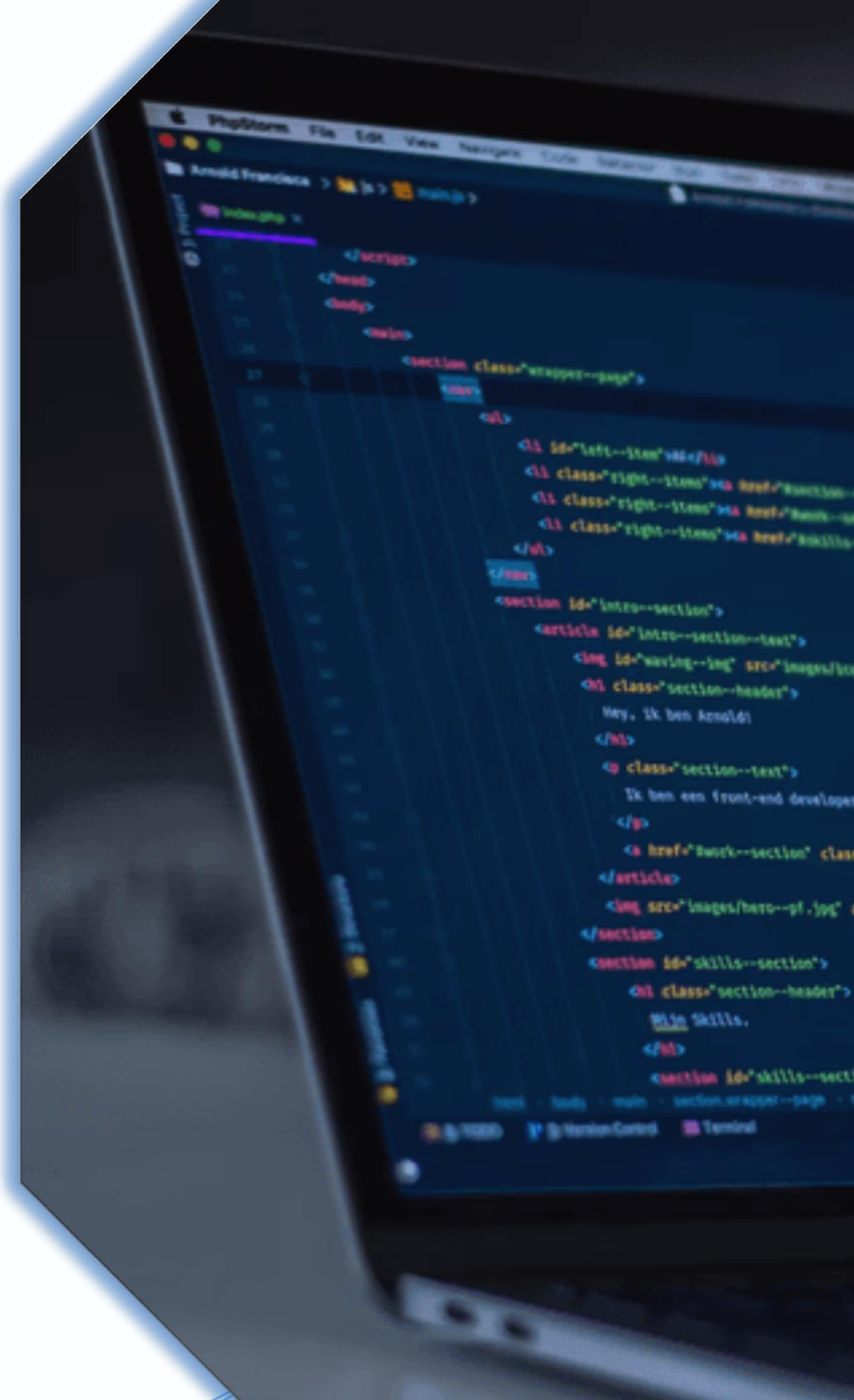
Dart

2025

م/ إدريس الهاملي

١



Dart

Dart: هي لغة برمجة أنشأتها شركة Google، تُستخدم لبناء تطبيقات سريعة وحديثة، خاصة لتطبيقات الهواتف باستخدام Flutter، لكنها تصلح أيضاً لتطبيقات الويب وسطر الأوامر.

