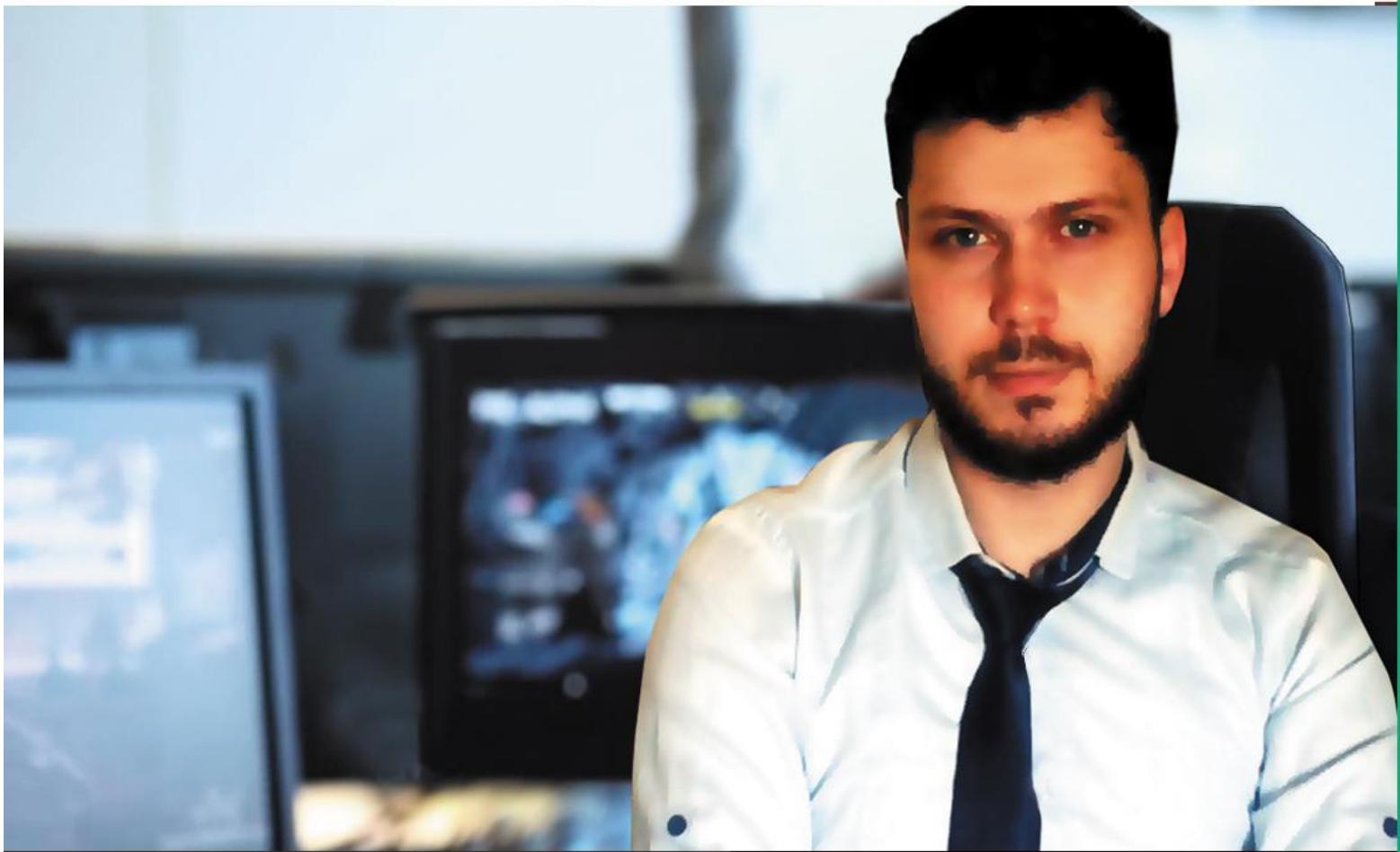


برمجة تطبيقات الجوال

Flutter



المهندس فيصل الاسود

باحث اكاديمي في جامعة SRM الدولية للعلوم والتكنولوجيا

كتاب عربي يتناول برمجة تطبيقات الجوال باستخدام لغة Dart

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

الأهداء

الى ابى وامى الذين كانوا معى في مسیرتى العلمية والى زوجتى التي تحملت مني كل ذلك الانشغال ، الى اخوتي رفاق دربى اهدي لكم هذا العمل المتواضع.

انا لا املك الحقيقة أنا مثلكم أحاول اللحاق بها الى كل من يشارکني ذلك اهدي اليه هذا الكتاب.



مقدمة الكتاب

أصبحت تطبيقات الجوال جزءاً من حياتنا اليومية مما أتاح لنا كبشر تأهيل كواذر لتلبية هذا الجزء المهم من الحياة والتفرد بأعمال برمجية خاصة بعلم برمجة التطبيقات ،سيمكناك هذا الكتاب ان شاء الله من دخول عالم برمجة التطبيقات من **اوسع ابوابه** ، كما سيعطيك القوة لكي تتأهل للعمل في اقوى الشركات او حتى الربح من التطبيقات بشكل مستقل.

لغة دارت هي لغة برمجية تم إنشاؤها من قبل شركة جوجل وتنستخدم في تطبيقات الويب أو سطح المكتب وتطبيقات الجوال.

تم ابتكار هذه اللغة من قبل Lars Bak و Kasper Lund وتم إطلاق أول اصدار منها في عام ٢٠١١ . من المهم أن نعلم أن Dart هي لغة Cross-Platform أي أنها تعمل على مختلف المنصات، كما أنها Native Language أي تتعامل مع العتاد مباشرة بدون مفسرات وسيطة وهذا يعطيها سرعة عالية جدا.

أما فلتر Flutter فهي منصة تمكنا من بناء تطبيقات جوال بواجهات رسومية معتمدين على لغة Dart

الذي يميز Flutter أنها تمكنا من بناء تطبيقات لأنظمة مختلفة منها الاندرويد أو ال IOS الخاص بأجهزة Apple والمذهل أكثر أنه يمكن أيضا استخدامها كاللغة الأولى لبرمجة تطبيقات نظام جوجل الجديد "فوشيا" Fuchsia والذي قد يزاح الاندرويد عن مكانه. يجب أن نعلم أيضاً أن Flutter تعتمد في تصمييمها Material Design التي تم بناؤها من قبل جوجل والتي تساعده في تصميم صفحات الويب.

تأليف، فيصل الأسود | مهندس برمجيات

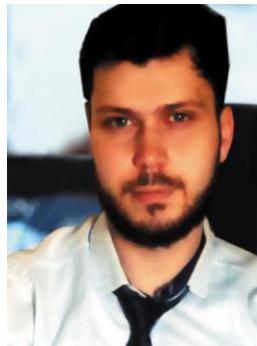
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

دول المؤلف



السلام عليكم انا فيصل الأسود مهندس برمجيات وباحث في جامعة SRM الدولية للعلوم والتكنولوجيا في هذا المجال ادرس الدكتوراه مروراً بالماستر واسخر جزءاً كبيراً من حياتي لأكون جسراً لإفادة الشغوفين مثلني يمكنني الحصول على شروحات كامل الكتب التي اقدمها عن طريق قناتي على اليوتيوب ، وأحياناً سعيائے وراء العلم.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

المحتويات

الوحدة الاولى: المدخل الى لغة دارت	7
الوحدة الثانية: اساسيات لغة دارت	17
الوحدة الثالثة: البرمجة المتقدمة في دارت	38
الوحدة الرابعة: برمجة التطبيقات باستخدام فلتر	45
التعرف على بيئه فلتر	46
أنظمة التصميم Layouts	52
الـ Widget المقدمة مع Flutter	55
القوالب الجاهزة Scaffold	70
التصفح والتوجيه Navigation	76
التعامل مع Json	84
استدامة البيانات	88
التعامل مع الصور	106
الاشعارات الداخلية	109
الحركة Animation	112
رسائل التنبيه السريعة Toast	121
قواعد البيانات Firebase	123
الوحدة الخامسة: النشر والربح من التطبيقات	136
إضافة إعلانات Admob	137
نشر تطبيقات اندرويد	142
المراجع	148

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

1

المدخل الى لغة دارت

Introduction to Dart Language



SUCCESS IS A LOUSY TEACHER . IT SEDUCES SMART PEOPLE
INTO THINKING THEY CAN'T LOSE

Bill Gates

المدخل الى لغة دارت

1- مقدمة:

لغة دارت هي لغة برمجية تم إنشاؤها من قبل شركة جوجل وتنستخدم في تطبيقات الويب أو سطح المكتب وتطبيقات الجوال.

تم ابتكار هذه اللغة من قبل Lars Bak و Kasper Lund و تم إطلاق أول اصدار منها في عام ٢٠١١. من المهم أن نعلم أن Dart هي لغة Cross-Platform أي أنها تعمل على مختلف المنصات، كما أنها Native Language أي تتعامل مع العتاد مباشرة بدون مفسرات وسيطة وهذا يعطيها سرعة عالية جدا.

أما فلاتر Flutter فهي منصة تمكنا من بناء تطبيقات جوال بواجهات رسومية معتمدين على لغة Dart.

الذي يميز Flutter أنها تمكنا من بناء تطبيقات لأنظمة مختلفة منها الاندرويد أو ال IOS الخاص بأجهزة Apple والمذهل أكثر أنه يمكن أيضا استخدامها كاللغة الأولى لبرمجة تطبيقات نظام جوجل الجديد "فوشيا" Fuchsia والذي قد يزبح الاندرويد عن مكانه.

يجب أن نعلم أيضاً أن Flutter تعتمد في تصميمها Material Design التي تم بناؤها من قبل جوجل والتي تساعده في تصميم صفحات الويب.

2- تاريخ اللغة:

دارت هي لغة برمجة مصممة لتطوير تطبيقات الويب، من تطوير شركة جوجل والتي تستهدف فيها مطوري الويب. أحد أهداف اللغة بأن تعمل على جميع متصفحات الويب المتقدمة والأجهزة المحمولة وصولاً إلى خوادم الويب.

الهدف من إنشاء لغة البرمجة دارت يكمن في المشاكل التي تواجهها لغة جافا سكريبت والتي يصعب حلها مثل أداء البرنامج والحماية من خطر البرمجة عبر المواقع.

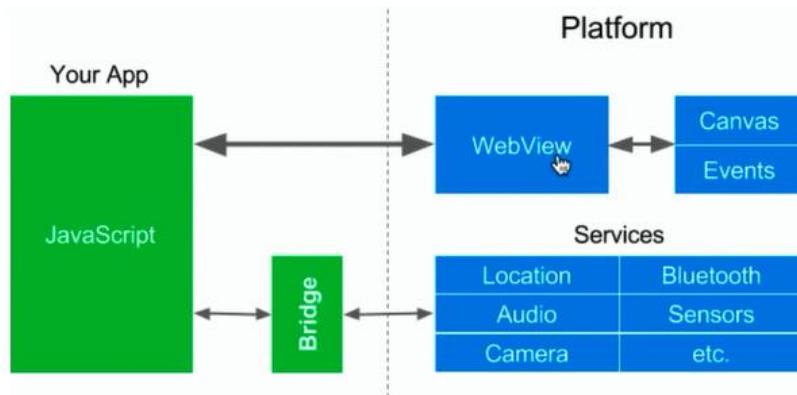
3- مقارنة بين React Native Vs Flutter Vs ionic

ايونيك (Ionic) هو منصة تسمح لك بكتابة التطبيق للجوال باستخدام JavaScript وتحصل العتاد عن طريق لغة Cardova ionic التي تشكل جسر بين الـ ionic والهاردويير وهذا يجعلها أكثر بطالاً.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكتاب محتويات الكتاب على قناته الـ Youtube

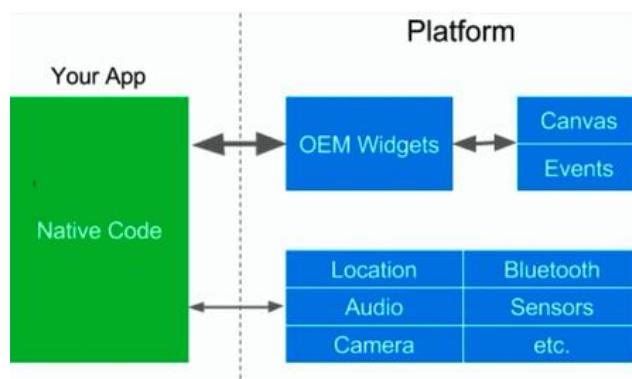




يعد خياراً جيداً لتطبيقات iOS ولكن بالنسبة للأندرويد فقد أثبت عدم كفاءة React Native واضح بالإضافة إلى حاجته لمكاتب كثيرة.



فلاتر (Flutter): لا تحتاج جسر بين التطبيق والعتاد ومنه نستنتج أن الفلاتر حالياً يشكل الحل الأفضل والأسرع لتطبيقات الجوال.



4- تنصيب Windows على Dart & Flutter

١. تنصيب IntelliJ IDEA أو Android studio

٢. نذهب للموقع التالي ونحمل الملف:

<http://Flutter.io/setup-windows/>

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ YouTube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

٣. يستحسن أن يتم حفظ الملف على قرص الـ C ضمن مجلد خاص
٤. تحميل حزم الجافا Java قبل البدء بأي شيء ان لم تكن متوفرة.
٥. تشغيل الملف داخل المجلد والمسما ب Flutter_console.bat
٦. بعد اكتمال التنصيب نشغل cmd ونكتب Flutter في حال لم يتعرف التعارف على الحزمة نضيف مجلد الحزمة في قرص الـ C إلى متغيرات النظام.
٧. في الخطوة التالية وبعد التعرف على Flutter نكتب في cmd عبارة flutter doctor والتي تخبر توفر كل متطلبات Flutter & Dart

٥- إضافة حزمة Dart إلى IDE

في هذه الخطوة سنتعلم إضافة حزمة Dart إلى IntelliJ IDEA أو Android studio

١. نفتح IDE المطلوب.
٢. نختار plug-ins ثم Configuration
٣. نكتب Flutter ثم نضغط Install
٤. نعيد تشغيل البرنامج IDE

٦- تحميل أدوات Dart

١. من الرابط التالي:

<http://www.dartlang.org/tools/sdk#Install>

٢. نختار تنصيب Dart حسب النظام المطلوب
٣. ثم تنصيب اعتيادي

٧- التطبيق الأول في Dart

سوف نستخدم في هذا الكتاب بيئة العمل IntelliJ Idea ولبناء أول مشروع في Dart

١. نفتح IDE ثم نختار مشروع جديد.

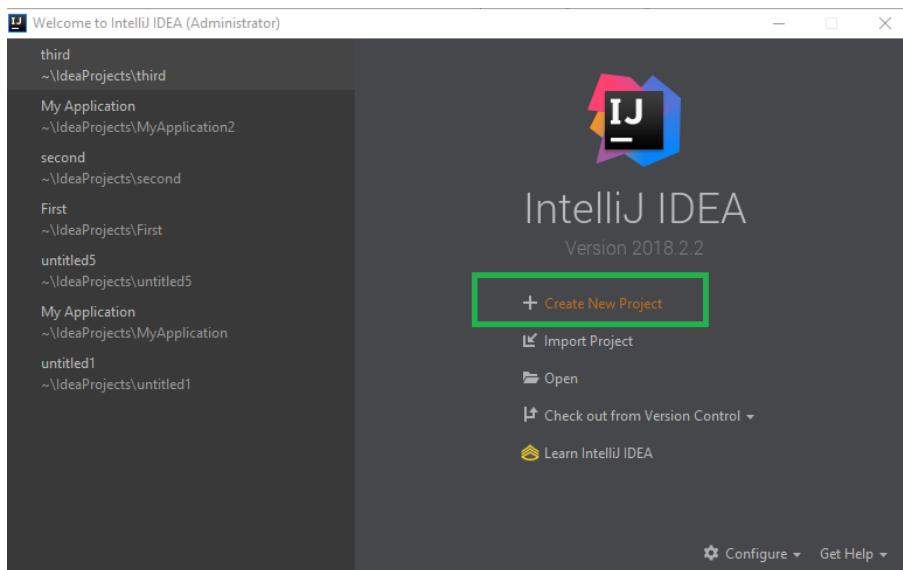
تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكتاب محتويات الكتاب على قناة الـ Youtube

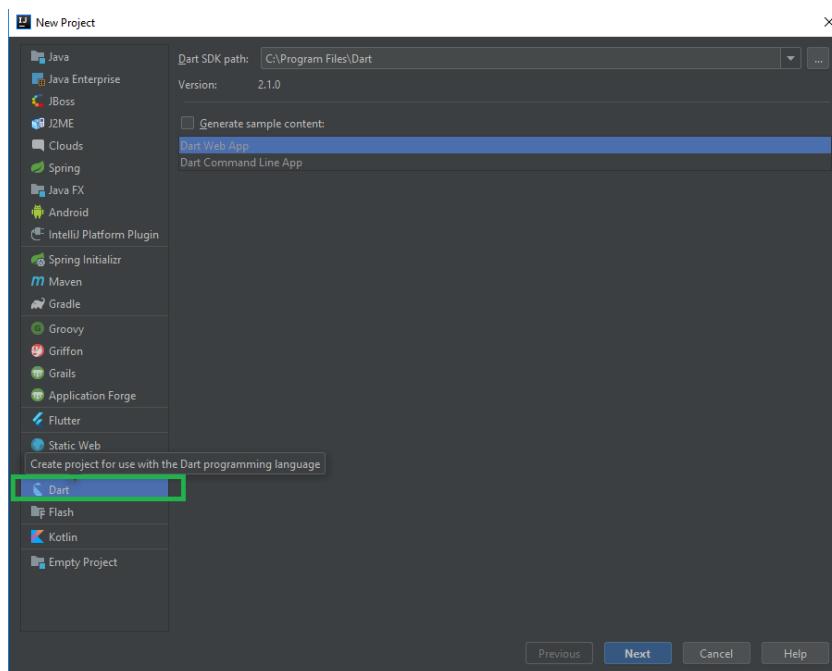


StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com



٢. اختيار حزمة Dart كما في الصورة .



٣. اختيار App Console .

