Curso básico de Flutter

Laboratorio de Multimedia e Internet
FI -UNAM

Identificadores

Reglas que se deben seguir para asignar los identificadores:

- Pueden contener letras y números, pero no debe comenzar con un número.
- Solo se acepta guión bajo (_) y signo de peso ($), cualquier otro símbolo especial no es aceptado.

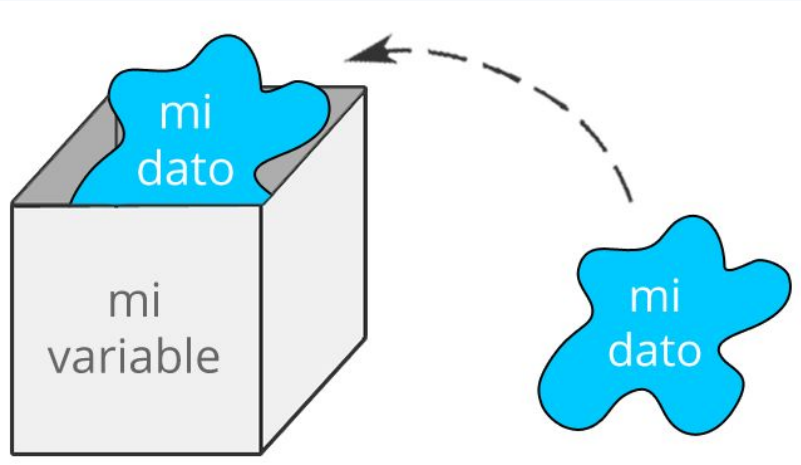
✔	❌
num1	12num
_resultado	%porcentaje
$nombre	!variable

- No pueden contener espacios
- Distingue entre mayúsculas y minúsculas
- No deben ser palabras reservadas.
- Deben ser únicos.

✔	❌
primerNombre	primer nombre
primer_nombre	Primer-nombre
	String int

¿Qué es una variable?

Son como pequeñas cajas que guardan dentro una referencia a un valor/dato, para poder acceder a ella se debe asignar un identificador



Tipos de Variables

Var

var es una expresión para hacer referencia a que la variable es de cualquier tipo.

El interprete de Dart se encarga de elegir el tipo de dato adecuado para la variable

```
int numero
var numero = 5;
```

```
bool numero
var numero = true;
```

```
String nombre
var nombre = 'Julio';
```

```
int nombre
var nombre = 5;
```

dynamic

Esta nos permite declarar una variable en la que el tipo puede cambiar en tiempo de ejecución.
También sirve para indicarle al compilador que la variable puede contener cualquier tipo de valor,

```dart
dynamic cambioDeTipo = "Cadena";
print(cambioDeTipo);
cambioDeTipo = 1234;
print(cambioDeTipo);
```

Númericas

Para referirse a valores numéricos se usan:

- int: Para un número entero
- double: Para un número que contenga decimales.

```
int numEntero = 10;

double numDoble = 10.5;
```

Strings - Cadenas

Las cadenas se definen con la palabra String , para que el valor sea valido debe ir entre comillas simples o dobles (‘’ , “”).

Si se requiere que el string tenga multiples lineas, se debe abrir y cerrar con 3 comillas simples.

```
String nombre = "Sandra";
```

```
String nombre = 'Sandra';
```

```
String multilinea = ''' Esto es un string
                        con varias
                        lineas ''';
```

Booleanas

Solo existen dos tipos de valores, true o false.
Este tipo de datos ayudan en la toma de desiciones.

```
bool luzPrendida = true;

bool paseElSemestre = false;
```

Listas

En Dart los arreglos se definen con el objeto List, sus valores van dentro de corchetes y separados por comas.

```
List <tipoDeDato> nombreLista = [valores];
```

Lista de tamaño dinámico:

```
List<tipoDeDato> nombreLista  = new List();
```

```
List<String> personajes = ['superman', 'batman'];
```

```
List<int> edades = new List();
```

Sets

Es similar a una lista, lo que la hace diferente a una lista es porque sus elementos nunca se repiten y van dentro de llaves separados por coma.

```
Set<tipoDeDato> nombreSet = {val1,val2};
```

```
Set<int> valores = {12,23,45,19,21};
```

```
Set<String> villanos = {'Malo1','Malo2','Malo3'};
```

Maps

Un mapa es un objeto que asocia una llave con un valor.

```
Map<tipoLlave, tipoValor> nombreMap = {

  llave1 : valor1,
  llave2 : valor2

};
```

```
Map <String,String> nombreCompleto = {

  'nombre' : 'Marcos',
  'apellidoPa' : 'Osorio',
  'apellidoMa' : 'Muñoz'

};
```

```
Map <String,int> ladas = {

  'mexico' : 52,
  'usa' : 1,
  'españa' : 34

};
```

final y const

Se usan para declarar variables cuyo valor no va a cambiar.