متطلبات التشغيل

حزمة Dart SDK عبر الرابط التالي :
Install manually | Flutter

محرر الأكواد الشهير

Abdoullah Ashgar EL-kharef

إعداد بيئة الدارت

1

حمِّل أولًا Dart SDK. ولأننا سنعمل على Flutter لاحقًا،

يكفي تثبيت Flutter SDK لأنه يحتوي على Dart مدمجًا.

ثم نستخرجه الى أي مكان مناسب .. ونحدد مساره

2

انتقل الآن إلى مجلّد الحزمة، انسخ مسار flutter\bin ،

ثم افتح متغيّرات البيئة، حرّر Path، وأضِف المسار واحفظ التغييرات .

3

تثبيت إضافات Dart في VS Code

وهي ضافتي:
Dart
و
dart-import

أنواع المتغيّرات في Dart

الأنواع الأساسية

1. int: لأعداد الصحيحة
2. double: للكسرية،
3. bool: للقيم المنطقية،
4. String: للنصوص.

ملاحظة:-

String أوّل حرف فيها كبير لأنها اسم صنف في اللغة

Var,dynamic

var ليست نوعاً؛ هي كلمة للاستدلال: إن كتبت var x = 5 سيصبح نوع x هو int ولا يتغيّر لاحقاً.

إن كتبت var x; بدون تهيئة فسيكون نوع المتغيّر dynamic تلقائياً ويمكنه استقبال أي نوع.

Dynamic نوعٌ يسمح بتغيير النوع وفحصه يكون في وقت التشغيل؛ من لكن أقل أماناً، فاستعمله عند الحاجة فقط.

❖ حدِّد النوع صراحةً متى استطعت (int, String...) لقراءة أوضح وأخطاء أقل. استخدم var فقط مع تهيئة واضحة، وتجنّب dynamic إلا عند الضرورة. وتذكّر var: ليست نوعاً، و Stringتُكتب بحرف أوّل كبير دائماً.

return AND print

print

- تظهر نصاً في الشاشة فقط.
- لا ترجع قيمة، تأثيرها بصري في وحدة الإخراج فقط.
- استخدم print للتتبّع المؤقّت ولا تبنِ منطق برنامجك عليها .

Return

- تُعيد قيمة من الدالة إلى منادِيها وتنهي تنفيذ الدالة.
- تُنهي الدالة فوراً وترسل قيمة يمكن تخزينها أو استخدامها.
- بعد return لا يُنفَّذ أي سطر داخل الدالة.

ملاحظات

الدوال ذات نوع غير void يجب أن تحتوي return بقيمة من نفس النوع.

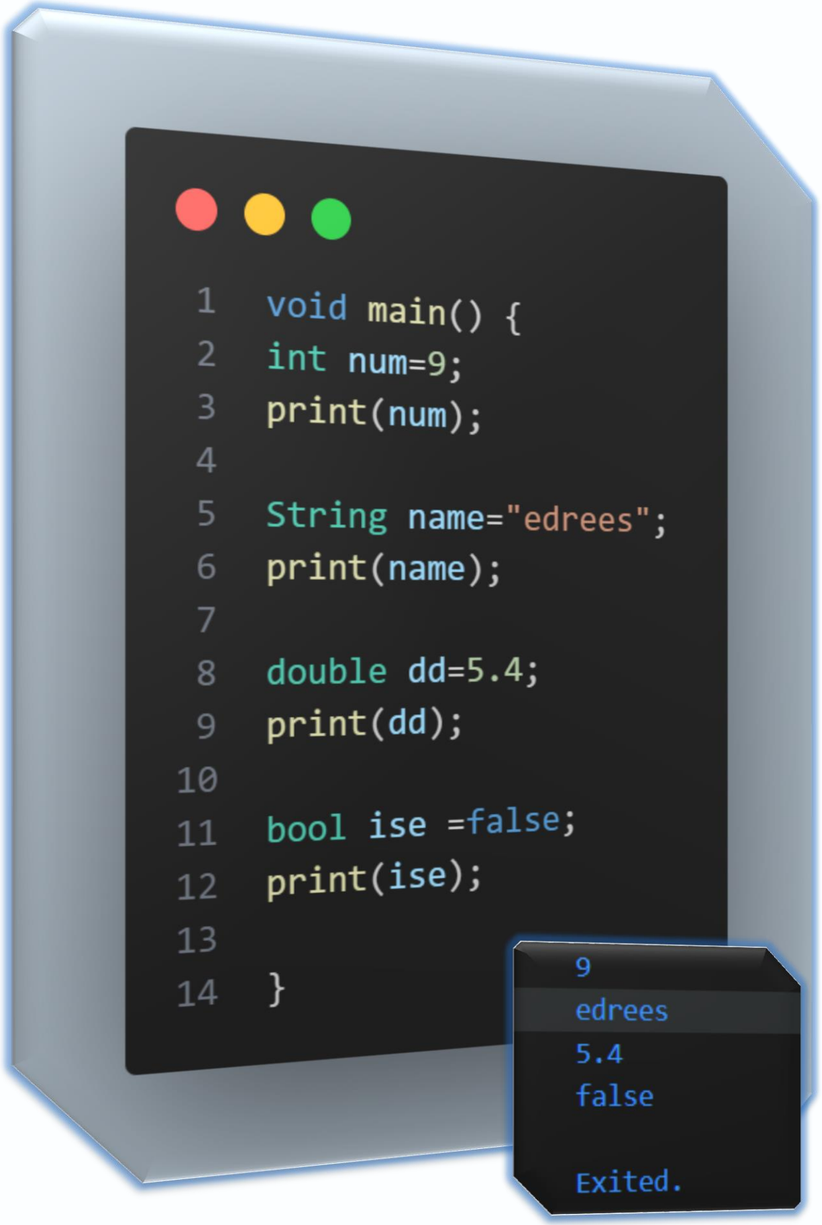
ابنِ النتيجة بـ return ثم استعملها في المكان المناسب. استخدم print لمراقبة سير التنفيذ أثناء التطوير فقط، واحذف الطباعة الزائدة قبل تسليم مشروعك .

