**Corso**

**Dart**

**Dedalo’s notes – Una serie di appunti da non prendere sul serio**

Serie di appunti sugli argomenti più svariati, fatti per imparare, apprendere e divertirsi, navigando per il magico mondo del…tutto!

Ogni errore o imprecisione sarà sistemata, oppure no 😊.

Attenzione, può contenere boiate, amenità o stramberie!

https://github.com/GiacomoBorsellino

Scritto da

**Giacomo Borsellino**

27-12-2024

**Cosa è?**

Dart è un linguaggio di programmazione **ad alto livello**, progettato per essere semplice e versatile. Le sue caratteristiche principali includono:

* **Tipizzazione forte e opzionale**: Dart supporta un sistema di tipi statici, ma consente anche di lavorare con tipi dinamici.
* **Orientato agli oggetti (OOP)**: Tutto in Dart è un oggetto, inclusi numeri, funzioni e null.
* **Compilazione mista**:
  + **Ahead-of-Time (AOT)**: Per creare applicazioni native ad alte prestazioni.
  + **Just-in-Time (JIT)**: Per velocizzare lo sviluppo con funzionalità come il **hot reload**.
* **Garbage collection**: Gestione automatica della memoria.
* **Multipiattaforma**: Ideale per sviluppare applicazioni per mobile, web e desktop.
* **Librerie standard e asincronicità**: Include strumenti avanzati per la gestione asincrona tramite Future e Stream.
* **Sintassi moderna**: Semplice e leggibile, simile a JavaScript, C# e Java.

Dart è un linguaggio **compilato**, ma può essere interpretato durante lo sviluppo. È progettato principalmente per lo sviluppo front-end, con particolare enfasi sul framework **Flutter**, per creare interfacce utente multipiattaforma.

Immagine che contiene Policromia, Elementi grafici, design

Descrizione generata automaticamente

Figura Link al corso: https://www.youtube.com/watch?v=7hzlhtuIgGE&list=PLP5MAKLy8lP9VP6KL4Q9ClYCuGhet59Od&index=1

Immagine che contiene illustrazione, arte

Descrizione generata automaticamente con attendibilità mediaCapitolo 1 - Installazione

1. Installa **powershell** (se non già presente)
2. Installare **Chocolatey\*** (<https://chocolatey.org/install#individual>)
3. Installa sdk dart (**choco install dart-sdk** **\*\***)

**\*** **Chocolatey** è un gestore di pacchetti, al pari di apt su linux

**\*\*** Un **SDK** (Software Development Kit) è un insieme di strumenti, librerie, documentazione e esempi forniti da un fornitore per aiutare gli sviluppatori a creare applicazioni o integrare servizi specifici all'interno di un determinato ecosistema o piattaforma.

Capitolo 2 - Introduzione

Immagine che contiene porta, illustrazione, design

Descrizione generata automaticamente

**Per creare un nuovo progetto**

1. Apri Powershell

**dart create -h** (vedi tutti i template disponibili per creare un primo mprogetto in dart)

**dart create -t console-full corso\_dart** (creare un primo progetto base)

**Nota**: se non viene trovato il percorso per dart, controlla sotto C in tools

1. Apri con vsCode il nuovo progetto, struttura

File/ Cartelle:

* pubspec.yaml, pubspec.lock, .dart\_tool servono a importare/gestire i pacchetti

core di dart

Per installare i pacchetti basta andare su [**https://pub.dev/**](https://pub.dev/)

* analysis\_lint: gestione del linting
* test: cartella di testing
* bin: core del codice
* lib: cartella con funziona di riuso

1. Avvia da console con **dart run**

**Estensioni** utili per **VSCode**:

* Dart
* Dart Data Class Generator
* Pubspec Assist

Capitolo 3 - Variabili

Immagine che contiene vestiti, disegno, persona, calzature

Descrizione generata automaticamente

Definizione e gestione delle variabili

Dart è un linguaggio fortemente tipizzato

È possibile dichiarare e poi definire una variabile, specificando il tipo all’inizio della dichiarazione.

