Volver

U4.1

Kotlin Básico

Volver

Rafael Alberti
Instituto de Educación Secundaria

Volver

Indice

Volver

Introducción

Volver

Kotlin

- Kotlin: un lenguaje de programación moderno y versátil.
- Desarrollado por JetBrains, lanzado en 2011.
- Interoperable con Java, popular en desarrollo Android.

Volver

Características de Kotlin

- Sintaxis concisa y expresiva.
- Seguridad de tipos nulos integrada.
- Soporta programación funcional y orientada a objetos.

Volver

Configuración del Entorno de Kotlin

- Kotlin puede ser usado con IntelliJ IDEA, Android Studio, o cualquier editor de texto.
- Compilador de Kotlin disponible para línea de comandos.
- Kotlin Playground: para experimentar en línea.

Volver

Estructura Básica de un Programa en Kotlin

- Todo programa en Kotlin comienza con la función `main`.
- `main` es el punto de entrada del programa.

kotlin Copy code

```kotlin
fun main() {
    println("Hola, Kotlin!")
}
```

Volver

Ejemplo de Programa en Kotlin

- Un programa simple que imprime un mensaje.
- Uso de `println` para mostrar salida en consola.

kotlin Copy code

```kotlin
fun main() {
    val saludo = "Bienvenidos a Kotlin"
    println(saludo)
}
```

Volver

Compilación y Ejecución

- Kotlin se compila a bytecode de Java, ejecutable en la JVM.
- Uso del comando `kotlinc` para compilar.
- Ejecución a través de la JVM o herramientas de Kotlin.

Volver

Variables en Kotlin

Volver

Introducción

- Kotlin maneja dos tipos de variables: `val` y `var`.
- `val` para valores inmutables, `var` para mutables.
- Fuerte inferencia de tipos.

Volver

Variables Inmutables: `val`

- `val` se usa para declarar una constante.
- Una vez asignado, su valor no puede cambiar.

kotlin Copy code

```kotlin
val pi: Double = 3.14159
val saludo = "Hola Mundo"
```

Volver

Variables Mutables: `var`

- `var` permite cambiar el valor de la variable.
- Útil cuando se necesita modificar el valor.

kotlin Copy code

```kotlin
var edad = 30
edad = 31
```

Volver

Tipos de Datos

Volver

Básicos

- Kotlin tiene tipos de datos similares a Java.
- Incluye `Int`, `Double`, `Float`, `Long`, `Short`, `Byte`, `Boolean`, `Char`, `String`.

kotlin Copy code

```kotlin
val entero: Int = 10
val decimal: Double = 10.5
val bandera: Boolean = true
val caracter: Char = 'A'
```

Volver

Strings en Kotlin

- Las cadenas de texto son representadas por `String`.
- Soportan interpolación de cadenas y operaciones comunes.

kotlin Copy code

```kotlin
val nombre = "Ana"
val saludo = "Hola, $nombre"
```

Volver

Tipos de Datos Nulos

- Kotlin maneja la nulabilidad de forma segura.
- Se especifica con `?` después del tipo de dato.

kotlin Copy code

```kotlin
var nombre: String? = null
nombre = "Carlos"
```

Volver

Colecciones en Kotlin

- Listas, conjuntos y mapas son comunes.
- Pueden ser mutables (`mutableListOf`, `mutableSetOf`, `mutableMapOf`) o inmutables.

kotlin Copy code

```kotlin
val numeros = listOf(1, 2, 3)
val mapa = mapOf("clave1" to "valor1", "clave2" to "valor2")
```

Volver

Operaciones

Volver

Números

- Kotlin ofrece operaciones matemáticas estándar.
- Incluye suma, resta, multiplicación y división.

kotlin Copy code

```kotlin
val a = 10
val b = 5
val suma = a + b
val resta = a - b
```

Volver

Cadenas de Texto

- Las cadenas tienen funciones para manipulación y consulta.
- Ejemplos incluyen `length`, `substring`, `contains`.

kotlin Copy code

```kotlin
val texto = "Kotlin es divertido"
val longitud = texto.length
val subcadena = texto.substring(0, 5)
```

Volver

Listas

- Las listas tienen funciones para agregar, eliminar y acceder a elementos.
- Ejemplos incluyen `add`, `remove`, `get`.

kotlin Copy code

```kotlin
val lista = mutableListOf(1, 2, 3)
lista.add(4)
val elemento = lista.get(2)
```

Volver

Conjuntos

- Los conjuntos se utilizan para operaciones de colección únicas.
- Ejemplos incluyen `add`, `remove`, `contains`.

kotlin Copy code

```kotlin
val conjunto = mutableSetOf("a", "b", "c")
conjunto.add("d")
val contieneB = conjunto.contains("b")
```

Volver

Mapas

- Los mapas almacenan pares clave-valor.
- Funciones comunes incluyen `put`, `remove`, `get`.

kotlin Copy code

```kotlin
val mapa = mutableMapOf("clave1" to "valor1", "clave2" to "valor2")
mapa.put("clave3", "valor3")
val valor = mapa.get("clave2")
```

Volver

Números Flotantes

- Operaciones específicas para números flotantes.
- Incluyen `round`, `ceil`, `floor`.

kotlin Copy code

```kotlin
val numero = 3.14
val redondeado = kotlin.math.round(numero)
```

Volver

Nulabilidad

- Kotlin maneja la nulabilidad con funciones seguras.
- Ejemplos incluyen `?.`, `?:`, `!!`.

kotlin Copy code

```kotlin
val nombre: String? = null
val longitud = nombre?.length ?: 0
```

