

Dart 快速参考

类型、函数、类、async、null 安全、集合

基础

Hello World

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

变量

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

字符串插值

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

类型

内置类型

int	64 位整数
double	64 位浮点
num	int 和 double 的父类型
String	UTF-16 字符串
bool	true 或 false
List	有序集合 (数组)
Set	无序唯一集合
Map	键值对
dynamic	任意类型, 禁用静态检查
void	无返回值

类型检查与转换

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

函数

函数语法

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

参数

int f(int a)	必需位置参数
int f([int a = 0])	可选位置参数 (带默认值)
f({required int a})	必需命名参数
f({int a = 0})	可选命名参数 (带默认值)

闭包与 Tearoff

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

控制流

条件判断

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

循环

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

Switch 与模式匹配

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

类

类定义

```
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));  
}
```

命名构造函数与工厂构造函数

```
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']);  
}
```

继承

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

Mixin 与扩展

Mixin

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

扩展方法

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

抽象类与实现

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

Async/Await

Future

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

Stream

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

错误处理

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

集合

List 操作

```
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 操作

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

展开与集合 if/for

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

null 安全

可空类型

int?	可空 int (可以为 null)
int	非空 int (永不为 null)
!	非空断言运算符
?.	空安全访问
??	为 null 时取默认值
??=	为 null 时赋值
late	延迟初始化

null 安全示例

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

Dart 快速参考

常用模式

带值枚举

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

Record 与解构

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

Sealed 类

```
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); }
```