Es.

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

È possibile usare la keyword **dynamic**, che permette di definire ciò che si vuole

Es.

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

È possibile usare anche **var**, questo, se non definito fin da subito (es. var test = 5), funzionerà come dynamic,



differentemente il suo tipo verrà bloccato fin da subito!

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

È possibile usare anche **final**, che indica che un valore è nel suo stadio finale, non posso più cambiarne tipo né valore.

Valori immutabili che sono determinati a tempo di **runtime**





È possibile usare anche **const**, che però deve essere dichiarato e definiti contestualmente

Valori immutabili che sono determinati a tempo di **compilazione**



Al pari con le **liste**, non è possibile modificare liste con const, si con final, anche se, anche in questo caso non posso riassegnare!

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Gerarchia della **flessibilità** delle keyword:

**dynamic > var > final > const**

**NOTA**: E’ possibile verificare se c’è un errore in fase di runtime usando l’attributo runtimeType, che restituisce il tipo effettivo della variabile durante l'esecuzione.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**Tipologie di errore:**

**Errore di compilazione**

* **Quando si verifica:** Durante la fase di compilazione, prima che il programma venga eseguito.
* **Cosa lo causa:** Problemi nel codice che impediscono al compilatore di tradurre il programma in codice eseguibile.

**Errore di runtime**

* **Quando si verifica:** Durante l'esecuzione del programma.
* **Cosa lo causa:** Problemi logici o situazioni impreviste che emergono mentre il programma è in esecuzione.

Capitolo 4 - Sound NULL Safety

Immagine che contiene disegno, cartone animato, clipart, illustrazione

Descrizione generata automaticamente

Null è una assenza di valore, da dart v2.12 è stata posto per definizione che **nessuna variabile può essere null (sound NULL Safety)**, a meno che non lo si decida esplicitamente.

Per ciò, se si dichiarazze, ma non si definisse una variabile che poi verrà usata, ci sarà un errore,



Per cui bisognerà specificare la variabile come **nullable** (**?**)



Capitolo 5 – Tipi di dati integrati

Immagine che contiene calzature, testo, cartone animato, schermata

Descrizione generata automaticamente

Quali sono **integrati** in Dart?

**1. Tipi di base**

Questi sono i tipi fondamentali forniti dal linguaggio Dart:

* **int**: Numeri interi.
  + Esempio:

int a = 10;

* **double**: Numeri a virgola mobile (decimali).
  + Esempio:

double b = 3.14;

* **num**: Superclasse di int e double. Può contenere sia numeri interi che decimali.
  + Esempio:

num c = 2.5;

* **String**: Sequenze di caratteri (testo).
  + Esempio:

String s = "Hello, Dart!";

* **bool**: Valori booleani (true o false).
  + Esempio:

bool isReady = true;

**2. Collezioni**

Dart fornisce strutture dati per contenere più valori:

* **List**: Una lista ordinata di elementi (array).
  + Esempio:

List<int> numbers = [1, 2, 3];

List<dynamic> mixed = [1, "Hello", true];

* **Set**: Una collezione di elementi unici (non ordinati).
  + Esempio:

Set<int> uniqueNumbers = {1, 2, 3};

* **Map**: Una collezione di coppie chiave-valore.
  + Esempio:

Map<String, int> ages = {'Alice': 30, 'Bob': 25};

**3. Tipi speciali**

Questi tipi hanno scopi particolari:

* **dynamic**: Una variabile che può contenere valori di qualsiasi tipo e cambiare il tipo durante l'esecuzione.
  + Esempio:

dynamic x = 5;

x = "Ciao";

* **Object**: La superclasse di tutti i tipi in Dart (compresi null e dynamic).
  + Esempio:

Object obj = "Test";

* **Object?**: Una versione nullable di Object (può essere null).
* **void**: Indica che una funzione non restituisce nulla.
  + Esempio:

void greet() {

print("Hello");

}

* **Null**: Un tipo che ha un solo valore, null. Usato per indicare l'assenza di un valore.