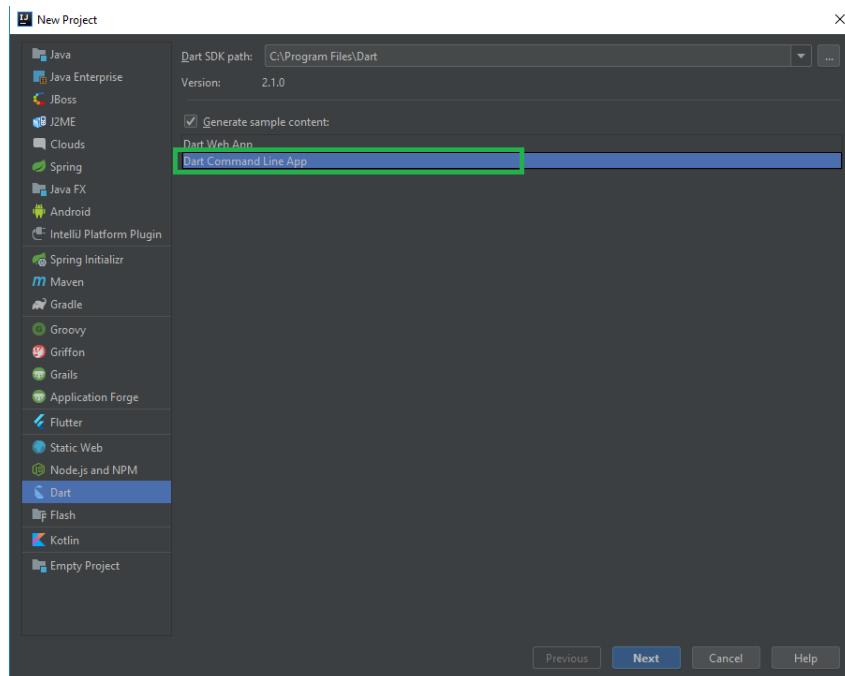
تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـ كامل محتويات الكتاب على قناة الـ Youtube

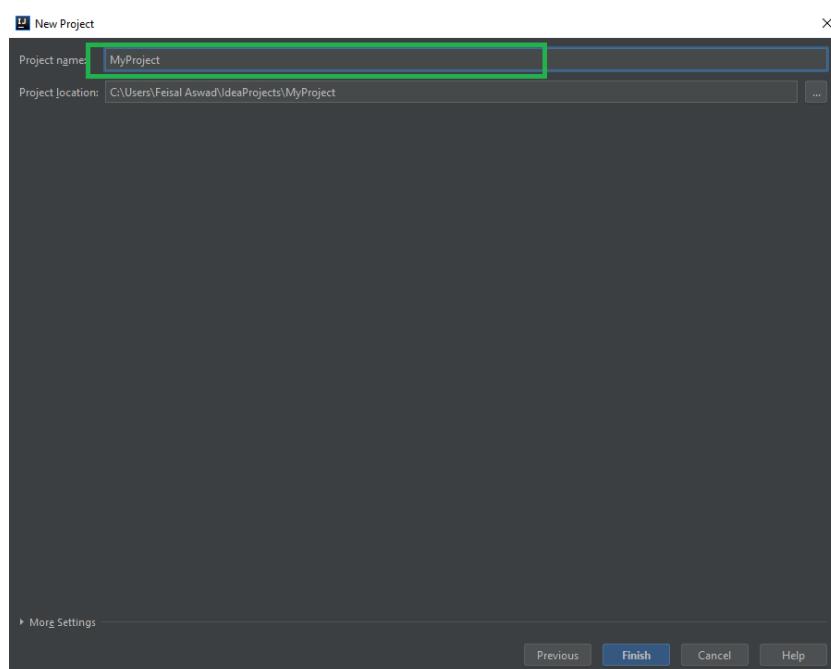


StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com



٤. تسمية المشروع وبدء العمل



تأليف فيصل الأسود | مهندس برمجيات

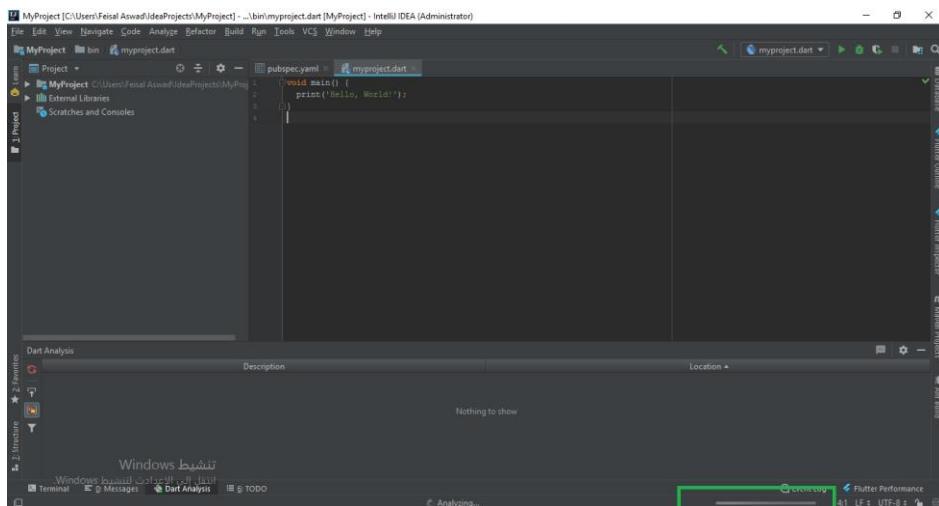
يمكنك الحصول على شرح لـKotlin كامل محتويات الكتاب على قناة الـYoutube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

٥. ننتظر شريط Indexing في الأسفل حتى الاتمام



نلاحظ أن تطبيق دارت يحوي مباشرة الدالة الرئيسية `.main()`.

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

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـة الـ [Youtube](#)

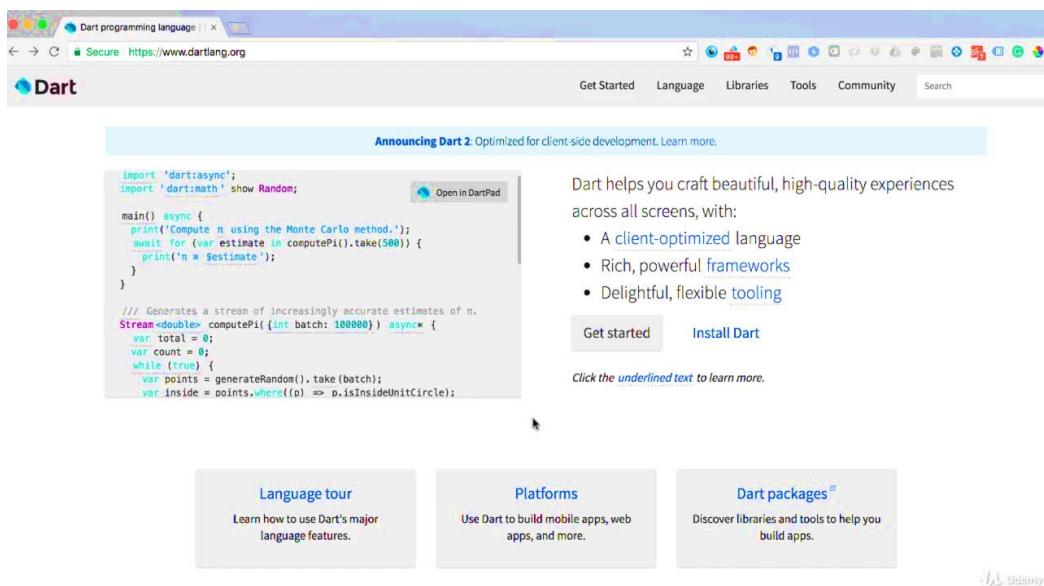


StackOverFlow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

8- التوثيق الخاص بـ Dart Documentation

يمكنك زيارة موقع Dart للطلاع على جميع مزايا لغة Dart أو مجالاتها. كما يمكنك استخدام ما يسمى Dart Pad والذي يقدم الموقع وهو مفسر اكواد دارت online.



يجب العلم أن Dart لا تعمل على جميع المحررات أو بيئات العمل مثل Atom أو emacs في حين آخر تعمل على كل محررات شركة Jet Brains.

9- ثقافة Dart

يتبع مبرمجو دارت تقليداً مثل كل المبرمجين في تسمية المتغيرات ومن هذه التقاليد كما في الجدول:

تسمية الكلاس بحرف كبير	الكلاس
تسمية المتتحول بحرف صغير	المتحول أو الكائن
تكتب بالشكل Lowercase_with_underscore	المكتبات
تكتب بطريقة الجمل LowerCamelCase	الثوابت

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

10- فهم تطبيق Dart

ما يهمنا بالدرجة الأولى هو التابع main والذي دونه لن يعمل التطبيق لذلك أغلب عملنا ضمن

أقواس التابع main

لنكتب تطبيقنا الأول الذي يطبع عبارة: "Hello Stackover Flow Arabi"

```
void main() {
  print('Hello StackoverFlow Arabi');
}
```

والخرج يكون:

Hello StackoverFlow Arabi

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverFlow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube

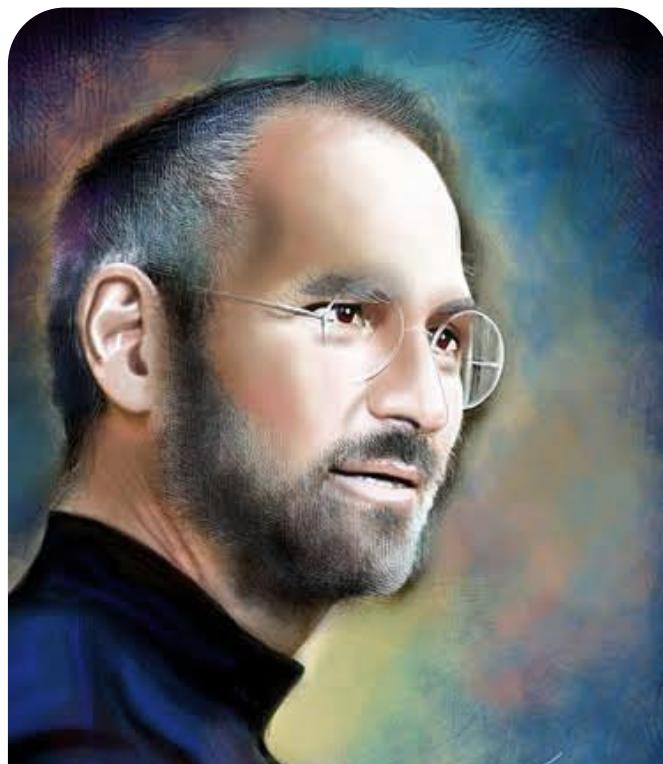


StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

اساسيات لغة دارت

Basics of Dart Language



YOUR TIME IS LIMITED , SO DON'T WASTE IT LIVING SOMEONE
ELSE 'S LIFE

Steve Jobs

اساسيات لغة دارت

1- المتغيرات في Dart

المتغير هو حجرة ذاكرةية بأطوال مختلفة حسب النوع كما يمكن تخزين قيمة ضمنها.

يعرف المتتحول في دارت بالشكل التالي:

```
var name;
```

وهنا عرفنا المتتحول باسم اختياري name دون إعطاءه قيمة. كما يمكن تعريف المتتحول وإعطاءه قيمة مباشرة.

```
var name = "Feisal";
```

والكود التالي يوضح طريقة تعريف المتغيرات في Dart:

```
void main() {
  var name = "Feisal";
  var LastName;
  LastName = "Aswad";
  print (name + ' ' + LastName);
}
```

كما يجب التنويه ان دارت لغة حساسة لحالة الاحرف بما يعني أن المتتحول A غير المتتحول a.

كما يمكن تغيير قيمة المتتحول كما تشاء حسب النوع.

والخرج يكون:

Feisal Aswad

- **أنواع المتغيرات الأساسية:**

<code>name = "Mohamad"</code>	نصية
<code>Num salary = 2000</code> <code>Int salary = 2000</code> <code>Var salary = 2000</code>	تعريف قيمة صحيحة
<code>Num x = 3.2</code> <code>Var P = 3.14</code> <code>Double d = 2.5</code>	عشرية
<code>IsReady = true</code> <code>IsReady = False</code>	بوليانية أو منطقية

كما لا يجوز تكرار تسمية متاحلين بنفس الاسم في أي لغة برمجية.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

٢- الثوابت Constant والمتغيرات النهائية:

الثوابت: هي حجرات ذكرية يجب إعطائها قيمة عند التعريف ولا يمكن تغيير قيمتها ، أبداً تبقى ثابتة.

```
const c = 10;
```

المتغيرات النهائية: هي حجرات ذاكرة عند إعطائها قيمة لا تتغير بعدها (تسند لها قيمة مرة واحدة)

```
final f=10;
```

يتم استخدام هذه الثوابت كقيم ثابتة لبعض المعادلات مثل $P=3.14$ المشتركة في الحسابات. في هذه الحالة يفضل تعريفه كثابت بدلاً من تعريفه كمتغير لضمان استحالة تغييره في أي وقت.

٣- المعاملات الرياضية Operators:

هي المعاملات التي تساعد في العمليات الرياضية.

الجدول التالي يوضح المعاملات في لغة Dart:

الشرح	المعامل
جمع الأعداد	+
طرح الأعداد	-
ضرب الأعداد	*
قسمة الأعداد	/
باقي القسمة	%

مثال:

```
void main() {
  int x = 10;
  int y = 20;
  var result1 = x + y;
  var result2 = x - y;
  var result3 = x * y;
  var result4 = x / y;
  var result5 = x % y;
  print(result1);
  print(result2);
  print(result3);
  print(result4);
  print(result5);
}
```

تأليف: فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

والخرج يكون:



4- المعاملات الشرطية :Arthmatic

هي معاملات تساعدنـي على التحقق من حالات معينة او الشروط المنطقية

الشرح	المعامل
هل x تساوي y	$x == y$
هل x لا تساوي y	$x != y$
هل x أكبر من y	$x > y$
هل x أكبر أو تساوي y	$x >= y$
هل x أصغر من y	$x < y$
هل x أصغر أو تساوي y	$x <= y$

وسوف نفهمها أكثر من خلال الأمثلة لاحقاً.

5- الجملة Is :

توجد في Dart فكرة التحقق من نوع أو قيمة معينة.

في هذا المثال سوف نتحقق هل المتـحول `var num` من نوع `String` سلسلة نصية.

```
void main() {
  var num = 10;
  print(num is String);
  print(num is ! String);
}
```

والخرج يكون:

```
false
true
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

6- الجملة الشرطية If – Else

تعرضنا حتى الان لبرامج تنفذ بشكل متالي حيث ينفذ الحاسب العبارات الموجودة في البرنامج بالترتيب الذي وردت به.

ولكن في التطبيقات العملية نحتاج للتخاذ بعض القرارات بعماً لشروط معينة، ومن هنا ظهرت الحاجة لوجود طرق لجعل البرنامج قادرًا على تغيير تسلسل تنفيذ التعليمات بعماً لشروط المطلوبة وهو ما يعرف بالجمل الشرطية.

ابسط الجمل الشرطية هي تلك التي تستخدم if – else وطريقة كتابتها بالشكل التالي:

```
void main() {
    var x=3;
    if(x==3)
    {
        print("Yes");
    }
    else
    {
        print("No");
    }
}
```

وهنا يتم التحقق من الشرط $x == 3$ ويتطابق لأن x فعلاً تساوي الـ 3 ويتم طباعة Yes فقط.

كما كان من الممكن كتابة البرنامج بالشكل التالي بدون اقواس للشرط وذلك لأن الشرط يحوي تعليمة واحدة.

```
void main() {
    var x=3;
    if(x==3)
        print("Yes");
    else
        print("No");
}
```

هنا يجب الانتباه انه في حال الشرط يحوي اكثر من تعليمة لا يجوز كتابته الا مع اقواس.

وكذلك يمكن وضع شرط if لوحده بدون وضع حالة else.



أيضاً يمكن وضع عدد لا نهائي من الشروط كما يلي:

```
void main() {
var x=3;
if(x==3)
{
    print("Yes");
}
else if(x==6 || x==1)
{
    print("x is 2");
}
else
{
    print("No");
}
```

الشرط الثاني يتحقق من كون X هو عبارة عن العدد 6 او العدد 1

وفي الحالات الثلاث للبرامج السابقة سيكون الخرج هنا عبارة Yes لأننا عرفنا x عدد بقيمة 3.

مثال: برنامج يقوم بإدخال القيمة الصحيحة X وإذا كانت أكبر او تساوي 100 يطبع "large value"

```
import 'dart:io';
void main() {
print('Enter x');
var x=stdin.readLineSync();
if(int.parse(x) > 100)
    print('large value');
```

هنا طلبنا من المستخدم ادخال قيمة x ثم قمنا بامر اللادخال عن طريق stdin وبعدها حولنا السلسلة النصية المدخلة الى قيمة وقارناها مع 100 وفي حال التحقق للشرط يطبع العبارة كما يلي:

```
Enter x
150
large value
```



7- الجملة الشرطية switch

تستخدم هذه الجملة الشرط المغلق (close condition) أي الشرط الذي يأخذ قيمة محددة (رقمية او حرفية) ولهذا يمكن القول بأن هذه الجملة تتعامل مع متغير واحد يمكن أن يأخذ في كل مرة قيمة محددة من خلال مجموعة من القيم.

الصيغة العامة للجملة:

```
switch(variable) {
    case 1: Statmets; break;
    case 2: Statmets; break;
    .....
    .....
    case n: Statmets; break;
    default: Statmets; break;
}
```

هذه الصيغة تمثل الاحتمالات التي يمكن أن يأخذها المتغير variable من 1 حتى n كل احتمال يتم وضعه داخل جملة باستخدام العبارة case ثم بعد ذلك كتابة الحدث الذي يمكن أن يكون تعليمية او مجموعة تعليمات ومن ثم ننهي التعليمات بالعبارة break والتي توقف تنفيذ الحالة وتنمنع الدخول في حالة أخرى.

أما الاحتمال default يمثل الحدث الملازم للحالة خارج النطاق من 1n

ملاحظات:

- إذا تطابقت قيمة Mi التي تلي case مع قيمة المتغير variable فإنه يبدأ التنفيذ عند بداية المطابقة ويكل للنهاية أو أن يتلقى بتعليمية تنقل التنفيذ إلى خارج هذه الجملة.
- تنفذ العبارة التي تلي default إذا لم تتطابق أي عبارة من Mi مع المتغير variable.
- إذا أردنا اختيار تنفيذ حالة واحدة فقط من case فيجب أن تتبعها بتعليمية break
- إذا لم تتطابق أي قيمة لـ Mi مع المتغير variable وكانت default غير موجودة فلا يتم تنفيذ أي تعليمية.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

مثال: اكتب برنامجاً للإدخال القيميتيين x و y ثم اجراء العمليات الحسابية (الجمع - الطرح - القسمة - الضرب - باقي القسمة) باستخدام `Switch` على القيميتيين المدخلات.

```
import 'dart:io';
void main() {
print('Enter x');
var x_String=stdin.readLineSync();
print('Enter y');
var y_String=stdin.readLineSync();
print('Enter operation');
var operation=stdin.readLineSync();
var x=int.parse(x_String);
var y=int.parse(y_String);
switch(operation){
  case '+':print(x+y);break;
  case '-':print(x-y);break;
  case '*':print(x*y);break;
  case '/':print(x/y);break;
  case '%':print(x%y);break;
  default:print('Input is wrong');
}
}
```

8- الجملة التكرارية المحدودة for-loop

في هذا النوع يتم تحديد عدد مرات التكرار بواسطة رقم صحيح ويبني هذا النوع على مفهوم العدد `counter` وهو عبارة عن متغير بأخذ قيمة ابتدائية Initial Value تتغير باستمرار بمعدل معين بالإضافة أو النقصان إلى أن تصل إلى القيمة النهاية Final value ولتصميم هذه البنية نستخدم الجملة `.for`.

مثال توضيحي لهذه الجملة لنفرض نريد طباعة الأعداد من 0 إلى 9 وبدون تكرار تعليمة الطباعة.

