



Dart Basics

- HelloWorld in Dart without VS Code
- Create an exe-file from Dart code
- Use command line arguments in Dart
- Built-In types in Dart (int, double, String, bool)
- Nullable types and “sound null safety”
- Variable declaration with var or final
- Functions with positional or named parameters or both
- Functions with the “arrow syntax”



HelloWorld in Dart without VS Code

1) Open a Text Editor (e.g. Notepad) and type

```
void main() {  
    print('Hello World!');  
}
```

2) Save this in a file hello.dart.

3) Open a Command Prompt in the directory where you saved hello.dart.

4) Type “dart run hello.dart”:

```
C:\flutter\code\dart_basics>dart run hello.dart  
Hello World!
```



Create an exe-file from Dart code

```
C:\flutter\code\dart_basics>dart compile exe hello.dart
Info: Compiling with sound null safety.
Generated: c:\flutter\code\dart_basics\hello.exe
```

This generates a 4 MB executable:

Name	Size
hello.dart	1 KB
hello.exe	4,670 KB

It can be executed on Windows, even if no Dart SDK was installed on that machine:

```
C:\flutter\code\dart_basics>hello.exe
Hello World!
```



Use command line arguments in Dart

```
void main (List<String> args) {  
    print('Hello World!');  
    for (int i=0; i<args.length; i++) {  
        print(args[i]);  
    }  
}
```

```
C:\flutter\code\dart_basics>dart run hello.dart a bb ccc 1234  
Hello World!  
a  
bb  
ccc  
1234
```



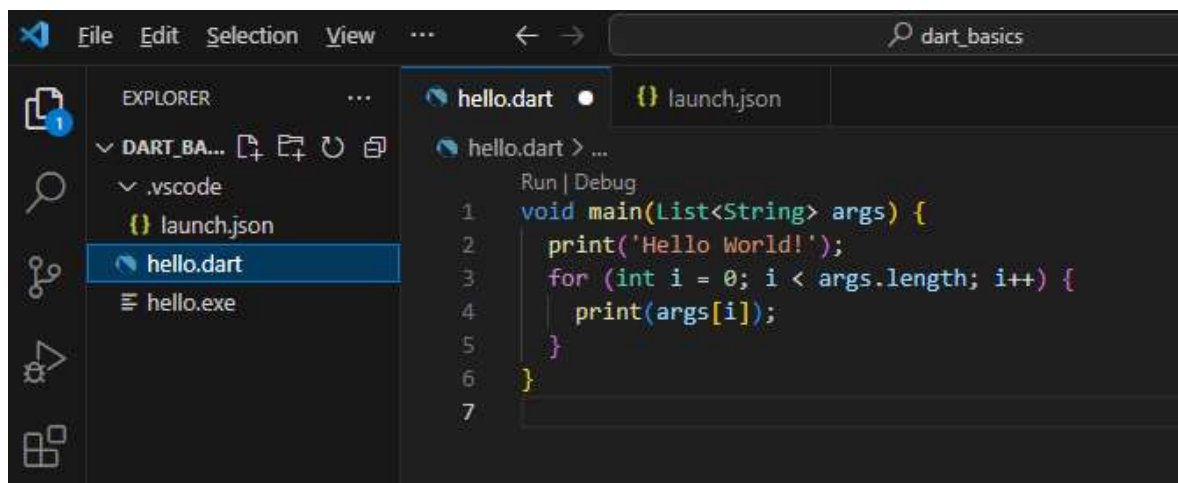
Switch from Notepad to VS Code

1) Open the directory with your hello.dart file in VS Code

(either via menu „File/Open Folder...” in VS Code

or via context menu in Windows Explorer

or by typing “code .” in the Command Prompt where you executed the last dart commands)



Colors and IntelliSense
make life easier

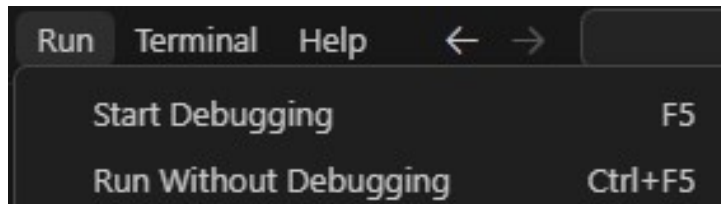


Tip: Shortcut Shift + Alt + F formats the document (see context menu of the editor).

