
Kotlin

¿Qué es Kotlin?

- Lenguaje de programación de código abierto.
- Desarrollo aplicaciones para dispositivos Android.
- Compatible con Java. Permite utilizar bibliotecas y código Java.

Declaración de variables

Kotlin

```kotlin
var texto = "Hola mundo"                    // Infiere el tipo
var texo: String = "Hola mundo"             // Indicar explícitamente
```

Java

```java
String texto = "Hola mundo";
```

Tipos de variables

```
val texto: String = "Texto"
val numero: Int = 1       // Long
val numeroDecimal: Float = 1.5f
val numeroDecimal2: Double = 1.5
val booleano: Boolean = true
```

Variables / Constantes

Kotlin

```kotlin
var editable: String = "Hola mundo"
editable = "Hasta luego"      // OK

val noEditable: String = "Hola mundo"
noEditable = "Hasta luego"    // ERROR
```

Java

```java
String editable = "Hola mundo";

final String noEditable = "Hola mundo";
```

Nulos / Lateinit

```
// El valor puede ser String o Nulo
var miApodo: String? = "Rey"

// Se inicializará más adelante
// No puede ser Nulo
lateinit var textoPendiente: String
```

Condición IF

```
val numero: Int = 1
if (numero < 10) {
    println("El número $numero es MENOR que 10")
} else {
    println("El número $numero es MAYOR que 10")
}
```

Condición WHEN / SWITCH

Java

```java
int diaDeLaSemana = 2;

switch (diaDeLaSemana) {
    case 1:
        System.out.println("Lunes");
        break;
    case 2:
        System.out.println("Martes");
        break;
    case 4:
        System.out.println("Jueves");
        break;
    case 6:
    case 7:
        System.out.println("Fin de semana");
        break;
    default:
        System.out.println("xxxxx");
}
```

Kotlin

```kotlin
val diaDeLaSemana: Int = 2
when (diaDeLaSemana) {
    1 -> { println("Lunes") }
    2 -> { println("Martes") }
    4 -> { println("Jueves") }
    6, 7 -> { println("Fin de semana") }
    else -> { println("xxxxx") }
}
```

```kotlin
when (diaDeLaSemana) {
    in 1..5 -> { println("Lunes - Viernes") }
    else -> { println("Fin de semana") }
}
```

Arrays / Listas

```
// Array --> Solo se pueden modificar los valores
var array = arrayOf("Uno", "Dos", "Tres")
array[0] = "UNO"

// ArrayList --> Editable
var arrayList = arrayListOf<String>("Uno", "Dos", "...")
arrayList.removeAt(2)
arrayList.add("Tres")
```

```
// List --> No editable
var lista = listOf("Uno", "Dos", "Tres")

// MutableList --> Editable
var listaEditable = mutableListOf<String>("Uno", "Dos", "...")
listaEditable.removeAt(2)
listaEditable.add("Tres")
```

Maps

Kotlin

```kotlin
var miMapa: Map<String, Int> = mapOf()
miMapa = mapOf("Uno" to 1, "Dos" to 2, "Tres" to 3) // No se podrán añadir más valores

var miMapa2: Map<String, Int> = mutableMapOf("Uno" to 1, "Dos" to 2, "Tres" to 3)
miMapa2["Cuatro"] = 4
miMapa2.put("Cinco", 5)
```

Java

```java
Map<String, Integer> miMapa = new HashMap<>();
miMapa.put("Uno", 1);
miMapa.put("Dos", 2);
miMapa.put("Tres", 3);
```

Bucles FOR / WHILE

```
val miArray = listOf("Uno", "Dos", "Tres", "Cuatro")
for (valorActual in miArray) {
    println(valorActual)
}
```

```
// De 0 a 4
for (i in 0 until 5) {
    println(i)
}

// De 0 a 5
for (i in 0..5) {
    println(i)
}
```

```
var contador = 1

while (contador <= 5) {
    println(contador)
    contador++
}
```

Funciones

Kotlin

```kotlin
fun funcion1() {}
funcion1()

fun funcion2(nombre: String) {
    println(nombre)
}
funcion2("Iker")

fun funcion3(primerNum: Int, segundoNum: Int) : Int {
    val suma: Int = primerNum + segundoNum
    return suma
}
```

