Dart Programming Language Cheatsheet

A comprehensive cheat sheet for the Dart programming language, covering syntax, data types, control flow, classes, and more. Perfect for quick reference and learning.

Dart Basics

Syntax Overview

Statements: End with a semicolon `;`.
Comments: `//` for single-line, `/* ... */` for multi-line.
Variables: Declared using `var`, `dynamic`, or specific types (e.g., `int`, `String`).

Functions: Defined using `returnType functionName(parameters) { ... }`.
Main Function: `void main() { ... }` is the entry point of a Dart program.

Data Types

`int`	Integers (whole numbers).
`double`	Floating-point numbers.
`String`	Sequence of characters.
`bool`	Boolean values: `true` or `false`.
`List`	Ordered collection of items.
`Map`	Collection of key-value pairs.

Variables

Declaring variables:

```
var name = 'Dart'; // Type is inferred
String language = 'Dart'; // Explicitly typed
dynamic anything = 123; // Can change type
final String constantName = 'Dart'; //
Constant - single assignment
const double compileTimeConstant = 3.14; //
Compile-time constant
```

Control Flow

Conditional Statements

If-Else Statement:

```
if (condition) {
  // Code to execute if condition is true
} else {
  // Code to execute if condition is false
}
```

Switch Statement:

```
switch (expression) {
  case value1:
    // Code to execute if expression == value1
    break;
  case value2:
    // Code to execute if expression == value2
    break;
  default:
    // Code to execute if no case matches
}
```

Loops

For Loop:

```
for (int i = 0; i < 10; i++) {
  // Code to execute
}
```

While Loop:

```
while (condition) {
  // Code to execute while condition is true
}
```

Do-While Loop:

```
do {
  // Code to execute at least once
} while (condition);
```

For-In Loop:

```
var list = [1, 2, 3];
for (var item in list) {
  print(item);
}
```

Exception Handling

```
try {
  // Code that might throw an exception
} catch (e) {
  // Code to handle the exception
} finally {
  // Code that always executes
}
```

Functions and Classes

Functions

```
// Function definition
returnType functionName(param1, param2) {
  // Function body
  return value;
}

// Example
String greet(String name) {
  return 'Hello, $name!';
}

// Arrow function (shorthand for single-expression functions)
String greetArrow(String name) => 'Hello, $name!';
```

Optional Parameters:

```
// Positional optional parameters
String greetOptional([String? name]) {
  if (name != null) {
    return 'Hello, $name!';
  } else {
    return 'Hello!';
  }
}

// Named optional parameters
String greetNamed({String? name}) {
  if (name != null) {
    return 'Hello, $name!';
  } else {
    return 'Hello!';
  }
}
```

Classes

```
class MyClass {
  // Instance variables
  String? name;
  int? age;

  // Constructor
  MyClass(this.name, this.age);

  // Method
  void sayHello() {
    print('Hello, my name is $name and I am $age years old.');
  }
}

// Creating an instance
void main() {
  var myObject = MyClass('Dart', 10);
  myObject.sayHello();
}
```

Inheritance:

```
class Animal {
  void makeSound() {
    print('Generic animal sound');
  }
}

class Dog extends Animal {
  @override
  void makeSound() {
    print('Woof!');
  }
}
```

Asynchronous Programming

Futures

```
Future<String> fetchData() async {
  // Simulate fetching data
  await Future.delayed(Duration(seconds: 2));
  return 'Data fetched!';
}

void main() async {
  print('Fetching data...');
  String data = await fetchData();
  print(data);
}
```

Streams

```
Stream<int> countStream(int to) async* {
  for (int i = 1; i <= to; i++) {
    await Future.delayed(Duration(seconds:
1));
    yield i;
  }
}

void main() async {
  await for (var number in countStream(5)) {
    print(number);
  }
}
```

Async/Await

Used to simplify asynchronous code, making it look and behave a bit more like synchronous code.

```
Future<void> myAsyncFunction() async {
  print('Start');
  await Future.delayed(Duration(seconds: 1));
  print('Middle');
  await Future.delayed(Duration(seconds: 1));
  print('End');
}

void main() {
  myAsyncFunction();
}
```