Dart Flash Cards

version: 1.1
by: Vandad Nahavandipoor
covers: Dart 2.14.2
Twitter, LinkedIn, YouTube

Data Types

Data types are ways of telling Dart what kind of data stored is stored in the memory. They make it easier for the programmer and for the computer to understand what they are dealing with and at the same time you will minimize the risk of calculating invalid operations such as adding a number to text, an opreation whose result might not be easily understood by the computer

More info:

- Dart - Data Types - GeeksforGeeks
- Dart Programming - Data Types - TutorialsPoint
- The Dart type system
- Dart Data Types - W3Schools
- Video: Basics of Dart, DataTypes

Examples

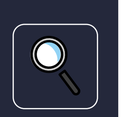
```dart
// This is an example of a integer
10
// this is a string inside single quotation marks
'Vandad Nahavandipoor'
// this is a double precision value
3.14
// and this is a list
[1, 2, 3]
// we can also have hashmaps
{'key1': 'value1', 'key2': 'value2'}
```

Variables

A variable is a named piece of information with a data type, that contains data in the memory, be it in the stack or in the heap. Variables usually have a name and a data type and they either contain a value or not (optionality). There are naming conventions for variables and other rules as to what data they can be assigned to based on their data types.

More info:

- Dart Programming - Variables - Tutorialspoint
- Dart Variables - W3Schools
- Variables and types in Dart - Suragch
- Dart - Variables - GeeksforGeeks
- Dart Variables - TutorialKart



Examples

```dart
/*
below are all examples of variables whose value can
be changed at compile and run time
*/
var foo = 10;
foo = foo + 1; // add 1 to foo

var bar = 'Bar';
bar *= 2; // Bar becomes BarBar now

var baz = 1.2;
baz = baz + 3; // baz is now 4.2
```

Dart

Dart is the language that sits behind Flutter, Google's revolutionary UI framework. Quoting Dart's own website: "Dart is a client-optimized language for developing fast apps on any platform. Its goal is to offer the most productive programming language for multi-platform development, paired with a flexible execution runtime platform for app frameworks."

Further reading:

- Dart programming language - dart.dev
- A tour of the Dart language - dart.dev
- Dart Tutorials - dart.dev
- Dart (programming language) - Wikipedia
- Dart packages - pub.dev

Examples

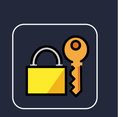
```dart
/*
Dart has many built in features, such as support for
many data types such as integers and doubles
*/
final ex1 = 1;
final ex2 = 1.2;
void func() {
  // and also support for functions
}
// and also function type definitions
typedef MyCallback = void Function(int);
// plus many more
```

Keywords

Keywords are words in a programming language that are reserved for the language itself. In Dart there are variaty of keywords that are reserved for the language and shouldn't be used otherwise as variable names for instance. Some of these words are, "break", "false", "true", "finally", "const", "enum", "else", "abstract" and many more which are documented by the good folks at Google.

Further reading:

- Language tour - keywords - dart.dev
- Dart keywords - w3adda
- Dart Keywords - Javatpoint
- Dart - Keywords - GeeksforGeeks
- Keywords (Reserved Words) in Dart - Codesansar



Examples

```dart
/*
in this case both the words "abstract" and
"class" are keywords, or as some may call them,
reserved words!
*/
abstract class Animal {

}

// here the word "const" is a keyword
const name = 'Foo Bar';

// and here the word "final" is a keyword
final age = 32;
```

Double

A double is a double precision floating point value. This is to say that the container of this type can contain values with decimal points, such as 1.2, or 3.14. Integer values on the other hand don't have the capacity to contain a decimal point. Values of type Double can also be "demoted" or type-cast to a value of type integer and vice versa. In the case of Integer to Double, the type will be promoted!

Further reading:

- Double-precision floating-point format - Wikipedia
- Double-Precision Floating Point - IBM
- double class - dart:core library
- What is a double in Dart? - Educative.io
- Floating-point arithmetic - Wikipedia

Examples

```dart
/*
a double value, or a double precision floating
point value refers to a value that can contain
decimals, in this example, the .2 is the decimal
value after the integer value of 1
*/
final fooBar = 1.2;
final plusTwo = fooBar + 2;
final againPlusTwo = fooBar + 2.0;
// a string cannot be added to a double by default
final invalid = fooBar + 'Foo Bar'; // compile-error
```

Integer

An integer is a value that can be described by a whole number, up to a limit, dictated by its container variable. An example of an integer is 1, 100, 1000, and so on. Integers can be either signed or unsigned. A signed integer can be either positive or negative, while an unsigned integer can only contain positive values, from and including 0 up to the limit dictated by the data type.

