# 1.8Dart

# Android Estudio V3.0.17/Flutter

Objetivo:Revisión de un programa Dart

F.Creación: 4-4-2018/4 horas

Programa:Ninguno

Fuente: [aprenda dart en y Minutos](https://learnxinyminutes.com/docs/dart/)

example1() {  
 example1nested1() {  
 example1nested2() => print(**"Example1 nested 1 nested 2"**);  
 example1nested2();  
 }  
 example1nested1();  
}

🡪Imprime una sola vez

example2() {  
 example2nested1(fn) {  
 fn();  
 }  
 example2nested1(() => print(**"Example2 nested 1"**));  
}

example3() {  
 example3nested1(fn(String informSomething)) {  
 fn(**"Example3 nested 1"**);  
 }  
 example3planB(fn) { *// Or don't declare number of parameters.* fn(**"Example3 plan B"**);  
 }  
 example3nested1((s) => print(s));  
 example3planB((s) => print(s));  
}

=>Es solo forma el parámetro

**var** example4Something = **"Example4 nested 1"**;  
example4() {  
 example4nested1(fn(informSomething)) {  
 fn(example4Something);  
 }  
 example4nested1((s) => print(s));  
}

**var** example5method = **"Example5 sayIt"**;  
**class** Example5Class {  
 sayIt() {  
 print(example5method);  
 }  
}  
example5() {  
 *// Create an anonymous instance of the Example5Class and call the sayIt  
 // method on it.* **new** Example5Class().sayIt();  
}

**class** Example7Class {  
 **static var** *example7ClassVariable* = **"Example7 class variable"**;  
 **static** *sayItFromClass*() {  
 print(*example7ClassVariable*);  
 }  
 sayItFromInstance() {  
 print(*example7ClassVariable*);  
 }  
}  
example7() {  
 Example7Class.*sayItFromClass*();  
 **new** Example7Class().sayItFromInstance();  
}

* Estático

**var** example8A = **const** [**"Example8 const array"**],  
 example8M = **const** {**"someKey"**: **"Example8 const map"**};  
example8() {  
 print(example8A[0]);  
 print(example8M[**"someKey"**]);  
}

**var** example9A = **const** [**"a"**, **"b"**];  
example9() {  
 **for** (**var** i = 0; i < example9A.**length**; i++) {  
 print(**"Example9 for loop '**${example9A[i]}**'"**);  
 }  
 **var** i = 0;  
 **while** (i < example9A.**length**) {  
 print(**"Example9 while loop '**${example9A[i]}**'"**);  
 i++;  
 }  
 **for** (**var** e **in** example9A) {  
 print(**"Example9 for-in loop '**${e}**'"**);  
 }  
 example9A.forEach((e) => print(**"Example9 forEach loop '**${e}**'"**));  
}

🡪Todas hacen lo mismo

**var** example10S = **"ab"**;  
example10() {  
 **for** (**var** i = 0; i < example10S.**length**; i++) {  
 print(**"Example10 String character loop '**${example10S[i]}**'"**);  
 }  
 **for** (**var** i = 0; i < example10S.**length**; i++) {  
 print(**"Example10 substring loop '**${example10S.substring(i, i + 1)}**'"**);  
 }

example12() {  
 **var** now = **new** DateTime.now();  
 print(**"Example12 now '**${now}**'"**);  
 now = now.add(**new** Duration(days: 1));  
 print(**"Example12 tomorrow '**${now}**'"**);  
}

example13() {  
 **var** s1 = **"some string"**, s2 = **"some"**, re = **new** RegExp(**"^s.+?g\$"**);  
 match(s) {  
 **if** (re.hasMatch(s)) {  
 print(**"Example13 regexp matches '**${s}**'"**);  
 } **else** {  
 print(**"Example13 regexp doesn't match '**${s}**'"**);  
 }  
 }  
 match(s1);  
 match(s2);  
}

example14() {  
 **var** v = **true**;  
 **if** (v) {  
 print(**"Example14 value is true"**);  
 }  
 v = **null**;  
 **try** {  
 **if** (v) {  
 *// Never runs* } **else** {  
 *// Never runs* }  
 } **catch** (e) {  
 print(**"Example14 null value causes an exception: '**${e}**'"**);  
 }

example15() {  
 **try** {  
 **try** {  
 **throw "Some unexpected error."**;  
 } **catch** (e) {  
 print(**"Example15 an exception si se imprime esta linea: '**${e}**'"**);  
 **throw** e; *// Re-throw* }  
 } **catch** (e) {  
 print(**"Example15 catch exception being re-thrown: '**${e}**'"**);  
 } **finally** {  
 print(**"Example15 Still run finally"**);  
 }  
}

example16() {  
 **var** sb = **new** StringBuffer(), a = [**"a"**, **"b"**, **"c"**, **"d"**], e;  
 **for** (e **in** a) { sb.write(e); }  
 print(**"Example16 dynamic string created with "  
 "StringBuffer '**${sb.toString()}**'"**);  
 print(**"Example16 join string array '**${a.join()}**'"**);  
}

example17() {  
 **print("Example17 "  
 "concatenate "  
 "strings "  
 "just like that"**);  
}

