Dart cheatsheet

A Dart cheat sheet with the most important concepts, functions, methods, and more. A complete quick reference for beginners.

Getting Started

hello.dart

```
// top-level function where app execution starts
void main(){
    print("Hello World!"); // Print to console
}
```

Every app has a main() function

Variables

```
int x = 2; // explicitly typed
var p = 5; // type inferred - Generic var with type inference

dynamic z = 8; // variable can take on any type
z = "cool"; // cool

// if you never intend to change a variable use final or const. So

final email = "temid@gmail.com"; // Same as var but cannot be reas
final String email = "temid@gmail.com"; // you can't change the va

const qty = 5; // Compile-time constant
```

Datatypes

```
int age = 20; // integers, range -2^63 to 2^63 - 1
double height = 1.85; // floating-point numbers

// You can also declare a variable as a num
num x = 1;  // x can have both int and double values
num += 2.5;
print(num); //Print: 3.5

String name = "Nicola";

bool isFavourite = true;
bool isLoaded = false;
```

String interpolation

```
// can use single or double qoutes for String type
var firstName = 'Nicola';
var lastName = "Tesla";

//can embed variables in string with $
String fullName = "$firstName $lastName";

// concatenate with +
var name = "Albert " + "Einstein";

String upperCase = '${firstName.toUpperCase()}';
print(upperCase); //Print: NICOLA
```

Comments

```
// This is a normal, one-line comment.

/// This is a documentation comment, used to document libraries,
/// classes, and their members. Tools like IDEs and dartdoc treat
/// doc comments specially.

/* Comments like these are also supported. */
```

Imports

```
// Importing core libraries
import 'dart:math';

// Importing libraries from external packages
import 'package:test/test.dart';

// Importing files
import 'path/to/my_other_file.dart';
```

Operators

Arithmatic Operators

```
print(2 + 3); //Print: 5
print(2 - 3); //Print: -1
print(2 * 3); //Print: 6
print(5 / 2);  //Print: 2.5 - Result is a double
print(5 ~/ 2); //Print: 2 - Result is an int
print(5 % 2); //Print: 1 - Remainder

int a = 1, b;
// Increment
b = ++a; // preIncrement - Increment a before b gets its value.
b = a++; // postIncrement - Increment a AFTER b gets its value.

//Decrement
b = --a; // predecrement - Decrement a before b gets its value.
b = a--; // postdecrement - Decrement a AFTER b gets its value.
```

Equality and relational operators

```
print(2 == 2);  //Print: true - Equal
print(2 != 3); //Print: true - Not  Equal
print(3 > 2); //Print: true - Grater than
print(2 < 3); //Print: true - Less than
print(3 >= 3); //Print: true - Greater than or equal to
print(2 <= 3); //Print: true - Less than or equal to
```

Logical operators

```
// !expr inverts the expression (changes false to true, and vice v
// ||   logical OR
// &&   logical AND
```

```dart
bool isOutOfStock = false;
int quantity = 3;
if (!isOutOfStock && (quantity == 2 || quantity == 3)) {
  // ...Order the product...
}
```

Control Flows : Conditionals

if and else if

```dart
if(age < 18){
    print("Teen");
} else if( age > 18 && age <60){
    print("Adult");
} else {
    print("Old");
}
```

switch case

```dart
enum Pet {dog, cat}
Pet myPet = Pet.dog;
switch(myPet){
    case Pet.dog:
        print('My Pet is Dog.');
        break;
    case Pet.cat:
        print('My Pet is Cat.');
        break;
    default:
        print('I don\'t have a Pet');
}
// Prints: My Pet is Dog.
```

Control Flows : Loops

while loop

```dart
while (!dreamsAchieved) {
  workHard();
}
```

while loop check condition before iteration of the loop

do-while loop

```dart
do {
  workHard();
} while (!dreamsAchieved);
```

do-while loop verifies the condition after the execution of the statements inside the loop

for loop

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

var numbers = [1,2,3];
// for-in loop for lists
for(var number in numbers){
    print(number);
}
```

Collections

Lists

```dart
// ordered group of objects
var list = [1, 2, 3];

print(list.length); //Print: 3
print(list[1]); //Print: 2

// other ways of list declaration and initializations

List<String> cities = <String>["New York", "Mumbai", "Tokyo"];

// To create a list that's a compile-time constant
const constantCities = const ["New York", "Mumbai", "Tokyo"];
```

Sets

```dart
// A set in Dart is an unordered collection of unique items.
var halogens = {'fluorine', 'chlorine', 'bromine', 'iodine', 'asta

// to create an empty set
var names = <String>{};
Set<String> names = {}; // This works, too.
//var names = {}; // Creates a map, not a set.
```

Maps

```dart
// a map is an object that associates keys and values
var person = Map<String, String>();
// To initialize the map, do this:
person['firstName'] = 'Nicola';
person['lastName'] = 'Tesla';

print(person); //Print: {firstName: Nicola, lastName: Tesla}
print(person['lastName']); //Print: Tesla


var nobleGases = {
  // Key: Value
  2: 'helium',
  10: 'neon',
  18: 'argon',
};
```

Functions

Functions

Arrow Syntax (=>)

```dart
// functions in dart are objects and have a type
int add(int a, int b){
    return a+b;
}

// functions can be assigned to variables
int sum = add(2,3); // returns: 5

// can be passed as arguments to other functions
int totalSum = add(2, add(2,3)); // returns : 7
```