Java

```java
public static void funcion2(String nombre) {
    System.out.println(nombre);
}
funcion2("Iker");

public static int funcion3(int primerNum, int segundoNum) {
    int suma = primerNum + segundoNum;
    return suma;
}
int resultado = funcion3(3, 5);
```

Clases

Kotlin

```kotlin
package...

class Persona(val nombre: String, val edad: Int) {
    fun saludo() {
        println("Hola, me llamo $nombre")
    }
}
```

```kotlin
val iker = Persona("Iker", 25)
println(iker.nombre) // Iker
iker.saludo()
```

Java

```java
public class Persona {
    private String nombre;
    private int edad;

    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    public void saludo() {
        System.out.println("Hola, me llamo " + nombre);
    }

    public String getNombre() {
        return nombre;
    }
}
```

```java
Persona iker = new Persona("Iker", 25);
System.out.println(iker.getNombre()); // Iker
iker.saludo();
```

Enum class

Estructura de datos con valores fijados.

```kotlin
enum class Direccion {
    NORTE, SUR, ESTE, OESTE;

    fun detalle() : String {
        return when (this) {
            NORTE -> "La dirección es NORTE"
            SUR -> "La dirección es SUR"
            ESTE -> "La dirección es ESTE"
            OESTE -> "La dirección es OESTE"
        }
    }
}

private fun enumClases() {
    var miDireccion: Direccion? = null
    miDireccion = Direccion.NORTE
    println("Dirección NAME: ${miDireccion.name}")          // NORTE
    println("Dirección ORDINAL: ${miDireccion.ordinal}")    // 0
    println("DETALLES: ${miDireccion.detalle()}")          // La dirección es NORTE
}
```

Java

```java
public enum Direccion {
    NORTE,
    SUR,
    ESTE,
    OESTE
}
```

Class / Data class / Interface

- Class ➔ Plantilla para la creación de objetos, junto con sus propios métodos.

- Data class ➔ Almacena datos de manera simple, generalmente inmutables.

```
data class Producto(val nombre: String, val precio: Double, val marca: String)
val camiseta = Producto("Camiseta", 12.99, "Primark")
```

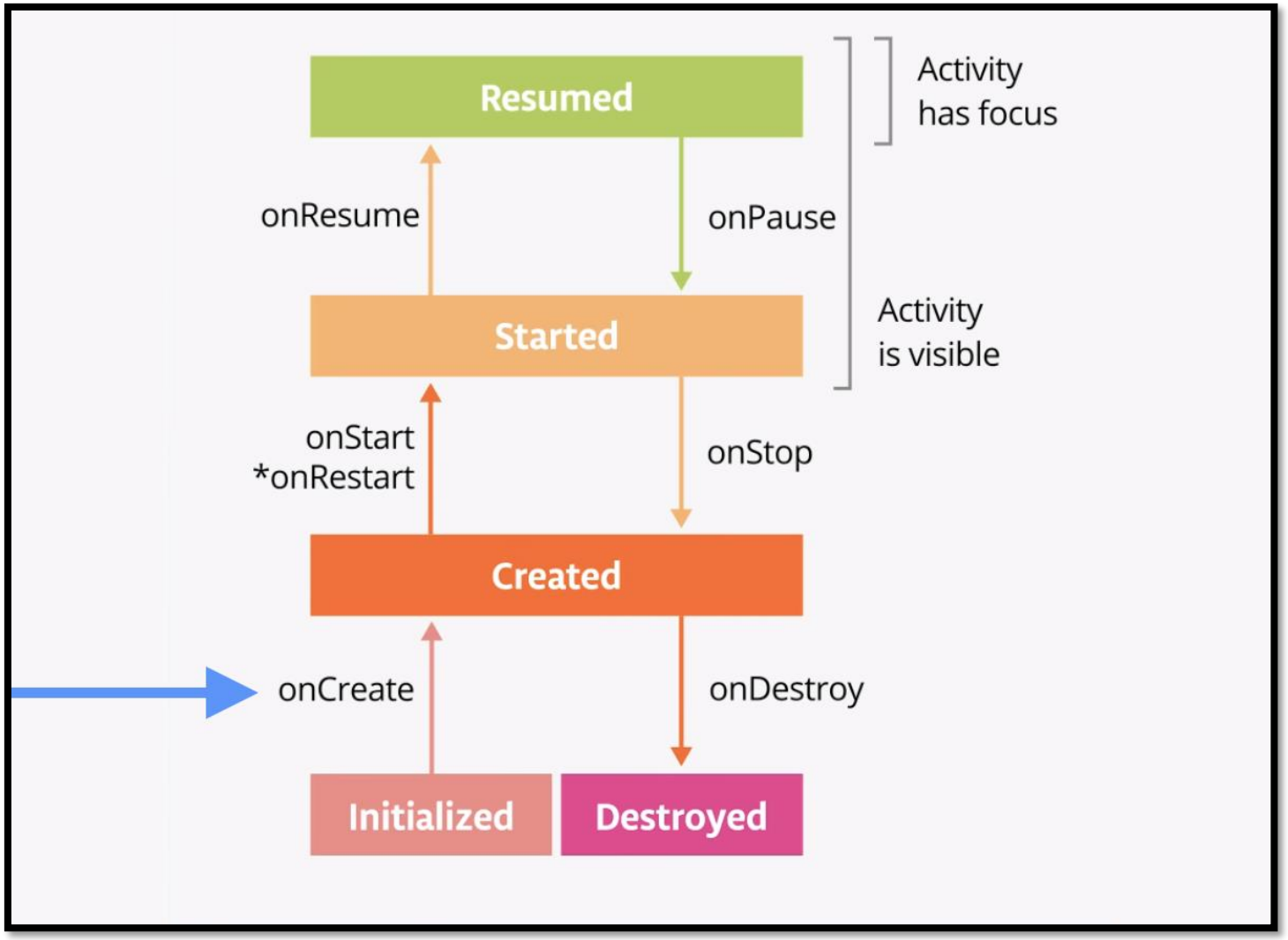
- Interface ➔ Define comportamientos y métodos.