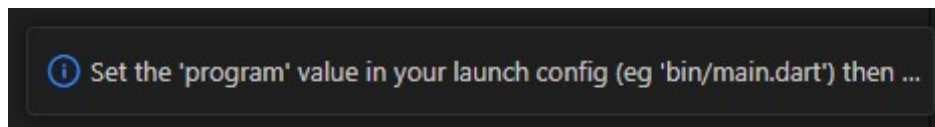


Run or Debug in VS Code

When you select in VS Code one of the menu entries



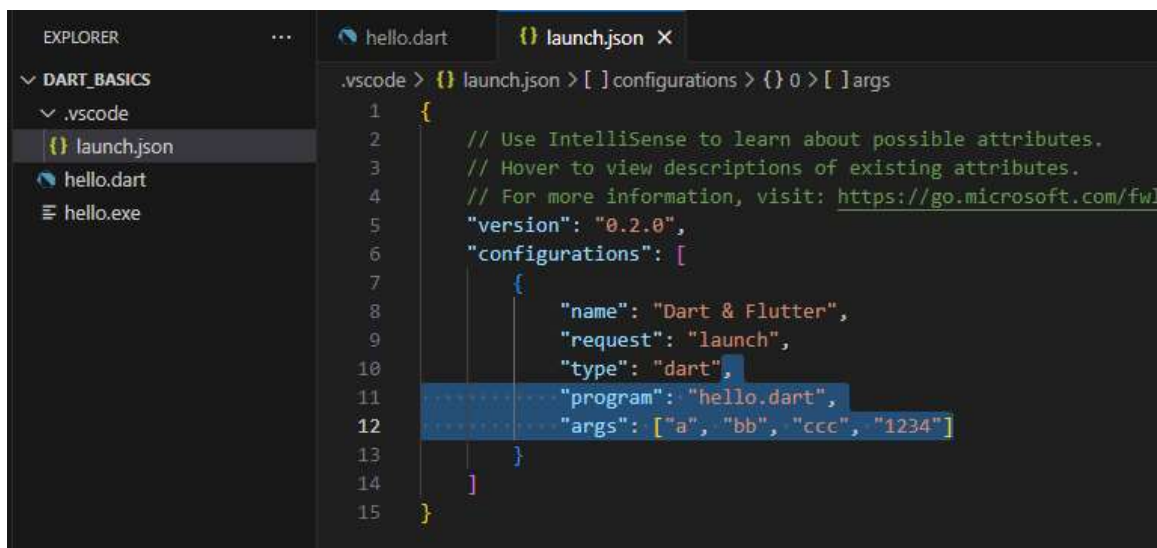
Then VS Code comes up with the message



and it creates a new folder `.vscode` and therein a file “`launch.json`”:

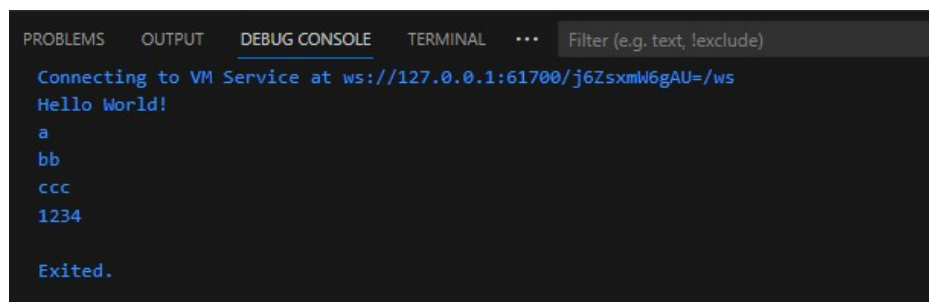


Run or Debug in VS Code (continued)



```
.vscode > {} launch.json > [ ] configurations > {} 0 > [ ] args
1 {
2     // Use IntelliSense to learn about possible attributes.
3     // Hover to view descriptions of existing attributes.
4     // For more information, visit: https://go.microsoft.com/fwlink/?linkid=833380
5     "version": "0.2.0",
6     "configurations": [
7         {
8             "name": "Dart & Flutter",
9             "request": "launch",
10            "type": "dart",
11            "program": "hello.dart",
12            "args": ["a", "bb", "ccc", "1234"]
13        }
14    ]
15 }
```

Add the blue marked text as shown above, save “launch.json”, open “hello.dart” and press F5.