```
void main() {
for(int i=0;i<10;i++)
{
  print(i);
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

والخرج يكون:

```
0
1
2
3
4
5
6
7
8
9
```

Process finished with exit code 0

مثال:

اكتب برنامج لحساب مجموع الاعداد الفردية sumOdd والاعداد الزوجية sumEven بدءا من العدد 20 الى العدد 50.

```
void main() {
    var sumOdd=0,sumEven=0;
    for(int i=20;i<51;i++)
    {
        if(i%2==0)
            sumEven=sumEven+i;
        else
            sumOdd=sumOdd+i;
    }
    print(sumEven);
    print(sumOdd);
}
```

حيث العدد الزوجي هو العدد الذي يقبل القسمة على 2 مع باقي 0.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

9- التعلیمة Continue و break

التعلیمة **break** تستخدم لإيقاف الحلقة التكرارية عندما يتحقق شرط معین.

```
void main() {
    for(int i=0;i<5;i++)
    {
        print(i);
        if(i==2)
        {
            print('i==2');
            break;
        }
    }
}
```

هنا نلاحظ أن الخرج سوف يطبع من 0 حتى 2 ثم يطبع عباره `i==2` ويتوقف عن اكمال الحلقة بسبب وجود تعلیمة `break`.

```
1
2
i==2
```

التعلیمة **Continue** تستخدم لاستبعاد شرط معین من الحلقة التكرارية.
هنا سوف يطبع الاعداد من 0 الى 4 مع تجاهل الرقم 2 وذلك لأنه عند ورود 2 سوف تنفذ تعلیمة `continue` التي تستبعد او تتجاهل التعلیمة التي تليها وهي الطباعة فلن يقوم بالطباعة في حالتها.

```
void main() {
    for(int i=0;i<5;i++)
    {
        if(i==2)
        {
            continue;
        }
        print(i);
    }
}
```

```
0
1
3
4
```

Process finished with exit code 0

تألیف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لکامل محتويات الكتاب على قناء الـ [Youtube](#)



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

10- الجملة التكرارية الغير محدودة while-loop & do-while

هذا النوع غير محدد الدورات تستمرة الحلقة في التنفيذ طالما الشرط محقق.

:مثال حلقة while

```
void main() {
var i=0;
while(i<10)
{
    print(i);
    i++;
}
}
```

وهنا سوف يطبع الاعداد من 0 الى 9 لأن الشرط $i < 10$ متحقق.

ننوه أن التعليمة $i = i + 1$ تماشل أي زيادة قيمة واحد للمتغير i .

:مثال حلقة do-while

```
void main() {
var i=0;
do{
    print(i);
    i++;
}while(i<10)
}
```

وهنا سوف يطبع الاعداد من 0 الى 9 لأن الشرط $i < 10$ متحقق كما في المثال السابق تماماً.

ولكن الاختلاف يكمن بين الحلقيتين أن الأولى تختبر الشرط ثم تدخل الحلقة ولن تدخل ولا مرة لو الشرط غير متحقق.

أما الثانية فإنها سوف تنفذ الحلقة وفي نهاية التنفيذ سوف تختبر الحلقة ، أي ستدخل مرة واحدة على الأقل.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

مثال:

برنامج لليجاد مربعات الأعداد من 1 الى 10 .

```
import 'dart:math';
void main() {
var i=0;
while(i<10)
{
    print(pow(i,2));
    i++;
}
}
```

11- القوائم Lists

لنفرض أنه طلب مني كتابة برنامج بسيط للغاية وهو إدخال درجات عشر طلاب، لكي تحل هذا البرنامج فإن عليك أن تقوم بالإعلان عن 12 متغيراً من وربما أن هذا مقبول نوعاً ماً، ولكن ماذا لو طلب منك إدخال أكثر من 1000 درجة طالب لحل هذه الإشكالية توفر لك لغة دارت القوائم. صحيح أننا قمنا بحل مسائل من هذا النوع لم تتطلب القوائم لكن ماذا لو طلب منك البحث عن درجة طالب معين فلن يكون هناك أي حل إلا بواسطة قائمة.

لغة دارت تزود القوائم والتي هي عبارة عن مصفوفات مع بعض الخصائص الإضافية.
يمكن تعريف قائمة كما يلي:

```
var l=[1,2,3];
```

مثال:

اكتب كود يحوي قائمة مخزن فيها أربع عناصر من 1 الى 4 ثم

- اطبع العنصر الثاني.
- غير قيمة العنصر الرابع الى 5.
- اطبع طول القائمة.

```
void main() {
var l=[1,2,3,4];
print(l[1]);
l[3]=5;
print(l.length);
}
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

هنا نلاحظ أن التغيير حصل على الحجرة رقم 1 والتي تكافئ العنصر الثاني ، وذلك لأن المصفوفة تبدأ بالفهرسة من الصفر دوماً.

12-التوابع Function

لقد تقدمنا كثيراً في دارت بعد مواضيع الحلقات والقوائم وربما لم يبقى لنا سوى عدة مواضيع حتى ننتقل إلى مرحلة البرمجة الكائنية وأحد أهم هذه المواضيع هي التتابع.

تقوم البرمجة الهيكلية على عدة تتابع بدلًا من تابع واحد هو وبإمكانه بعد هذا الموضوع تجزئة برنامجك إلى عدة تتابع كل تابع منها يقوم بوظيفة محددة ثم يسلمهما للآخر بعد أن يكون قد أنجز ما هو مطلوب ؛ ومن الممكن النظر إلى التتابع على أنها عبارة عن اتحاد عدة أوامر برمجية في كتلة واحدة ولهذا الاتحاد وظيفة معينة يقوم بأدائها وبالتالي فسيصبح بإمكانك الاستفادة من هذه التتابع في جميع برامجك ، فكما رأيت إحدى وما تقوم به من أعمال ، بإمكانك أن تقوم `math` تتابع المكتبة الرياضية بصنع تتابع وضمها في مكتبة واحدة ، وأيضاً فعن طريق التتابع بإمكانك تجزئة عمل برنامجك إلى أجزاء كثيرة وصغيرة للغاية بدلًا من أن تكون واحدة وبصراحة فإن أغلب البرامج تركت أسلوب التجزئة إلى تابع واحد هو تتابع وأبدلته ب التقسيم البرنامج إلى كائنات والكائنات نفسها تشمل على تتابع كثيرة ، من الضروري للغاية أن تدرك أهمية هذه الوحدة إذا ما أردت التقدم في البرمجة فأولًا هي مدخل إلى الكائنات وثانياً هي أحد أهم مواضيع لغة دارت .

بعد هذه المقدمة البسيطة سندخل في اختصاص هذه الوحدة.

فكرة التتابع هي بكتابة كود في تابع منفصل واستدعائه عند الحاجة ، لنفرض أنه ليس لدينا تابع جاهز لإيجاد مربع العدد ونريد إنشاء واحد فسيكون كما يلي:

```
int power(int num)
{
    return num*num;
}
```

وهو تابع عند استدعائه عن طريق اسمه سيقوم بالعملية الرياضية السابقة وهي إيجاد مربع العدد ، كما ان الكلمة `int` تعني ان التابع سيرجع عبارة من نوع `int` أي عدد صحيح ويتم عملية الارجاع للتابع المستدعي بالعبارة `.return`.

```
void main() {
var x=power(5);
}
```

وهذا التابع الرئيسي `main` استدعى التابع `power` بالقيمة 5 وسيذهب التنفيذ للتابع `power` ويقوم تأليف فيصل الأسود | مهندس برمجيات يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ [Youtube](#)



بالجاء ويعيد نتيجة الجاء للتابع main لتوسيع النتيجة في المتحول x. بعض التوابع لا ترجع شيء فنضع عند تعريفها الكلمة void وهذا لا يحتاج للإرجاع او عبارة return.

13- التعليقات:

”التعليقات Comments“ هي عبارة عن جمل توضيحية يقوم المبرمج بإضافتها ضمن أي مكان في المشروع البرمجي أثناء بنائه، وتُعتبر التعليقات كالمفتاح، فهي تهدف إلى تسهيل قراءة المشروع البرمجي سواءً للشخص الذي قام ببنائه أو حتى لأي شخص آخر يحاول فهمه أو التعديل عليه.

التعليقات يتم تجاهلها بشكل نهائي أثناء تنفيذ المشروع البرمجي حتى وإن كانت تحتوي على أكواد برمجية، فهي لا تظهر نهائياً ولا يستطيع رؤيتها إلا من يحصل على ملفات المشروع البرمجي.

ولتبسيط فهم فوائد التعليقات في لغات البرمجة لنقل إنك قمت ببناء مشروع برمجي مؤلف من مئة سطر برمجي على سبيل المثال ثم أردت أن تقوم بالتعديل على المشروع بعد فترة زمنية؛ لنقل شهر على سبيل المثال، أليس الأمر بغاية الصعوبة أن تتذكر كل ما قمت بفعله أثناء كتابة مشروع البرمجي؟ أليس من الصعب البحث في مئة سطر للتعديل على جزء معين؟ ناهيك عمّا إذا كنت قد طرحت المشروع لمبرمجين آخرين لإكمال المشروع البرمجي أو الاعتماد عليه. لهذا السبب وجدت التعليقات ببساطة، لشرح المشروع البرمجي وتقسيمه بطريقة تجعل من التعديل على المشروع أمر بغاية البساطة لأي شخص، حتى وإن لم يكن هو من قام ببناء المشروع من الأساس.

لعل السؤال في ذهنك الآن هو كيفية وضع التعليقات في لغات البرمجة المختلفة؟ في الحقيقة تختلف طريقة إضافة التعليقات من لغة لأخرى، ولكن تقريباً معظم التعليقات في لغات البرمجة مستمدة من نظام التعليقات في لغتي C و C++, وهو إضافة تعليق بعد شرطتين مائلتين لليسار //) ثم كتابة التعليق بعد هاتين الشرطتين بهذا الشكل:

```
//here will set comment
```

```
//here will set comment
```

وبمجرد النزول لسطر جديد ينتهي التعليق، ونفس هذا النوع من التعليقات الذي ينتهي بمجرد

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

النَّزُولُ لِسُطُرِ جَدِيدٍ هُوَ أَنْ نَقُومُ بِاستِخْدَامِ الرَّمْزِ (#)، وَمَثَلٌ عَلَى ذَلِكَ:

```
#here will set comment
```

```
#here will set comment
```

أَمَّا النَّوْعُ الثَّانِي هُوَ لِلتَّعْلِيقَاتِ الطَّوِيلَةِ وَالَّتِي نَقُومُ بِتَمْيِيزِهَا، وَهُوَ بِاسْتِخْدَامِ الشَّرْطَةِ الْمَائِلَةِ لِلِّيْسَارِ ثُمَّ عَلَمَةِ النَّجْمَةِ لِفَتْحِ التَّعْلِيقِ، ثُمَّ نَقُومُ بِإِغْلَاقِ التَّعْلِيقِ عَكْسَ الْفَتْحِ وَهُوَ بِإِضَافَةِ عَلَمَةِ النَّجْمَةِ مَتَّبِعَةٍ بِالشَّرْطَةِ الْمَائِلَةِ لِلِّيْسَارِ لِيُصْبِحَ بِالشَّكْلِ التَّالِيِّ:

```
/* here will comments */
1
/* here will comments */
```

تُعَتَّبُ إِضَافَةُ التَّعْلِيقَاتِ أَمْرٌ يُزِيدُ مِنْ احْتِرَافِيَّةِ الْعَمَلِ الْخَاصِ بِكَ حَتَّىٰ وَإِنْ كَانَ الْمَشْرُوْعُ الْبَرْمَجيُّ الْخَاصُ بِكَ بِسِيَطٍ، وَعَلَى الرَّغْمِ أَنَّ لِكَ الْحَرِيَّةَ الْمَطْلُقَةَ فِي إِضَافَةِ التَّعْلِيقَاتِ فِي أَيِّ مَكَانٍ فِي مَشْرُوْعِكَ الْبَرْمَجيِّ وَلَكِنَّ هُنَّاكَ قَوَاعِدٌ عَامَّةٌ يَتَّبَعُهَا مِنْ قَبْلِ الْمَبْرَمِجِينَ، هَذِهِ الْقَوَاعِدُ لَيْسَتْ بِالْحَيْرَةِ أَنْ تَلْتَزِمَ بِهَا إِلَّا عِنْدَمَا تَقُومُ بِطَرْحِ الْمَشْرُوْعِ الْبَرْمَجيِّ الْخَاصِ بِكَ لِمَبْرَمِجِينَ آخَرِينَ، وَلَعَلَّ أَهْمَّ هَذِهِ الْقَوَاعِدِ هِيَ مَا يَلِي:

- استخدم كلمات واضحة ومفهومة للجميع في شرح التعليق، ولا تستخدم رموز وطلاسم غير قابلة للفهم.
- استخدم اللغة الإنجليزية وذلك لتسمح لأي شخص بفهم التعليق، لأن اللغة الإنجليزية هي اللغة الأكثر انتشاراً بين لغات العالم.
- أضف التعليقات ليشرح التعليق ما بعده من أكواد برمجية، وهذا ما يجعل قراءة السكريبت أكثر يسر، ولا تقوم بإضافة التعليقات بمناطق عشوائية من المشروع البرمجي لأن هذا الأمر يزيد الوضع تعقيداً.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ [Youtube](#)



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

14-جدول الكلمات المحجوزة:

هي كلمات محجوزة للغة دارت ولا يجب تسمية المتغيرات او الكائنات بأسماء مشابهة الجدول التالي يذكر أهمها:

assert	default	finally	rethrow	try
break	do	for	return	var
case	else	if	super	void
catch	enum	in	switch	while
class	extends	is	this	with
const	false	new	throw	
continue	final	null	true	

15-أنواع العناصر في Dart:

تقدم دارت عدد جيد من الأدوات للمتحولات اهم المتغيرات المقدمة مع مكتبة دارت:

الأعداد: تقدم dart نوعين من الأعداد ذكرها.

- الأعداد الصحيحة int: وهي قيم مجالها ذاكريا من . حتى ٦٤ بت.

مثال:

```
var x = 1
var nex = 0xdefafe
```

حيث الرقم الثاني مكتوب بطريقة النظام الست عشري.

- الأعداد ذات الفاصلة double:

هي أعداد بفاصلة أكبر من ٦٤ بت حسب معيار IEEE754.

مثال:

```
var y = 1.1;
var exponents = 1.42e5
```

وكلًا من Int, double ينتمي إلى النوع num والذي يتضمن التعامل مع المؤشرات والمعاملات الرياضية مثل (+,-,*,/). كذلك التوابع الرياضية مثل ()Abs للقيمة المطلقة ()ceil للتقرير يمكن التعامل مع هذه التعامل باستدعاء المكتبة .math.dart.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قناة الـYoutube



السلالس النصية: string

السلالس النصية هي عبارة عن تسلسل مهارف من ترميز UTF.

أي يمكن استخدام مختلف اللغات.

نستخدم في dart لكتابه سلسلة كلًّا من علاقتي التنصيص المفردة والمضاعفة مثل:

```
var s3 = 'H\'s astring';//It's a string
var s4 = "It's a string";
```

العلامة \ تستخدم للهروب وذلك لإظهار العلامات المحجوزة.

```
var s = 'hello' + 'world';
```

كما يوجد طريقة لكتابه السلسلة على عدة أسطر باستخدام علامة التنصيص الثلاثة مثل:

```
var s = " hello we"
      "Are study dart";
```

وكما في لغات البرمجة إن أردت تجاوز سطر عند الطباعة لسلسلة نستخدم n :

```
var s = "hello world\n I am here";
```

وهنا سيطبع hello world في سطر و I am here في سطر آخر.

المتحولات المنطقية: Booleans

المتحول البوليني أو المنطقي هو متحول يأخذ قيمتين فقط هما true or false ودائما عند اختبار

الشروط يختبر عبارات منطقية مثل:

```
if (x == true)
  print ("yes");
if(x == 5)
  print ("x = 5");
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

16- اللوائح والقوائم :List

من الاستخدامات الشائعة في البرمجة هي المصفوفات ، دارت تقدم المصفوفات على شكل قوائم كائنات.

القائمة هي ببساطة مجموعة مرتبة من الكائنات ، والمكتبة الأساسية مع دارت تزود أصناف التعامل مع القوائم .

يبين الشكل تمثيل منطقي لاستخدام القوائم في دارت:

0	1	2
12	44	64

هنا القائمة تحوي ثلات قيم هم 12-44-64 وتعرف على أنها عناصر القائمة.

كل عنصر من القائمة يمكن الوصول له عن طريق رقم مميز وفرید له يدعى الفهرس index . الفهرس في دارت يبدأ من الصفر وينتهي حتى (طول المصفوفة - 1) وهذا لدينا الفهرس بدء من الصفر وانتهت عند القيمة 2 .

يمكن ان تكون القوائم:

- بطول ثابت:

أي لها طول محدد عند التصريح عنها ولا يمكن ان تتسع اكثراً من طولها المحدد.

- بطول متناهي:

هي لوائح لا نهاية تتسع للعديد من الكائنات ضمنها ولا يوجد لها حدود اتساع.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

بعض التوابع الخاصة بالتعامل مع اللوائح:

التابع	الشرح
first()	يعيد العنصر الأول من اللائحة
isEmpty()	يعيد قيمة بوليانية تحدد في حال اللائحة فارغة أم لا. true = empty false =not empty
isNotEmpty()	يعيد قيمة بوليانية تحدد في حال اللائحة ليست فارغة أم لا. true = not empty false =empty
length	يعيد حجم اللائحة.
last	يعيد العنصر الأخير من اللائحة
reversed	يعيد كائن لنفس اللائحة مرتبة بترتيب عكسي.
single	يفحص إذا كانت اللائحة تحوي عنصر واحد فقط وترجعه.

17 - الخرائط: Map

هي عموماً كائنات لكل منها قيمة ومفتاح. وكل من القيمة أو المفتاح يمكن أن يكون أي نوع من الكائنات.

كل مفتاح فريد ونادر لا يتكرر ضمن الخريطة أما القيمة يمكن أن تكون نفسها لمفتاحين.

```
var names = {
  'first': 'Ahmad',
  'second': 'Feras',
  'third': 'Nour'
};
```

حيث العناصر في العامود الأيسر هي المفاتيح والعامود الأيمن هي القيم.