Volver

Estructuras Condicionales

Volver

If

- `if` se usa para decisiones básicas.
- Puede ser usado como una expresión.

kotlin Copy code

```kotlin
val temperatura = 18
if (temperatura < 20) {
    println("Hace frío")
} else {
    println("Temperatura agradable")
}
```

Volver

If-Else

- `if-else` para múltiples caminos de decisión.
- Estructura clásica de control de flujo.

kotlin Copy code

```kotlin
val calificacion = 85
if (calificacion >= 90) {
    println("Excelente")
} else if (calificacion >= 75) {
    println("Bueno")
} else {
    println("Necesita mejorar")
}
```

Volver

When

- `when` reemplaza al `switch` de Java.
- Más flexible y potente.

kotlin Copy code

```kotlin
val dia = 3
when (dia) {
    1 -> println("Lunes")
    2 -> println("Martes")
    3 -> println("Miércoles")
    else -> println("Otro día")
}
```

Volver

When con Rangos

- `when` es especialmente útil con rangos.
- Permite agrupar casos.

kotlin Copy code

```kotlin
val edad = 25
when (edad) {
    in 0..12 -> println("Niño")
    in 13..19 -> println("Adolescente")
    else -> println("Adulto")
}
```

Volver

When como Expresión

- `when` también puede ser usado como expresión.
- Retorna un valor basado en el caso que coincida.

kotlin Copy code

```kotlin
val estado = "activo"
val mensaje = when (estado) {
    "activo" -> "Usuario activo"
    "inactivo" -> "Usuario inactivo"
    else -> "Estado desconocido"
}
println(mensaje)
```

Volver

Estructuras repetitivas

Volver

Bucle For

- Itera sobre rangos, arrays, o colecciones.
- Sintaxis simple y versátil.

kotlin Copy code

```kotlin
for (i in 1..5) println(i) // Imprime números del 1 al 5
```

Volver

Bucle While

- Bucle basado en una condición.
- Se ejecuta mientras la condición sea verdadera.

kotlin Copy code

```kotlin
var contador = 5
while (contador > 0) {
    println(contador)
    contador--
}
```

Volver

Bucle Do-While

- Similar a `while`, pero se ejecuta al menos una vez.
- La condición se evalúa después de la primera ejecución.

kotlin Copy code

```kotlin
var contador = 0
do {
    println(contador)
    contador++
} while (contador < 5)
```

Volver

Bucle For con Colecciones

- `for` es ideal para recorrer colecciones.
- Puede usar índices o elementos directamente.

kotlin Copy code

```kotlin
val frutas = listOf("manzana", "banana", "cereza")
for (fruta in frutas) println(fruta)
```

Volver

Bucle For con Índices

- `for` también puede iterar con índices.
- Usa `indices` o `withIndex()` para acceder a índices.

kotlin Copy code

```kotlin
val frutas = listOf("manzana", "banana", "cereza")
for (i in frutas.indices) {
    println("Fruta ${i + 1}: ${frutas[i]}")
}
```

Volver

Funciones en Kotlin

- Definidas con `fun`, parámetros y retorno.
- Ejemplo: `fun sum(a: Int, b: Int) = a + b`.

kotlin Copy code

```kotlin
fun greet(name: String) = "Hola, $name!"
```

Volver

Programación Funcional

Volver

Introducción

- Paradigma de programación centrado en funciones puras.
- Evita el estado compartido y los datos mutables.
- Enfatiza la inmutabilidad y las operaciones de alto nivel.

Volver

Funciones Puras en Kotlin

- Una función pura devuelve el mismo resultado para los mismos argumentos.
- No tiene efectos secundarios (como modificar estados globales).

kotlin Copy code

```kotlin
fun sumar(a: Int, b: Int): Int {
    return a + b
}
```

Volver

Inmutabilidad en Kotlin

- La inmutabilidad reduce los errores y efectos secundarios.
- `val` en Kotlin crea variables inmutables.

kotlin Copy code

```kotlin
val lista = listOf(1, 2, 3)
```

Volver

Funciones de Orden Superior

- Kotlin soporta funciones de orden superior.
- Estas funciones pueden tomar funciones como argumentos o retornarlas.

kotlin Copy code

```kotlin
fun operar(x: Int, y: Int, op: (Int, Int) -> Int): Int {
    return op(x, y)
}
```

Volver

Lambdas en Kotlin

- Las lambdas son funciones anónimas que pueden ser pasadas alrededor.
- Utiles para operaciones de colecciones, como `filter`, `map`, `reduce`.

kotlin Copy code

```kotlin
val numeros = listOf(1, 2, 3, 4)
val cuadrados = numeros.map { it * it }
```

Volver

Programación Funcional y Colecciones

- Kotlin proporciona una rica API para trabajar con colecciones de manera funcional.
- Operaciones comunes incluyen `map`, `filter`, `fold`.

kotlin Copy code

```kotlin
val filtrados = numeros.filter { it > 2 }
val suma = numeros.fold(0) { acc, i -> acc + i }
```

Volver

Ventajas de la Programación Funcional

- Facilita el razonamiento sobre el código.
- Promueve un código más limpio y mantenible.
- Útil para la concurrencia y paralelismo.

Volver

Interoperabilidad con Java

- Kotlin y Java pueden coexistir y colaborar.
- Uso de bibliotecas Java en Kotlin.

kotlin Copy code

```kotlin
val list = java.util.ArrayList<string>()
list.add("Kotlin")
</string>
```

Volver

Conclusión sobre Kotlin

- Kotlin: eficiente, seguro y fácil de aprender.
- Ideal para aplicaciones modernas y desarrollo Android.

Volver

¡Gracias por su atención!