مثال عملي

ملاحظة هاااااااامة:

تنفيذ الأوامر في لغة البرمجة من الداخل الى

الخارج و من اليمين الى اليسار



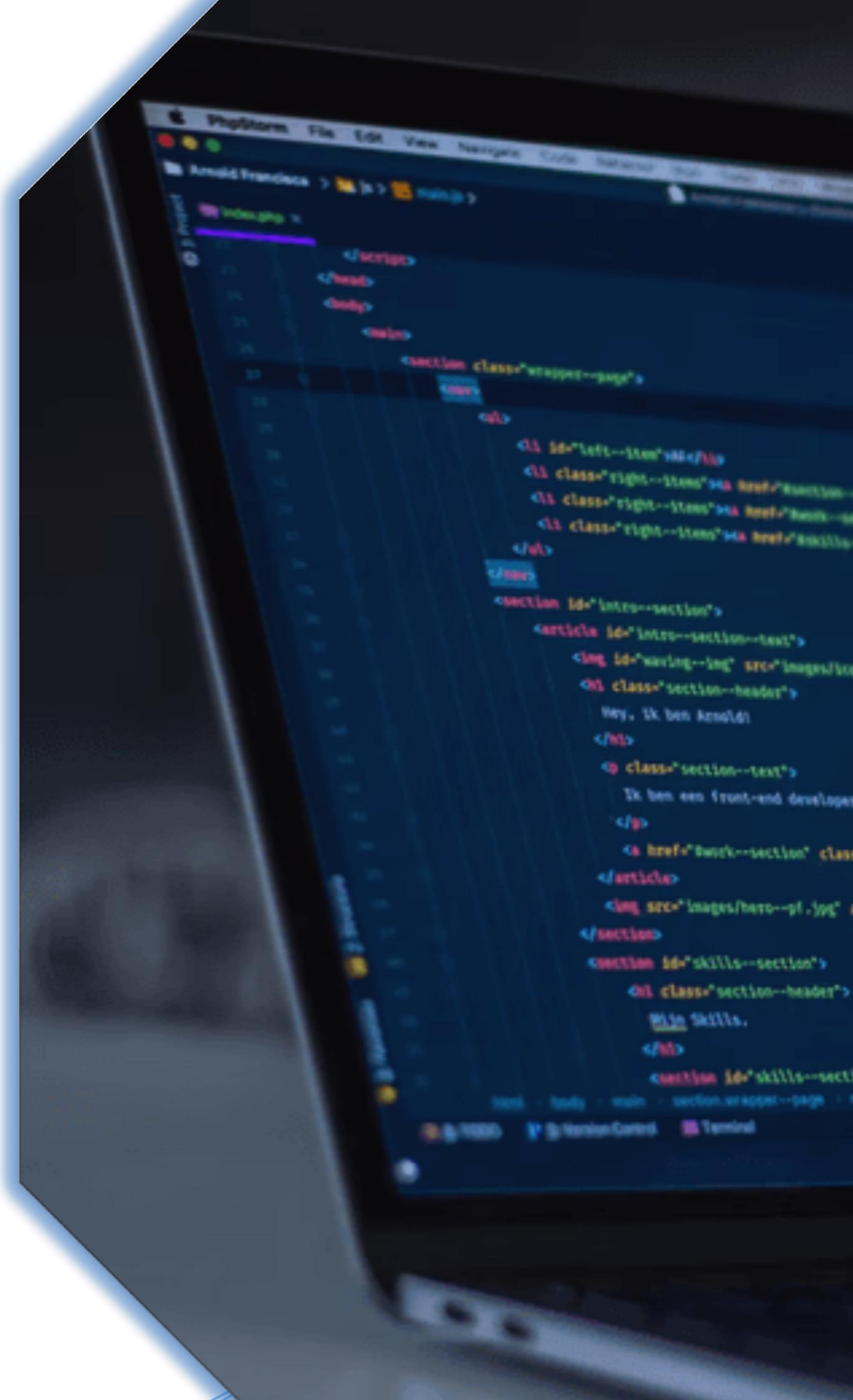
انتهت المحاضرة الأولى

Dart

2025

م/ إدريس الهاملي

2



ملاحظات وتوصيات

01 الإلتزام بالمعايير

- اسما الكلاس من نوع مفرد
- يكون اسم الكلاس بالأحرف الكبيرة او اقل شيء اول حرف
- الكود المتماسك افضل بكثير من الكود المترابط

02 طريقة إنشاء الكلاس واستدعائه

```
1
2  class Student {
3  //هذه هو الكلاس
4  }
5
6  Student S=Student();
7  //هذه طريقة الاستدعاء☝
```

03 Null safety

قيمة المتغير الافتراضيه في اغلب الغات تكون null

عداء الدارت لا يمكن إلا اذا سمح المبرمج بذلك

04 مواضيع أخرى

- حالات null safety
- Constructors
- If
- for

Null safety

هي فكرة تمنع وقوعك في أخطاء الفراغ. النوع افتراضاً غير قابل للفراغ ولا يصبح null إلا إذا سمحت أنت بإضافة ?.

١- القاعدة: المتغيّر من نوع عادي مثل int أو String لا يقبل null ، ويجب تهيئته قبل الاستعمال.

٢- جعل النوع قابلاً للفراغ: أضف ? مثل String? name. الآن قد يساوي null.