مثال:

```
var nobleGases = {
  Z: 'helum',
  10: 'neon',
```

تأليف: فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```
18: 'argon'
};
```

كما يمكن كتابة الخريطة بالشكل:

```
var Names = Map();
Names ['first'] = 'Ahmad';
Names[second] = 'Feras';
Names['third'] = 'Nour';
```

كما يمكن إضافة عنصر جديد للخريطة في أي وقت:

```
Names['fourth'] = 'farah';
```

18- جدول المؤثرات الزيادة والنقصان:

Operator	Result
+	Addition
-	Subtraction (also unary minus)
*	Multiplication
/	Division
%	Modulus
++	Increment
+=	Addition assignment
-=	Subtraction assignment
*=	Multiplication assignment
/=	Division assignment
%=	Modulus assignment
--	Decrement

19 - جدول العمليات المنطقية:

Input1	Input2	And	Or
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

20 - التعلیمة try – catch

تستخدم هذه التعلیمة لتجنب الوقوع في حالات فشل للبرنامج عند ورود خطأ ويتم فيها وضع كل من البرنامج المطلوب أو الجزء المطلوب في try مثل العمليات المهمة فتح قاعدة البيانات – الأشكال في الشبكة وفي حال وجود أي خطأ سوف يكمل التنفيذ في catch

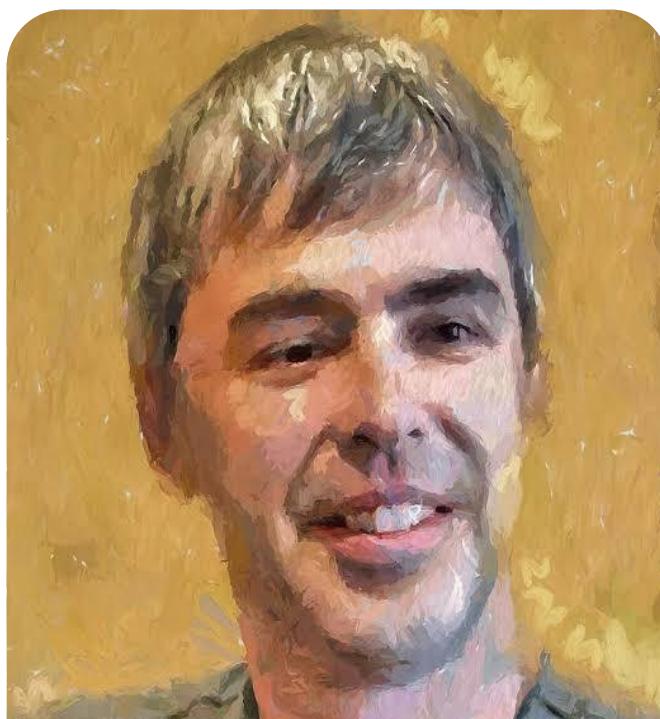
مثال:

```
try {
    var x = 5;
    var y = 0;
    var z = x/y;
    print (z);
}
catch (e){
    print ('divide by zero');
}
```



البرمجة المتقدمة في دارت

Advanced Programming Dart



YOU NEVER LOSE A DREAM , IT JUST INCUBATES AS A HOBBY
Larry Page

البرمجة المترددة في دارت

1- الأصناف:

دارت هي لغة كائنية تتعامل مع الأصناف والعمليات عليها.

كل كائن هو نسخة من صنف أساسى.

لنبوسط الأمر الصنف هو عبارة عن نوع جديد أريد أنا إنشاءه وبعد الانتهاء من بناءه آخذ منه نسخ تسمى كائنات.

مثال: لنفرض أننا أردنا إنشاء صنف نقطة ونعلم أن هذه النقطة تتكون من احداثيات x,y وهذا لابد أن نذكر أن الكلاس يتكون من خصائص أفعال (طرق) وباني.

صنف نقطة هو:

```
class Point {
    num x;
    num y;
    point (num x, num y)
    {
        this.x = x;
        this.y = y;
    }
    void printXAndY()
    {
        print(x);
        print(y);
    }
}
```

بعد أن شكلنا الصنف يمكن استخدامه عدد لا نهائي من المرات بتشكيل كائن منه كما يلى:

Point p1 = new Point(5,3);

وهنا أنشأنا كائن p1 من نوع Point ولها القيمة 3

وفي حال كتبنا :

p1.printXAndY();

سوف يقوم بطباعة القيم كما يلى:

5
6

في dart يمكن كتابة الباني بالشكل المختصر:

Point (this.x, this.y):

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

أي القيمة الأولى الواردة عند إنشاء الكائن هي نفسها قيمة `x` للصنف `point` والقيمة الثانية هي قيمة `y` للصنف.

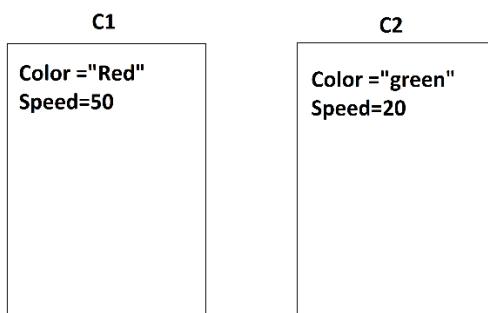
كما رأينا يستدعي الفعل (الطريقة) في `dart` عن طريق اسم الكائن `1` ثم اسم الطريقة المطلوبة. مثال: ننشأ صنف لسيارة لها (سرعة - لون) وعند إعطائهما مساحة تقوم بتقدير الزمن لـك عن طريق فعل خاص لها.

```
class Car {
  num speed;
  String color;
  car (this.speed, this.color);
  double giveMeTime(int distance)
  {
    return speed/distance;
  }
}
```

وفي البرنامج الرئيسي ننشأ سيارتين

```
Car c1 = new Car(50, 'Red');
var z1 = c1.giveMeTime(25);
print (z1);
Car c2 = new Car(30, 'Green');
c2.speed = 20;
var z2 = c2.giveMeTime(20);
print (z2);
```

أنشأنا كائنين لسيارتين الأولى بسرعة ٥٠ ولون أحمر والثانية بسرعة ٣٠ ولون أخضر غيرنا سرعة السيارة الثانية لـ ٢٠



الخرج سيكون

2
1

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

التعليمية `return` مهمتها إرجاع قيمة من الطريقة إلى التابع. كما رأينا عندما طلبت `c` الزمن بطريقة() تم ارجاع القيمة للبرنامج الرئيسي عن طريق `return` وهناك بعض الطريق لا ترجع شيء.

2- معرفات الوصول:

على خلاف الكثير من لغات البرمجة dart لا تحوي معرفات وصول الكلاسيكية إنما تحوي فقط معرف الخصوصية `private` على مستوى الحزمة ككل. أي إذا أردنا إنشاء متحول ضمن صنف `class` وهذا المتحول لا يوجد أي كلاس آخر يمكنه معرفة قيمته نكتب المتحول بالشكل:

```
var _x = 5;
```

بحيث نضع قبل اسمه `_` وهذا الكلام ينطبق أيضاً على التوابع والطرق بحيث لا استطيع استدعاء أي طريقة من خازنة المكتبة في حال بدأت بالمعرف `_` مثلاً

```
int _giveMeTime ()
```

3- المتحول الستاتيكي:

الكثير يعني من فهم المتحول الستاتيكي إنما هو يحمل مفهوماً بسيطاً جداً. لنعود لمثالنا السابق بخصوص إنشاء صنف سيارة ونفرض أننا عرفنا متحول السرعة بشكل `static` كما يلي:

```
class car {
  static num speed;
  String color;
  car (this.speed, this.color);
  double giveMeTime (int distance)
  {
    return speed/distance;
  }
}
```

تأليف، فيصل الأسود | مهندس برمجيات

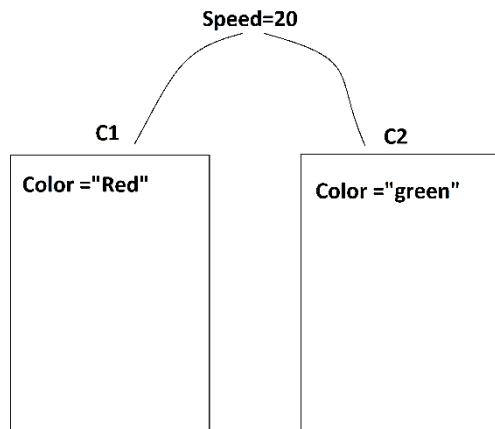
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ [Youtube](#)



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

هنا يكون شكل الكائنات عند إنشاء سيارتين كما يلي:



فنلاحظ أنه عند تعريف متحول ستاتيكي يصبح قيمة نفسها لكل الكائنات بحيث إذا عدلتها لأي كائن (سيارة) سوف تتعدي للجميع.
طبعا يوجد أيضا التوابع الستاتيكية و يجب أن نعلم أن التابع الستاتيكي يتعامل حسرا مع متحول ستاتيكي.

4- الوراثة:

الوراثة مفهوم بسيط يساعدني على بناء الأصناف بالاعتماد على أصناف سابقة ويريحني من عناء صناعتها من جديد.

لنفرض أريد إنشاء صنف سيارة لديها (سرعة، لون، وزن) وتتابع ليعطييني الوقت كما في المثال السابق هنا نلاحظ أنه لدينا الصنف السيارة القديم فيه كل من السرعة واللون والتابع المطلوب ولكننا بحاجة بالإضافة خاصة الوزن. فهنا نستخدم الوراثة ويكون الصنف الجديد كما في الشكل:

```

class car2 extends car {
    car2(num speed, String color) :
        super(speed, color);
        var weight;
}
  
```

تأليف، فيصل الأسود | مهندس برمجيات

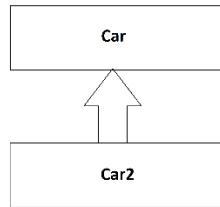
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



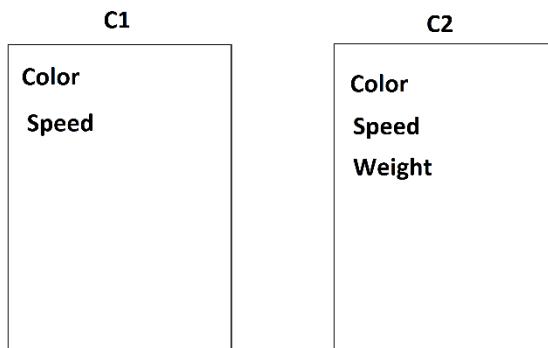
StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

أي صنف السيارة Car2 يرث من صنف السيارة Car ويأخذ جميع خصائصها وأفعالها.



فلو أنشأنا الآن الكاثنين الأول من نوع Car والثاني من Car2 يكونا بالشكل:



تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube

[Youtube](#)



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

برمجة التطبيقات باستخدام فلتر

Applications programming with
flutter



THE BIGGEST RISK IS NOT TAKING ANY RISK

Mark Zuckerberg

التعرف على بيئة فلاتر

مقدمة:

فلاتر هي منصة تطوير تطبيقات جوال مفتوحة المصدر أنشأت من قبل جوجل وتستخدم هذه المنصة من أجل تطوير تطبيقات أنظمة اندرويد و ios وكذلك نظام فوشا الجديد. النسخة الأولى من فلاتر والمعروفة بـ sky السماء قد بنيت لتطوير تطبيقات الاندرويد عام ٢٠١٥ وهكذا تطورت حتى باقي أنظمة التشغيل.

يجب أن ننوه أن أي شيء في فلاتر هو widget على سبيل المثال: النص، الصورة والزر هي widget. بعد أن نكون جهزنا أحد IDE (Android studio, InteliJ IDEA) كما يمكن زيارة التوثيق الخاص ب Flutter من خلال الموقع التالي :

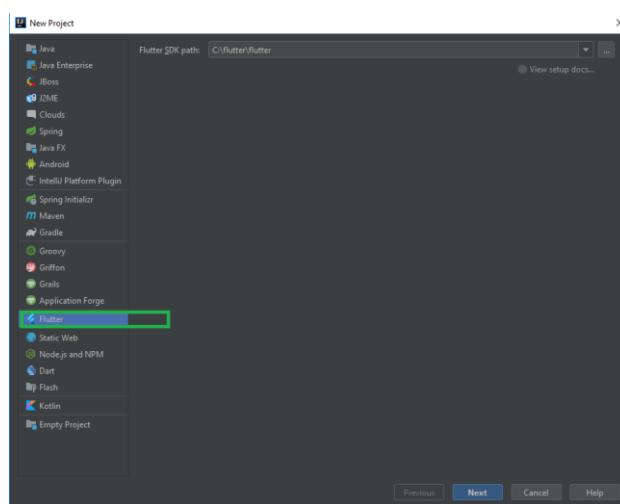
<https://flutter.io>

سوف نستخدم في كتابة البرامج بيئة العمل IntelliJ IDEA لأنها تمكنا من بناء تطبيقات نظامي android والاندرويد كما ننوه أيضاً أن دارت هي اللغة التي ستكون لنظام فوشا البديل من جوجل. كما يجب التنويه إلى ضرورة تنصيب Android studio في جميع الأحوال للاستفادة من المحاكيات المتضمنة مع حزمته.

المشروع الأول:

1. بعد فتح IntelliJ IDEA نختار Create new project ونختار من الشريط الأيسر الخيار

Flutter كما في الصورة.



تأليف، فيصل الأسود | مهندس برمجيات

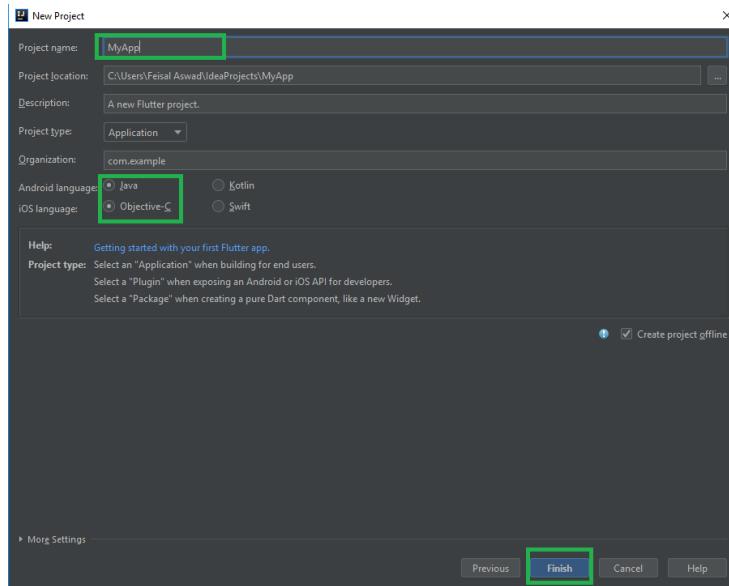
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

٢. نعطي المشروع اسم ومسار محدد ثم نختار أساس عمل النظام ويفضل أن يبقى كما هو محدد في الصورة وثم .Finish



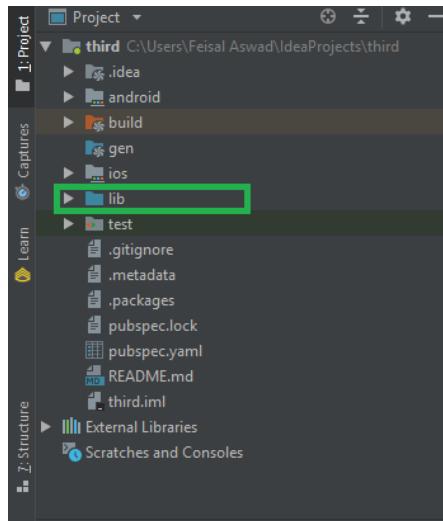
٣. بعد فتح المشروع سنلاحظ قالب لأي مشروع Flutter بالشكل التالي.

```

1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MyApp());
4
5 class MyApp extends StatelessWidget {
6   // This widget is the root of your application.
7   @override
8   Widget build(BuildContext context) {
9     return MaterialApp(
10       title: 'Flutter Demo',
11       theme: ThemeData(
12         // This is the theme of your application.
13         //
14         // Try running your application with "flutter run". You'll see the
15         // application has a blue toolbar. Then, without quitting the app, try
16         // changing the primarySwatch below to Colors.green and then invoke
17         // "hot reload" (press "r" in the console where you ran "flutter run",
18         // or simply save your changes to "hot reload" in a Flutter IDE).
19         // Notice that the counter didn't reset back to zero; the application
20         // is not restarted.
21         primarySwatch: Colors.blue,
22       ),
23       home: MyHomePage(title: 'Flutter Demo Home Page'),
24     );
25   }
26 }

```

٤. في الشريط الأيسر من مجموعة المجلدات والملفات التابعة لمشروع Flutter والتي لن نتطرق لها كثيرا، ما يهمنا هو بعض المجلدات مثل lib الذي يحوي المكتبات في المشروع



التطبيق الأول:

أي تطبيق Flutter يجب أن يحوي الأمور الأساسية التالية:

١. استدعاء المكتبة material.dart والمستخدمة في التصميم.
٢. التابع main لبدء المشروع منه.
٣. التابع runAPP() والذي يعتبر التابع المنشط (المشغل) للتصميم.

```
import 'package:flutter/material.dart';

void main() {
  runApp(
    new Center(
      child: new Text('Hello Stack Overflow Arabi',
          textDirection: TextDirection.ltr),
    )
  );
}
```

هذا أنشأنا برنامجنا الأول والذي يطبع العبارة التالية كما في الصورة.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com



يجب أن نعلم أن تصميم واجهات تطبيقات Flutter يعتمد بشكل أساسي على Material Design المدعوم أيضاً من جوجل ولكي نبدأ تصميم الواجهة سوف نستخدم أصناف classes الخاصة بـ .Material Design

```
import 'package:flutter/material.dart';

void main() {
  runApp(
    new Material(
      color: Colors.greenAccent,
      child: new Center(
        child: new Text('Feisal Aswad',
          textDirection: TextDirection.ltr,
          style: new TextStyle(
            fontSize: 25,
            fontWeight: FontWeight.bold,
            fontStyle: FontStyle.italic
          )),
    ),
  );
}
```

تأليف، فيصل الأسود | مهندس برمجيات

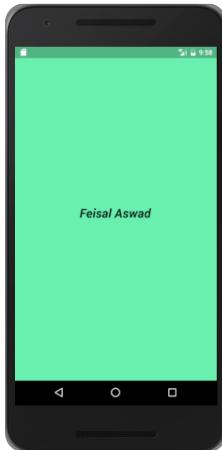
يمكنك الحصول على شرح للكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

والناتج هنا هو عبارة عن خلفية بلون "أخضر" مع كتابة تظهر بالوسط بخط Bold italic كما في الصورة التالية:



كما يمكن عزل التصميم بصنف خاص وهذا ما يسمى بعزل العناصر الثابتة stateless حيث نقوم بوضع كل العناصر الثابتة والتي لا تطلب منها الاستجابة للأحداث أو التفاعل مع المستخدم بعزلها بصنف خاص بها واستدعاءها بالتتابع الأساسي كما في الشكل:

main.dart

```
import 'package:flutter/material.dart';
import 'MyUi.dart';

void main() {
  runApp( new MaterialApp(
    home:MyText(),
  )
); // MaterialAPP
}
```

MyUi.dart

```
import 'package:flutter/material.dart';

class MyText extends StatelessWidget{
@override
Widget build (BuildContext context){
  return new Material(
    color : Colors.green,
    child : new Center(
      child : new Text('Feisal Aswad',
        textDirection: TextDirection.ltr,
        style : new TextStyle(fontSize : 23.0,
        fontWeight : FontWeight.bold) //Text style
    )
  );
}
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```

    ) , //Text
    ) //center
) ; //Material
}
}

```

والناتج سوف يكون مماثل للتطبيق السابق مع الاستفادة من كوننا أصبح لدينا widget جاهزة ويمكن استدعائها أكثر من مرة ويكون التعديل عليها سهل للغاية. كما يجب الانتباه أننا وضعناه داخل المسار lib في ملف جديد اسمه MyUi.dart وهذا ما يفسر الاستدعاء MyUi.dart في البرنامج الرئيسي.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube

[Youtube](#)



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

أنظمة التصميم Layouts

توفر لنا Flutter مجموعة من الحاويات Container وهي أيضا widget ومن الأفضل استخدامها لاحتواء الـ widgets الأخرى فالمستحسن في برنامجنا السابق استبدال الكلمة new Material بـ new container لكي تكون حاوية للنص التي بداخلها وسيبقى البرنامج كما هو بالطبع لأن هنا تمايل Material و يوجد Layouts مختلفة سترها في الدروس اللاحقة.

كما يمكن أن تحوي الـ Layouts أو مخطط التصميم مخططات تصميم أخرى داخلها .Layouts

من أنواع Layouts :

- **التخطيط العمودي Column**

في مثالنا التالي سوف ننشأ Container وبداخلها Column وهو أيضا Layouts وهذا العمود column سيكون له عدة أولاد من نوع text وهكذا نفهم أنه يمكن لأي Layouts يحوي عدد غير محدد من الأبناء.

MyUi.dart

