

Historia de Kotlin

Kotlin es un lenguaje de programación moderno, conciso y seguro, creado por JetBrains en 2011. Inicialmente diseñado como un lenguaje alternativo para la Máquina Virtual de Java (JVM), Kotlin se ha expandido rápidamente y ahora también es compatible con JavaScript y Nativo.



Versión actual de Kotlin

La versión más reciente de Kotlin es la versión 2.0, lanzada en mayo 23 del 2024. Esta versión incluye mejoras en el rendimiento del compilador, nuevas características del lenguaje, y mejor interoperabilidad con Java. Sigue siendo compatible con versiones anteriores.













Kotlin

Kotlin Playground is an online s
programming language. Browse

```
() {  
    println("World")  
}
```

```
(name: String = "Kotlin"  
    println("Hello, $name!")
```

Posición de Kotlin en el ranking de lenguajes de programación

<div><div>TIOBE <small>(the software quality company)</small></div><div>About us Knowledge News Coding Standards TIOBE Index Contact</div><div>Products Quality Models Markets Schedule a demo</div></div>						
10	15	⬆️		Fortran	1.53%	+0.53%
11	11			Delphi/Object Pascal	1.52%	+0.27%
12	19	⬆️		Swift	1.27%	+0.33%
13	10	⬇️		Assembly language	1.26%	-0.03%
14	12	⬇️		MATLAB	1.26%	+0.14%
15	8	⬇️		PHP	1.22%	-0.52%
16	13	⬇️		Scratch	1.17%	+0.15%
17	20	⬆️		Rust	1.17%	+0.26%
18	18			Ruby	1.11%	+0.17%
19	29	⬆️		Kotlin	1.01%	+0.50%
20	22	⬆️		COBOL	0.96%	+0.22%

Utilidad de Kotlin

Aplicaciones Android

Kotlin es el lenguaje de elección para el desarrollo de aplicaciones móviles Android, gracias a su concisión, seguridad y alta interoperabilidad con Java.

Desarrollo Backend

Kotlin también se utiliza cada vez más en el desarrollo de aplicaciones backend, gracias a su integración con frameworks como Spring Boot y Ktor.

Multipropósito

Además de Android y backend, Kotlin se puede utilizar para desarrollar aplicaciones web, de escritorio, e incluso scripts de servidor con Kotlin/Scripting.

Conceptos de Clases y Objetos (métodos, propiedades, encapsulamiento)

```
1
2 // Conceptos de Clases y Objetos (métodos, propiedades, encapsulamiento)
3 class Persona(private val nombre: String, private var edad: Int) {
4
5     // Método público que permite acceder a los detalles de la persona
6     fun mostrarDetalles() {
7         println("Nombre: $nombre")
8         println("Edad: $edad")
9     }
10
11     // Método público que permite incrementar la edad
12     fun cumplirAños() {
13         edad += 1
14     }
15 }
16
17 fun main() {
18     val persona = Persona(nombre: "Juan", edad: 25)
19
20     // Llamada al método mostrarDetalles
21     persona.mostrarDetalles()
22
23     // Incrementar la edad y mostrar los detalles nuevamente
24     persona.cumplirAños()
25     println("Después de cumplir años:")
26     persona.mostrarDetalles()
27 }
28
```

Conceptos de Contenedores / Asociación

```
1
2 class Profesor(val nombre: String)
3
4 class Curso(val titulo: String) {
5     // Propiedad que asocia un curso con un profesor
6     var profesor: Profesor? = null
7 }
8
9 fun main() {
10     val profesor = Profesor(nombre: "Dr. Smith")
11     val curso = Curso(titulo: "Biología")
12
13     // Asociación del curso con el profesor
14     curso.profesor = profesor
15
16     println("El curso ${curso.titulo} es impartido por ${curso.profesor?.nombre}")
17 }
18
```

Conceptos de Arrays - Listas, / Agregación

```
Exposicipn.iml  Estudiante.kt  Main.kt x  Colegio.kt
13 |
14 |
15 | */
16 |
17 | fun main(){
18 |     //LISTAS ESTATICAS
19 |
20 |     var milista = listOf(10, 11, 12)
21 |     println(milista)
22 |
23 |     var milista2 = mutableListOf("Juan","Pedro","Jesús")
24 |     println(milista2)
25 |     milista2.add("Carlos")
26 |     println(milista2)
27 |     milista2.removeAt(index: 1)
28 |     println(milista2)
29 |
30 |     //AGREGACIÓN
31 |
32 |     var e1 = Estudiante( name: "Brayan", edad: 89)
33 |     var e2 = Estudiante( name: "Mauricci", edad: 100)
34 |     var c1= Colegio()
35 |     c1.agregarEstudiante(e1)
36 |     println(c1.estudiantes)
37 |     c1.agregarEstudiante(e2)
38 |     println(c1.estudiantes)
39 |
40 |     println("Existen: " + c1.estudiantes.size + " Estudiantes")
41 |
42 | }
```


Conceptos de Contenedores / Composición

```
1
2 class Motor(val tipo: String)
3
4 class Coche(val modelo: String) {
5     // El coche tiene un motor que es creado en el constructor
6     val motor: Motor = Motor(tipo: "V8")
7
8     fun detallesCoche() {
9         println("Modelo: $modelo, Motor: ${motor.tipo}")
10    }
11 }
12
13 fun main() {
14     val coche = Coche(modelo: "Ford Mustang")
15
16     // Mostrar detalles del coche
17     coche.detallesCoche()
18 }
19
```


Conceptos de Herencia / Polimorfismo

```
1 // Conceptos de Herencia / Polimorfismo
2
3 // Superclase
4 open class Animal(val nombre: String) {
5     open fun hacerSonido() {
6         println("El animal hace un sonido")
7     }
8 }
9
10 class Perro(nombre: String) : Animal(nombre) {
11     override fun hacerSonido() {
12         println("El perro ladra")
13     }
14 }
15
16 class Gato(nombre: String) : Animal(nombre) {
17     override fun hacerSonido() {
18         println("El gato maúlla")
19     }
20 }
21
22 fun main() {
23     val animales: List<Animal> = listOf(Perro(nombre: "Carlos"), Gato(nombre: "Juan David"))
24
25     for (animal in animales) {
26         animal.hacerSonido()
27     }
28 }
29
```