

Funciones

Higher-order

Control de Flujo e Iteración,
Funciones y métodos



Yosafat Coronel
GDSC ESCOM IPN
GitHub: YosafatM



 Developer Student Clubs
ESCOM - IPN

Itinerario

- Abstracción de objetos
- Funciones Higher-Order
- Método forEach
- Ciclo for each
- Función de flecha
- Función anónima
- Tipo de dato: Function



Todo es un Objeto



Todo es un Objeto

Hasta ahora, todo lo que hemos usado tiene alguna definición dentro de Dart, esas definiciones son llamadas **Clases** (y nosotros podemos hacer las nuestras), por ejemplo las clases:

Number, String, List

Todo es un Objeto

Cuando nosotros hemos estado creando enteros, listas y cualquier cosas, estamos **creando un Objeto** (para ser precisos una **instancia**). Excepto null si usamos null safety.

```
int valor = 10;
```

```
String name = 'Lalo';
```

Todo es un Objeto

Si seguimos extendiendo esto, entonces, las funciones también son objetos (Y LO SON), por lo que podemos pasarlas como argumentos y asignarlas a variables (aunque se puede, el linter nos va a recomendar declarar la función y no una variable):

```
var fun = () {
  print('Hola');
};
```

Use a function declaration to bind a function to a name.

```
void main() {
  fun();
}
```

Funciones Higher-Order

Si las funciones son objetos, quiere decir que las podemos recibir como parámetros y regresar con return. Cuando esto sucede (alguno de los dos casos o ambos a la vez), estas funciones reciben el nombre de funciones Higher-Order

Funciones Higher-Order

Existen varias funciones HO ya hechas en Dart y podemos hacer las nuestras. Una muy usada es el método `forEach`, donde recibe una función que regresa `void` y tiene como parámetro un genérico `E` (de `Element`):

```
void forEach(void action(E element)) {  
  for (E element in this) action(element);  
}
```

For each y forEach



forEach en acción

En este ejemplo, notemos que **nombre** no tiene tipo (es dynamic al ser inferido). Y su salida es esta:

Console

Ana
Javier
Lalo

```
const names = [  
  'Ana',  
  'Javier',  
  'Lalo'  
];  
  
void iterar (nombre) {  
  print(nombre);  
}  
  
void main() {  
  names.forEach(iterar);  
}
```

For each en acción

Dentro del método encontramos un ciclo diferente, este ciclo es conocido como el for each (no confundir con el método que usamos antes, el anterior es un método, este es un ciclo):

```
for (<type> <id> in <iterable>)
```

For each en acción

El equivalente usando un ciclo for each, var será inferido como String, y aquí no podemos usar un índice, sino que se itera por dentro (en ciertas ocasiones más rápido).

```
const names = <String>[
  'Ana',
  'Javier',
  'Lalo'
];
void main() {
  for (var name in names) {
    print(name);
  }
}
```

Función anónima y flecha



Función anónima y flecha

En la clase anterior, vimos que cuando solamente tenemos un **statement (instrucción)** en la función, podemos pasarla a **función de flecha** para ahorrar espacio y ser más claro en la lectura:

```
int s(int a, int b) => a+b;
```

```
int s(a, b) => a+b;
```

Infiere que son tipo num

Función anónima y flecha

Hasta ahora, hemos usado funciones nombradas o con nombre, sin embargo, **a veces no queremos volverlas a usar**, como en las funciones Higher-Order (ya sea con return o como parámetro), para ello simplemente usamos las funciones anónimas:

Simplemente le quitamos el nombre

`() {[body]}`

`() => <stm>;`

Función anónima y flecha

```
void main() {  
    //Se infiere que e es String  
    names.forEach((e) => print(e));  
  
    //No imprime, infiere print()  
    names.forEach((e) => print);  
  
    //Infiere que print(element)  
    names.forEach(print);  
}
```

Dart puede inferir los argumentos en la última forma.

Sin embargo, puede hacer cosas que no queremos si no somos cuidadosos, como en la forma 2.

Tipo Function



Tipo Function

El tipo de dato de una función es... Function, si solo ponemos Function entonces recibiremos (o retornaremos) cualquier función, desde la v1.23 podemos ser específicos de qué esperamos:

```
int Function(int, int) op  
[type] Function([list]) <id>
```

Dudas y preguntas



¡Muchas gracias!

