Dart Cheat Sheet

200Lab Education

Starting Point

```dart
void main() {
  print('Hello World!');
}
```

Variables

```dart
int year = 2021; // explicitly typed int
var month = 01; // inferred int created with var
double day; // null

p = 'aaa'; // error

dynamic numberOfCats; // dynamic variable

numberOfCats = '3'; // dynamic String
numberOfCats = 0.2; // dynamic double
bool isCat = true; // explicitly typed boolean
```

Constants

```dart
const hourInDay = 24; // Compile-time constants
hourInDay = 23; // error can't assign to the const variable

final age = 23; // Immutables with final
age = 24; // error can't assign to the final variable
```

Conditional Expressions

```dart
var hero = 'Batman';

if (hero == 'Superman' || hero == 'Flash'){
  print('Hero belongs to DC Universe.');
} else if (hero == 'Iron man' || hero == 'Hawkeye'){
  print('Hero belongs to Marvel Universe.');
} else{
  print('Hero belongs to another Universe.');
}

switch(hero){
  case 'Superman':
  case 'Flash':
  case 'Batman':
    print('Hero belongs to DC Universe.');
    break;
  case 'Iron man':
  case 'Black Panther':
  case 'Captain America':
    print('Hero belongs Marvel DC Universe.');
    break;
  default:
    print('Hero belongs to another Universe.');
    break;
}
```

Loops

```dart
for (int i = 0; i < 69; i++) {
  print(i);
} // prints 0 to 69

final list = ['a', 'b', 'c', 'd', 'e'];

for (final i in list) {
  print(i);
} // prints: a, b, c, d, e

final numbers = [1, 2, 3, 4, 5, 6];
numbers.forEach(print);

int i = 100;
while (i < 105) {
  print(i);
  i++;
} //prints: 100, 101, 102, 103, 104

i = 100;
do {
  print(i);
  i++;
} while (i < 105);
//prints: 100, 101, 102, 103, 104
```

Dart Cheat Sheet

200Lab Education

String Interpolation

```dart
int x = 9;
int y = 6;

print('${x + y}'); // 15
print('${"flutter".toUpperCase()}'); // 'FLUTTER'
print('$x$y'); // concat x & y: 96
```

Parsing

```dart
var s1 = "456";
var s2 = "12.21";
var s3 = "18.01";
var s4 = "12.a56";

print(int.parse(s1)); // 456
print(double.parse(s2)); // 12.21
print(num.parse(s3)); // 18.01
print(num.parse(s4)); // error FormatException
```

Null-aware Operators

```dart
int y; // initial null value
y ??= 10; // if y null, y will be 10

print(y ?? 8);
// result is 8 if the value of y is null,
// or else display the value of y
```

Functions

```dart
// Named Function
String fullName() {
  String firstName = "Barry";
  String lastName = "Allen";
  return '$firstName $lastName'; // 'Barry Allen'
}

int lengthOfText(String text) {
  return text.length; // returns length of text
}

int length = lengthOfText('200lab'); // 6

// Arrow Syntak Function
int sum(int a, int b) => a + b;

// Optional named arguments
int sumNumber(int a, int b, {int c = 0}) {
  return a + b + c;
}
print(sumNumber(1, 2, c: 3)); // 6

// Lamda Function
var list = [1, 4, 3, 2, 5];
var odds = list.where((n) => n % 2 == 1).toList();
print(odds); // [1, 3, 5]

// High Order Function
list.sort((a,b){
  return a > b ? 1 : -1;
});
print(list); // prints: [1, 2, 3, 4, 5]
```

Collections: List

```dart
// dynamic list
var countryList = ['Vietnam', 'German', 'Poland'];
print(countryList.length); // 3
print(countryList[1]); // German
countryList[2] = countryList[2].toUpperCase();
print(countryList[2]); // POLAND

countryList.insert(countryList.length, 'Thailand');
print(countryList); // [Vietnam, German, POLAND, Thailand]
print('${countryList.isEmpty}'); // false
print('${countryList.last}'); // Thailand

var fixedList = List<int>(2); // fixed list size
```

Collections: Set

```dart
var anotherSet = {1, 3, 5, 7};

print('${anotherSet.contains(69)}'); // false

anotherSet.add(9);
print(anotherSet); // {1, 3, 5, 7, 9}

anotherSet.remove(1);
print(anotherSet); // {3, 5, 7, 9}

anotherSet.addAll([11, 13]);
print(anotherSet); // {3, 5, 7, 9, 11, 13}

var declareSet = <int>{15, 17};
anotherSet = anotherSet.union(declareSet);
print(anotherSet); // {3, 5, 7, 9, 11, 13, 15, 17}
```

Dart Cheat Sheet

200Lab Education

Collections: Map

```dart
// initial Map
final student = {
  "id": 1,
  "name": "Barry Allen",
  "age": 23,
};

print(student); // {id: 1, name: Barry Allen, age: 23}

// element access by key
print(student['name']); // print: Barry Allen

// access all keys
student.keys.forEach(print); // prints: id, name, age

// update value by key
student.update("age", (value) => 18);

print(student); // {id: 1, name: Barry Allen, age: 18}

// Loop over key-value
student.forEach((key, value) {
  print('$key - $value');
}); // id - 1, name - Barry Allen, age - 18
```

