Dart Cheat Sheet

Variables

```
int n1 = 5; // explicitly typed
var n2 = 4; // type inferred
// n2 = "abc";  // error
dynamic n3 = 4; // dynamic means n3
                // can take on any
                // type
n3 = "abc";
double n4;  // n4 is null
String s1 = 'Hello, world!';
var s2 = "Hello, world!";
```

Constants

```
const PI = 3.14; // const is used
// for compile-time constant

final area = PI * 5*5;
// final variables can only be set
// once
```

Conditional Expressions

```
var grade = 3;
var reply = grade > 3 ? "Cool":"Not
cool";

var input; // input is null
var age = input ?? 0;
print(age); // 0
```

Functions

```
void doSomething() {
  print("doSomething()");
}
int addNums1(num1, num2, num3) {
  return num1+num2+num3;
}

  doSomething();
  print(addNums1(1,2,3));
```

Arrow Syntax

```
void doSomethingElse() {
   doSomething();
}

// the above can be rewritten using
// arrow syntax
void doSomethingElse() =>
  doSomething();
```

Optional Positional Parameters

```
int addNums2(num1, [num2=0, num3=0])
{
  return num1+num2+num3;
}

  print(addNums2(1));
  print(addNums2(1,2));
  print(addNums2(1,2,3));
```

Named Parameters

```
// named parameters
int addNums3({num1, num2, num3}) {
  return num1+num2+num3;
}

  print(addNums3(
    num1:1,num2:2,num3:3));
```

Optional Named Parameters

```
int addNums4(num1, {num2=0, num3=0})
{
  return num1+num2+num3;
}
  print(addNums4(1));
  print(addNums4(1,num3:2));
  print(addNums4(1,num2:5,num3:2));
```

Parsing

```
var s1 = "123";
var s2 = "12.56";
var s3 = "12.a56";
var s4 = "12.0";
print(num.parse(s1)); // 123
print(num.parse(s2)); // 12.56
print(num.parse(s3));
  // FormatException: 12.a56

print(num.tryParse(s3)); // null
```

String Interpolation

```
var s1 = "Hello";
var s2 = "world";
var s3 = s1 + ", " + s2;
var s = "${s3}!";
print(s);   // Hello, world!
print("Sum of 5 and 6 is ${5+6}");
// Sum of 5 and 6 is 11
```

List (Arrays)

```
// dynamic list
var arr = [1,2,3,4,5];
print(arr.length); // 5
print(arr[1]);     // 2
arr[4] *= 2;
print(arr[4]);     // 10
arr.add(6);
print(arr);  // [1, 2, 3, 4, 10, 6]

List arr2;
arr2 = arr;
arr2[1] = 9;

print(arr);  // [1, 9, 3, 4, 10, 6]
print(arr2); // [1, 9, 3, 4, 10, 6]

// fixed size list
var arr3 = new List(3);
print(arr3); // [null, null, null]
arr3.add(5);
// Uncaught exception:
// Unsupported operation: add
```

Map

```
var details = {"name":"Sam",
               "age":"40"};
print(details);

var devices = new Map();
var apple = ["iPhone","iPad"];
var samsung = ["S10","Note 10"];
devices["Apple"] = apple;
devices["Samsung"] = samsung;
for (String company in
  devices.keys) {
  print(company);
  for (String device in
    devices[company]) {
    print(device);
  }
}
```

Lambda Functions

```
var nums = new
  List<int>.generate(10, (i) => i);
print(nums);
// [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

var odds = nums.where(
  (n) => n % 2 == 1).toList();
print(odds); // [1, 3, 5, 7, 9]

var sum = nums.reduce(
  (s,n) => s + n);
print(sum); // 45

var prices = nums.map(
  (n) => "\$$n").toList();
print(prices);
// [$0, $1, $2, $3, $4, $5, $6, $7,
// $8, $9]
```

Higher Order Functions

```
var names = ["Jimmy","TIM","Kim"];
// sort alphabetically with case
// insensitivity
names.sort(
    (a, b) =>
      a.toUpperCase().compareTo(
        b.toUpperCase())
);
print(names);
// [Jimmy, Kim, TIM]

// sort by length of name
names.sort((a,b) {
  if (a.length > b.length)
    return 1;
  else
    return -1;
});
print(names);
// [Kim, TIM, Jimmy]
```

```
List bubbleSort(List items, bool
 Function (int,int) compareFunction)
{
  for (var j=0; j<items.length-1;
    j++) {
    var swapped = false;
    for (var i=0;
      i<items.length-1-j; i++) {
      if (!compareFunction(items[i],
          items[i+1])) {
        var t = items[i+1];
        items[i+1] = items[i];
        items[i] = t;
        swapped = true;
      }
    }
    if (!swapped) break;
  }
  return items;
}

var nums = [5,2,8,7,9,4,3,1];

// sort in ascending order
var sortedNums = bubbleSort(nums,
  (n1,n2) => n1<n2);
print(sortedNums);

// sort in descending order
sortedNums = bubbleSort(nums,
  (n1,n2) => n1>n2);
print(sortedNums);
```

