Apunte: Principales Funciones de Orden Superior en Dart

¿Qué es una función de orden superior?

Una función de orden superior es una función que:

- **Recibe una o más funciones como argumento**, o
- **Devuelve una función como resultado**.

Estas funciones permiten escribir código más expresivo, reutilizable y conciso, sobre todo cuando trabajamos con **listas** y **colecciones** en Dart.

Principales funciones de orden superior en Dart (Colecciones)

Las funciones de orden superior más utilizadas trabajan sobre **List**, **Set** y **Map**.
A continuación, las más importantes:

Función	Descripción	Ejemplo
`forEach()`	Ejecuta una función por cada elemento de la colección.	`lista.forEach((e) => print(e));`
`map()`	Transforma cada elemento y retorna una nueva colección.	`lista.map((e) => e * 2).toList();`
`where()`	Filtra los elementos que cumplen una condición.	`lista.where((e) => e % 2 == 0).toList();`
`reduce()`	Combina todos los elementos en un solo valor usando una función acumuladora.	`lista.reduce((a, b) => a + b);`
`fold()`	Similar a `reduce`, pero permite especificar un valor inicial.	`lista.fold(0, (a, b) => a + b);`
`any()`	Retorna `true` si **algún** elemento cumple la condición.	`lista.any((e) => e > 5);`
`every()`	Retorna `true` si **todos** los elementos cumplen la condición.	`lista.every((e) => e > 0);`
`take(n)`	Retorna los primeros `n` elementos.	`lista.take(3).toList();`
`skip(n)`	Omite los primeros `n` elementos.	`lista.skip(2).toList();`

Ejemplos prácticos

forEach()

```
var lista = [1, 2, 3];

lista.forEach((e) => print(e));
```

map()

```
var lista = [1, 2, 3];

var duplicada = lista.map((e) => e * 2).toList();

print(duplicada); // [2, 4, 6]
```

where()

```
var lista = [1, 2, 3, 4, 5];

var pares = lista.where((e) => e % 2 == 0).toList();

print(pares); // [2, 4]
```

reduce() y fold()

```
var lista = [1, 2, 3, 4];

// Suma total con reduce

var suma = lista.reduce((a, b) => a + b);

print(suma); // 10

// Suma total con fold (valor inicial 0)

var sumaFold = lista.fold(0, (a, b) => a + b);

print(sumaFold); // 10
```

Funciones anónimas y expresiones lambda

Se usan mucho con funciones de orden superior.

Ejemplo de **lambda**:

```
lista.map((e) => e * 3).toList();
```

Ventajas de usar funciones de orden superior

* Código más **limpio** y **conciso**
* Mejor manejo de listas y colecciones
* Ideal para **callbacks** y **eventos** en Flutter
* Fomenta un estilo de programación **funcional**

Resumen rápido

Función	Uso típico
`forEach()`	Imprimir o ejecutar acción por cada elemento
`map()`	Transformar elementos
`where()`	Filtrar elementos
`reduce()` / `fold()`	Combinar elementos
`any()` / `every()`	Verificar condiciones
`take()` / `skip()`	Seleccionar subconjuntos

Tip

Usa **`.toList()`** al final de `map()` o `where()` si quieres una lista nueva.