٣- التعامل الآمن:
- ?. نداء/وصول آمن: يتوقف ويعيد null بدلاً من الخطأ.
- ?? قيمة بديلة عند الفراغ.
- ??= إسناد بديل إذا كان المتغيّر فارغاً.

٤- التأكيد القسري !: تقول للمترجم "متأكد أنه ليس null". استخدمه بحذر لأنه يَرمي خطأ إن كان فعلاً null.

٥- أدوات مساعدة:
- late لتأجيل التهيئة مع ضمان التعيين قبل الاستخدام.
- في المجموعات يمكن استعمال الانتشار الواعي ...?listNullable لتجاهل الفراغ.

نصيحة عملية:

اجعل الأنواع غير قابلة للفراغ دائماً، ولا تضف ? إلا لسبب واضح. ابدأ بـ ?. و ?? و??=، وتجنّب ! إلا عند يقين تام.

تذكير: في Dart لا تكون قيمة المتغيّر null افتراضاً إلا إذا سمحت بذلك بإضافة ?.

تطبيق عملي عن ال null safety

code

```dart
void main() {
  String? name;
  name="alhamli";
  int? age, phone;

  String n = name ?? "edrees";
  print(n);
  print(age ?? 33);
  phone ??= 777777;
  print(phone);
  print(n?.toUpperCase());
}

```

Outputs

```
alhamli
33
777777
ALHAMLI

Exited.
```

Null Safety

الفرق بين ! و؟

?تعني "قد يكون فارغاً" أو "تصرّف بحذر."

