# **Dart**

Fue creado por Google en 2011 y liberado en su versión 1.0 en diciembre 2018, es un lenguaje de propósito general.

* Es un lenguaje de POO, el cual utiliza los cuatro pilares:
  + Abstracción.
  + Encapsulamiento.
  + Herencia.
  + Polimorfismo.
* Se traba con definición de clases.
* Cuenta con un recolector de basura.
* Su sintaxis es en base a C.
* Tiene un transpilador a JavaScript, esto significa que podemos convertir código Dart en código JavaScript el cual se puede ejecutar en un navegador.
* Es **JIT** (Justo en el momento) & **AOT** (Antes del momento), esto permite la potencia a flutter para desarrollar apps móviles.
  + **JIT**: Compila la aplicación en el dispositivo en momento de ejecución (interpretado).
  + **AOT**: Compila la aplicación en el momento de construcción (compilación).

**Puede ser utilizado**

* Servidor.
* Web.
* Móvil.

**La documentación**

En Dart la documentación es muy importante lo cual nos ayudará a resolver muchas dudas ya que está es excelente, y no olvidar la característica principal de este lenguaje, **todo es un objeto**.

El lenguaje Dart sirve para trabajar en varios entornos, entre ellos:

* **Back End**: Aqueduct.
* **Front End**: AngularDart.
* **Móvil**: Flutter.

Es importante destacar que se puede transpilar a código Javascript utilizando **Dart2JS**, utilizando las herramientas **webdev** y **stagehand**.

¿Por qué Dart para Flutter?

* AOT (Ahead Of Time): Compilado a un rápido y predecible código nativo. (Totalmente personalizable)
* Puede ser JIT (Just In Time): Compilado para una velocidad excepcional de desarrollo. (Esto incluye el popular Hot Reload).
* Hace fácil la creación de animaciones y transiciones que corren a 60fps (frames per second)
* Al ser compilado a código nativo, no hay puentes innecesarios para correr el código.
* Dart le permite a Flutter evitar el desarrollo de diseños en archivos independientes como JSX, XML o bien interfaces separadas.
* Dart es relativamente fácil de aprender.

# **DartPad y qué editores utilizar**

[DartPad](https://dartpad.dev/) es una herramienta web en la que puedes editar código Dart, nos sirve para empezar bastante rápido, esto porque no necesita que instales ningún complemento, para este curso DartPad es el mínimo requerido.

Pero si eres de los que quieren utilizar un programa de Escritorio, los editores de código e IDEs modernos te permiten programar en Dart, sólo necesitas instalar el plugin correspondiente a cada uno.

# **Variables: Números (int,double), String y booleanos**

Las variables son espacios reservados en la memoria que pueden cambiar de contenido a lo largo de la ejecución de un programa.

Para declarar una variable podemos utilizar la palabra reservada **var**, seguido del nombre de la variable o también podemos escribir el tipo de variable (**int**, **double**, **String** y **bool**) seguido del nombre de la variable.

En Dart tenemos los tipos básicos de variables que encontramos en casi todos los lenguajes tales como **int**, **double**, **String** y **bool** y funcionan de la siguiente manera:

* **int** son valores numéricos enteros que no aceptan punto decimal.
* **double** son valores numéricos decimales que tienen punto decimal.
* **String** son cadenas de caracteres que aceptan textos.
* **bool** son constantes en tiempo de compilación que permiten valores verdaderos (true) o falsos (false).

Las variables de tipo **dynamic** aceptarán cualquier tipo de variable y cambiarán según la necesidad.

# **Colecciones**

Las colecciones son objetos de tipo estructuras de datos que almacenan diversos elementos, existen 2 tipos de colecciones, las listas y los sets.

En Dart los arreglos son Objetos ordenados de tipo List los cuales almacenan con un índice a cada uno de los elementos, mientras que los sets son colecciones que contiene valores únicos (no se pueden repetir) no ordenados (no se pueden recuperar mediante índices).

Una lista permitirá agregar los elementos que se desees, mientras que un set no permitirá volver a agregar un valor ya existente, pero tampoco mandará un error.

**Colecciones**

**List**

* Una de las colecciones más comunes en casi todos los lenguajes es el Array, en Dart los arreglos son objetos tipo List.
* Trabaja con index 0
* Tiene muchas funciones por defecto.

**Declaración**

List list; //forma 1 List list<tipo>; //forma 2

**Inicialización**

list = ["negro","blanco"];

**Agregar elemento**

list.add("Rosa");

**Borrar elemento**

list.removeAt(indice);

**Borrar último elemento**

list.removeLast();

**Añadir elementos de otra lista**

list.addAll(lista\_auxiliar);

**Set**

* En este tipo de colección un elemento puede existir una sola vez. Si intentamos enviar uno repetido no lo permitirá (pero tampoco mandará un error)
* Esta no regresará en el mismo orden y por este mismo motivo no se puede pedir por index

**Declaración**

Set set;

**Inicialización**

set = set.from(['Argel', 'Juan']);

