Manual de Referencia del Lenguaje

Documentación de Sintaxis y Ejemplos

18 de noviembre de 2025

Índice

1. Introducción

Este documento sirve como manual de referencia para el lenguaje de programación, basado en la gramática BNF proporcionada. Cada sección describe una característica del lenguaje y proporciona uno o más ejemplos de código sobre cómo utilizarla.

2. Programa Básico

Un programa consiste en una secuencia de cero o más expresiones, seguidas de la palabra clave `return`.

```
# Un programa simple que define una variable y retorna
var x = 10;
print(x)

return
```

3. Variables y Constantes

El lenguaje permite la definición de variables mutables (`var`) e inmutables (`const`).

3.1. Variables Mutables (var)

Las variables declaradas con `var` pueden cambiar su valor después de la inicialización.

```
# Declaración de una o más variables
var x = 10, y = "hola";

# Reasignación de una variable
x = 20;

print(x) # Imprime 20
return
```

3.2. Constantes (const)

Las constantes declaradas con `const` no pueden ser reasignadas.

```
const PI = 3.14159;
const MENSAJE = "Hola";

# Esto generaría un error en tiempo de ejecución:
# PI = 3.15;

return
```

4. Tipos de Datos y Literales

- **Números**: `10`, `-5`, `12.5`

- **Texto (Strings)**: `"hola"`, `"mundo"`
- **Booleanos**: `true`, `false`
- **Nulo**: `null`

```
var n = 10;
var s = "texto";
var b = true;
var nada = null;
return
```

5. Operadores (Primitivas Binarias)

Las operaciones binarias deben estar envueltas en paréntesis.

```
var suma = (10 + 5);        # 15
var resta = (20 - 2);       # 18
var mult = (5 * 5);         # 25
var div = (100 / 10);     # 10
var modulo = (10 % 3);      # 1

var saludo = ("Hola" concat " Mundo");

var esMayor = (10 > 5);      # true
var esIgual = (5 == 5);      # true
var esDiferente = (true != false); # true

var logicaY = (true && (1 > 0));   # true
var logicaO = (false || true);   # true

return
```

6. Estructuras de Datos

6.1. Listas

Las listas son colecciones ordenadas de valores.

```
var miLista = [1, "dos", 3, true, [10, 20]];

# Acceso a índice (basado en la gramática id-tail)
var primerItem = miLista.[0]; # 1
var segundoItem = miLista.[1]; # "dos"

# Asignación por índice
miLista.[1] = 2;
# miLista ahora es [1, 2, 3, true, [10, 20]]

print(miLista)
```

```
return
```

6.2. Diccionarios

Los diccionarios son colecciones de pares clave-valor.

```
var miDic = {"a": 1, "b": "dos", "c": true};

# Obtener un valor
var valorB = get(miDic, "b"); # "dos"

# Establecer un valor (añadir o modificar)
set(miDic, "c", false); # Modifica "c"
set(miDic, "d", 100);   # Añade "d"

print(miDic)
return
```

7. Control de Flujo

7.1. Condicionales (if / else if / else)

```
var x = 10;

if (x > 10) {
    print("Mayor a 10")
} else if (x == 10) {
    print("Igual a 10")
} else {
    print("Menor a 10")
}

return
```

7.2. Bucle While

```
var i = 0;
while (i < 5) {
    print(i)
    i = (i + 1)
}
# Imprime 0, 1, 2, 3, 4

return
```

7.3. Bucle For

El bucle `for` itera sobre los elementos de una expresión iterable (como una lista).

```
var miLista = ["a", "b", "c"];

for elemento in miLista {
    print(elemento)
}
# Imprime "a", "b", "c"

return
```

8. Funciones (Procedimientos)

Las funciones se crean usando `lambda` y se llaman usando la sintaxis de comilla simple `'`.

```
# Definición de una función (procedimiento)
const sumar = lambda(a, b) => {
    (a + b)
};

# Llamada a la función
var resultado = 'sumar(5, 3); # resultado es 8

print(resultado)
return
```

9. Objetos (Prototipos)

El lenguaje utiliza un sistema de prototipos para objetos.

```
# 1. Crear un objeto prototipo base
object Animal = {
    "nombre": "generico",
    "saludar": lambda() => {
        print("Hola, soy un animal")
    }
};

# 2. Clonar el prototipo para crear una instancia
object Perro = clone(Animal);

# 3. Modificar la instancia
Perro.["nombre"] = "Fido";
Perro.["saludar"] = lambda() => {
    print("Guau!")
};

# 4. Acceder a propiedades y métodos
print(Perro.["nombre"]) # "Fido"
```

```
var metodoSaludo = Perro.["saludar"];
'metodoSaludo() # "Guau!"

return
```

10. Primitivas N-Arias (Funciones Integradas)

Ejemplos de llamadas a funciones integradas del lenguaje.

```
# Imprimir múltiples valores
print("Hola", 1, 2, 3, true)

# --- Listas ---
var miLista = [10, 20, 30];
print( length(miLista) )   # 3
print( head(miLista) )     # 10
print( tail(miLista) )     # [20, 30]
print( list?(miLista) )    # true
print( empty?(miLista) )   # false

# --- Diccionarios ---
var miDic = {"a": 1, "b": 2};
print( keys(miDic) )       # ["a", "b"]
print( values(miDic) )     # [1, 2]

# --- Conversión ---
var numStr = "123";
var numInt = toInt(numStr);
print( (numInt + 7) )      # 130

return
```

11. Ejemplo Completo: Función Recursiva (Factorial)

Este ejemplo combina constantes, lambdas, condicionales, recursión y primitivas binarias.

```
# Definimos la función factorial
const factorial = lambda(n) => {

    # Condición base
    if (n == 0) {
        1 # Retorna 1 si n es 0
    }
    # Condición recursiva
    else {
        # n * factorial(n-1)
        (n * 'factorial( (n - 1) ))
    }
```

```
};

# La llamamos
var fac5 = ’factorial(5);

print("El factorial de 5 es:")
print(fac5) # 120

return
```