!تعني "أنا متأكّد أنه ليس فارغاً الآن" وتتحمّل العاقبة إن كان العكس .

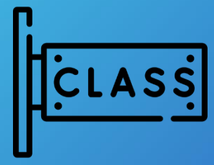
الفرق بين String و String؟

- من String إلى String? **مسموح** (آمن)؛ القيمة دائماً غير فارغة.
- من String? إلى String **غير مسموح** إلا بعد معالجة: بديل ?? أو فحص ,(if (x != null أو تأكيد ! بحذر.
- في الدوال: وسيطا String يرفض null ، بينما وسيطا String? يقبله.

❖ اجعل الأنواع غير قابلة للفراغ افتراضاً. إن شككت، استخدم ? مع .? و ??و .=??لا تستخدم ! إلا بعد خطوة تحقق واضحة (مثلاً شرط .((if (x != null بهذه الطريقة تتجنّب الأخطاء وتبقي الشفرة آمنة وسهلة الفهم .

return AND print

class



- قالب يصف البيانات والسلوك.
  نعرّفه مرة واحدة لنحدّد الخصائص والدوال والبائي.
- يجمع خصائص (بيانات) ودوال (سلوك).
- له بائي لتهيئة القيم عند الإنشاء.
- لا ينشئ شيئاً بحدّ ذاته حتى نُنشئ كائناً منه.
- يوفّر تنظيم الشفرة وإعادة الاستخدام.

object

- نسخة حيّة من الصنف لها قيمها الخاصة. نُنشئه باستدعاء بائي الصنف.
- لكل كائن حالة مستقلّة.
- نستخدم النقطة للوصول للخصائص والدوال.
- يمكن إنشاء عدّة كائنات من نفس الصنف.
- تغيير كائن لا يؤثّر في الآخرين.

code

```dart
class User {
  String name;
  User(this.name);
}
void main() {
  var u = User("Lena");
  print(u.name);
}
```

Outputs

```
Lena

Exited.
```

code

```dart
class User {
 late String name ;
  int? age;
  void greet() => print("Hi $name age:$age");
}
void main() {
  User u = User();
  u.name = "Edrees";
  u.age = 22;
  u.greet();
}

```

Outputs

```
Hi Edrees age:22

Exited.
```

constructor

الـ **constructor** هو أول شيء يلمسه البرنامج عند إنشاء كائن من الصنف class بدونه لا يمكن تهيئة القيم الأولية ولا ضمان حالة صحيحة للكائن.

- Constructor: دالة خاصة اسمها مثل اسم الصنف، لا تملك نوع إرجاع، تُستدعى تلقائياً عند كتابة ClassName(...). وظيفتها : حجز مساحة للكائن في الذاكرة وتهيئة خصائصه.
- يمكنك في Dart إنشاء أكثر من constructor عبر المُنشئات المسماة (named constructors) و/أو factory constructors.
- لا يوجد Overloading في Dart لا يمكنك إنشاء مُنشئين بالاسم نفسه مع معاملات مختلفة. عِوَضاً عن ذلك استخدم named constructors بأسماء مختلفة مثل User.guest(), : User.fromMap(...) أو parameters اختيارية بقيم افتراضية.

- أنواع شائعة من الـ constructors:

١.) Generative العادي : ينشئ كائناً جديداً ويهيئ الخصائص.

٢. Named constructor بدائل مسماة لطرق إنشاء متعددة.

٣. Redirecting constructor يستدعي مُنشئاً آخر داخل نفس الصنف.

٤. Factory constructor قد يعيد كائناً موجوداً بدلاً من إنشاء جديد مفيد للتخزين المؤقت/Singleton

```
class User {
  String name;
  int age;

  // Default constructor: initializes both fields
  User(this.name, this.age);

  // Named constructor: a preset "guest" user
  User.guest()
      : name = "Guest",
        age = 0;

  User.withAge(String name, int age)
      : name = name,
        age = age >= 0 ? age : 0;
}

void main() {
  var u1 = User("Alice", 25);        // default constructor
  var u2 = User.guest();             // named constructor
  var u3 = User.withAge("Bob", -3);  // named constructor with check

  print("${u1.name} is ${u1.age}");
  print("${u2.name} is ${u2.age}");
  print("${u3.name} is ${u3.age}");

}

```

WARDIERE

THANK YOU

reallygreatsite.com