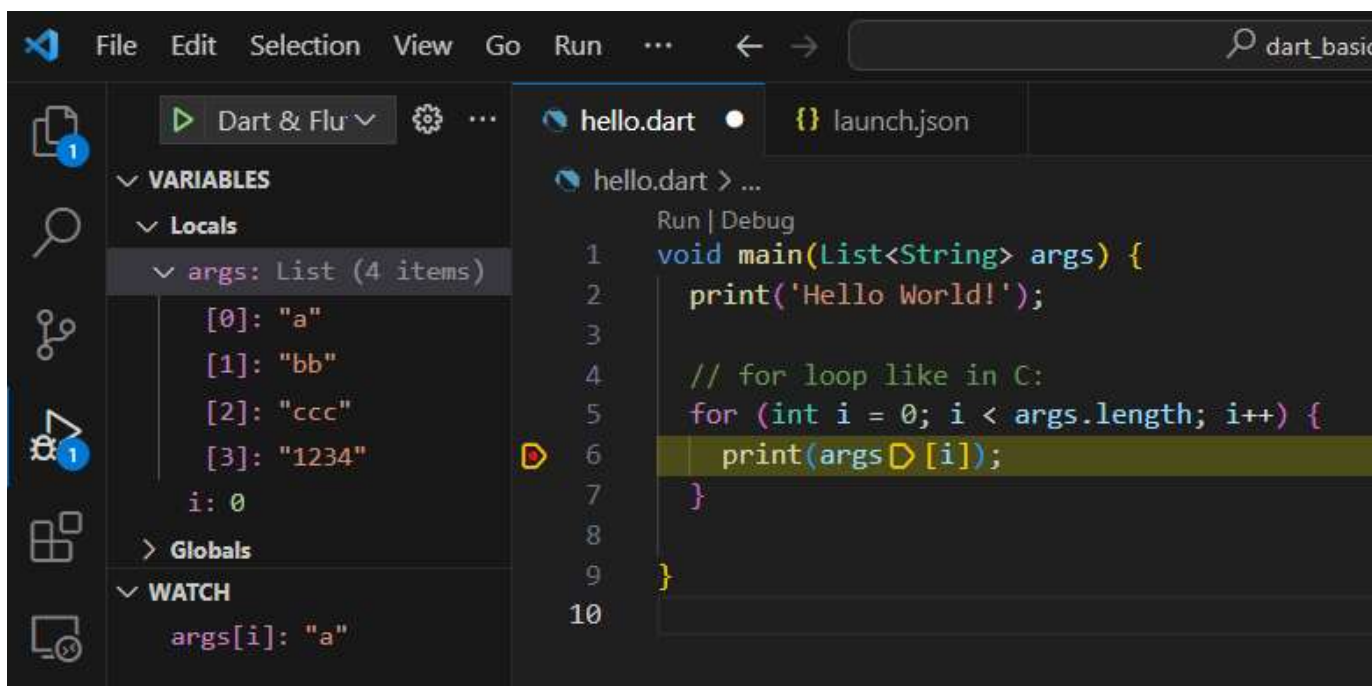


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... Filter (e.g. text, !exclude)
Connecting to VM Service at ws://127.0.0.1:61700/j6ZsxmW6gAU=/ws
Hello World!
a
bb
ccc
1234

