokémon ID = 1, the app should display "Bulbasaur" along with its profile picture.

Tips:
You may use the .font(.custom("Pokemon-Pixel-Font", size: 16)) modifier to set a Pokemon font.
How would you handle invalid input (try using nil-coalescing operator ??)
How would you handle edge cases where the user input exceeds 151 or falls below 0?

Submission:
No submission needed.

Q & A