```dart
// functions that contain just one expression, you can use a short
bool isFav(Product product) => favProductsList.contains(product);
```

Anonymous (lambda) functions

```dart
// small one line functions that dont have name
int add(a,b) => a+b;

// lambda functions mostly passed as parameter to other functions
const list = ['apples', 'bananas', 'oranges'];
list.forEach(
(item) => print('${list.indexOf(item)}: $item'));
//Prints: 0: apples 1: bananas 2: oranges
```

Classes and Objects

Class

```dart
class Cat {
    String name;

    // method
    void voice(){
        print("Meow");
    }
}
```

Object

```dart
// instance of a class
// below myCat is Object of class Cat

void main(){
    Cat myCat = Cat();
    myCat.name = "Kitty";
    myCat.voice(); // Prints: Meow
}
```

Constructors

```dart
class Cat {
    String name;
    Cat(this.name);
}
void main(){
    Cat myCat = Cat("Kitty");
    print(myCat.name); // Prints: Kitty
}
```

Abstract Classes

```dart
// abstract class—a class that can't be ir
// This class is declared abstract and thu
abstract class AbstractContainer {
  // Define constructors, fields, methods.

  void updateChildren(); // Abstract methc
}
```

Getters Setters

```dart
// provide read and write access to an obj
class Cat {
    String name;

    // getter
    String get catName {
        return name;
    }

    // setter
    void set catName(String name){
        this.name = name;
    }
}
```

Implicit interfaces

A basic interface

```dart
// A person. The implicit interface contains greet().
class Person {
  // In the interface, but visible only in this library.
  final String _name;

  // Not in the interface, since this is a constructor.
  Person(this._name);

  // In the interface.
  String greet(String who) => 'Hello, $who. I am $_name.';
}

// An implementation of the Person interface.
class Impostor implements Person {
  String get _name => '';

  String greet(String who) => 'Hi $who. Do you know who I am?';
}

String greetBob(Person person) => person.greet('Bob');

void main() {
  print(greetBob(Person('Kathy'))); // Hello, Bob. I am Kathy.
  print(greetBob(Impostor())); // Hi Bob. Do you know who I am?
}
```

Extending a class

```dart
class Phone {

    void use(){
        _call();
        _sendMessage();
    }
}
// Use extends to create a subclass
class SmartPhone extends Phone {
    void use(){
        // use super to refer to the superclass
        super.use();
        _takePhotos();
        _playGames();
    }
}
```

Exceptions

Throw

```
// throws or raises and exception
throw IntegerDivisionByZeroException();

// You can also throw arbitrary objects
throw "Product out of stock!";
```

Catch

```
try {
    int c = 3/0;
    print(c);
} on IntegerDivisionByZeroException {
    // A specific exception
    print('Can not divide integer by 0.')
} on Exception catch (e) {
    // Anything else that is an exception
    print('Unknown exception: $e');
} catch (e) {
    // No specified type, handles all
    print('Something really unknown: $e');
}
```

Finally

```
// To ensure that some code runs whether c
try {
  cookFood();
} catch (e) {
  print('Error: $e'); // Handle the except
} finally {
  cleanKitchen(); // Then clean up.
}
```

Futures

Async Await

```
// functionswhich are asynchronous: they r
// The async and await keywords support as

Future<String> login() {
 String userName="Temidjoy";
 return
  Future.delayed(
    Duration(seconds: 4), () => userName);
}

// Asynchronous
main() async {
 print('Authenticating please wait...');
 print(await userName());
}
```

Miscellaneous

Null and Null aware

```
int x; // The initial value of any object is null

// ?? null aware operator

x ??=6; // ??= assignment operator, which assigns a value of a var
print(x); //Print: 6

x ??=3;
print(x); // Print: 6 - result is still 6

print(null ?? 10); // Prints: 10. Display the value on the left if
```

Ternary Operator

```
// condition ? exprIfTrue : exprIfFalse
bool isAvailable;

isAvailable ? orderproduct() : addToFavourite();
```

Spread Operator (...)

```
// to insert multiple values into a collection.
var list = [1, 2, 3];
var list2 = [0, ...list];

print(list2.length); //Print: 4
```

Cascade notation (..)

```
// allows you to make a sequence of operations on the same object

// rather than doing this
var user = User();
user.name = "Nicola";
user.email = "nicola@g.c";
user.age = 24;

// you can do this
var user = User()
  ..name = "Nicola"
  ..email = "nicola@g.c"
  ..age = 24;
```

Conditional Property Access

```
userObject?.userName

//The code snippet above is equivalent to following:
(userObject != null) ? userObject.userName : null
```

```
//You can chain multiple uses of ?. together in a single expressio
userObject?.userName?.toString()

// The preceeding code returns null and never calls toString() if
```

Top Cheatsheet

- **Python** Cheatsheet — Quick Reference
- **Vim** Cheatsheet — Quick Reference
- **JavaScript** Cheatsheet — Quick Reference
- **Bash** Cheatsheet — Quick Reference

Recent Cheatsheet

- **Google Search** Cheatshee — Quick Reference
- **Kubernetes** Cheatsheet — Quick Reference
- **ES6** Cheatsheet — Quick Reference
- **ASCII Code** Cheatsheet — Quick Reference

QuickRef.ME

Share quick reference and cheat sheet for developers.

中文版 #Notes