Una variable final solo se puede asignar una vez.
const es una constante en tiempo de compilación, su valor es conocido en tiempo de compilación.

```
final horaActual = '13:00';

const horaDelSistema = 0;
```

Estructuras de Selección
y
Control de flujo

Operadores Condicionales

Operador	Propósito
>	mayor que
>=	mayor o igual que
<	menor que
<=	menor o igual que
==	igual
!=	distinto
&&	AND lógico
||	OR lógico
!	NOT Lógico

Operadores Aritméticos

Operador	Propósito
+	suma
-	resta
-exp	negación
*	multiplicación
/	division
~	rdivide y retorna un resultado entero
%	modulo, regresa el resto de la division

Operadores de Incremento y Decremento

Operador	Propósito
++var	var = var +1
var ++	var = var + 1
--var	var =var -1
var--	var = var -1

If - Else

Esta estructura ayuda a la toma de decisión.

Se define con la palabra reservada if, entre (), se coloca la condición que queremos validar si se cumple o no.

el else se puede omitir si lo deseamos que se cumpla la condición del if

```
if(condición){
   si se cumple la condición se ejecutará el código
   que está entre las llaves
  }else {
   si la condición del if no se cumple, se realiza
   el código que esté entre las llaves del else
 }
```

Ejemplo

En el if la condición que debe cumplir para realizar el bloque/línea de código, es que edad sea mayor o igual a 18, si no se cumple la condición realizará el bloque/línea de código del else.

Caso 1:
edad = 19, cumple con la condición de ser igual o mayor que 18 así que, se mostrará el mensaje Eres mayor de edad.

Caso 2 :
edad = 12, no cumple con la condición, así que pasará al else y mostrará Eres menor de edad

```
var edad = 19;

if(edad>=18){
  print('Eres Mayor de Edad');
}else{
  print('Eres menor de Edad');
}
```

```
var edad = 12;

if(edad>=18){
  print('Eres Mayor de Edad');
}else{
  print('Eres menor de Edad');
}
```

For

Se utiliza para ejecutar un código una limitada cantidad de veces, hasta que cumpla con la condición que se le asignó.

(variable a evaluar; condición; cuanto aumenta después del ciclo)

```
for(var contador=0; contador<5; contador ++){
   Se va a realizar el código que esté dentro de las llaves,
   hasta que se cumpla con la condición,
   Cada que se realiza un ciclo el contador aumenta su valor.
 }
```

Ejemplo

En este ejemplo el ciclo se va a realizar hasta que se cumpla con la condición menor que 10.

Cuando entra por primera vez al ciclo i vale 0, al realizar el print toma el valor de + 1, cuando vuelve al inicio del ciclo ahora su valor es 2, sigue cumpliendo con la condición de ser menor a 10.
cuando i tiene el valor de 10 y vuelve al inicio del ciclo, ya no cumple con la condición de ser menor que 10, por lo tanto se sale del ciclo.

```
for (int i=0; i<10;i++){

print('Hola Mundo');

}
```

For … in

Se utiliza para recorrer las propiedades de un objeto.

Por ejemplo en Listas y Sets, realizará el bloque de código que se indique, finalizará hasta que haya recorrido todos los elementos.

```
 for( String recorrer in objetoDondeVaARecorrer){
 va a realizar el código que esté dentro de las
llaves,
 esto lo realizará hasta que termine de recorrer
todos los valores del objeto
}
```

Ejemplo

En este ejemplo tenemos una lista llamada nombres.
Con el for in, se recorreré la lista, el elemento que va leyendo lo almacena en recorrer y lo imprime.
Cuando ya no quedan más elementos que leer, se sale del ciclo.

```
List<String> nombres = ['Maria',  'Pepe',  'Alberto' ];

for (var recorrer in nombres) {
  print(recorrer);

}
```

While

Se realiza el código sólo si la condición se cumple, se sale del ciclo y continúa con la siguiente línea de código.

```
while(condición){
 si cumple la condición realiza
 el bloque de código entre las llaves
}
```

si no se cumple la condición se sale del ciclo y continua con otra linea de código.