Class

```dart
class Cat {
  // field
  String name = 'Milo';

  // function
  void call(){
    print('Meow meow');
  }
}
```

Getters and Setters

```dart
class Location {
  // Properties
  double _lat;
  double _lng;

  // get value
  double get long => _lng;
  double get lat => _lat;

  // set value
  set latitude (double value) => _lat = value;
  set longitude (double value) => _lng = value;

  // constructor
  Location(this._lat,this._lng);

  // named constructor
  Location.withPosition(double lat,double lang){
    _lat = lat;
    _lng = lang;
  }
}

final location = Location(69.96,96.69);
final location2 = Location.withPosition(12.23,23.21);

location2.latitude = 11.0;
location2.longitude = 12.21;

print('new latitude: ${location2.lat}'); // 11.0
print('new longitude: ${location2.long}'); // 12.21
```

Static Members / Methods

```dart
class Channel{

  static String name = 'Channel name' ;

  static void changeChannelName(String name){
    print('New name: $name');
  }

}
```

Enum

```dart
enum WeekDays { MON, TUES, WED, THURS, FRI, SAT,SUN, }

void main(){
  print(WeekDays.values); // [WeekDays.MON, WeekDays.TUES, WeekDays.WED,
  // WeekDays.THURS, WeekDays.FRI, WeekDays.SAT, WeekDays.SUN]

  var today = WeekDays.SUN;

  switch(today){
    case WeekDays.MON:
    case WeekDays.TUES:
    case WeekDays.WED:
    case WeekDays.THURS:
    case WeekDays.FRI:
      print('Working day');
      break;
    case WeekDays.SAT:
    case WeekDays.SUN:
      print('Relaxing day');
      break;
  }
}
```

Dart Cheat Sheet

200Lab Education

Inheritance

```dart
class Person {
  String name;
  int age;

  Person(String name, int age) {
    this.name = name;
    this.age = age;
  }
}

class Student extends Person {
  String id;

  Student({String id, String name, int age}) : super(name, age) {
    this.id = id;
  }
}

void main() {
  final student1 = Student(id: "123", name: "Barry", age: 22);

  print('id: ${student1.id}'); // id: 123
  print('name: ${student1.name}'); // name: Barry
  print('age: ${student1.age}'); // age: 22
}
```

Abstract / Interface

```dart
class Person {
  String name;
  int age;
  int money = 0;

  Person(String name, int age) {
    this.name = name;
    this.age = age;
  }
}

abstract class LenderBehavior {
  bool lendMoney(BorrowerBehavior borrower, int money);
  requestPayment();
}

abstract class BorrowerBehavior {
  void askForMoney(LenderBehavior lender, int money);
  void receiveMoney(LenderBehavior lender, money);
  int payMoneyBack();
}
```

```dart
class Lender extends Person implements LenderBehavior {
  BorrowerBehavior borrower;
  Lender(String name, int age) : super(name, age);

  @override
  bool lendMoney(BorrowerBehavior borrower, int money) {
    // if lender doesn't have enough money => false;
    if (this.money < money) return false;

    this.borrower = borrower;
    this.money -= money;

    // give money for borrower
    borrower.receiveMoney(this, money);

    return true;
  }

  @override
  requestPayment() {
    int returnMoney = borrower.payMoneyBack();

    if (returnMoney == null) {
      print("Borrower doesn't have enough money");
    } else {
      print("Yeah he is awesome guy");
      this.money += returnMoney;
      this.borrower = null;
    }
  }
}
```

Dart Cheat Sheet

200Lab Education

Abstract / Interface

```
class Borrower extends Person implements BorrowerBehavior {
  LenderBehavior lender;
  int debt = 0;

  Borrower(String name, int age) : super(name, age);

  @override
  void askForMoney(LenderBehavior lender, int money) {
    if (lender.lendMoney(this, money)) {
      print("what a hero");
    } else {
      print("I have to find someone else");
    }
  }

  @override
  int payMoneyBack() {
    int returnMoney;
    // if borrower has enough money for pay back to lender
    if (money >= debt) {
      money -= debt;
      returnMoney = debt;
      debt = 0;
      this.lender = null;
    } else {
      return null;
    }
    return returnMoney;
  }

  @override
  void receiveMoney(LenderBehavior lender, money) {
    this.lender = lender;
    debt = money;
    this.money += money;
  }
}
```

```
void main() {
  Lender lenderObj = Lender("Barry", 22);
  lenderObj.money = 2000;
  Borrower borrowerObj = Borrower("Steve", 18);

  // Example 1
  borrowerObj.askForMoney(lenderObj, 1000); // what a hero
  lenderObj.requestPayment(); // Yeah he is awesome guy

  // Example 2
  borrowerObj.askForMoney(lenderObj, 1000);
  borrowerObj.money = 800;
  lenderObj.requestPayment(); // Borrower doesn't have enough money
}
```