Introducción a estructuras de datos: Diccionarios

Diccionarios

> Son una colección de datos como los arrays, pero cada elemento se divide en una clave (la cual funciona como un identificador) y un valor.

Estructura:

Al igual que los arrays, comienzan y terminan con corchetes. Los elementos dentro se dividen con una coma y para separar la clave del valor, usamos dos puntos (:).

> En los diccionarios, definimos los tipos de datos que van a poseer la clave y el valor. Se dice que un diccionario `“va de un tipo de dato a otro”`.

Generamos un diccionario vacío para ver esto:

```
var dic:[String:String] = [:]

//Definimos la variable y el diccionario. Luego, indicamos los tipos de datos que seran las claves y los valores.
```

En el diccionario vacío especificamos que el diccionario va de String a String, es decir, todas las claves y los valores serán de tipo String.

Si generamos un diccionario vacío, Swift solicita obligatoriamente que se especifiquen los tipos de dato.

También podemos generar un diccionario de esta manera:

Acá, Swift automáticamente va a inferir que es un diccionario de String a String, dado que esta vez cuenta con valores que le indican los posibles tipos de datos.

```
var dic = ["Jorge":"Argentino"]

//En este caso, ya cargamos directamente los datos.
```

¿Cómo accedemos a los valores?

Podemos utilizar las claves para acceder a la información que se encuentra en el diccionario.

```
let edades = ["Jorge":56,"Ana":35]

print(edades["Jorge"])
```

¿Qué pasa cuando intentamos leer un valor que no se encuentra en el diccionario?

Swift nos devuelve `nil (null)` * .
A veces nos puede servir, pero podemos asignar también valores por default en caso de que falte una clave.

Ejemplo:

Podemos saber la edad de "Jorge" accediendo a la clave.

Sin embargo, si queremos saber el valor de algún elemento que no sabemos si esta en la lista, podemos hacer esto:

```
let edades = ["Jorge":56,"Ana":35]

print(edades["Jorge"])
print(edades["Mario", default:0])
```

En el ejemplo de arriba, lo que va a pasar es que si Swift no encuentra esa clave en el diccionario, nos va a devolver 0 (default value) en lugar de nil.

💡 **Super importante:** El tipo de dato del default tiene que coincidir con el tipo de dato que tienen los valores en el diccionario. Si es un diccionario de String a Int, entonces el default value debe ser un Int.

💡 Aclaración sobre Nil :

Nil por el momento vamos a interpretarlo como la ausencia de valor, pero más adelante vamos a investigar realmente su significado y propósito

No siempre vamos a utilizar los valores por default, pero si pueden ser útiles en ciertas ocasiones.