Ejemplo

Tenemos una variable de tipo entera con un valor de 0.
El While se va a realizar hasta que vueltas sea menor que 5, cuando entra al ciclo imprime vueltas (valor de 0), aumenta su valor en 1, cuando vuelve a la condición ahora vueltas vale 1, con el ++ le indicamos que aumente su valor en una unidad.

```
int vueltas = 0;

while (vueltas < 5){
  print('vueltas');
  vueltas++;
}
```

do-While

Este ciclo a diferencia del while, realiza por lo menos una vez, lo que está dentro de las llaves del do y luego evalúa la condición que tenga el while, si se cumple, realiza lo que esté entre sus llaves y si no vuelve al código del do.

```
do{
 Código que se realiza por lo menos una vez
 cuando llega a la ultima linea de código dentro de
las llaves, se revisa si se cumple o no con la
condición del while
 }while(condición){
 si cumple la condición realiza
 el bloque de codigo entre las llaves
}
```

Ejemplo

En este ejemplo hora vale 0,
Cuando entra en el do va a imprimir el mensaje, al llegar al while se va a salir del ciclo, debido a que la condición dice que mientras sea distinto de 0 puede realizar lo que esté en do, pero en este caso vale 0.

```
int hora = 0;

do{
  print ('Esto se va a imprimir una vez');
} while(hora != 0);
```

Switch

Funcionan con variables en vez de funcionar con una condición.
El Switch va caso por caso comparando la variable con el valor del case y en caso de que sean iguales el código dentro del case será ejecutado.

```
switch (opcion) {
  case 1:
  si el valor de la variable es igual al caso, realiza el código que esté dentro de él, y si no es igual sigue evaluando otros casos.
    break;// el break tiene la funcionalidad de salirse del caso o del ciclo.
  default: este realiza el codigo que tenga, si se da el caso que la variable
   no es igual a ninguno de los casos posibles

}
```

Ejemplo

En este ejemplo la variable es de tipo cadena, con Fresa como valor.

Al evaluar el sabor en los diferentes casos disponibles (Limón, Chocolate, Fresa) , va entrar en el de Fresa e imprimirá el mensaje El sabor es fresa,
Si el sabor fuera Durazno, entraría en default porque no hay un caso definido para ese sabor.

```
String sabor = 'Fresa';

switch (sabor) {
  case 'Limon' :
   print('El sabor es limon');

    break;

  case 'Chocolate':
  print('El sabor es chocolate');

    break;

  case 'Fresa':
  print('El sabor es fresa');


    break;
  default:
  print('De ese sabor no hay');
}
```

Funciones

¿Qué son las funciones ?

Son un conjunto de declaraciones que realizan una tarea específica.

Organizan el programa en bloques lógicos de código. Una vez definidas pueden ser llamadas desde cualquier parte del código, lo que hace que el código sea reutilizable.

Estructura

Esta sería una función básica, no devuelve ningún valor. Se utiliza cuando no se necesita que la función devuelva algún valor, void es el encargado de indicarlo.

```
void nombreFuncion() {

  //bloque de codigo

}
```

Esta otra forma de declarar la función, devuelve un valor el cual tipo que se igual al tipo de dato.

```
tipoDato nombreFuncion() {

  //bloque de codigo

  return valor;
}
```

```
int numeros(){

  return 5;

}
```

Funciones con Parámetros

Dentro del paréntesis, se especifica el tipo de parámetro que va a recibir y su identificador.

```
tipo nombre (tipoParametro identificadorParametro1){
  //bloque de codigo
  return valortipo;
}
```

Ejemplo

En este ejemplo la función es de tipo int, su identificador es suma y recibe dos parámetros de tipo int, llamados num1 y num2.

Lo que realiza la función es sumar los dos números enviados como parámetros y regresa el resultado, el cual es la suma de los números.

```
//Dentro del main se manda llamar
suma(3, 4);

}

int suma (int num1, int num2){

  int resultado;
  resultado = num1 + num2;
  return resultado;

}
```

Funciones Flecha o Arrows

Estas funciones solo pueden tener una línea de código.

Su sintaxis es la siguiente:

```
[tipo_dato] nombre_función (parámetros) => expresión;
```

Se crean funciones pequeñas y se mejora la legibilidad del código.

Laboratorio de Multimedia e Internet
FI -UNAM

Ejemplo

Vamos a convertir el ejemplo de la función suma a tipo flecha.
Ambas formas hacen exactamente lo mismo.

```
//Dentro del main se manda llamar
suma(3, 4);

}

int suma (int num1, int num2){

  int resultado;
  resultado = num1 + num2;
  return resultado;

}
```

```
int suma (int num1, int num2) => num1 + num2;
```

Funciones Anónimas

Todas la funciones deben tener un nombre, pero en las funciones anónimas no es así.
Esta función es llamada en el momento y se pasa como parámetro a otras funciones.

```
(nombreParametro){
  //Bloque de codigo
}
```

```
var lista = List.from([1,2,3,4,5,6,7,]);
lista.forEach((parametro) {
  //Aqui va el codigo
});
```