Further reading:

- Integer - Wikipedia
- Numbers in Dart - Dart.dev
- int class in Dart - api.flutter.dev
- Integer in Dart - Educative.io
- Integer class - Pub.dev

Examples

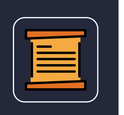
```dart
// a person's age is a good example of an integer
final age = 20;

// you can perform various operators such as
// division on an integer
final dividedByTwo = age / 2;

// multiplication is done through the * operator
final multipliedByTwo = age * 2;

// functions or getters can also return
// values of type integer
final lengthOfName = 'Foo Bar'.length;
```

String

A string in Dart is text usually surrounded by single quotation marks 'like so'. If the string object itself contains single quotation marks such as "let's", then it's usually surrounded by double quotation marks. Dart recomments that all strings by default be surrounded by single quotation marks unless there is a good reason for them to not to. Strings can be compile time constants.

Further reading:

- String class - dart:core library - api.dart.dev
- Dart Programming - String - Tutorialspoint
- Exploring String methods in Dart - Darshan Kawar
- Working with Strings in Dart/Flutter - FrontBackend
- Dart/Flutter String Functions & Operators - Bezkoder



Examples

```dart
// a string is text
final myName = 'Vandad Nahavandipoor';
/*
it can contain numbers as well but this number
will not be considered an integer anymore
and is from this point on a string, meaning that
you cannot add another integer to it
*/
final myAge = '20';
// so this code will not compile
final invalid = myAge + 30;
```

Class

A class is metadata around a type or multiple types. A class can have properties and methods. Imagine a Person class, a property of this class could be "age" while a metohd could be "run". Classes can either be abstract or concrete. Classes can then be instantiated in order to create objects. An object is therefore a copy of that class in memory and can be individually modified.

Further reading:

- Dart Classes and Object - Javatpoint
- Dart Programming - Classes - Tutorialspoint
- Language samples - dart.dev
- Dart - Classes And Objects - GeeksforGeeks
- Dart Class - TutorialKart

Examples

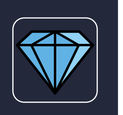
```dart
/*
this is an example of a class with 3 properties namely
age, address and name. here the properties are final
meaning that their values cannot be changed after being
assigned to. the class also has a constructor with
the prefix of const
*/
class Person {
  final String name;
  final String address;
  final int age;
  const Person(this.name, this.address, this.age);
}
```

Constants

A constant, or a compile-time constant, is a value whose internals do not change during the entire execution of the program. An example of a constant value in Dart is the value "1", or a constant instance of a class called Person whose first name is "Foo" and last name is "Bar". Those values will not change for the entire lifetime of the program. They can however be changed before compiling the project.

Further reading:

- Dart final and const - dart.dev
- Compile-time constants and variables - Newbedev
- Constants and Variables - Rebus
- Constant (computer programming) - Wikipedia
- Constants in Programming Language - Toppr



Examples

```dart
/*
foo in this example is a compile-time constant
meaning that its value cannot be changed after
it has been assigned to.
*/
const foo = 'Foo';
/*
so this code is invalid because
foo is a compile-time constant
*/
foo = foo + 'Bar'; // this will not compile
```

Final

Final is a variable modifer that makes a variable assignable only once. Notice that final has nothing to do with constant. A final variable can only be assigned to once, whereas a constant variable can not only be assigned only once, but also has to be a constant! For instance, if you read a user's name from the console into a variable, that value can be stored into a final variable, but not a constant!

Further reading:

- Final and const - dart.dev
- Dart - Const And Final Keyword - GeeksforGeeks
- Difference between Const and Final in Dart - Jeroen Ouwehand
- Difference Between the const and final keywords - Somyarajsinh J
- Difference Between Constant and Final Explained Example

Examples

```dart
/*
this will create an integer variable named
"age" whose value is initially set to 20
and cannot be changed after it has been declared
*/
final age = 20;
final otherAge = age + 20;
/*
if you try to change the value of this variable
after it has been assigned to for the first
time, you will receive a compilation error
*/
age = age + 30; // this will not compile
```

Lists

Lists, as their name denote, can contain more than one object at once. Objects are not named. They are accessible by their index. The first item is always at index 0, item 2 is at index 1 and so on. That's why they call lists zero-based, meaning that their indexes start at 0. A list can contain heterogenous objects, meaning that objects inside the list don't necessarily have to be of the same type.

