Enums

Hay una estructura de datos más que no presentamos, y estos son los `Enumerators` (o simplemente enums).

> 💡 Estos son una forma de definir un universos de posibilidades a un grupo limitado.

Supongamos que queremos representar un caso de éxito o error en un proceso.

Podemos hacer esto:

```
let resultado = "Exito"
```

Pero que pasa cuando alguien utiliza otro nombre y pasa esto?:

```
let resultadoDos = "exitoso"
let resultadoTres = "Exitoso"
```

Los tres resultados que obtuvimos están escritos diferente, por lo que significan cosas diferentes.

Los enums nos permiten solucionar este problema.

Estructura:

> 💡 Declaramos la palabra Enum y el nombre del caso (con mayúscula). Luego, entre llaves, definimos los diferentes casos, como en el Switch.

Ahora, podemos declarar los resultados de la siguiente manera:

```
enum Resultado: String {
    case exito
    case error
}
```

Luego, cuando usemos el enum, elegimos una de las opciones!

```
let resultadoCuatro = Resultado.error
```

Veamos como lo podemos combinar con un tema que ya vimos: Switch

> 💡 En lugar de definir múltiples condiciones en el switch, podemos llamar al caso del enum y dependiendo cual sea, indicar una funcionalidad.

Por ejemplo:

```
enum CompassPoint {
    case north
    case south
    case east
```

```
    case west
}
```

Ahora, creamos el `Switch` y llamamos a cada caso del `Enum`.

```
directionToHead = .south

switch directionToHead {
case .north:
    print("Lots of planets have a north")
case .south:
    print("Watch out for penguins")
case .east:
    print("Where the sun rises")
case .west:
    print("Where the skies are blue")
```

Esto permite que la cantidad de errores sea menor y que el código sea más claro!

Veamos un tema más relacionado con Enums: Valores asociados.

Además de tener un valor simple, los Enums pueden tener valores asociados para cada caso.

Esto te permite agregar información adicional a los Enums.

Veamos como funciona:

```
enum Activity {
    case bored
    case running(destination: String)
    case talking(topic: String)
    case singing(volume: Int)
}
```

Esto nos permite aclarar información y ser más precisos acerca del caso.

Podemos indicar la información asociada cuando lo declaramos:

```
let talking = Activity.talking(topic: "Football")
```