Iterations

```
for (int i=0;i<5; i++) {
  print(i);
} // prints 0 to 4

var list = [1,2,3,4,5];
for (final i in list) {
  print(i);
} // prints 1 to 5

int i=0;
while (i < 5) {
  print(i);
  i++;
} // prints 0 to 4

i = 0;
do {
  print(i);
  i++;
} while (i<5);
// prints 0 to 4
```

Class

```
class MyLocation {
}

// type inference
var loc1 = new MyLocation();

// declare and initialize
MyLocation loc2 = new MyLocation();
```

Properties

```
class MyLocation {
  // read/write properties
  var lat;
  var lng;

  // read-only property
  final arrived = false;
}

loc1.lat = 57.123;
loc1.lng = 37.22;
// loc1.arrived = true; // error
var arr = loc1.arrived;
```

Methods

```
class MyLocation {
  // read/write properties
  var lat;
  var lng;
  // read-only property
  final arrived = false;
  void someMethod() {
  }
}

loc1.someMethod();
```

Constructors

```
class MyLocation {
  // read/write properties
  var lat;
  var lng;

  // read-only property
  final arrived = false;

  // unnamed constructor
  MyLocation() {
    this.lat = 0;
    this.lng = 0;
  }

  // named constructor
  MyLocation.withPosition(
    var lat, var lng) {
    this.lat = lat;
    this.lng = lng;
  }
  void someMethod() {
  }
}

  var loc1 = new MyLocation();
  var loc2 = new
    MyLocation.withPosition(
      57.123,37.22);
```

Getters and Setters

```
class MyLocation {
  double _lat;
  double _lng;
  double get lat => _lat;
  set lat (double value) {
    if (value > 90 || value < -90) {
      throw("Invalid latitude");
    }
    _lat = value;
  }

  double get lng => _lng;
  set lng (double value) {
    if (value > 180 ||
        value < -180) {
      throw("Invalid longitude");
    }
    _lng = value;
  }

  // read-only property
  final arrived = false;

  // unnamed constructor
  MyLocation() {
    this.lat = 0;
    this.lng = 0;
  }

  // named constructor
  MyLocation.withPosition(
    var lat, var lng) {
    this.lat = lat;
    this.lng = lng;
  }

  void someMethod() {
  }
}

  var loc1 = new MyLocation();
  var loc2 = new
    MyLocation.withPosition(
      57.123,37.22);

  loc1.lat = 57.123;
  loc1.lng = 37.22;

  loc2.lat = 999;
  // Uncaught exception:Invalid
  // latitude
```

Inheritance

```
// abstract class cannot be
// instantiated directly
abstract class Shape {
  double length;
  double width;

  // without this zero-argument
  // constructor, class cannot be
  // extended
  Shape() {
    this.length = 0;
    this.width = 0;
  }

  // constructor with another name
  Shape.withDimension(double length,
                      double width){
    this.length = length;
    this.width = width;
  }
  double perimeter() {
    return 2 * (this.length +
                this.width);
  }
  double area() {
    return this.length * this.width;
  }
}

class Rectangle extends Shape {
  Rectangle() {}
  Rectangle.withDimension(
    double length, double width):
    super.withDimension(
      length, width);
}
```

Final Class

```
// Square cannot be extended (it
// does not have a zero-argument
// constructor)
class Square extends Rectangle {
  Square(double length):
    super.withDimension(
      length, length);
}

  Square s = new Square(5);
  print(s.area());      // 25
  print(s.perimeter()); // 20
```

Overriding

```
class Circle extends Shape {
  Circle(double radius):
    super.withDimension(
      radius, radius);
  double area() {
    return 3.14 * this.length *
                  this.length;
  }
  double perimeter() {
    return (2 * 3.14 * this.length);
  }
  // overloading of methods not
  // supported in Dart
}

  Circle c = new Circle(6);
  print(c.area());
  // 113.03999999999999

  print(c.perimeter()); // 37.68
```

Static Members/Methods

```
class Car {
  static var MilesToKM = 1.60934;
  static double kilometersToMiles(
    double km) {
    return km / 1.60934;
  }
  void accelerate() {}
  void decelerate() {}
  void stop() {}
  void printSpeed() {}
}
```

Interfaces

```
class CarInterface {
  void accelerate() {
    // default implementation
    ...
  }
  void decelerate() {}
  void accelerateBy(int amount) {}
}

class MyCar implements
  CarInterface {
  void accelerate() {
  }
  void decelerate() {
  }
  void accelerateBy(int amount) {
  }
}
```