Further reading:

- Lists - dart.dev
- Dart Programming - Lists - Tutorialspoint
- Dart/Flutter List Tutorial with Examples - BezKoder
- List class - dart:core library - Flutter API
- Dart Lists - W3Schools

Examples

```dart
/*
here is an example of 3 strings placed
inside a list, the list has 3 items. using
the index of 0 you can access 'foo', with the
index of 1 you can get to 'bar' and with
the index of 2 you can get to 'baz'
*/
const names = ['foo', 'bar', 'baz'];
const ages = [20, 30, 44];
/*
this grabs the values inside both "names" and
"ages" and flattens them inside the new list
*/
const all = [...names, ...ages]; // List<Object>
```

Sets

While a list can contain duplicate items, a set's job is to filter out duplicate items. Objects have hash values using which a set determines if two objects are alike. These hash values are usually integers that custom objects can override and define manually. Built-in types have hash values that you don't have to play with in order to have them work together with sets, objects such as strings and integers for instance.

Further reading:

- Sets - dart.dev
- Set class - dart:core library
- Dart Sets - W3Schools
- Dart Programming - Collection Set - Tutorialspoint
- Dart Sets - Javatpoint

Examples

```dart
/*
a set definition starts with curly brackets and ends
with a closing bracket. it cannot contain duplicate values.
duplication of values is defined by their hash values.
*/
const ages = {
  10, 20, 30
};
// this is not a valid set, and will not compile
const names = {
  'foo',
  'foo' // this is not allowed
}
```

Maps

Maps are key value containers, meaning that for a value to be stored inside the map you need to associate a key to it. The key is then used to retrieve the value. A map or hash map or dictionary as it may be named in other languages is usually used to store structured data. Keys of the map need to be hashable and unique. In Dart you can retrieve a value by key, or just retrieve all the keys or all the values separately.

Further reading:

- maps - dart.dev
- Dart Programming - Map - Tutorialspoint
- Dart Map - zetcode.com
- Maps in dart - Jay Tillu
- Dart/Flutter Map, HashMap Tutorial - Bezkoder

Examples

```dart
/*
personInfo in this case is a Map<Object, Object>
where keys are of type Object and the values
are of type Object as well.
*/
final personInfo = {
  'name': 'Foo bar',
  'age': 20,
  'address': 'dummy',
  10: 20,
};
// access the name out of personInfo with its key
final name = personInfo['name'];
```

Functions

A function is a group of lines of code, or even a single line of code, that has a name, and optionally a return value and arguments. Functions are used for giving contextual meanings to code that are related to each other which perform a specific task. For instance, a Person object might have a function called "run" that performs the running task for that particular person object.

Further reading:

- functions - dart.dev
- the main function - dart.dev
- anonymous functions - dart.dev
- Dart Programming - Functions - Tutorialspoint
- Dart function - working with functions in Dart - ZetCode

Examples

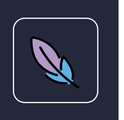
```dart
/*
here is an example of a person class with an empty
function called run, that returns no values (denoted
by the void result type) and takes in an argument
of type double that denotes the speed in kilometers
by which the given person should run when this
function is invoked
*/
class Person {
  void run(double speedInKilometers) {
    // perform the running task here
  }
}
```

Arguments

Arguments are values that are passed to functions in order to give them more information and context as how they should perform their tasks. An argument might have a data type and must have a name that the function internally can use in order to refer to that argument. An argument can optionally specify whether it is a required argument or not. Required arguments are prefixed with the "required" keyword.

Further reading:

- parameters - dart.dev
- Dart: Optional Function Parameters - Zaiste
- Function arguments - Flutter by example
- Dart Optional Parameters - W3Schools
- Optional Named Parameter - Tutorialspoint



Examples

```dart
/*
this function takes in two arguments of type  int and
 it returns the result of adding them together, and the
 result is of the same data type as the incoming arguments
*/
int add(int value1, int value2) {
  return value1 + value2;
}

// another example of a function, a short one!
int subtract(int value1, int value2) => value1 - value2;
```

Operators

Operators are special functions whose names are symbols. Operators can be prefix, infix or suffix meaning that they can perform operations on data by being specified before that data (prefix), between two pieces of data (infix) or by appearing after an object (suffix). Dart comes with its own built-in operators but you are allowed to define your own operators for existing and new objects.

Further reading:

- A tour of the Dart language (Operators) - dart.dev
- Dart Programming - Operators - Tutorialspoint
- Dart Operators - W3Schools
- Operators in Dart - GeeksforGeeks
- Dart Operators - Javatpoint

Examples

```dart
var foo = 10;
/* this is an example of an infix operator that
sits between two values */
final fooPlusTwo = foo + 2; // fooPlusTwo = 12
/* this is a prefix operator that first adds 1
to "foo", and then assigns the result to "bar" */
final bar = ++foo;
/* this is an example of a suffix operator that
first assigns the value of foo to baz and then
adds 1 to foo */
final baz = foo++;
```