*// Strings with triple single-quotes or triple double-quotes span  
// multiple lines and include line delimiters.*example19() {  
 print(**'''Example19 <a href="etc">  
Example19 Don't can't I'm Etc  
Example19 </a>'''**);  
}

🡪las tres comillas

*// Strings have the nice interpolation feature with the $ character.  
// With $ { [expression] }, the return of the expression is interpolated.  
// $ followed by a variable name interpolates the content of that variable.  
// $ can be escaped like so \$ to just add it to the string instead.*example20() {  
 **var** s1 = **"'\${s}'"**, s2 = **"'\$s'"**;  
 **var** s3=12;  
 print(**"Example20 \$ interpolation** ${s1} **or** $s2 **works.[**$s3**+1,**${s3+1}**]"**);  
}

**class** Example21 {  
 List<String> **\_names**;  
 Example21() {  
 **\_names** = [**"a"**, **"b"**];  
 }  
 List<String> **get names** => **\_names**;  
 **set names**(List<String> list) {  
 **\_names** = list;  
 }  
 int **get length** => **\_names**.**length**;  
 **void** add(String name) {  
 **\_names**.add(name);  
 }  
}  
void example21() {  
 Example21 o = **new** Example21();  
 o.add(**"c"**);  
 print(**"Example21 names '**${o.**names**}**' and length '**${o.**length**}**'"**);  
 o.**names** = [**"d"**, **"e"**];  
 print(**"Example21 names '**${o.**names**}**' and length '**${o.**length**}**'"**);  
}

*/ Class inheritance takes the form of class name extends AnotherClassName {}.***class** Example22A {  
 **var \_name** = **"Some Name!"**;  
 **get name** => **\_name**;  
}  
**class** Example22B **extends** Example22A {}  
example22() {  
 **var** o = **new** Example22B();  
 print(**"Example22 class inheritance '**${o.**name**}**'"**);  
}

*// Class mixin is also available, and takes the form of  
// class name extends SomeClass with AnotherClassName {}.  
// It's necessary to extend some class to be able to mixin another one.  
// The template class of mixin cannot at the moment have a constructor.  
// Mixin is mostly used to share methods with distant classes, so the  
// single inheritance doesn't get in the way of reusable code.  
// Mixins follow the "with" statement during the class declaration.***class** Example23A {}  
**class** Example23Utils {  
 addTwo(n1, n2) {  
 **return** n1 + n2;  
 }  
}  
**class** Example23B **extends** Example23A **with** Example23Utils {  
 addThree(n1, n2, n3) {  
 **return** addTwo(n1, n2) + n3;  
 }  
}  
example23() {  
 **var** o = **new** Example23B(), r1 = o.addThree(1, 2, 3),  
 r2 = o.addTwo(1, 2);  
 print(**"Example23 addThree(1, 2, 3) results in '**${r1}**'"**);  
 print(**"Example23 addTwo(1, 2) results in '**${r2}**'"**);  
}

*/// The Class constructor method uses the same name of the class and  
// takes the form of SomeClass() : super() {}, where the ": super()"  
// part is optional and it's used to delegate constant parameters to the  
// super-parent's constructor.***class** Example24A {  
 **var \_value**;  
 Example24A({value: **"someValue"**}) {  
 **\_value** = value;  
 }  
 **get value** => **\_value**;  
}  
**class** Example24B **extends** Example24A {  
 Example24B({value: **"someOtherValue"**}) : **super**(value: value);  
}  
example24() {  
 **var** o1 = **new** Example24B(),  
 o2 = **new** Example24B(value: **"evenMore"**);  
 print(**"Example24 calling super during constructor '**${o1.**value**}**'"**);  
 print(**"Example24 calling super during constructor '**${o2.**value**}**'"**);  
}

*// There's a shortcut to set constructor parameters in case of simpler classes.  
// Just use the this.parameterName prefix and it will set the parameter on  
// an instance variable of same name.***class** Example25 {  
 **var value**, **anotherValue**;  
 Example25({**this**.**value**, **this**.**anotherValue**});  
}  
example25() {  
 **var** o = **new** Example25(value: **"a"**, anotherValue: **"b"**);  
 print(**"Example25 shortcut for constructor '**${o.**value**}**' and "  
 "'**${o.**anotherValue**}**'"**);