**4. Tipi di supporto**

Questi tipi sono inclusi nella libreria di base di Dart e forniscono funzionalità specifiche:

* **Future**: Rappresenta un valore o un errore che potrebbe essere disponibile in futuro (operazioni asincrone).
  + Esempio:

Future<int> fetchNumber() async { return 42; }

 **Stream**: Una sequenza di valori asincroni.

* Esempio:

Stream<int> numberStream = Stream.fromIterable([1, 2, 3]);

 **Iterable**: Una sequenza generica di elementi che può essere iterata.

* Esempio:

Iterable<int> numbers = [1, 2, 3];

**5. Tipi di costanti**

* **const**: Valori immutabili che sono determinati a tempo di compilazione.
  + Esempio:

const pi = 3.14;

* **final**: Valori immutabili che sono determinati a runtime.
  + Esempio:

final currentTime = DateTime.now();

**6. Altri tipi**

* **Runes**: Rappresenta caratteri Unicode. Usato per manipolare stringhe a livello di carattere Unicode.
  + Esempio:

Runes emoji = Runes('\u{1F600}');

* **Symbol**: Rappresenta un simbolo nel codice Dart, utile per metaprogrammazione.
  + Esempio:

Symbol symbol = #mySymbol;

**7. Null safety**

Con Dart 2.12+ è stata introdotta la **null safety**, che distingue tra tipi nullable e non-nullable:

* **Non-nullable:** Una variabile deve avere sempre un valore.
  + Esempio:

int a = 10;

* **Nullable:** Una variabile può essere null.
  + Esempio:

int? a = null;

Capitolo 6 – Operazioni con i numeri

Immagine che contiene cartone animato, clipart, disegno, orologio

Descrizione generata automaticamente

Operazioni con i numeri

**Num** è una classe numero che è un insieme più grande di **int** e **double**.

E’ possibile effettuare operazioni tra loro.

Oltre a ciò, è possibile fare il **casting** del valore di un tipo ad un altro

Es.

num numero = 6.6;

int intero = 0;

double decimale = 1.1;

print("ciao " + numero.toString()); // ciao 6.6

Consersione da numero a stringa (.toString()

String stringa = '5';

print(double.parse(stringa)); // 5.0

Conversione da string a double (double.parse(stringa))

Con i metodi **ceil**, **round** e **floor** è possibile, rispettivamente, arrotondare per eccesso, arrotondare e arrotondare per difetto

  double qwerty = 6.6;

  print(qwerty.ceil()); // 7

  print(qwerty.round()); // 7

  print(qwerty.floor()); // 6

**Operatori aritmetici**:

* **+**
* **-**
* **/**
* **\***
* **%**
* **~/** (il resto dell’intero, es. 10 / 3 = 3.333, con questo operatore avremo 3)

**Operatori di assegnamento**:

* **+=**
* **-=**
* **\*=**
* **/=**

**Es.**

  int intero = 5;

  intero += 5;

  print(intero); // 10

**Operatori di incremento**: **\***

* **++intero**
* **intero++**

  int intero = 5;

  // intero++;

  ++intero;

  print(intero); // 6

**Operatori di decremento**: **\***

* **--intero**
* **Intero--**

  int intero = 5;

  // intero--;

  --intero;

  print(intero); // 4

**\*Nota:** Se è un **pre-incremento**, l’incremento avviene nella riga stessa, come prima operazione, mentre se messo in **post**, l’incremento avverrà nella riga successiva.

Es.

  int intero = 8;

  int qw = ++intero;

  print(qw); // 9

Risultato 9, intero è stato incrementato prima che venisse incamerato dalla variabile qw

Es 2.

  int intero = 8;

  int qw = intero++;

  print(qw); // 8

Risultato 8, intero **non** è stato incrementato prima che venisse incamerato dalla variabile qw, ma dopo.