Exited.
```



Debugging in VS Code



Shortcuts in VS Code:

F5: Start Debugging

F9: Toggle Breakpoint

F10: Step over



Built-in Types

Built-in types



The Dart language has special support for the following:

- Numbers (`int`, `double`)
- Strings (`String`)
- Booleans (`bool`)
- Records (`((value1, value2))`)
- Lists (`List`, also known as *arrays*)
- Sets (`Set`)
- Maps (`Map`)
- Runes (`Runes`; often replaced by the `characters` API)
- Symbols (`Symbol`)
- The value `null` (`Null`)

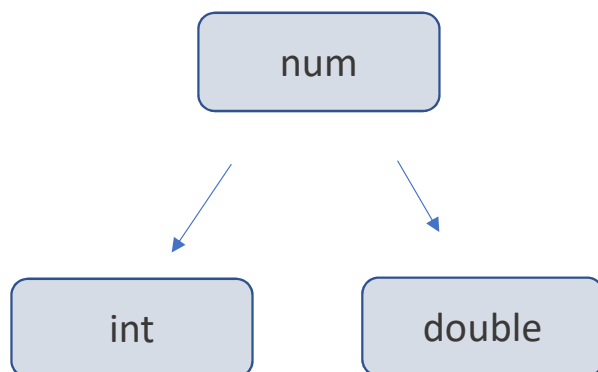
This support includes the ability to create objects using literals. For example, `'this is a string'` is a string literal, and `true` is a boolean literal.

For some more info see <https://www.flutter.de/artikel/dart-basics-datentypen>

BTW: Günther has till now nearly no experience with Records, Sets, Runes and Symbols !



Numbers (int and double)



```
/// Adds [other] to this number.  
///  
/// The result is an [int], as described by [int.+],  
/// if both this number and [other] is an integer,  
/// otherwise the result is a [double].  
num operator +(num other);
```



```
int i = 23;  
double d = 24;  
  
double d1 = 2.34;  
int i1 = 2.34;           // double cannot be assigned to int  
  
d = i;                   // int cannot be assigned to double  
d = i.toDouble();  
  
i = d;                   // double cannot be assigned to int  
i = d.toInt();           // toInt() truncates the decimal places  
  
d = i + i;               // int cannot be assigned to double  
d = i + d;  
i = i + d;               // double cannot be assigned to int
```



Strings

```
String s = 'hello world';

s = "It's me"; // you can either use " or ' to surround strings
s = 'It\'s me'; // backslash is the "escape character" in strings
s = 'c:\\flutter\\sdk'; // \\ stands for \ inside the string
s = '1st line\n2nd line'; // \n is new line for multi-line strings

s = "hello";
s = s + " world"; // "hello world"
s += "!"; // "hello world!"

double d = 1.234567;
s = d; // double cannot be assigned to string
s = d.toString();

s = "d is " + d.toString(); // "d is 1.234567"
s = "d is " + d.toStringAsFixed(2); // "d is 1.23"

num n = 3;
s = "n is " + n.toStringAsFixed(2); // "n is 3.00"
```



String Interpolation

You can access the value of an expression inside a string by using `${expression}`.

```
String greeting = "Hello";  
String person = "Fritz";  
  
print("${greeting} ${person} !"); // prints Hello Fritz !
```

If the expression is an identifier, the `{}` can be skipped.

```
print("$greeting $person !");
```

If the variable inside the `{}` isn't a string, the variable's `toString()` method is called:

```
s = "d is $d !"; // "d is 1.234567 !"  
s = "d + 1 is ${d + 1} !"; // "d + 1 is 2.234567 !"  
s = "d is ${d.toStringAsFixed(1)}"; // "d is 1.2"
```

The text above was copied from <https://shailen.github.io/blog/2012/11/14/dart-string-interpolation/>



Number systems and shift operation for int

```
i = 30;
print(i.toRadixString(16));
print(i.toRadixString(8));
print(i.toRadixString(7));
print(i.toRadixString(2));
i = i >> 1;
// next line makes the same as last line:
// i >>= 1;
print(i.toRadixString(2));
print(i);
i = i << 3;
print(i.toRadixString(2));
print(i);
```

1e
36
42
11110

1111
15

1111000
120

Using “hex” in Assembler:

R0194	6A90	STH	(R9, LEIN)
	0570R		
R0198	20C0	LIH	(R12, LAKTRT)
	02CER		
R019C	21C0	AIH	(R12, J2)
	0020		
R01A0	2BF0	BU	(15, BHIASC)
	0000F		
R01A4	0004	ADT	(4)
R01A6	056AR	ADT	(LPARFL)
R01A8	7A10	BL	(FEHWAN)
	0098R		



What is defined where ?

