

Dart Quick Reference

Types, functions, classes, async, null safety, collections

Basics

Hello World

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

Variables

```
var name = 'Dart';      // type inferred  
String lang = 'Dart';  // explicit type  
final pi = 3.14;       // runtime constant  
const max = 100;       // compile-time constant
```

String Interpolation

```
var name = 'World';  
print('Hello, $name!');  
print('1 + 1 = ${1 + 1}');
```

Types

Built-in Types

int	64-bit integer
double	64-bit floating point
num	Supertype of int and double
String	UTF-16 string
bool	true or false
List	Ordered collection (array)
Set	Unordered unique collection
Map	Key-value pairs
dynamic	Any type, disables static checking
void	No return value

Type Checks & Casts

```
if (obj is String) print(obj.length);  
var s = obj as String; // cast  
print(obj.runtimeType); // runtime type
```

Functions

Function Syntax

```
int add(int a, int b) => a + b;  
void greet({required String name}) {  
  print('Hello, $name!');  
}
```

Parameters

int f(int a)	Required positional parameter
int f([int a = 0])	Optional positional with default
f({required int a})	Required named parameter
f({int a = 0})	Optional named with default

Closures & Tearoffs

```
var square = (int n) => n * n;  
[1, 2, 3].map(e) => e * 2);  
[1, 2, 3].forEach(print); // tearoff
```

Control Flow

Conditionals

```
if (x > 0) { print('pos'); }  
else if (x == 0) { print('zero'); }  
else { print('neg'); }  
var result = x > 0 ? 'pos' : 'neg';
```

Loops

```
for (var i = 0; i < 5; i++) { }  
for (var item in list) { }  
while (x > 0) { x--; }  
do { x--; } while (x > 0);
```

Switch & Pattern Matching

```
switch (color) {  
  case 'red': print('R'); break;  
  case 'blue': print('B'); break;  
  default: print('?');  
}
```

Classes

Class Definition

```
class Point {  
  final double x, y;  
  Point(this.x, this.y);  
  double distanceTo(Point p) =>  
    sqrt(pow(x - p.x, 2) + pow(y - p.y, 2));  
}
```

Named & Factory Constructors

```
class Point {  
  double x, y;  
  Point(this.x, this.y);  
  Point.origin() : x = 0, y = 0;  
  factory Point.fromJson(Map j) =>  
    Point(j['x'], j['y']);  
}
```

Inheritance

```
class Animal { void speak() {} }  
class Dog extends Animal {  
  @override  
  void speak() => print('Woof');  
}
```

Mixins & Extensions

Mixins

```
mixin Flyable {  
  void fly() => print('Flying');  
}  
class Bird with Flyable {}
```

Extension Methods

```
extension StringX on String {  
  String capitalize() =>  
    '${this[0].toUpperCase()}${substring(1)}';  
}  
print('hello'.capitalize()); // Hello
```

Abstract & Implements

```
abstract class Shape {  
  double area();  
}  
class Circle implements Shape {  
  final double r;  
  Circle(this.r);  
  @override double area() => pi * r * r;  
}
```

Async/Await

Futures

```
Future<String> fetchData() async {  
  var res = await http.get(uri);  
  return res.body;  
}
```

Streams

```
Stream<int> count(int n) async* {  
  for (var i = 0; i < n; i++) {  
    yield i;  
  }  
}
```

Error Handling

```
try {  
  var data = await fetchData();  
} on HttpException catch (e) {  
  print('HTTP error: $e');  
} catch (e) {  
  print('Error: $e');  
}
```

Collections

List Operations

```
var nums = [1, 2, 3];  
nums.add(4);  
nums.where((n) => n > 2); // [3, 4]  
nums.map((n) => n * 2); // [2,4,6,8]  
var sorted = nums.sort();
```

Map Operations

```
var m = {'a': 1, 'b': 2};  
m['c'] = 3;  
m.containsKey('a'); // true  
m.entries.map(e) => '${e.key}=${e.value}';
```

Spread & Collection If/For

```
var all = [0, ...nums];  
var nav = ['Home', if (isAdmin) 'Admin'];  
var sq = [for (var i in nums) i * i];
```

Null Safety

Nullable Types

int?	Nullable int (can be null)
int	Non-nullable int (never null)
!	Null assertion operator
?.	Null-aware access
??	Default if null
??=	Assign if null
late	Deferred initialization

Null Safety Examples

```
String? name; // nullable  
int len = name?.length ?? 0;  
late final String title; // set before use  
name ??= 'default'; // assign if null
```

Dart Quick Reference

Common Patterns

Enum with Values

```
enum Color {  
  red('FF0000'), green('00FF00');  
  final String hex;  
  const Color(this.hex);  
}
```

Records & Destructuring

```
(String, int) userInfo() => ('Alice', 30);  
var (name, age) = userInfo();  
print('$name is $age');
```

Sealed Classes

```
sealed class Shape {}  
class Circle extends Shape { final double r; Circle(this.r); }  
class Rect extends Shape { final double w, h; Rect(this.w,  
this.h); }
```