Capitolo 7 – Operazioni con le stringhe

Immagine che contiene testo, cartone animato, schermata, linea

Descrizione generata automaticamente

Le stringhe sono definibile tramite **singoli apici** (‘’) o **doppi apici** (“”), eventuali singoli apici vengono inseriti tramite lo **slash** inveso (\) con l’escape dei caratteri

**Le stringhe sono**…

… composte da caratteri, *indicizzati*, naturalmente, a partire da 0

… *concatenabili* (string1 + string 2 = strin1string2)

…*interpolabili* con altri tipi, ovvero è possibile inserire le variabili in doppi apici, iterpolandole

Es.

String string4 = 'Ciao, sono in classe ';

int classe = 5;

print("${string4}${classe}"); // Ciao, sono in classe 5

…mandabili a capo con *\n*

Es.

String string5 = 'Ciao,\n mi trovo bene';

… o con le *multiline*, usando i 3 apici singoli,

Es.

String string6 = '''ciao

  come

  va

  ?''';

…o visualizzare in modo *raw* la stringa, usando come prefisso *r*

Es.

String string7 = r'Ciao,\n mi trovo bene';

**Metodi** **base**:

* indexof
* lastindexof
* touppercase
* tolowercase
* split
* substring

Es.

  String testing = "Sono una stringa";

  dynamic formatString0 = testing.contains('w');

  dynamic formatString1 = testing.indexOf('o');

  dynamic formatString2 = testing.lastIndexOf('o');

  dynamic formatString3 = testing.toUpperCase();

  dynamic formatString4 = testing.toLowerCase();

  dynamic formatString5 = testing.split(' ');

  dynamic formatString6 = testing.substring(1);

  print(formatString0); // false

  print(formatString1); // 1

  print(formatString2); // 3

  print(formatString3); // SONO UNA STRINGA

  print(formatString4); // sono una stringa

  print(formatString5); // [Sono, una, stringa]

  print(formatString6); // ono una stringa

Capitolo 8 – Booleani



È **true** o **false**, cos’altro vorresti sapere?

Capitolo 9 – List

Immagine che contiene forniture per ufficio, schermata, testo, grafica

Descrizione generata automaticamente

Struttura dati di dati, equivalenti agli array (es. in js)

Es.

  List<int> testListNumeri = [1, 2, 3];

Es 2.

List<Object> testListOggetti = [1, 2, 3, '', true, null];

**Nota**: in questo caso il tipo Oggetto fa funzionare la lista poiché, i dati inseriti, appartengono alle rispettive classi ttipo di appartenenza (es. true alla classe Boolean, ‘’ alla classe String, ecc…)

Differente è per **null**, che non può essere inserito, poiché assenza di valore e non apprtenente a Classe Object.

Casistiche peculiari:

  List<int> a = [1, 2, 3];

  List<int> b = [1, 2, 3, null];

  List<int?> c = [1, 2, 3, null];

  List<int>? d;

a) Nel caso **a** la lista è definite e dichiarata **corretamente**

b) Nel caso **b** la lista **non** può contenere un **null**, poiché esso non appartiene al tipo intero

c) Nel caso **c**, ponendo come nullable gli **interi** inseriti in lista, sarà possibile inserire un null

d) Nel caso **d**, non essendoci definizione, l’intera **lista** darà null, indi per cui bisognerà dichiarare la lista come nullable

Le liste sono **indicizzabili**, hanno una **lunghezza** e possono esser **concatenate** con altre liste, al pari di come si è già visto con le stringhe

  List<int> lista = [2, 4, 6, 8];

  List<int> lista2 = [1, 3, 5, 7];

  lista.add(100);

  lista.addAll(lista2);

  print(lista);

  print(lista[0]);

  print(lista.length);

  print(lista[lista.length - 1]);

  List<int> lista = [2, 4, 6, 8];

  List<int> lista2 = [1, 3, 5, 7, ...lista];

print(lista2); // [1, 3, 5, 7, 2, 4, 6, 8]

Nel caso sopra, **add** e **addAll**, permettono, rispettivamente, di aggiungere singoli **item o iterabili**, come altre liste, al pari dello **spread** **operator** [lista, …lista2] o rimuoverli (.remove(), .removeAt(), ecc…).