Lambdas

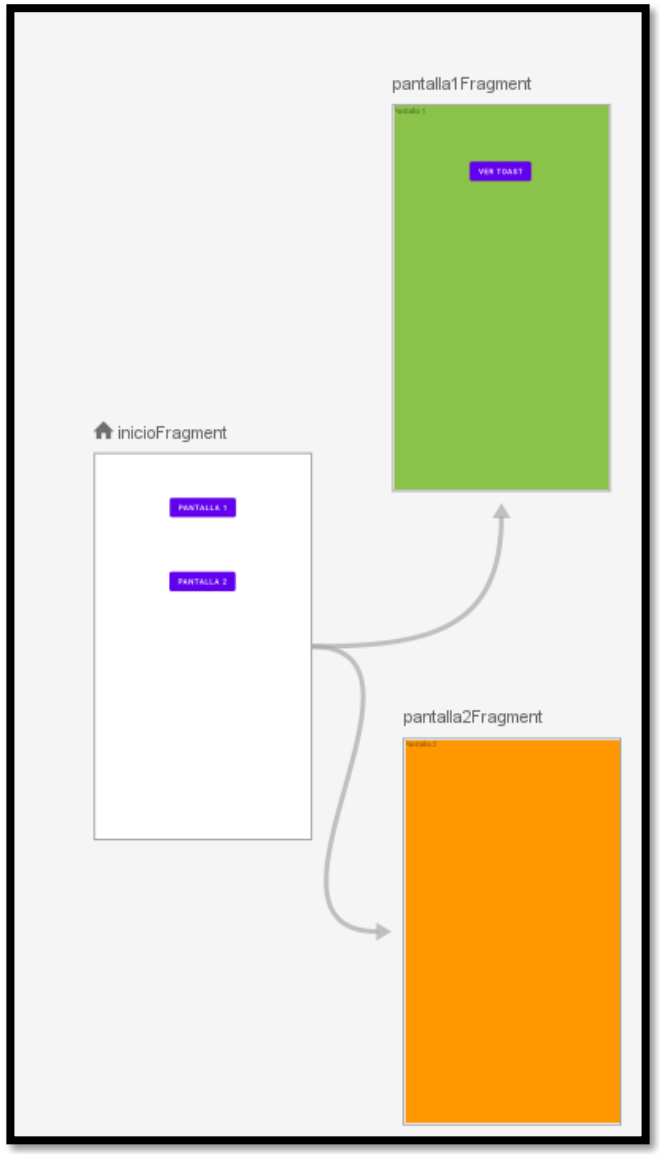
Funciones que se pueden crear en cualquier sitio y se definen en una sola línea.

