Cheat Sheet: Exploring Dart Language

Variables and types

Type	Description	Code example
Var	Inferred type variable	`var name = 'Dart';`
Dynamic	Variable types can change dynamically	`dynamic x = 42;`
Final	Constant at runtime	`final cityName = 'New York';`
Const	Compile-time constant	`const PI = 3.14;`

Functions and methods

Type	Description	Code example
Function	Defines a function that adds two numbers	`int add(int a, int b) => a + b;`
Arrow syntax	Uses arrow syntax for concise function declaration	`void printItem(item) => print(item);`
Required parameters	Must be explicitly provided in the function call	`int multiply(int a, int b) => a * b;`
Optional positional parameters	Can be omitted; enclosed in square brackets	`String fullName(String firstName, [String middleName, String lastName])`
Named parameters	Specified by name; can be required or optional with default values, enclosed in curly braces	`void greet({required String name, String greeting = 'Hello'}) => print('$greeting, $name!');`
Default parameters	Allows default values if not provided in the function call	`String describe(String name, {int age = 30, String city = 'Unknown'})`
Closures	Anonymous functions that can capture variables from their context	`List<int> numbers = [1, 2, 3]; numbers.forEach((number) => print(number * 2));`

Classes and OOP

Type	Description	Code example
Class	Defines a simple class with properties	`class Person { String name; int age; }`
Inheritance	Demonstrates basic inheritance in Dart	`class Employee extends Person { int salary; }`
Encapsulation	Uses class methods and visibility to enforce encapsulation	`class Person { String name; // Public property int _age; // Private property, underscore prefix in Dart }`
Public properties	Can be accessed from any location where the object is visible	`class Person { String name; // Public property }`
Private properties	Prefixed with an underscore and can only be accessed within the class	`class Person { int _age; // Private property }`
Getters and Setters	Control access to class properties	`class Person { int _age; int get age => _age; // Getter set age(int value) { // Setter _age = value; } }`
Static methods	Belong to the class rather than any instance of the class and can be called without an object	`class Utility { static int add(int a, int b) { return a + b; } }`

Anonymous functions	Used for single-expression functions; also known as lambdas or closures	`var list = ['apples', 'bananas', 'oranges']; list.forEach((item) { print(item); });`

Common data structures

Type	**Description**	**Code example**
List	Ordered collection of items	`List<int> numbers = [1, 2, 3];`
Map	Key-value pairs collection	`Map<String, int> ages = {'Alice': 18, 'Bob': 20};`
Set	Unordered collection of unique items	`Set<String> names = {'Alice', 'Bob'};`
Queues	FIFO collection for elements	`Queue<int> queue = Queue(); queue.addAll([1, 2, 3]);`
LinkedLists	Sequence of elements where each element points to the next	`LinkedList<int> linkedList = LinkedList(); linkedList.add(1);`

Libraries and command line utilities

Type	**Description**	**Code example**
Import	Access built-in Dart libraries	`import 'dart:math';`
CLI commands	Compile Dart code to native executable	`dart compile exe test.dart`
Dart SDK	Essential tool for running and managing Dart applications	`dart run, dart create`
Pub tool	Package manager to handle dependencies	`dart pub get, dart pub add http`
Dart DevTools	Suite for debugging and performance profiling	`Performance profiling, memory analysis, and widget inspection`
dart:core	Fundamental classes and functions	`Handling strings, numbers, collections like List and Map.`
dart:math	Provides mathematical constants and functions	`sin, cos, sqrt, and constants like pi.`
dart:async	Supports asynchronous programming	`Future, Stream for handling asynchronous operations.`
dart:convert	Handling JSON, UTF-8 encoding/decoding	`jsonEncode, jsonDecode for JSON data manipulation.`
Custom libraries	Creating and using your own library	`library my_utils; int add(int a, int b) => a + b;`

Skills Network