*// Named parameters are available when declared between {}.  
// Parameter order can be optional when declared between {}.  
// Parameters can be made optional when declared between [].*example26() {  
 **var** \_name, \_surname, \_email;  
 setConfig1({name, surname}) {  
 \_name = name;  
 \_surname = surname;  
 }  
 setConfig2(name, [surname, email]) {  
 \_name = name;  
 \_surname = surname;  
 \_email = email;  
 }  
 setConfig1(surname: **"Doe"**, name: **"John"**);  
 print(**"Example26 name '**${\_name}**', surname '**${\_surname}**', "  
 "email '**${\_email}**'"**);  
 setConfig2(**"Mary"**, **"Jane"**);  
 print(**"Example26 name '**${\_name}**', surname '**${\_surname}**', "  
 "email '**${\_email}**'"**);  
}

*// Variables declared with final can only be set once.  
// In case of classes, final instance variables can be set via constant  
// constructor parameter.***class** Example27 {  
 **final color1**, **color2**;  
 *// A little flexibility to set final instance variables with syntax  
 // that follows the :* Example27({**this**.**color1**, color2}) : **color2** = color2;  
}  
example27() {  
 **final** color = **"orange"**, o = **new** Example27(color1: **"lilac"**, color2: **"white"**);  
 print(**"Example27 color is '**${color}**'"**);  
 print(**"Example27 color is '**${o.**color1**}**' and '**${o.**color2**}**'"**);  
}

*// To import a library, use import "libraryPath" or if it's a core library,  
// import "dart:libraryName". There's also the "pub" package management with  
// its own convention of import "package:packageName".  
// See import "dart:collection"; at the top. Imports must come before  
// other code declarations. IterableBase comes from dart:collection.***class** Example28 **extends** IterableBase {  
 **var names**;  
 Example28() {  
 **names** = [**"a"**, **"b"**];  
 }  
 **get iterator** => **names**.iterator;  
}  
example28() {  
 **var** o = **new** Example28();  
 o.forEach((name) => print(**"Example28 '**${name}**'"**));

}

*/ For control flow we have:  
// \* standard switch with must break statements  
// \* if-else if-else and ternary ..?..:.. operator  
// \* closures and anonymous functions  
// \* break, continue and return statements*example29() {  
 **var** v = **true** ? 30 : 60;  
 **switch** (v) {  
 **case** 30:  
 print(**"Example29 switch statement"**);  
 **break**;  
 }  
 **if** (v < 30) {  
 } **else if** (v > 30) {  
 } **else** {  
 print(**"Example29 if-else statement"**);  
 }  
 callItForMe(fn()) {  
 **return** fn();  
 }  
 rand() {  
 v = **new** DM.Random().nextInt(50);  
 **return** v;  
 }  
 **while** (**true**) {  
 print(**"Example29 callItForMe(rand) '**${callItForMe(rand)}**'"**);  
 **if** (v != 30) {  
 **break**;  
 } **else** {  
 **continue**;  
 }  
 *// Never gets here.* }  
}

*// Parse int, convert double to int, or just keep int when dividing numbers  
// by using the ~/ operation. Let's play a guess game too.*example30() {  
 **var** gn, tooHigh = **false**,  
 n, n2 = (2.0).toInt(), top = int.*parse*(**"123"**) ~/ n2, bottom = 0;  
 top = top ~/ 6;  
 gn = **new** DM.Random().nextInt(top + 1); *// +1 because nextInt top is exclusive* print(**"Example30 Guess a number between 0 and** ${top}**"**);  
 guessNumber(i) {  
 **if** (n == gn) {  
 print(**"Example30 Guessed right! The number is** ${gn}**"**);  
 } **else** {  
 tooHigh = n > gn;  
 print(**"Example30 Number** ${n} **is too "  
 "**${tooHigh ? **'high'** : **'low'**}**. Try again"**);  
 }  
 **return** n == gn;  
 }  
 n = (top - bottom) ~/ 2;  
 **while** (!guessNumber(n)) {  
 **if** (tooHigh) {  
 top = n - 1;  
 } **else** {  
 bottom = n + 1;  
 }  
 n = bottom + ((top - bottom) ~/ 2);  
 }  
}

main() {  
 print("Learn Dart in 15 minutes!");  
 [example1, example2, example3, example4, example5, example6, example7,  
 example8, example9, example10, example11, example12, example13, example14,  
 example15, example16, example17, example18, example19, example20,  
 example21, example22, example23, example24, example25, example26,  
 example27, example28, example29, example30  
 ].forEach((ef) => ef());