**Agregar elemento**

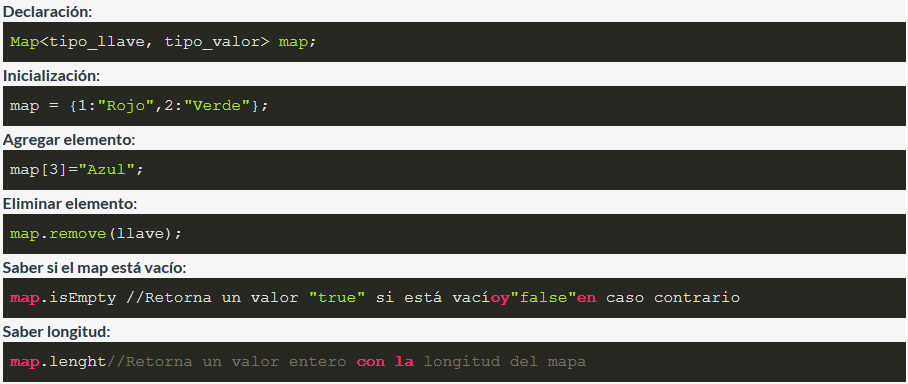
set.add("Erika");

**Eliminar elemento**

set.remove("Argel");

# **Diccionarios**

Los **Maps**, también llamadas **Hash** o **Diccionarios** son objetos que tienen una asociación entre **llaves** y **valores**.



# **Constantes (final y const)**

Las variables de tipo **const** y **final** sólo pueden ser declaradas al mismo tiempo de ser creadas, su diferencia mayor es que **final** es **inicializada** al momento de **usarse**, mientras **const** al momento de **escribirse**.

**final**: solo se pueden declarar una vez y es inicializada cuando se accede a ella.



**const**: Son implícitamente tipo **final**, pero son constantes de tiempo de compilación. Si se requiere una constante a nivel de clase, debe anteponerse el prefijo “**static**”.



**Principal diferencia**

* **final** únicamente se alojará en memoria al momento de utilizarla, es decir, que el código haga uso **realmente** de ella.
* **const** es declara y alojada en memoria al momento de ponerla en nuestro código.

# **Operadores aritméticos, de asignación y lógicos**

**Operadores**

Los operadores utilizados en Dart son los siguientes:

**Unario PostExpr**

****

**Unario PreExpr**

****

**Multiplicativo**

****

**Adición**

****

**Shift**

****

**Bitwise**

****

**Relacional y tipo prueba**

****

**Igualdad**

****

**Lógicos**

****

**Si Nulo**

****

**Condicional**

****

**Cascada**

****

**Asignación**

****

Con los operadores puedes crear expresiones, algunos ejemplos son:

**a++**

**a + b**

**a = b**

**a == b**

**c ? a : b**

**a is T**

En la tabla de operadores, cada operador tiene mayor precedencia que los operadores de las filas siguientes. Por ejemplo, el operador multiplicativo % tiene mayor precedencia que (y por lo tanto ejecuta antes) el operador de igualdad ==, que tiene mayor precedencia que el operador lógico AND (&&). Esa precedencia significa que las siguientes dos líneas de código se ejecutan de la misma manera:

// Los paréntesis mejoran la lectura



// Mas difícil de leer, pero equivalente.



**Operadores Aritméticos**

Dart soporta los operadores aritméticos habituales:

* Suma ( +)
* Resta ( - )
* -expr Unario menos también conocido como negación (invierte el signo de la expresión)
* Multiplicación
* / División
* ~/ Divide, regresando el valor entero
* % regresa el restante del entero en una división (modulo)

Dart también soporta operadores de incrementos y decrementos de prefijos y postfijos.

* **++var** El cual se leería; var = var + 1 (expression value is var + 1)
* **var++** El cual se leería; var = var + 1 (expression value is var)
* **--var** El cual se leería; var = var – 1 (expression value is var – 1)
* **var--** El cual se leería; var = var – 1 (expression value is var)

**Operadores de Igualdad y relacionales**

Enumeraremos el significado de la igualdad y de los operadores relacionales.

* == Igual
* != No igual (diferente)
* > Mayor que
* < Menor que
* >= Mayor que o igual a
* <= Menor que o igual a

**Operadores de prueba de tipo**

Los operadores **as**, **is**, **and is!** son útiles para verificar los tipos en tiempo de ejecución.

**as** Typecast (También utilizado para especificar prefijos de biblioteca)

**is** Verdadero si el objeto es el tipo especificado.

**is!** Falso si el objeto tiene el tipo especificado.

**Operadores de Asignación**

Como ya has visto, puedes asignar valores usando el operador =. Para asignar sólo si la variable asignada es **nula**, utilice el operador **??=**

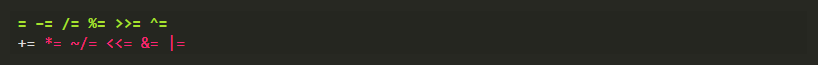
**Asigna valor a variable a**

a = value;

**Asigna valor a b, si b es nulo; de lo contrario b se mantiene igual**

b ??= value;

**Operadores que existen**



//Así es como trabajan

Donde **op** es un operador a **op**= b a = a **op** b

Ejemplo a += b a = a + b

**Operadores Lógicos**

Puedes invertir o combinar expresiones booleanas utilizando los operadores lógicos.

* **!expr** invierte la expresión que le sigue (cambia falso a verdadero y viceversa)
* **||** OR lógico
* **&&** AND lógico

**Operadores Bitwise y Shift**

Puedes manipular los bits individuales de los números en Dart. Por lo general, se utilizan estos operadores de bits y de desplazamiento con enteros.

* **&** AND
* **|** OR
* **^** XOR
* **~expr** Complemento unario bitwise (0s a 1s; 1s a 0s)
* **<<** Desplazamiento hacia la izquierda.
* **>>** Desplazamiento hacia la derecha.