E’ possibile concatenare anche utilizzando i **..add(8)** a seguire.

Metodi utili:

* forEach
* indexOf
* clear()
* contains()
* map
* collection if **\***
* collection for **\*\***

…ecc

**\*** **collection if**: permette di inserire un if in una lista

  int qwerty = 5;

  List<String> lista = [

    'Agamennone',

    'Penelope',

    'Anassimandro',

    if (qwerty > 5) 'Anna',

    'Ettore'

  ];

  print(lista); // [Agamennone, Penelope, Anassimandro, Ettore]

**\*\*** **collection for**: permette di inserire un for in una lista

  List<int> prova = [1, 2, 3];

  List<String> lista2 = ['A', 'B', 'C', for (var i in prova) '${i}', 'D'];

  print(lista2); // [A, B, C, 1, 2, 3, D]

Capitolo 10 – Set



**Cosa sono?**

Struttura dati in cui una **lista** ha dei **valori** sono **unici** e **non ordinati**.

Es.

  var set = {1, 1, 2, 3, 4, 5, 6, 7, 8, 8};

// {1, 2, 3, 4, 5, 6, 7, 8}

Il set **non garantisce** l’ordine, quindi non è possibile usare il suo indice per selezionare un suo item.

Esistono vari modi di **dichiarare** un **Set**:

  var set1 = Set();

  Set<String> set2 = {"ciao", "hello"};

  var set3 = {"ciao", "hello"};

  var set4 = <int>{1, 2};

  print(set1); // {}

  print(set2); // {ciao, hello}

  print(set3); // {ciao, hello}

  print(set4); // {1}

* tramite un construttore Set()
* Dichiarandolo come tale
* ~~Usando le graffe~~
* Dichiando il tipo e poi le graffe

Il **terzo** modo è **errato**, perché, di default, il linguaggio usa dichiarazioni di questo come sintatticamente dei **Map**.

**Metodi utili:**

* add()
* addAll()
* …{} (**sprepad operator**)
* remove()
* removeAll()
* removeWhere()
* collection if
* collection for
* intersect **\***
* union **\*\***
* difference **\*\*\***

var set = <int>{1, 2, 3, 4, 5, 6};

var set2 = <int>{2, 88, 101};