```
import 'package:flutter/material.dart';

class MyColumn extends StatelessWidget{
@override
Widget build (BuildContext context){
return new Container(
color : Colors.green,
child : new Column(
mainAxisAlignment : MainAxisAlignment.center,
children : <Widget>[
new Text('Feisal Aswad',
textDirection : TextDirection.ltr),
new Text('Ahmad Afandi',
textDirection : TextDirection.ltr),
],
),
);
}
}
```

تأليف، فيصل الأسود | مهندس برمجيات

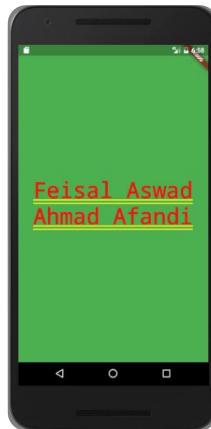
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

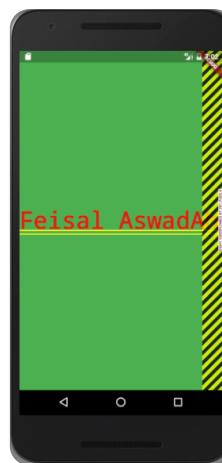
والملف الرئيسي main.dart هو نفسه الملف للبرنامج السابق.



والناتج كما ظاهر بالصورة أعلاه.

- **التخطيط الأفقي (السطري) Row:**

رأينا في التخطيط السابق كيف ترتتب الـ widget عموديا ضمن عمود وماذا لو أردنا ترتيبها أفقيا بجانب بعضها البعض هناك نستخدم الـ Layouts المسماة Row باستبدال الكلمة column في البرنامج السابق بكلمة Row سوف يكون ناتج البرنامج كما يلي: ويجب الانتباه أنه لا يجب أن تمتلئ الشاشة عرضيا بحيث نضع عناصر مناسبة لحجم الشاشة.



ملاحظة: يمكن أن يجعل العرض النص الأخير في التخطيط يأخذ كامل المساحة المتبقية واستدعاء التابع Expanded للعنصر الأخير كما يلي:

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

www.Learn-Barmaga.com : تعلم المزيد عن برمجة التطبيقات



MyUi.dart

```
import 'package:flutter/material.dart';

class MyColumn extends StatelessWidget{
@override
Widget build (BuildContext context){
    return new Container(
        color : Colors.greenAccent,
        child : new Row(
            mainAxisAlignment : MainAxisAlignment.center,
            children : <Widget>[
                new Text('Feisal Aswad',
                    textDirection : TextDirection.ltr),
                Expanded( child: new Text('Ahmad Afandi',
                    textDirection : TextDirection.ltr)
                )
            ],
        ),
    );
}
}
```

يوجد المزيد من الـ Layouts التي ربما ستناولها في كتابنا للحقائق

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكامل محتويات الكتاب على قناته الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

الـ Widget المقدمة لـ Flutter

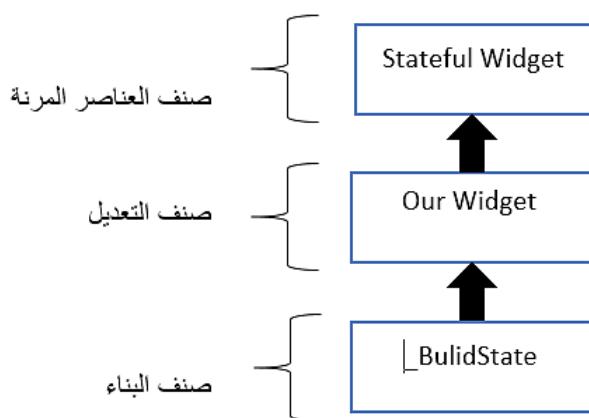
هي مجموعة من العناصر المقدمة مع حزمة Flutter sdk الأساسية.

حيث تقدم Flutter نوعين من العناصر الثابتة والمرنة وهي:

- Stateless widget
- Stateful widget

سنبدأ مع العنصر Stateful كما سيمر معنا في المثال التالي.

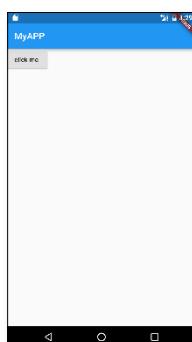
يجب الانتباه إلى فكرة أن إنشاء زر مرن يمر بمراحلتين أو يتكون من صنيفين الأول للبناء والثاني للتعديل كل مرة.



الآن سنقوم بعرض أهم Widget المقدمة مع حزمة Flutter

- Raised Button

الكود التالي هو لإنشاء زر كما بینا في المخطط السابق:



تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

main.dart

```
import 'package:flutter/material.dart';
void main(){
  runApp(new MaterialApp(
    home: new ourWidget(),
  ));
}
```

ourWidgetFile.dart

```
class ourWidget extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _Buildstate();
  }

}
class _Buildstate extends State<ourWidget>
{
  String names= "";

  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      appBar : new AppBar(
        backgroundColor: Colors.blue,
        title: new Text ("MyAPP"),
      ),
      body: new Container(
        child: new RaisedButton(
          onPressed: () => clickfunc('Hi'),
          child: new Text ('click me ${names}')
        ),
      ),
    );
  }
  void clickfunc(String txt){
    setState((){
      names = txt;
    });
  }
}
```

في الكود السابق أنشأنا زر داخل تابع البناء وعند كل استدعاء لتابع التعديل سوف يعيد بناءه وهكذا يكون الزر حرف. نلاحظ أن الزر يتطلب التابع clickfunc عن طريق تابع مج هو لم يمرر له نص معين كما أن التابع يعرف في نهاية الصنف.



- الزر المسطح FlatButton: هو زر يأخذ شكل نص عادي فقط، كما يبين الشكل التالي الزر من نوع

.FlatButton



ويمكن تعريف كما في الكود التالي:

```
ourWidgetFile.dart
class ourWidget extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _Buildstate();
  }
}

class _Buildstate extends State<ourWidget>
{
  String names= '';
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      appBar : new AppBar(
        backgroundColor: Colors.blue,
        title: new Text ("MyAPP"),
      ),
      body: new Container(
        child: new FlatButton(
          onPressed: () => clickfunc('Hi'),
          child: new Text ('click me ${names}')
        )
      ),
    );
  }

  void clickfunc(String txt){
    setState(() {
      names = txt;
    });
  }
}
```

هنا الاستدعاء عن طريق التابع main كما في المثال السابق.

تأليف: فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube

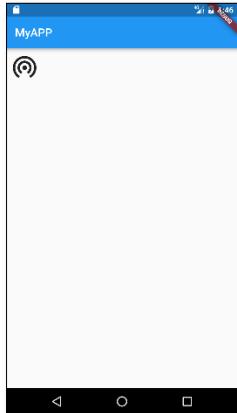


StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

- الزر الأيقونة :`IconButton`

هو زر يحوي داخله صورة على شكل أيقونة كما في الشكل:



يمكن تعريفه كما يلي:

ourWidgetFile.dart

```
class ourWidget extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _Buildstate();
  }
}

class _Buildstate extends State<ourWidget>
{
  String names= '';
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      appBar : new AppBar(
        backgroundColor: Colors.blue,
        title: new Text ("MyAPP"),
      ),
      body: new Container(
        child: new IconButton(
          onPressed: () => clickfunc('Hi'),
          icon: Icon(Icons.wifi_tethering,size: 50,),
        )
      ),
    );
  }

  void clickfunc(String txt){
    setState((){
      names = txt;
    });
  }
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قناة الـ [Youtube](#)



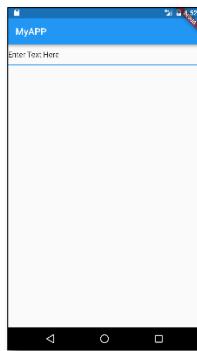
StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

حيث تتيح لنا فلاتر العديد من الرموز والأشكال الجاهزة.

- نص الإدخال `TextField`

هو حقل نصي يسمح للمستخدم بإدخال نص معين مثل كلمة مرور أو اسم مستخدم كما هو مبين في الشكل:



ويمكن تعريفه كما في الكود التالي

```
ourWidgetFile.dart
class ourWidget extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _Buildstate();
  }
}

class _Buildstate extends State<ourWidget>
{
  String names= '';
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      appBar : new AppBar(
        backgroundColor: Colors.blue,
        title: new Text ("MyAPP"),
      ),
      body: new Container(
        child: new TextField(
        )
      ),
    );
  }
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

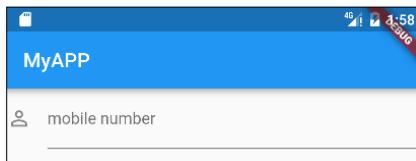
تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

كما يحوي الـ `TextField` بعض الخصائص الهامة نذكر منها:

الخاصية	الشرح
Auto correct : true	تجعل الحقل النصي يصحح الأخطاء اللغوية تلقائياً
keyboardType: TextInputType	تسمح لنا بتحديد المدخلات (رقمي - ايميل) وتساعد في عملية اختيار الدخل
decoration	تسمح لك بتشكيل صنف لإظهار الحقل النصي بعدة أشكال

مثال تطبيقي:

في مثالنا التالي سوف نقوم بتصميم واجهة تحوي حقل ادخال له العنوان mobile number ويجانبه صورة المستخدم كما أن نوع المدخلات هي من نوع رقم هاتف



```
ourWidgetFile.dart
class ourWidget extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _Buildstate();
  }
}
class _Buildstate extends State<ourWidget>
{
  String names= '';
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      appBar : new AppBar(
        backgroundColor: Colors.blue,
        title: new Text ("MyAPP"),
      ),
      body: new Container(
        child: new TextField(
          autocorrect: true,
          keyboardType: TextInputType.phone,
          decoration: new InputDecoration(
            icon: Icon(Icons.perm_identity),
            labelText: 'mobile number',
            hintText: 'Enter Here',
          ), // Input Decoration
        ) // text field
      ),
    );
}
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكامل محتويات الكتاب على قناته الـ Youtube

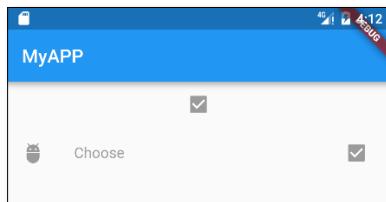


StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

- مربع الاختيار Checkbox

جميعنا نعرف مربع الاختيار المتعدد الذي يسمح لنا اختيار بعض العناصر وترك أخرى كما هو مبين في الصورة.



ourWidgetFile.dart

```
class ourWidget extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _Buildstate();
  }

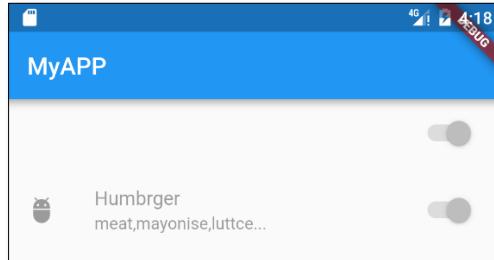
}

class _Buildstate extends State<ourWidget>
{
  String names= '';
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      appBar : new AppBar(
        backgroundColor: Colors.blue,
        title: new Text ("MyAPP"),
      ),
      body: new Container(
        child: new Column(
          children: <Widget>[
            new Checkbox(
              value: true,
              onChanged: null,
              activeColor: Colors.greenAccent,),
            new CheckboxListTile(value: true,
              onChanged: null,
              title: new Text("Choose"),
              secondary: new Icon(Icons.adb),)
          ],
        )
      ),
    );
}
}
```

هنا قمنا ببناء عنصر من نوع CheckBox و مربع الاختيار المطورو CheckBoxListTile

• مبدل الحالة :switch

هو مشابه تماماً لـ checkbox مع اختلاف في الشكل، الصورة التالية توضح العنصر .switch



والكود المطلوب لتعريف switch كما يلي:

ourWidgetFile.dart

```
class ourWidget extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _Buildstate();
  }
}

class _Buildstate extends State<ourWidget>
{
  String names= '';
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      appBar : new AppBar(
        backgroundColor: Colors.blue,
        title: new Text ("MyAPP"),
      ),
      body: new Container(
        child: new Column(
          children: <Widget>[
            new SwitchListTile(
              value: true,
              onChanged: null,
              activeColor: Colors.greenAccent,),
            new SwitchListTile(value: true,
              onChanged: null,
              title: new Text("Humburger"),
              secondary: new Icon(Icons.adb),subtitle: new
Text("meat,mayonise,luttce..."),)
          ],
        )
      );
}
}
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

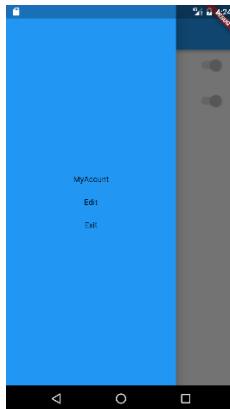


StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

- **الشريط الجانبي Drawer**

هو الشريط المستخدم غالباً لإظهار خيارات المستخدم وملفه الشخصي ويكون موجود في الصفحة الرئيسية للبرنامج. الصورة التالية تبين عنصر Drawer يحوي داخله بعض العناصر.



لإنشاء عنصر من نوع Drawer نكتب الكود التالي:

```
ourWidgetFile.dart
class ourWidget extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _Buildstate();
  }
}

class _Buildstate extends State<ourWidget>
{
  String names= '';
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      drawer: new Drawer(
        child: new Container(
          color: Colors.blue,
          padding: EdgeInsets.all(12.0),
          child: new Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Padding(padding: EdgeInsets.only(bottom: 25)),
              new Text ('MyAccount'),
              Padding(padding: EdgeInsets.only(bottom: 25)),
              new Text ('Edit'),
              Padding(padding: EdgeInsets.only(bottom: 25)),
              new Text ('Exit'),
            ],
          ),
        ),
      ),
      appBar : new AppBar(
        backgroundColor: Colors.blue,
        title: new Text ("MyAPP"),
      ),
      body: new Container(
      ),
    );
  }
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

مربع الحوار (Alert Dialog) Notification Dialog

هو مربع رسالة لتنبيه المستخدم لأجراء معين مع إمكانية إعطائه عدة خيارات.



يجب أن تعلم أن مربع الحوار لا يستخدم إلا عند اعلام شيء ما أو سماحية المستخدم للتخاذل القرارات، ومنه هذا المربع يستدعي دوماً عن طريق تابع أو طريقة ما.

ourWidgetFile.dart

```
import 'package:flutter/material.dart';
import 'dart:io';
class ourWidget extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _Buildstate();
  }
}
class _Buildstate extends State<ourWidget>
{
  String names= '';
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      appBar : new AppBar(
        backgroundColor: Colors.blue,
        title: new Text ("MyAPP"),
      ),
      body: new Container(
        child: new FlatButton(onPressed: (){DialogBuild(context);},
      child: new Text("Exit"))
      ),
    );
}
```

ننوه ان التابع exit(0) للخروج من البرنامج موجود في المكتبة .dart:io

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

بقي لدينا انشاء الدالة DialogBuild(context) والتي تقوم بناء مربع التنبية عند طلبها وإظهاره.

ourWidgetFile.dart

```
Future<Null> DialogBuild(BuildContext context) async {
  return showDialog<Null>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return new AlertDialog(
        title: new Text("Exit"),
        content: new SingleChildScrollView(
          child: new ListBody(
            children: <Widget>[
              new Text("Do you want to Exit"),
            ],
          ),
        ),
        actions: <Widget>[
          new FlatButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            child: new Text("No")),
          new FlatButton(
            onPressed: () {
              exit(0);
            },
            child: new Text("Yes")),
        ],
      );
    });
}
```

• مربع الحوار البسيط :singleDialog

هو رسالة مشبهة لـ مربع الحوار العادي إلا أنها تقدم مجموعة من الخيارات لتساعد المستخدم على الاختيار ولا تتطلب تعقيد برمجي.



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

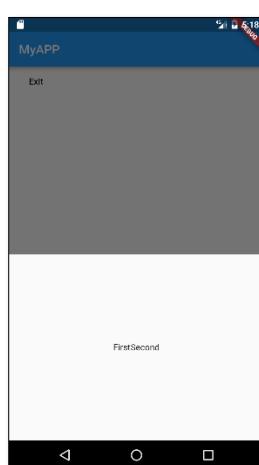
والكود التالي لإنشاء تابع (دالة) يبني مربع الحوار البسيط

ourWidgetFile.dart

```
Future simpleShowSimpleDialog(BuildContext context) async {
  switch (await showDialog(context: context, builder: (BuildContext context) {
    return new SimpleDialog(
      title: new Text ('options'),
      children: <Widget>[
        new SimpleDialogOption (child: new Text ('No'),
          onPressed: () {
            Navigator.pop(context);
          },
        new SimpleDialogOption (child: new Text('Yes'),
          onPressed: () {
            exit(0);
          })
      ]
    );
  }));
}
```

• القائمة السفلية :Bottom sheet

تشبه الـ Drawer إلا أنها تظهر في الأسفل :



تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكامل محتويات الكتاب على قناة الـ Youtube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

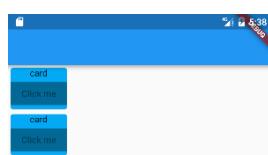
والكود التالي لإنشاء قائمة من نوع BottomSheet ، كما أنها تشكل ضمن تابع مستقل أيضاً وتنسق عند الطلب.

ourWidgetFile.dart

```
Bottomsheet open(BuildContext context) async{
  await showModalBottomSheet(context: context,
    builder: (BuildContext context) {
      return new Container(
        height: 300,
        child: new Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            new Text ('First'),
            new Text('Second'),
          ],
        ),
      );
    });
}
```

• الـ card layout :

هو إحدى العناصر المهمة في Flutter لكونه يساعدنا على إنشاء مثل البطاقات التي تحوي تفاصيل عن معلومات عدة أنواع.



من المؤكد أن البطاقات يمكن ان تأخذ كامل عرض الشاشة.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

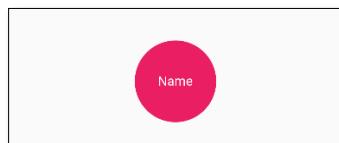
يمكن إنشاء مثل هذا النوع من الـ layout كما في الكود التالي:

ourWidgetFile.dart

```
class _Buildstate extends State<ourWidget> {
  String names = '';
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      appBar: new AppBar(),
      body: new Container(
        child: new Column(
          children: <Widget>[
            new Card(
              color: Colors.amber,
              child: new Column(
                children: <Widget>[
                  new Text('card'),
                  new RaisedButton(onPressed: null, child: new Text("Click me")),
                ],
              ),
            ),
            new Card(
              color: Colors.amber,
              child: new Column(
                children: <Widget>[
                  new Text('card'),
                  new RaisedButton(onPressed: null, child: new Text("Click me")),
                ],
              ),
            ),
            new Card(
              color: Colors.amber,
              child: new Column(
                children: <Widget>[
                  new Text('card'),
                  new RaisedButton(onPressed: null, child: new Text("Click me")),
                ],
              ),
            ),
          ],
        ),
      );
    }
}
```

- دائرة المستخدم Circle Avatar

هي غالباً دائرة تستخدم لوضع صورة المستخدم مع معلوماته أو صورة أي عنصر مع معلوماته كما في الصورة.



تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

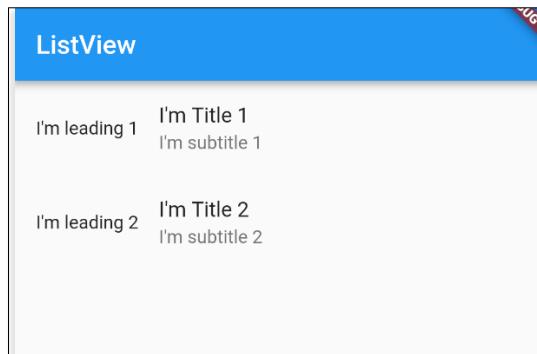
```

        child: new CircleAvatar(
            child: new Text('Name'),
            backgroundColor: Colors.pink,
            foregroundColor: Colors.white,
            radius: 50,
        ),
    ),

```

• العنصر :List View

هذا العنصر يستخدم لعرض مجموعة من العناصر على شكل قائمة بعده لا نهائي من السجلات ويمكننا الاطلاع على السجلات بالتحريك للأعلى والأسفل كما في الصورة.