```
Object o;  
  
o.toString();  
int i = o.hashCode();  
Type t =  
o.runtimeType;
```

```
num n;  
  
n.toInt();  
n.ToDouble();  
n.toStringAsFixed(2);  
int i = n.ceil();  
int i = n.floor();
```

```
int i;  
  
i.toRadixString(2);  
i.isEven;  
i.isOdd;
```



Parse strings for numeric values

```
print(int.tryParse("2"));  
print(int.tryParse("a"));
```

```
2  
null
```

```
int parsed = int.tryParse("2");
```

A value of type 'int?' can't be assigned to a variable of type 'int'.

```
int? parsed = int.tryParse("2");  
print(parsed.isEven);
```

The property 'isEven' can't be unconditionally accessed because the receiver can be 'null'.

```
int? parsed = int.tryParse("2");  
if (parsed != null) {  
    print(parsed.isEven);  
}
```




Nullable Types in Dart

Since version 3.0, Dart provides “sound null safety” („solide null Sicherheit”).

It should avoid null pointer exceptions often seen in Java or C++, e.g. in

```
1 public class Temp {
2
3     public static void main(String[] args) {
4
5         foo(null);
6
7     }
8
9     public static void foo(String s) {
10         System.out.println(s.toLowerCase());
11     }
12 }
13
```

Problems @ Javadoc Declaration Console

<terminated> Temp [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_...
Exception in thread "main" java.lang.NullPointerException
 at Temp.foo(Temp.java:10)
 at Temp.main(Temp.java:5)

```
nullable.dart > ...
Run | Debug
1 void main(List<String> args) {
2     foo1(null);
3 }
4
5 void foo1(String? s) {
6     print(s.toLowerCase());
7 }
8
```

The method 'toLowerCase' can't be unconditionally invoked because the receiver can be 'null'.
Try making the call conditional (using '?..')



Operator ?. (conditional access)

In the expression “s?.toLowerCase()” the method toLowerCase is only called when variable s is not null. When s is null, the whole expression is null:

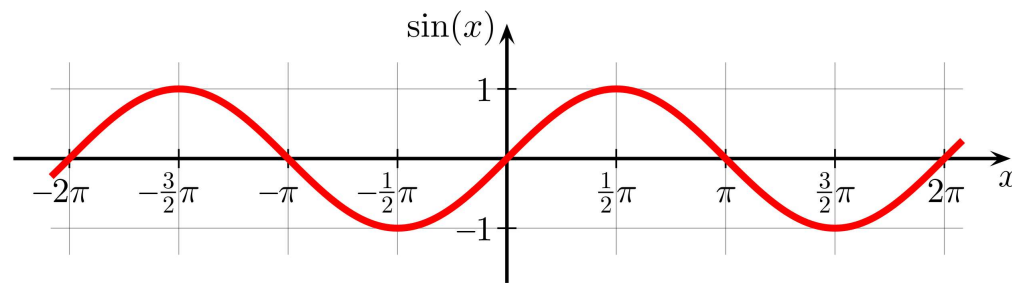
```
Run | Debug
void main(List<String> args) {
    foo2(null);    // prints "null"
    foo2("Hello"); // prints "hello"
}

void foo2(String? s) {
    print(s?.toLowerCase());

    //String lower = s?.toLowerCase();    // compile-error
    String? lower = s?.toLowerCase();
}
```



Math reminder



Kreisfunktionen

φ	x	$\sin x$	$\tan x$	$\cot x$	$\cos x$
171°53'	3,00	0,1411	0,1425	7,015	0,9900
172 28	,01	,1312	,1324	7,555	,9914
173 02	,02	,1213	,1222	8,184	,9926
173 36	,03	,1114	,1121	8,924	,9938
174 11	,04	,1014	,1019	9,809	,9948
174°45'	3,05	0,0915	0,0918	10,89	0,9958
175 20	,06	,0815	,0818	12,23	,9967
175 54	,07	,0715	,0717	13,94	,9974
176 28	,08	,0616	,0617	16,22	,9981
177 03	,09	,0516	,0516	19,37	,9987
177°37'	3,10	0,0416	0,0416	24,03	0,9991
178 11	,11	,0316	,0316	31,64	,9995
178 46	,12	,0216	,0216	46,30	,9998
179 20	,13	,0116	,0116	86,26	0,9999
179 55	,14	0,0016	0,0016	627,9	1,0000
180°29'	3,15	0,0084	0,0084	118,9	1,0000
181 03	,16	,0184	,0184	54,32	0,9998
181 38	,17	,0284	,0284	35,19	,9996
182 12	,18	,0384	,0384	26,02	,9993
182 46	,19	,0484	,0484	20,64	,9988
183°21'	3,20	0,0584	0,0585	17,10	0,9983
183 55	,21	,0684	,0685	14,60	,9977
184 30	,22	,0783	,0786	12,73	,9969
185 04	,23	,0883	,0886	11,28	,9961
185 38	,24	,0982	,0987	10,13	,9952

Copied from <https://upload.wikimedia.org/wikipedia/commons/thumb/a/a2/Sine.svg/2560px-Sine.svg.png>



Library math.dart

This library provides trigonometric, exponential, logarithmic and other functions.

To use them you need to import “math.dart”:

