

RÉFÉRENCE RAPIDE DART

Types, fonctions, classes, async, null safety, collections

Bases	<pre>abstract class Shape { double area(); } class Circle implements Shape { final double r; Circle(this.r); @override double area() => pi * r * r; }</pre>
Hello World	<pre>void main() { print('Hello, Dart!'); }</pre>
Variables	<pre>var name = 'Dart'; // type inféré String lang = 'Dart'; // type explicite final pi = 3.14; // constante d'exécution const max = 100; // constante de compilation</pre>
Interpolation de chaînes	<pre>var name = 'World'; print('Hello, \$name!'); print('1 + 1 = \${1 + 1}');</pre>
Types	
Types intégrés	
int	Entier 64 bits
double	Virgule flottante 64 bits
num	Supertype de int et double
String	Chaîne UTF-16
bool	true ou false
List	Collection ordonnée (tableau)
Set	Collection unique non ordonnée
Map	Paires clé-valeur
dynamic	Tout type, désactive la vérification statique
void	Pas de valeur de retour
Vérifications de type et casts	
	<pre>if (obj is String) print(obj.length); var s = obj as String; // cast print(obj.runtimeType); // type à l'exécution</pre>
Fonctions	
Syntaxe des fonctions	
	<pre>int add(int a, int b) => a + b; void greet({required String name}) { print('Hello, \$name'); }</pre>
Paramètres	
int f(int a)	Paramètre positionnel requis
int f([int a = 0])	Positionnel optionnel avec valeur par défaut
f({required int a})	Paramètre nommé requis
f({int a = 0})	Nommé optionnel avec valeur par défaut
Fermetures et tearoffs	
	<pre>var square = (int n) => n * n; [1, 2, 3].map(e) => e * 2); [1, 2, 3].forEach(print); // tearoff</pre>
Flux de contrôle	
Conditions	
	<pre>if (x > 0) { print('pos'); } else if (x == 0) { print('zero'); } else { print('neg'); } var result = x > 0 ? 'pos' : 'neg';</pre>
Boucles	
	<pre>for (var i = 0; i < 5; i++) { } for (var item in list) { } while (x > 0) { x--; } do { x--; } while (x > 0);</pre>
Switch et correspondance de motifs	
	<pre>switch (color) { case 'red': print('R'); break; case 'blue': print('B'); break; default: print('?'); }</pre>
Classes	
Définition d'une classe	
	<pre>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)); }</pre>
Constructeurs nommés et factory	
	<pre>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']); }</pre>
Héritage	
	<pre>class Animal { void speak() {} } class Dog extends Animal { @override void speak() => print('Woof!'); }</pre>
Mixins et extensions	
Mixins	
	<pre>mixin Flyable { void fly() => print('Flying'); } class Bird with Flyable {}</pre>
Méthodes d'extension	
	<pre>extension StringX on String { String capitalize() => '\${this[0].toUpperCase()}\${substring(1)}'; } print('hello'.capitalize()); // Hello</pre>
Abstrait et Implements	

Async/Await

```
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;
  }
}
```

Gestion des erreurs

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

Collections

Opérations sur les listes

```
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();
```

Opérations sur les maps

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

Spread et collection if/for

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

Null safety

Types nullables

```
int? int nullable (peut être null)
int int non nullable (jamais null)
! Opérateur d'assertion non-null
? Accès null-aware
?? Valeur par défaut si null
??= Assigner si null
late Initialisation différée
```

Exemples de null safety

```
String? name; // nullable
int len = name?.length ?? 0;
late final String title; // à définir avant usage
name ??= 'default'; // assigner si null
```

Patterns courants

Enum avec valeurs

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

Records et destructuration

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

Classes sealed

```
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); }
```