**\*** **intersect**: Fa intersezione di valori uguali

  print(set.intersection(set2)); // {2}

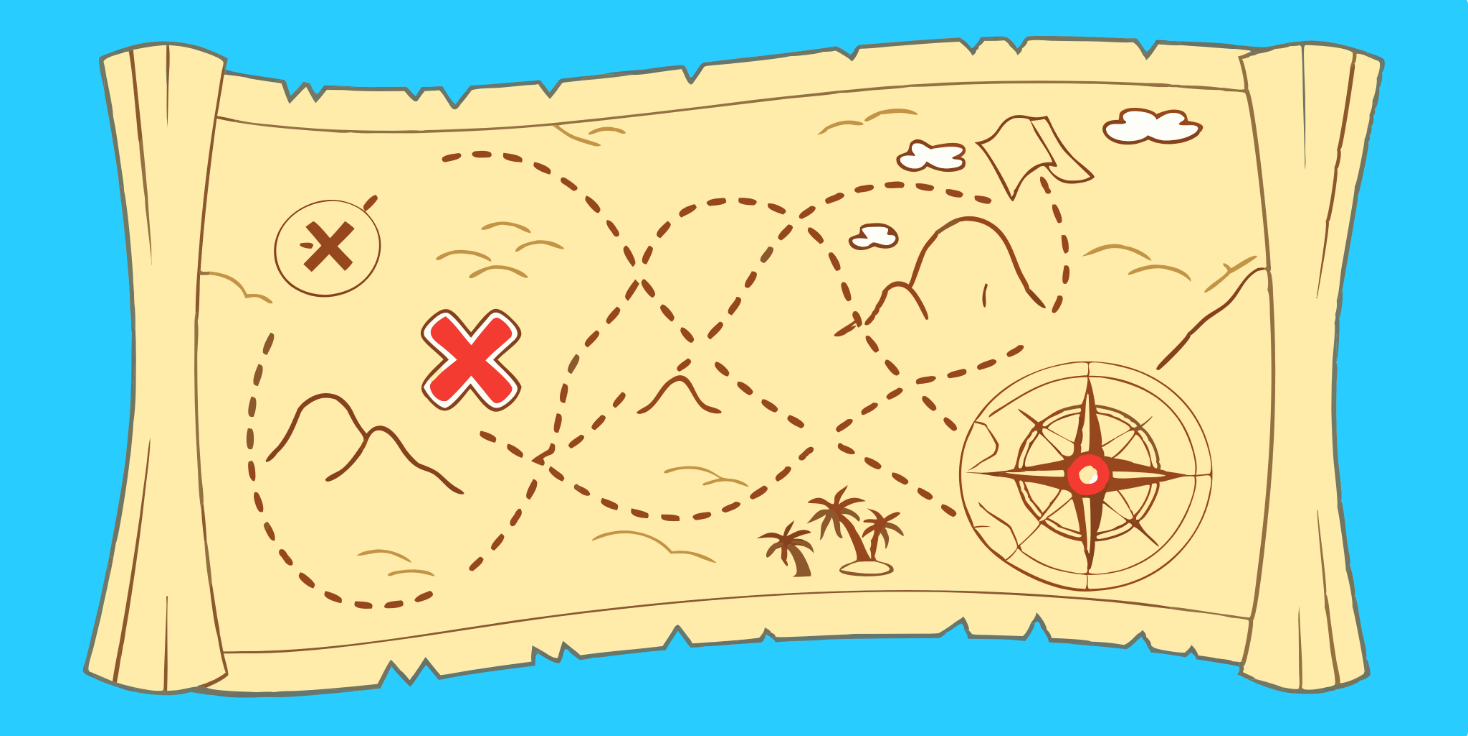
**\*\*** **union**: fa l’unione dei due set

  print(set.union(set2)); // {1, 2, 3, 4, 5, 6, 88, 101}

**\*\*\*** **difference**: stampa i non differenti

  print(set.difference(set2)); // {1, 3, 4, 5, 6}

Capitolo 11 – Map



Struttura dati formata da una lista di item **chiave-valore** (key-value).

Es.

  var list = {

    "name": "Jack",

    "surname": "O'Lantern",

    "age": 28,

  };

Al pari degli altri, è possibile definire il tipo per la chiave e il tipo per il valore:

  var mapVuota = {};

  print(mapVuota.runtimeType); // \_Map<dynamic, dynamic>

  Map<String, int> test = {"numOggetti": 88};

  print(test); // {numOggetti: 88}

*// Esempio tipizzato*

**Aggiunta/modifica valore:**

  Map<String, int> test = {"numOggetti": 88};

  test["qualitaOggetti"] = 8;

  print(test); // {numOggetti: 88, qualitaOggetti: 8}

accedendo dinamicamente ad una proprietà.

**Nota**: al pari degli altri, collectionIf, add, spread operator, remove, foerEach, ecc…sono metodi ammessi e utilizzabili anche con i Map

Es.

  Map<String, int> test = {"numOggetti": 88, if (2 < 5) "tipologiaOggetti": 3};

  test["qualitaOggetti"] = 8;

  print(test); // {numOggetti: 88, tipologiaOggetti: 3, qualitaOggetti: 8}

  var prova = {"a": 1, "b": 2, "c": 3, "d": 4, "e": 5, "f": 6, "g": 7, "h": 8};

  prova.remove("e");

  prova.forEach((key, value) => print(key));

// a

// b

// c

// d

// f

// g

// h

Capitolo 12 – Funzioni