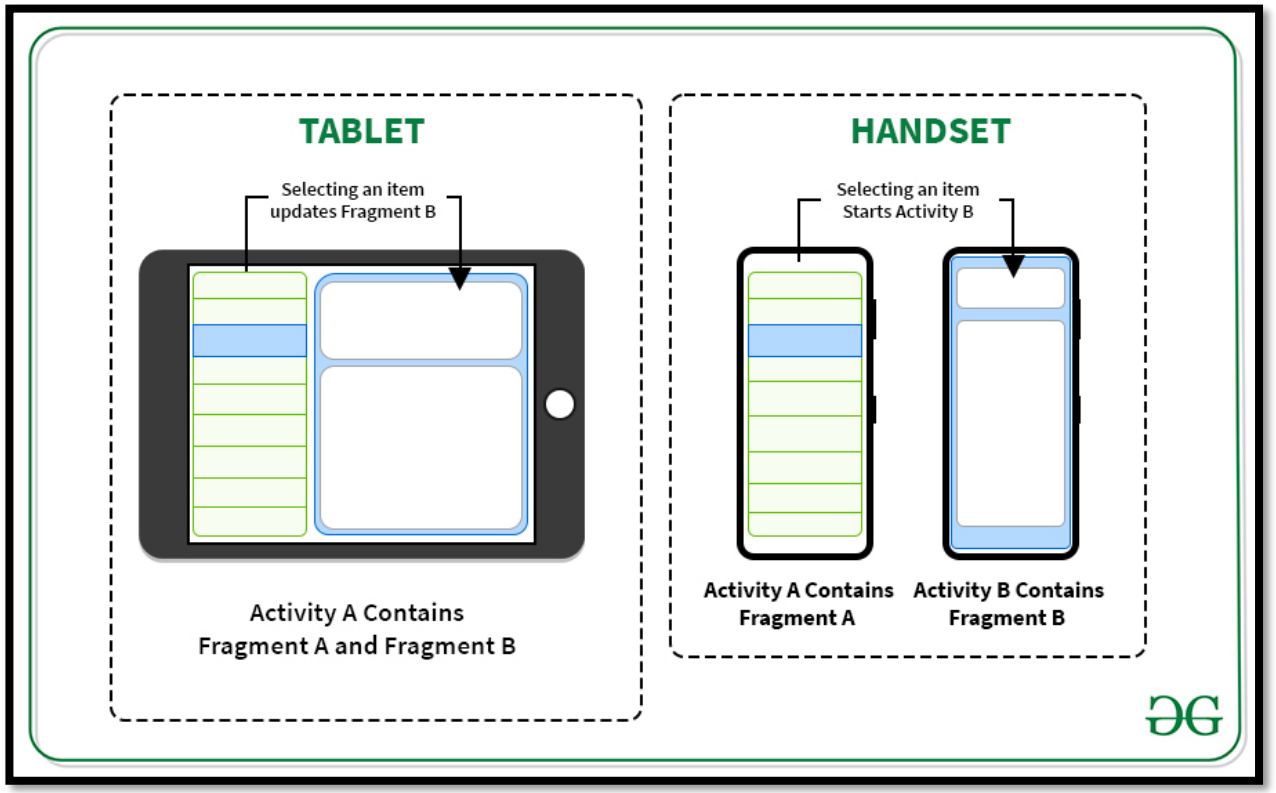
```
val suma: (Int, Int) -> Int = { a, b -> a + b }
val resultado = suma(3, 5)
```

Ciclo de vida - Activity



Fragment


pantalla1Fragment
VER TOAST
inicioFragment
PANTALLA 1
PANTALLA 2
pantalla2Fragment


TABLET
Selecting an item updates Fragment B
Activity A Contains Fragment A and Fragment B
HANDSET
Selecting an item Starts Activity B
Activity A Contains Fragment A
Activity B Contains Fragment B

Activity vs Fragment

- Puede contener múltiples Fragments
- Para interfaces de usuario completas (pantalla de inicio o configuración)
- No necesita un Fragment para existir

- Permite diseños más flexibles y adaptables
- Navegación más cómoda
- Necesita una Activity para existir