```
math_functions.dart > ...
1  import 'dart:math';
2
   Run | Debug
3  void main() {
4      double d = sin(3.14);
5      print(d);
6  }
```

Context-Menu “Go to definition (F12)” for sin shows:

```
C: > FlutterSDK > flutter_windows_3.3.10-stable > flutter > bin > cache > dart-sdk > lib > math > math.dart > tan
194
195  /// Converts [radians] to a [double] and returns the sine of the value.
196  ///
197  /// If [radians] is not a finite number, the result is NaN.
198  external double sin(num radians);
199
```

```
void main() {
  double d = sin(3.14);
  print(d);
  print(d.toStringAsFixed(10));
  d = sin(pi);
  print(d);
  print(d.toStringAsFixed(10));
}
```

```
0.0015926529164868282
0.0015926529
1.2246467991473532e-16
0.0000000000
```

Definition von pi in math.dart:

```
/// The PI constant.
const double pi = 3.1415926535897932;
```



Exercise

Use dart to create a file “wurzel.exe”.

When this file is executed in a Windows command prompt, it calculates and prints for each of the given command arguments the square root.

Test wurzel.exe also for negative numbers as arguments and for arguments that are no numbers.

2,00 ≤ x ≤ 2,49								
$\frac{1}{x}$	x^2	x	\sqrt{x}	$\sqrt[3]{10x}$	x^3	$\sqrt[3]{x}$	$\sqrt[3]{10x}$	$\sqrt[3]{100x}$
0,50000	4,0000	2,00	1,4142	4,472	8,000	1,2599	2,714	5,848
,49751	,0401	,01	,4177	,483	,121	,2620	,719	,858
,49505	,0804	,02	,4213	,494	,242	,2641	,723	,867
,49261	,1209	,03	,4248	,506	,365	,2662	,728	,877
,49020	,1616	,04	,4283	,517	,490	,2683	,732	,887
0,48780	4,2025	2,05	1,4318	4,528	8,615	1,2703	2,737	5,896
,48544	,2436	,06	,4353	,539	,742	,2724	,741	,906
,48309	,2849	,07	,4387	,550	,870	,2745	,746	,915
,48077	,3264	,08	,4422	,561	8,999	,2765	,750	,925
,47847	,3681	,09	,4457	,572	9,129	,2785	,755	,934



Possible solution

```
wurzel.dart > main
1  import 'dart:math';
2
3  Run | Debug
4  void main(List<String> args) {
5      for (int i = 0; i < args.length; i++) {
6          String arg = args[i];
7          double? d = double.tryParse(arg);
8          if (d == null) {
9              print("argument '$arg' is not a number.");
10             } else if (d < 0) {
11                 print("cannot calculate square root from negative number '$arg'.");
12             } else {
13                 String rounded = sqrt(d).toStringAsFixed(4);
14                 print("square root of '$arg' rounded to 4 decimal places is '$rounded'.");
15                 // instead of the last 2 lines you could also write the following quite long line
16                 //print("square root of $arg rounded to 4 decimal places is '${sqrt(d).toStringAsFixed(4)}'.");
17             }
18         }
19     }
```

```
C:\flutter\code\dart_basics>dart compile exe wurzel.dart
Generated: c:\flutter\code\dart_basics\wurzel.exe

C:\flutter\code\dart_basics>wurzel 4 10 -2 x
square root of '4' rounded to 4 decimal places is '2.0000'.
square root of '10' rounded to 4 decimal places is '3.1623'.
cannot calculate square root from negative number '-2'.
argument 'x' is not a number.
```