كل عنصر في هذه القائمة يسمى ListTile ويأخذ شكل معين حسب تصميمه، غالبا له الشكل التالي:



والكود التالي لإنشاء ListView:

```

child:new ListView(
    children:<Widget>[new ListTile(
        title: new Text("I'm Title 1"),
        subtitle: new Text("I'm subtitle 1"),
        leading: new Text("I'm leading 1")
    ) ,
    new ListTile(
        title: new Text("I'm Title 2"),
        subtitle: new Text("I'm subtitle 2"),
        leading: new Text("I'm leading 2")
    )],// ListTile
)

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



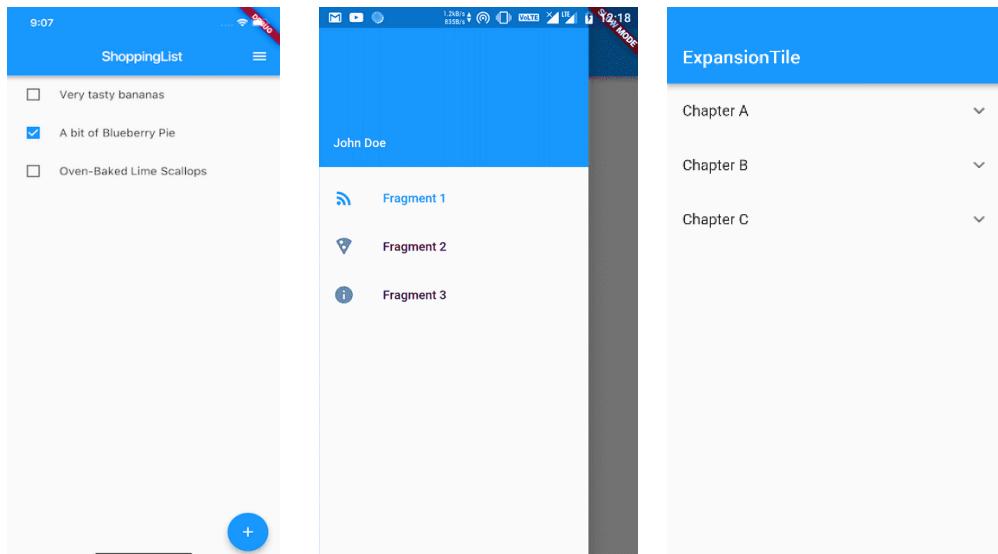
StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

Scaffold القوالب الجاهزة

هي widgets جاهزة تقدمها Flutter تساعد في بناء واجهات قوية دون الخوض في عناوين تصميم الواجهات.

الصور التالية تبين مجموعة قوالب جاهزة



:AppBar •

المثال التالي لصنف مستخدم scaffold القالب

MyUi.dart

```
import 'package:flutter/material.dart';

class MyColumn extends StatelessWidget{
@override
Widget build (BuildContext context){
return new Scaffold(
appBar :new AppBar(
title: new Text('AppBar Title'),
backgroundColor : Colors.greenAccent,
actions : <Widget>[
new IconButton(icon :new Icon(Icons.adb),
onPressed : (){}),
new IconButton(icon:new Icon(Icons.verified_user),
onPressed: (){}),
],
)
);
}
}
```

ويكون ناتج البرنامج عبارة عن تطبيق يستخدم الـ widget AppBar. التي تظهر شريط في أعلى البرنامج مع مجموعة اوامر معينة على شكل أيقونات كما في الشكل التالي.

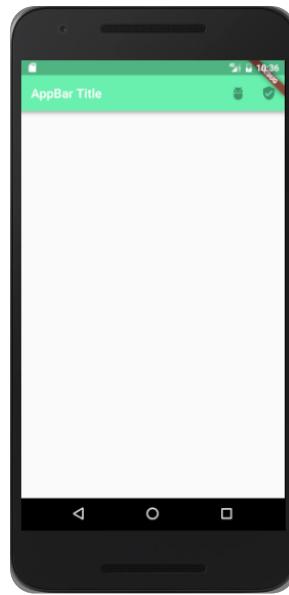
تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com



كما يمكن إضافة أحداث لتلك الأيقونات عن طريق تعريف تابع داخل الصنف واستدعاء هذا التابع بالشكل التالي:

```
new IconButton(icon: new Icon(Icon.alarm),
onPressed: myFunction),
```

وهنا نحتاج إلى دالة `myFunction` ضمن الصنف لتنفيذ الأمر المطلوب ويمكن إنشاء دالة مجهرولة مباشرة كما يلي:

```
new IconButton(icon: new Icon(Icon.alarm),
onPressed: () { //write code here}),
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على [Youtube](#)



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

كما يمكنك وضع widgets داخل جسم Scaffold اى Body وهذا يمكن وضع أجزاء جديدة في الواجهة في الكود التالي جزء من كود الـ widget داخل الـ Scaffold وبالتحديد خاصة Body ثم إنشاء نص من نوع معين InkWell وهو نص عادي قابل للستجابة للأحداث وقد تم ربطه بحيث عند الضغط عليه ينفذ تابع معين.

MyUi.dart

```
import 'package:flutter/material.dart';

class MyColumn extends StatelessWidget{
@override
Widget build (BuildContext context){
    return new Scaffold(
        body: new Container(
            alignment : Alignment.center,
            child : new Column (
                mainAxisAlignment : MainAxisAlignment.center,
                children:<Widget>[
                    new InkWell(
                        child: new Text ('Press Me'),
                        onTap: (){print('Hello')};,
                    ),
                ],
            ),
        ),
        appBar :new AppBar(
            title: new Text('AppBar Title'),
            backgroundColor : Colors.greenAccent,
            actions : <Widget>[
                new IconButton(icon :new Icon(Icons.adb),
                onPressed : null ),
                new IconButton(icon:new Icon(Icons.verified_user),
                onPressed:null),
            ],
        )
    );
}
}
```





• شريط التصفح :Navigation Bar

يمكن إنشاء Bottom Navigation Bar بسهولة عن طريق scaffold كما في الشكل واعطاء كل زر حدث يستجيب له.



MyUi.dart

```
import 'package:flutter/material.dart';

class MyColumn extends StatelessWidget{
@override
Widget build (BuildContext context){
return new Scaffold(
bottomNavigationBar:new BottomNavigationBar(items:[
new BottomNavigationBarItem(icon: new
Icon(Icons.access_alarms), title: new Text('Bus')) ,
new BottomNavigationBarItem(icon: new
Icon(Icons.ac_unit), title: new Text('adb')) ],
onTap:(int x) => print('index $x'),
type: BottomNavigationBarType.fixed,),

appBar :new AppBar(
title: new Text('AppBar Title'),
backgroundColor : Colors.greenAccent,
actions : <Widget>[
new IconButton(icon :new Icon(Icons.adb),
onPressed : null ), // Icon1
new IconButton(icon: new Icon(Icons.verified_user),
onPressed:null),
],
)
);
}
}
```

تأليف، فيصل الأسود | مهندس برمجيات

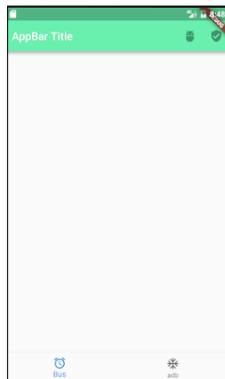
يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflowArabi

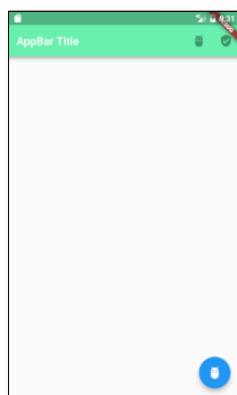
تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

والناتج يكون:



- **Floating Button**

هو زر يظهر في زاوية الشاشة وغالباً يساعد في العمليات التكرارية كما في الصورة الكود التالي يبين طريقة إنشاء "زر عائم" كما يلي مع إعطائه استجابة لحدث معين:



هنا عند الضغط على الزر سوف يطبع في الـ `console` عبارة `I am float button`

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته الـ `Youtube`



StackOverflowArabi

www.Learn-Barmaga.com : تعلم المزيد عن برمجة التطبيقات

كما في الكود التالي:

MyUi.dart

```
import 'package:flutter/material.dart';

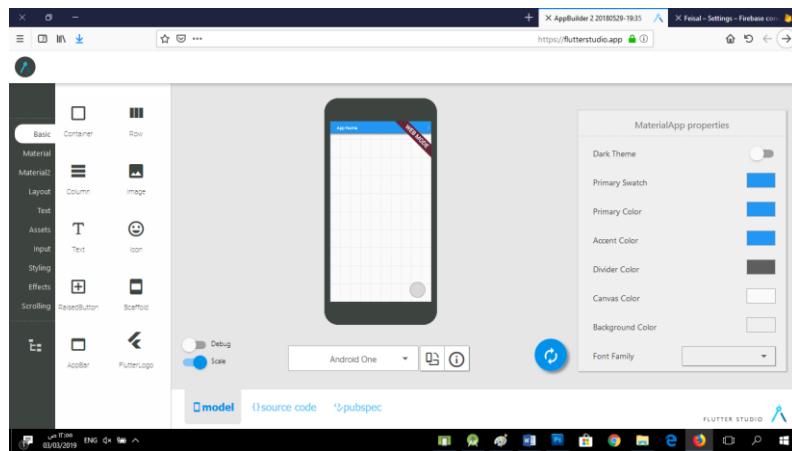
class MyColumn extends StatelessWidget{
@override
Widget build (BuildContext context){
    return new Scaffold(
        floatingActionButton: new FloatingActionButton(
            onPressed :(){print('I am float button');},
            child : new Icon(Icons.adb),
            backgroundColor: Colors.blue
        ), // floatingActionButton

        appBar :new AppBar(
            title: new Text('AppBar Title'),
            backgroundColor : Colors.greenAccent,
            actions : <Widget>[
                new IconButton(icon :new Icon(Icons.adb),
                onPressed : null ), // Icon
                new IconButton(icon:new Icon(Icons.verified_user),
                onPressed:null),
            ],
        )
    );
}
}
```

• التصميم Online

تتيح لنا منصة Flutter Studio على الانترنت تصميم واجهات عن طريق أدوات وازرار يمكننا الوصول لتلك المنصة عن طريق الرابط:

<https://www.flutterstudio.app>



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـة الـYoutube

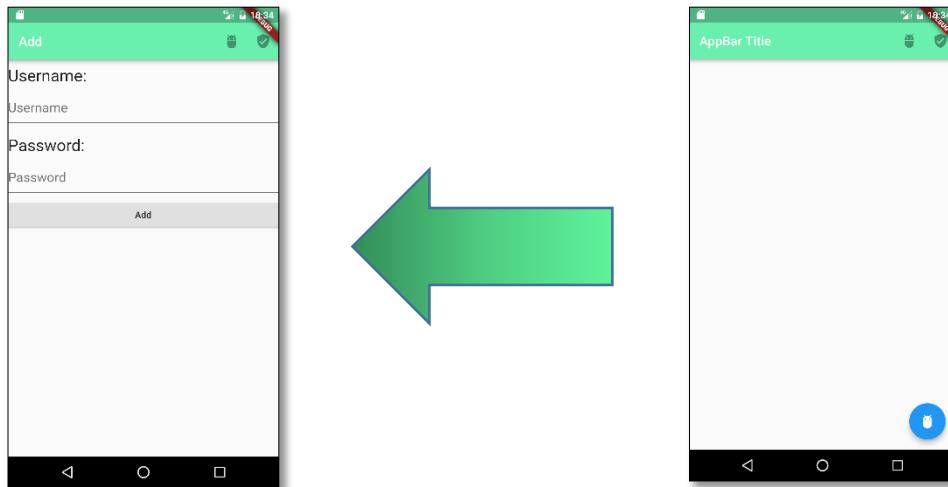


StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

التصفح والتوجيه Navigation

في هذا القسم من الكتاب سوف نركز على كيفية التنقل بين الصفحات أو الواجهات وهذا ما يسمى بمفهوم الـ **Navigation**.



في مثالنا التالي لدينا واجهتين سوف ننتقل من الواجهة الأولى (الأساسية) إلى الواجهة الثانية باستخدام مفاهيم التصفح.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكتاب على قناته على [Youtube](#)



StackOverflow Arabi

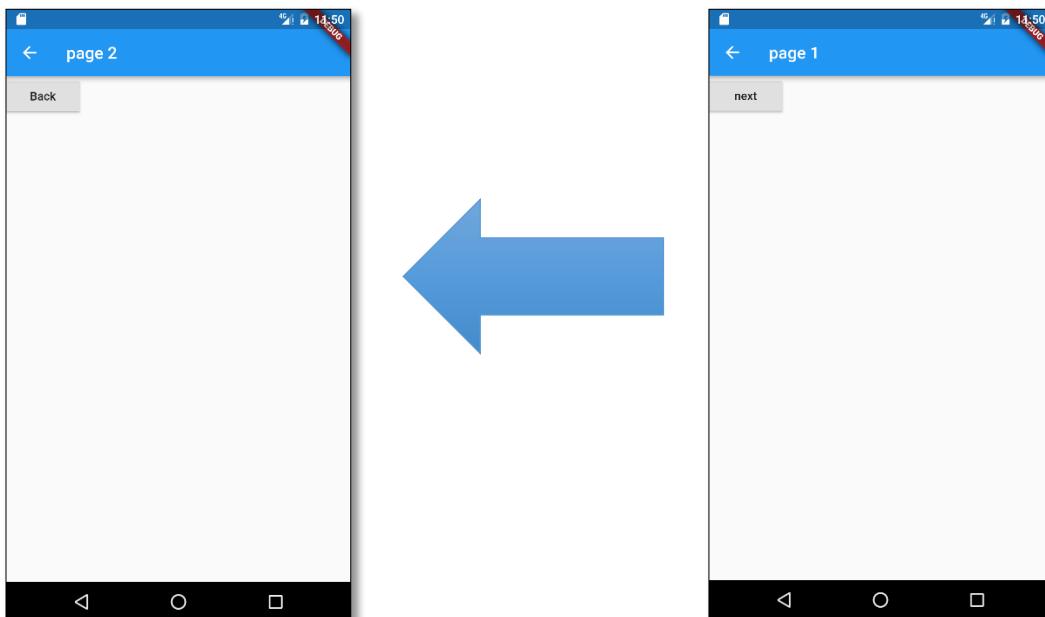
تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

بالبداية سوف نشكل صنف class لكل واجهة ونبني ضمنها التصميم الصحيح ، ثم نكتب التابع main كما يلي مستعينين بصنف ثالث يحدد لنا المسارات:

main.dart

```
import 'package:flutter/material.dart';
import 'package:navigationProject/PageOneFile.dart';
import 'package:navigationProject/PageTwoFile.dart';
void main(){
  runApp(new MaterialApp(
    home: new MyApp()
  );
}
class MyApp extends StatefulWidget{

  @override
  State<StatefulWidget> createState() {
// TODO: implement createState
    return new _MyApp();
  }
}
class _MyApp extends State<MyApp>{
  @override
  Widget build(BuildContext context) {
// TODO: implement build
    return new MaterialApp(
      title: 'Navigation',
      routes: <String, WidgetBuilder>{
        '/First': (BuildContext context) => new PageOne(),
        '/Second': (BuildContext context) => new PageTwo(),
        home: new PageOne();
    } //MaterialApp
  }
}
```



تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

ونكتب كود الصنفين بحيث الأول يطلب الثاني والثاني يطلب الأول

الصنف الأول:

PageOneFile.dart

```
import 'package:flutter/material.dart';
class PageOne extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _PageOne();
  }
}

class _PageOne extends State<PageOne> {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      appBar: new AppBar(
        title: new Text("page 1"),
      ),
      body: new Container(
        child: new Column(
          children: <Widget>[
            new RaisedButton(
              onPressed: () {
                Navigator.of(context).pushNamed('/Second');
              },
              child: new Text('next')
            ),
          ],
        ),
      );
    }
}
```

والصنف الثاني:

PageOneFile.dart

```
import 'package:flutter/material.dart';
class PageTwo extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _PageTwo();
  }
}

class _PageTwo extends State<PageTwo> {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      appBar: new AppBar(

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```

        title: new Text("page 2"),
),
body: new Container(
child: new Column(
children: <Widget>[
new RaisedButton(
onPressed: () {
Navigator.of(context).pushNamed('/First');
},
child: new Text('Back'))
],
),
),
);
}
}
}

```

نلاحظ بالخاصية routes بنينا موجهاً للصفحات وكما في الخاصة home وضمنها الصفحة `PageOne` كصفحة أولى.

نلاحظ عند الانتقال للصفحة الثانية ظهور زر الرجوع أتوماتيكياً وهذا ما يوفره الـ Scaffold بالطريقة `.PushNamed`.



يمكن عدم إظهار هذا الزر باستخدام طريقة تصفح أخرى هي كما في الكود التالي:

```

 onPressed: () {
Navigator.of(context).
pushNamedAndRemoveUntil('/Second', (Route<dynamic> route) => false);
},
}

```

وهنا عند الذهاب من الصفحة الأولى إلى الصفحة الثانية لن يظهر زر الرجوع أما في باقي حالات التصفح فسوف يظهر تلقائياً.

هناك أيضاً طريقة للرجوع للصفحة السابقة كما يلي:

```

 onPressed: () {
Navigator.of(context).pop();
},
}

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـ كامل محتويات الكتاب على قناته الـ [Youtube](#)



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

وهي تنبء تماماً عن زر الرجوع في حال كانت إمكانية الرجوع مقبولة أي أرسلت بالطريقة `PushNamed` للتحقق من هذا يجب كتابة الكود السابق بالشكل:

```
onPressed: () {
  if (Navigator.of(context).canPop())
  {
    Navigator.of(context).pop();
  }
}
```

ملاحظة: تستخدم عملية من الرجوع كما في حالات صفحة الـ `Login` حيث لا يسمح للمستخدم بالرجوع إليها مرة أخرى.

تبادل القيم والمتغيرات بين الصفحات:

تبادل القيم والمتغيرات بين الصفحات:

رأينا في الفقرة السابقة كيف يمكن الانتقال بين الصفحات ولكن ماذا لو أردنا أن نرسل قيم من صفحة إلى أخرى.

هناك طريقتين للقيام بذلك:

- الطريقة الأولى: إرسال القيم عن طريق الباقي للصفحة `construcrter`

في البداية يجب استخدام التابع التالي للتنقل كما في الكود:

```
toPageTwo(BuildContext context) {
  Navigator.push(context, new MaterialPageRoute(builder:
    (BuildContext context)
      => new PageTwo("hello")
  ));
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ `Youtube`



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

ويكون شكل التابع للصفحة الأولى كما في الكود التالي:

PageOneFile.dart

```
import 'package:firebase_tutorial/PageTwoFile.dart';
import 'package:flutter/material.dart';

class PageOne extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _PageOne();
  }
}

class _PageOne extends State<PageOne> {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
      appBar: new AppBar(
        title: new Text("page 1"),
      ),
      body: new Container(
        child: new Column(
          children: <Widget>[
            new RaisedButton(
              /*onPressed: () {
                Navigator.of(context).pushNamed('/Second');
              },*/
              /*onPressed: () {
                Navigator.of(context).
                pushNamedAndRemoveUntil('/Second', (Route<dynamic>
route) => false);
              },*/
              onPressed: () {
                toPageTwo(context);
              },
            ),
            child: new Text('next')
          ],
        ),
      );
  }

  toPageTwo(BuildContext context) {
    Navigator.push(context, new MaterialPageRoute(builder:
      (BuildContext context)
      => new PageTwo("hello")
    )));
  }
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

هنا أرسلنا العبارة "hello" من الصفحة الأولى إلى الصفحة الثانية وعلينا الآن أن نجهز الصفحة الثانية لاستقبال البيانات.

وهنا عند الضغط على زر معين من الصفحة الأولى سوف تنفذ الطريقة `toPageTwo` ومنه سوف ترسل العبارة "hello" وتقوم الصفحة `second` باستقبالها عن طريق الباني.

ونكتب الصنف الثاني كما في الكود التالي:

PageTwoFile.dart

```
import 'package:flutter/material.dart';

class PageTwo extends StatefulWidget {
    String Message;
    PageTwo(this.Message);
    @override
    State<StatefulWidget> createState() {
        // TODO: implement createState
        return new _PageTwo(Message);
    }
}

class _PageTwo extends State<PageTwo> {
    String Message;
    _PageTwo(this.Message);
    @override
    Widget build(BuildContext context) {
        // TODO: implement build
        return new Scaffold(
            appBar: new AppBar(
                title: new Text("page 2"),
            ),
            body: new Container(
                child: new Column(
                    children: <Widget>[
                        new RaisedButton(
                            onPressed: () {
                                Navigator.of(context).pushNamed('/First');
                            },
                            child: new Text('Back')
                        ),
                    ],
                ),
            );
    }
}
```

هنا اعتبرنا ان الصفحة الأولى فقط ترسل للصفحة الثانية وإذا اردنا أن نرسل من الثانية للأولى نشكل نفس التابع في الثانية وكذلك ننشئ باني في الأولى لاستقبال البيانات كما فعلنا مسبقاً.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ [Youtube](#)



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

- الطريقة الثانية: نلخص الطريقة الثانية بإنشاء صنف خاص يحوي متحولات ستاتيكية عامة هذا المتحولات يمكن تخزين البيانات فيها من قبل الصفحة المرسلة وتقوم الصفحة المستقبلة باستدعاء هذه البيانات من الصنف وذلك عن طريق مفتاحها الخاص .key

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

التعامل مع Json

مقدمة:

في البداية دعونا نشرح Json وما الفائدة منها، تعد Json هي الوسيلة الأفضل للحصول من البيانات من المخازن الالكترونية أو مواقع الويب. بحيث علينا طلبها عن طريق رابط يدعى API خاص لهذه البيانات فيعود لي البيانات مكتوبة بلغة Json.

الحقيقة إن Json ليست لغة برمجة إنما هي لغة هيكلة بيانات مثل XML ولكن تساعده جداً في فهم كيفية استقبال البيانات من أي موقع.

كما يجدر الذكر أن هناك العديد من المواقع التي تقدم بيانات جاهزة تساعده المبرمجين في بناء التطبيقات منها بيانات عن أسعار العملات أو بيانات الطقس.

يجب أن نعلم أن طلب هذه الخدمة عن طريق API الخاص بها يسمح لنا فقط باستعراضها وعرضها ضمن تطبيقنا، حيث أي ملف Json يكتب على شكل كائنات وخصائص لكل خاصية أو كائن مفتوح (اسم معين) وقيمة.

لدينا هنا ملف Json يحوي بيانات مستخدمين حيث لكل مستخدم رقم معرف id، اسم name، العنوان Address، الآباء parents.

JSON File: data.json
[{ id: 1, name: "Ahmad", Address: "Syria", Parents: { father: "Mohamad", mother: "Nour" } }, { id: 2, Name: "Farah", Address: "Damascus", Parents:{ Father = "Feisal", Mother = "Batoul", } }]

يمكن الوصول للمعلومات بالشكل التالي :

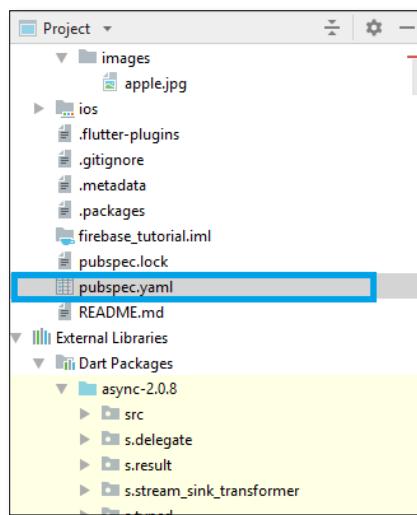
```
data [0] ['Name'] = "Ahmad"
data [1] ['Parents'] ['Father'] = " Feisal ",
```

حيث data ملف الـ Json الذي لدينا .

الآن لنعد إلى Flutter ونتعلم كيفية جلب مثل تلك البيانات وعرضها على تطبيقنا.

بالبداية يجب جلب مكتبة الـ http التي تتعامل مع json الى مشروعنا كما يلي:

- نفتح الملف pubspec.yaml من ضمن ملفات المشروع.



- نضيف فقط السطر `http: ^0.12.0+1` كما في الصورة التالية :

```
dependencies:
  flutter:
    sdk: flutter
  firebase_database: ^2.0.1
  http: ^0.12.0+1
```

- نضغط Package get لاستدعاء المكتبة من الانترنت:

[Packages get](#) [Packages upgrade](#) | [Flutter upgrade](#) | [Flutter doctor](#)

وننتظر حتى اكتمال التحميل.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته الـ Youtube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

الطريقة التالية مهمتها جلب البيانات بطريقة Json إلى تطبيقنا:

```
Future<List> getData() async{
  String url = "our API ";
  http.Response r= await http.get(url);
  return json.decode(r.body);
}
```

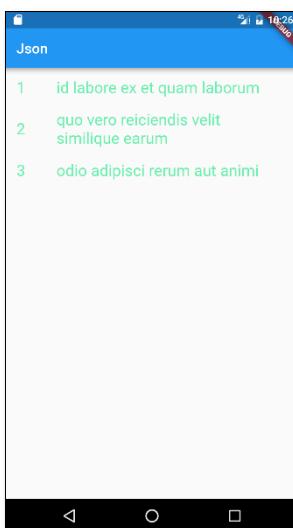
نلاحظ أنها متزامنة إنها تنتظر قدوم البيانات من الشبكة.

كما يجب لتطبيق تلك الطريقة استخدام المكتبات:

```
import 'dart:convert';
import 'package:http/http.dart' as http;
```

مثال:

ليكن لدينا API تعيد لي الكثير من البيانات وأريد وصفها ضمن قائمة.
سنبدأ الآن بקוד كامل لحل تلك المشكلة وتظهره لدى البيانات كما في الصورة.



```
import 'dart:async';
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:flutter/material.dart';

void main() async{
  List data = await getData();
  runApp(new MaterialApp(
    title: 'Json Example',
    home:new Scaffold(
      appBar: AppBar(
        title: new Text ('Json')
      ),
      body: new Center(
        child:new ListView.builder(

```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```

        itemCount: data.length,
        itemBuilder: (BuildContext context, int Position){
      return new ListTile(
        title: new Text("${data[Position]['name']}"), style: TextStyle(
          color: Colors.greenAccent,
          fontSize: 22,
        ),
        leading: new Text("${data[Position]['id']}"), style: TextStyle(
          color: Colors.greenAccent,
          fontSize: 22,
        ),
      );
    },
  );
}
)
)
)
)
)
);

}

Future <List> getData() async{
  String url = 'https://jsonplaceholder.typicode.com/comments';

  http.Response r= await http.get(url);
  return json.decode(r.body);
}

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

استدامة البيانات

توفر فلاتر أكثر من طريقة لحفظ البيانات الخاصة بالتطبيقات أهم هذه الطرق تشمل ما يلي:

- الذاكرة الداخلية (Internal Storage):** وفيها يتم تخزين البيانات في الذاكرة الداخلية للتطبيق.
- الذاكرة الخارجية (External Storage):** وفيها يتم تخزين البيانات في الذاكرة الخارجية العامة بحيث أي تطبيق يمكن الوصول لها

التفضيلات المشتركة (Shared Preferences): تشمل حفظ بيانات أساسية بمفاتيح محددة (Key-Value pairs).

قواعد البيانات (SQLite Databases): وفيها يتم تخزين البيانات في قواعد بيانات خاصة بالتطبيق.

استخدام أي من هذه الطرق يعتمد على احتياجاتك، وكذلك المساحة المطلوبة لتخزين البيانات وسيتم تاليًا تفصيل كل طريقة على حدى.

- **الذاكرة الداخلية (internal Storage):** يمكن تخزين البيانات مباشرة في الذاكرة الداخلية للجهاز، وتكون البيانات مخزنة خاصة بالتطبيق الذي تم التخزين من خلاله ولا يمكن للتطبيقات الأخرى الوصول لهذه البيانات. وعند حذف التطبيق تمحى معه ملفاته وبياناته مباشرة.
- لإنشاء وتخزين واسترجاع بيانات تطبيق الذاكرة نستخدم الطريقتين الرئيسيتين التاليتين.

يجب أن ننوه بالبداية أنه كي نستطيع استخدام التخزين يجب أن نضيف حزمة `io` إلى مشروعنا بحيث نضيف في الملف `Pubspec.yaml` وتحت السطر `Cupertino_icons` سطر جديد يحتوي على `Path_Provider: ^0.4.1`.

وأيضاً نضيف المكتبة `io` للمشروع كما يلي:

```
import 'package:flutter/material.dart';
import 'package: Path_Provider/Path_provider.dart';
import 'dart:io';
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ [Youtube](#)



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

التابع أو الطريقة (method) الأولى المستخدم هي AppPath والتي تعيد مسار تطبيقنا كما يلي:

```
Future<String> AppPath() async{
  final path = await getApplicationDocumentsDirectory();
  return path.path;
}
```

الطريقة الثانية تعيد الملف الذي سنستخدمه لحفظ أو استرجاع الملفات كما يلي:

```
Future <File> AppFile() async {
  final file = await AppPath();
  return new File('$file/dart.txt');
}
```

الطريقة الثالثة والتي تكتب على الملف تكون كما يلي:

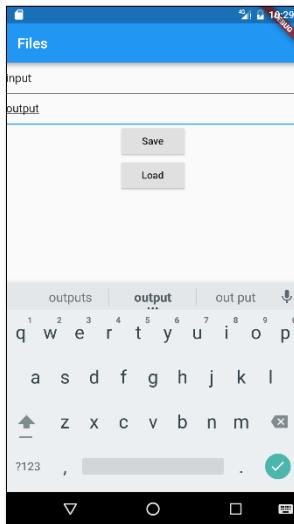
```
Future <File> writefile(String data) async
{
  final file = await (AppFile());
  return writefile('$data');
}
```

الطريقة الرابعة وهي قراءة البيانات من ملف:

```
Future <String> readFile() async {
  try {
    final File = await (AppFile());
    String data = await (File.readAsString());
  } catch (e) {
    return 'there is N file';
  }
}
```



الآن سوف نقوم ببناء كود يقوم بحفظ نص معين داخل ملف عن الضغط على الزر save ويستدعي الأمر ويظهره على label عند الضغط على زر load كما في الصورة.



main.dart

```
import 'dart:async';
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:flutter/material.dart';
import 'dart:io';
import 'package:path_provider/path_provider.dart';

Future<String> AppPath() async {
  final path = await getApplicationDocumentsDirectory();
  return path.path;
}

Future<File> AppFile() async {
  final file = await AppPath();
  return new File('$file/dart.txt');
}

Future<File> WriteFile(String data) async {
  final file = await (AppFile());
  return file.writeAsString('$data');
}

Future<String> ReadFile() async {
  try {
    final file = await (AppFile());
    String data = await (file.readAsString());
    return data;
  } catch (e) {
    return 'there is N file';
  }
}

final TextEditingController input = new TextEditingController();
final TextEditingController output = new TextEditingController();
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```

void main() async {
    //List data = await getData();
    runApp(new MaterialApp(
        title: 'Json Example',
        home: new Scaffold(
            appBar: AppBar(title: new Text('Files')),
            body: new Center(
                child: new Column(
                    children: <Widget>[
                        new TextField(controller: input),
                        new TextField(controller: output),
                        RaisedButton(
                            child: new Text("Save"),
                            onPressed: () {
                                WriteFile(input.text);
                            },
                        ),
                        RaisedButton(
                            child: new Text("Load"),
                            onPressed: () async {
                                output.text = await ReadFile();
                            }
                        ),
                    ],
                ))));
}

```



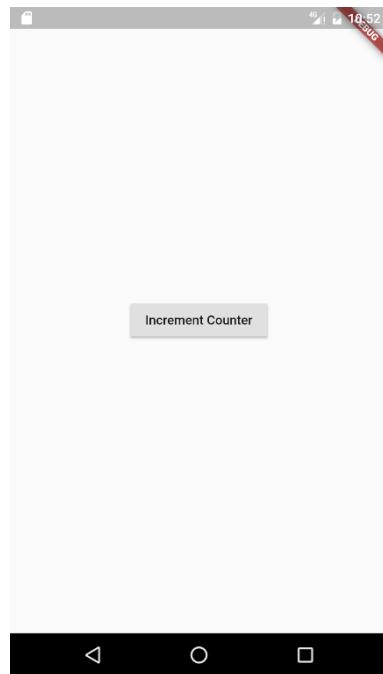
- التفضيلات المشتركة (Shared Preferences) :

الفضيلات المشتركة (SharedPreferences) هي مكون يوفر حفظ واسترجاع بيانات أساسية بسيطة . كل معلومة يتم حفظها يجب أن تكون على شكل مفتاح وقيمة (key-value) بحيث يحدد المفتاح (key) اسم مميز ليتم استعادة القيمة (value) فيما بعد عن طريقه . يمكن استخدام التفضيلات المشتركة لحفظ بيانات من الأنواع البدائية (primitive types) وتشمل : booleans, floats, ints, longs, and Strings . يتم الاحتفاظ بالبيانات المخزنة من تطبيق ما في التفضيلات المشتركة حتى بعد إغلاق هذا التطبيق.

ننوه بالبداية أنه كي نستطيع استخدام التخزين بالفضيلات المشتركة يجب أن نضيف حزمة `shared_preferences` إلى مشروعنا بحيث نضيف في الملف `Pubspec.yaml` وتحت السطر `Cupertino_icons` سطر جديد يحتوي

`shared_preferences: ^0.5.1+1`
ثم نضغط `package get` كما فعلنا مع حزمة `.json`

المثال التالي يظهر زر عند الضغط عليه سيجلب قيمة محفوظة بالمفتاح `counter` ويزيد عليها واحد وبطبعها على الكونسول ثم يعيد حفظها ، واجهة البرنامج هي:



تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

main.dart

```

import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';

void main() {
  runApp(MaterialApp(
    home: Scaffold(
      body: Center(
        child: RaisedButton(
          onPressed: _incrementCounter,
          child: Text('Increment Counter'),
        ),
      ),
    )));
}

_incrementCounter() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  int counter = (prefs.getInt('counter') ?? 0) + 1;
  print('Pressed $counter times.');
  await prefs.setInt('counter', counter);
}

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

• قواعد البيانات Sqlite

تعتبر SQLite (اس كيو لايit) نظام إدارة قواعد بيانات علائقية مثل MySQL و PostgreSQL) مضمونة في مكتبة مبرمجة بلغة C صغيرة الحجم تقربياً ٥٠٠ كليوبait . وخلافاً لأنظمة إدارة قواعد البيانات التي تتبع نظام (عميل - خادم) فإن محرك اس كيو لايit غير مستقل عن البرنامج التي يتخاطب ويتوافق معه. وبدلاً عن ذلك في مكتبة اس كيو لايit تربط بداخل ذلك البرنامج و هكذا تصبح متكاملة مع البرنامج. ويقوم البرنامج باستدعاء وظائف اس كيو لايit بواسطة باستدعاءات داخلية بسيطة مما يقلل الزمن التأخير في الوصول إلى قاعدة البيانات. قاعدة البيانات اس كيو لايit تحفظ البيانات و التعريفات والجداول في ملف واحد (قابل للنقل بين أنظمة التشغيل) على الجهاز المستضيف. وهذا التصميم البسيط يسمح بقفل ملف قاعدة البيانات عند بداية عملية نقل البيانات.

اس كيو لايit طورها الدكتور ريتشارد هب ، و يقدم و يبيع دورات تعلمية عليها و يقدم عقود الدعم الفني و الإضافات مثل الضغط و التشفير. ومصدر قاعدة البيانات اس كيو لايit مرخص تحت الترخيص الملكية العامة public domain ، بحيث يمكنه من استعمالها بحرية من قبل أي شخص لأي غرض كان.

SQLite مميزات

-دعم معظم مقاييس SQL-92 والتي شملت المناقلات قاعدة البيانات والتي تحوي على ثلاثة مميزات Atomicity وتعني قدرة قاعدة البيانات على إنجاز كافة المهام أو عدم انجازها بالكامل مثل القدرة على نقل الودائع بشكل كامل أو فشلها بالكامل بسبب أي سبب من الأسباب. الميزة الثانية isolated وهي تعني قدرة التطبيق على جعل المناقلة تظهر منفصلة عن بقية العمليات، وهذا يعني أنه لا توجد عملية خارج المناقلة تستطيع باي شكل من الأشكال رؤية البيانات في وسط المناقلة.

الميزة الثالثة durable وهي تعني ضمان أن المناقلات التي تمت بنجاح تبقى حية باستمرار ولا تلغى بسبب فشل النظام، مثال ذلك إذا أخبر نظام قواعد البيانات لجز الرحلات بأن مقعد ما حجز بنجاح فإن المقعد سيبقى محجوزاً حتى لو انهار النظام.

-صغر حجمها.

-سهولة التركيب.

-سهولة نقل البيانات من مزود إلى آخر.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



- لا توجد مشاكل بالترميز لا سيما مع اللغة العربية.
- قاعدة البيانات عبارة عن ملف واحد فقط.
- تدعم حجم قاعدة البيانات إلى ٢٠٤٨ جيجابايت (٢٠٤٨ جيجابايت) - ما قبل الإصدارة ٢٠٨ كان الحد الأقصى: ٢ جيجابايت.
- (шиفرة الاتصال والاستعلام بها سهلة) مشابهة لـ MySQL على نحو أبسط.
- يمكن استخدامها على المواقع التي لا تدعم MySQL.

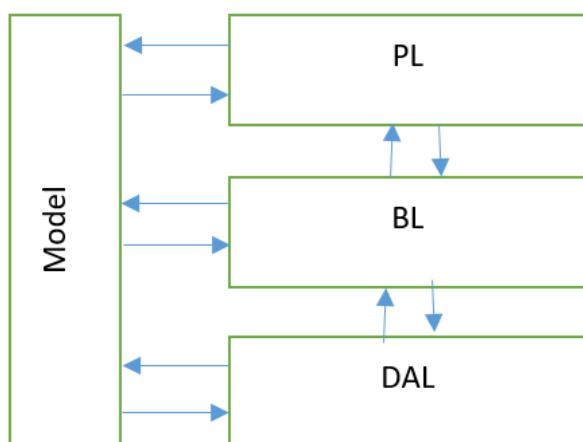
استدعاء مكتبة SQLite:

- نفتح الملف pubspec.yaml من ضمن ملفات المشروع.
- نضيف السطر packages get sqflite: any ثم نضغط Ctrl + S.

هرمية العمل:

سوف نعتمد في هذا الكتاب على هرمية N-tier للتعامل مع قواعد البيانات SQLite والتي تعتبر من أفضل وأبسط هرميات هندسة البرمجيات والتي تعتمد على تقسيم الكود إلى طبقات حسب وظيفة كل جزء من الكود.

الشكل التالي يبين الهرمية المتبعة.



تأليف، فيصل الأسود | مهندس برمجيات

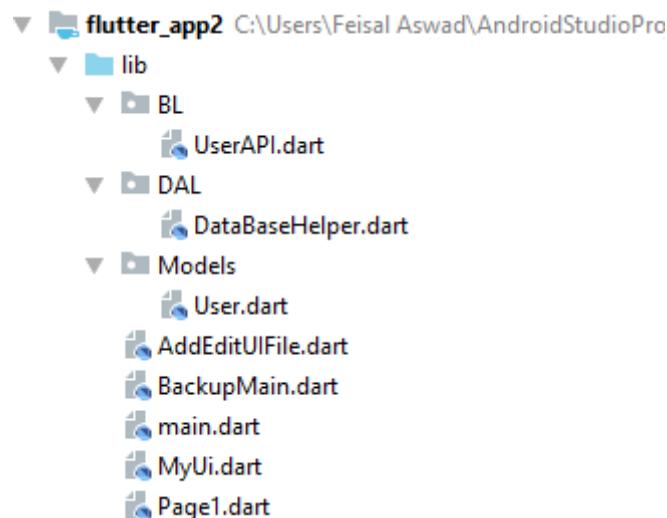
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

كما ننشئ ملفات المشروع بشكل يوافق الهرمية كما يلي:



شرح الطبقات:

الطبقة DAL: يتم وضع فيها كل تابع الاتصال مع قاعدة البيانات بالإضافة لتابع إنشاء القاعدة أول مرة.

الطبقة BL: يتم وضع فيها كل الأصناف والتوابع التي تقوم بالعمليات من نوع CRUD والتي هي أي جميع العمليات التي يمكن تطبيقها على قاعدة البيانات لصنف معين مثلاً (مستخدم، موظف، مخزن).

الطبقة PL: يتم فيها بناء واجهة المستخدم التي ستتعامل مع الجدول والأصناف المقابلين لها.

الطبقة Models: هي طبقة مساعدة تسهل التعامل مع بيانات نوع الجداول فمثلاً لدى جدول الموظفين فهنا أنشأنا في طبقة Models صنف موظف بخصائص معينة.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

الآن لنطبق هذه الهرمي بشكل مفصل وعلى جدول واحد هو جدول المستخدمين ولتسهيل الأمر فرضنا لديهم فقط (معرف مستخدم Id ، اسم مستخدم username، كلمة مرور password) بالبداية ننشئ الصنف user كما يلي:

User.dart

```
class User {
    int _Id;
    String _Username;
    String _Password;
    User(this._Id, this._Username, this._Password);

    int getId() => _Id;
    set Id(id) => _Id = id;

    String getUsername() {
        return _Username;
    }

    setUsername(String username) {
        _Username = username;
    }

    String getPassword() {
        return _Password;
    }

    setPassword(String password) {
        _Password = password;
    }
}
```

الطبقة DAL تحوي الصنف DataBaseHelper.dart والذي يقوم بجميع عمليات الاتصال والإنشاء لقاعدة البيانات.

سنقوم بشرح كل جزء من الكود على حدى.

أولاً استدعاء المكتبات، حيث نجد في بداية الصنف استدعاء لعدة مكتبات كما في الكود:

```
import 'package:path_provider/path_provider.dart';
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';
```

والتي ستساعدنا في التعامل مع الملفات – قواعد البيانات و التزامن التابع الأول الذي سنتحدث عنه وهو getdb

```
static Future<Database> getdb() async {
    if (_db != null) {
        return _db;
    } else {
        _db = await OpenDb();
        return _db;
    }
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قناة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

والذي مهمته أن يقوم بإرجاع قاعدة البيانات في حال كانت موجودة أما في حال كانت غير موجودة

سيقوم بتحميلها مستدعاً تابع آخر وهو OpenDb

```
static Future<Database> OpenDb() async {
    Directory dir = await getApplicationDocumentsDirectory();
    String path = join(dir.path, 'Userdb.db');

    var Userdb = await openDatabase(path, version: 1, onCreate:
    _FirstCreate);

    return Userdb;
}
```

وهذا بدوره سيقوم بالتقاط مسار التطبيق ثم تحميل Database من مسارها في حال وجودها مسبقاً

وعدا ذلك يقوم باستدعاء مرة ثانية لتابع هو FirstCreate والذي يقوم بإنشاء قاعدة البيانات المطلوبة.

```
static void _FirstCreate(Database db, int version) async {
    var CreateUserTableStatement =
        "create table User(Id INTEGER PRIMARY KEY AUTOINCREMENT NOT
NULL,Username Text,Password Text)";

    await db.execute(CreateUserTableStatement);
}
```

والتابع الكلي يكون كما يلي:

DataBaseHelper.dart

```
import 'package:path_provider/path_provider.dart';
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';
import 'dart:io';

class DataBaseHelper {
    static Database _db;

    static Future<Database> getdb() async {
        if (_db != null) {
            return _db;
        } else {
            _db = await OpenDb();
            return _db;
        }
    }

    static Future<Database> OpenDb() async {
        Directory dir = await getApplicationDocumentsDirectory();
        String path = join(dir.path, 'Userdb.db');

        var Userdb = await openDatabase(path, version: 1, onCreate:
        _FirstCreate);

        return Userdb;
    }
}
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```

static void Restart() async {
  Directory dir = await getApplicationDocumentsDirectory();
  String path = join(dir.path, 'Userdb.db');
  File f = new File(path);

  if (!f.existsSync()) {
    deleteDatabase(path);
    print("DataBase has deleted");
  }
}

static void _FirstCreate(Database db, int version) async {
  var CreateUserTableStatement =
    "create table User(Id INTEGER PRIMARY KEY AUTOINCREMENT NOT
NULL,Username Text,Password Text)";

  await db.execute(CreateUserTableStatement);
}

static Future<void> CloseDb() async {
  var db = await getdb();
  db.close();
  _db = null;
}
}

```

إلى هنا تكون قد بنينا تابع مهمته الاتصال والتحكم بقاعدة المعطيات والمطلوب الآن بناء تابع UserAPI والذي مهمته إجراء العمليات على قاعدة البيانات كما ذكرنا وهو موجود ضمن الطبقة BL ويحوي جميع التوابع الخاصة بالتعامل مع المستخدم كما يلي:

UserAPI.dart

```

import 'package:flutter_app2/DAL/DataBaseHelper.dart';
import 'package:flutter_app2/Models/User.dart';

class UserAPI {
  static Future<int> AddUser(User user) async {
    var db = await DataBaseHelper.getdb();
    Map<String, dynamic> UserMap = new Map();
    UserMap["Username"] = user.getUsername();
    UserMap["Password"] = user.getPassword();
    int result = await db.insert("User", UserMap);

    return 0;
  }

  static Future<List> GetAllUser() async {
    var db = await DataBaseHelper.getdb();
    String SelectAllUserStatement = "Select * from User";
    List result = await db.rawQuery(SelectAllUserStatement);

    return result.toList();
  }

  static Future<int> EditUser(User user) async {
    var db = await DataBaseHelper.getdb();

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قناة الـYoutube



```

Map<String, dynamic> UserMap = new Map();
UserMap["Username"] = user.getUsername();
UserMap["Password"] = user.getPassword();
UserMap["Id"] = user.getId();
int result = await db
    .update("user", UserMap, where: "Id=?", whereArgs:
[user.getId()]);
return result;
}

static Future<int> DeleteUser(int Id) async {
var db = await DataBaseHelper.getdb();

int result = await db.delete("user", where: "Id=${Id}");
return result;
}

static Future<void> DeleteAllUser() async {
var db = await DataBaseHelper.getdb();
db.execute("delete from user");
}
}

```

يجب أن نذكر أن أي تابع ينتظر أمرا من خارج التطبيق نضع له `async` وعند طلبه للبيانات نضع `.Future Method` ويكون مستقبلي أي `await` أما بخصوص الطبقة PL ف تكون كما يلى:

MyUi.dart

```

import 'package:flutter/material.dart';
import 'package:flutter_app2/BL/UserAPI.dart';
import 'package:flutter_app2/Models/User.dart';
import 'package:flutter_app2/AddEditUIFile.dart';

class MyListView extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _MyList();
  }
}

class _MyList extends State<MyListView> {
List Users = new List();

@Override
void initState() {
super.initState();
UserAPI.GetAllUser().then((usrs) {
setState(() {
usrs.forEach((usr) {
Users.add(usr);
});
});
});
}
}

@Override

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـة الـYoutube



```

Widget build(BuildContext context) {
  return new MaterialApp(
    home: new Scaffold(
      floatingActionButton: new FloatingActionButton(
        onPressed: () => _NavigateToAdd(context),
        child: new Icon(Icons.add),
        backgroundColor: Colors.blue,
      ),
      appBar: new AppBar(
        iconTheme: new IconThemeData(
          color: Colors.green,
        ),
        title: new Text('AppBar Title'),
        backgroundColor: Colors.greenAccent,
      ),
      body: new Column(
        children: <Widget>[
          new TextField(
            decoration: new InputDecoration(hintText:
"Search")),
          new Expanded(
            child: new ListView.builder(
              itemCount: Users.length,
              itemBuilder: (BuildContext context, int
position) {
              return new Card(
                color: Colors.white,
                child: new ListTile(
                  title: new Text(
"${Users[position]['Username']}"),
                  subtitle:
                    new Text("Id:
${Users[position]['Id']}"),
                  trailing: new Row(
                    mainAxisAlignment: MainAxisAlignment.min,
                    children: <Widget>[
                      new IconButton(
                        onPressed: () {
                          _NavigateToEdit(context,
position);
                      },
                      icon: new Icon(Icons.edit,
color:
Colors.lightBlue)),
                      new IconButton(
                        onPressed: () =>
                          _deleteUser(context,
position),
                        icon: new Icon(Icons.delete,
color:
Colors.redAccent)),
                    ],
                  );
                )));
            },
          )));
}
}

void _NavigateToAdd(BuildContext context, int position) async {
  String result = await Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => AddEditUI(
        "Edit",
        new User(Users[position]["Id"]),
    ),
  );
}

```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قناة الـYoutube



```

    Users[position][ "Username" ],
    Users[position][ "Password" ])));
    RefreshList();
}

void _NavigateToAdd(BuildContext context) async {
    String result = await Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => AddEditUI("Add", new User(0, "", ""),
        "")));
    RefreshList();
}

void RefreshList() {
    Users.clear();
    UserAPI.GetAllUser().then((usrs) {
        setState(() {
            usrs.forEach((usr) {
                Users.add(usr);
            });
        });
    });
}

void _deleteUser(BuildContext context, int position) {
    UserAPI.DeleteUser(Users[position][ "Id" ]);
    setState(() {
        Users.removeAt(position);
    });
}

void _AddUser(BuildContext context, int position) {
    UserAPI.DeleteUser(Users[position][ "Id" ]);
    setState(() {
        Users.removeAt(position);
    });
}

void _EditUser(BuildContext context, int position) {
    UserAPI.DeleteUser(Users[position][ "Id" ]);
    setState(() {
        Users.removeAt(position);
    });
}
}

```

Main.dart

```

import 'package:flutter/material.dart';
import 'MyUi.dart';
import 'package:flutter_app2/MyUi.dart';

List Users;
void main() async {
    runApp(new MaterialApp(home: new MyListView()));
}

```

تأليف، فيصل الأسود | مهندس برمجيات

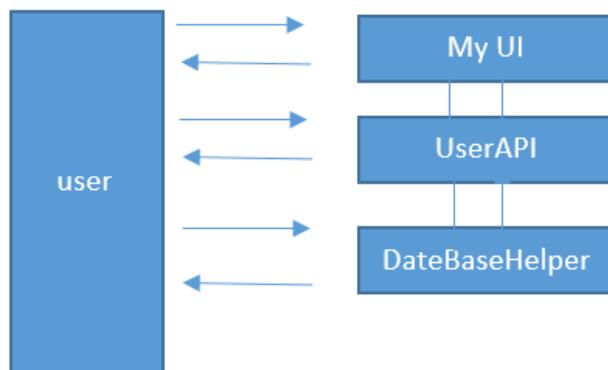
يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

ويكون المخطط لمثالنا كما يلي:



صنف واجهة الإضافة والتعديل كما يلي:

AddEditUIFile.dart

```

import 'package:flutter/material.dart';
import 'package:flutter_app2/BL/UserAPI.dart';
import 'package:flutter_app2/Models/User.dart';

class AddEditUI extends StatefulWidget {
    User user;
    String command;
    AddEditUI(this.command, this.user);
    @override
    State<StatefulWidget> createState() {
        // TODO: implement createState
        return new _AddEditUI(command, user);
    }
}

class _AddEditUI extends State<AddEditUI> {
    User user;
    String command;
    _AddEditUI(this.command, this.user);

    final _txtUsername = new TextEditingController();
    final _txtPassword = new TextEditingController();

    @override
    void initState() {
        super.initState();

        if (command == "Edit") {
            setState(() {
                _txtUsername.text = user.getUsername();
                _txtPassword.text = user.getPassword();
            });
        }
    }
}
  
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```

}

@Override
Widget build(BuildContext context) {
    return new MaterialApp(
        home: new Scaffold(
            appBar: new AppBar(
                title: new Text(command),
                backgroundColor: Colors.greenAccent,
            ),
            body: new Center(
                child: new Column(
                    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                    children: <Widget>[
                        Padding(
                            padding: EdgeInsets.all(5),
                        ),
                        new Text(
                            "Username:",
                            style: TextStyle(
                                fontSize: 25,
                            ),
                            textAlign: TextAlign.left,
                        ),
                        Padding(
                            padding: EdgeInsets.all(5),
                        ),
                        new TextField(
                            controller: _txtUsername,
                            decoration: new InputDecoration(hintText: "Username"),
                            style: TextStyle(fontSize: 20, color: Colors.black),
                        ),
                        Padding(
                            padding: EdgeInsets.all(5),
                        ),
                        Padding(
                            padding: EdgeInsets.all(5),
                        ),
                        new Text(
                            "Password:",
                            style: TextStyle(
                                fontSize: 25,
                            ),
                            textAlign: TextAlign.left,
                        ),
                        Padding(
                            padding: EdgeInsets.all(5),
                        ),
                        new TextField(
                            controller: _txtPassword,
                            decoration: new InputDecoration(hintText: "Password"),
                            style: TextStyle(fontSize: 20, color: Colors.black),
                        ),
                        Padding(
                            padding: EdgeInsets.all(5),
                        ),
                        new RaisedButton(
                            onPressed: () => AddOrEdit(),
                            child: new Text(command),
                            color: Colors.greenAccent,
                        )
                    ],
                ),
            ),
        ),
    );
}

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

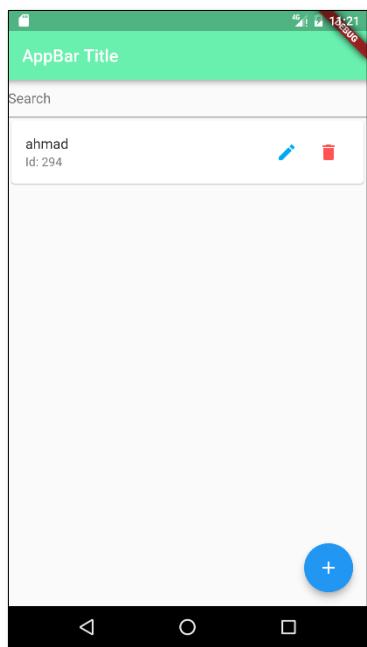
```

    }

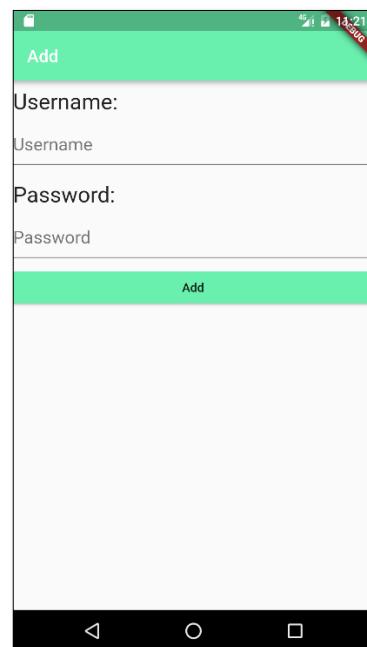
    void AddOrEdit() {
        if (_txtUsername.text.length > 0 && _txtPassword.text.length > 0) {
            if (command == "Add") {
                UserAPI.AddUser(new User(0, _txtUsername.text,
                _txtPassword.text));
            } else {
                UserAPI.EditUser(
                    new User(user.getId(), _txtUsername.text,
                _txtPassword.text));
            }
            Navigator.pop(context);
        }
    }
}

```

واجهة التطبيق بالكامل تكون:



واجهة عرض المستخدمين



واجهة الإضافة والتعديل

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflowArabi

www.Learn-Barmaga.com : تعلم المزيد عن برمجة التطبيقات

التعامل مع الصور

مقدمة:

تتيح لنا فلاتر بسهولة الية للتعامل مع الصور سواء اكانت مخزنة في الاستديو او التقاطها من الكاميرا مباشرة.

كما يوفر المحاكي في الاندرويد استديو كاميرا افتراضية وهمية لاختبار عمل الكاميرا داخل التطبيق.

استدعاء مكتبة picker:

قد يتاح لنا الصنف imagepicker في flutter التعامل وجلب أي نوع من الصور أو التقاطها من الكاميرا ويجب أن نعلم أن التوابع ضمنه متزامنة لذلك يجب تكتيب أيضا ضمن توابع متزامنة.

مثال تطبيقي:

التابع التالي بسيط جدا يتاح عند استدعائه جلب صورة من الاستديو ووضعها ضمن كائن file ومن ثم وضعه في widget لعرض الصور.

```
picker() async{
  File img =await ImagePicker.pickImage(source: ImageSource.gallery);
  if(img !=null){
    image=img;
    setState(() {
      });
  }
}
```

نلاحظ أن التابع تعامل مع ملفات النظام وهذا يجب أن تقوم بتضمين مكتبة dio في استدعاء المكتبات كما يلي:

```
import 'package:flutter/material.dart';
import 'dart:io';
import 'package:image_picker/image_picker.dart';
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

الآن نكتب ملف الواجهة كاملاً وعند الضغط على الزر ضمنها سيأخذنا إلى الاستديو لجلب الملف وسيوضعه في widget لإظهار الصور وهي image كما في الكود التالي:

CameraUi.dart

```
import 'package:flutter/material.dart';
import 'dart:io';
import 'package:image_picker/image_picker.dart';

class CameraUi extends StatefulWidget {
    @override
    State<StatefulWidget> createState() {
        // TODO: implement createState
        return _CameraUi();
    }
}

class _CameraUi extends State<CameraUi> {
    File image;
    picker() async {
        File img = await ImagePicker.pickImage(source:
ImageSource.gallery);
        if (img != null) {
            image = img;
            setState(() {});
        }
    }

    @override
    Widget build(BuildContext context) {
        // TODO: implement build
        return new Scaffold(
            appBar: new AppBar(title: new Text("ImageDemo")),
            body: new Center(
                child: image == null ? Text("There Is No Image") :
Image.file(image),
            ),
            floatingActionButton: new FloatingActionButton(
                onPressed: picker,
                child: new Icon(Icons.add),
            ),
        );
    }
}
```

والتابع main بالشكل التالي:

main.dart

```
import 'package:flutter/material.dart';
import 'package:image_demo/	CameraDeal.dart';

void main() {
    runApp(new MaterialApp(home: new CameraUi(),));
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ [Youtube](#)



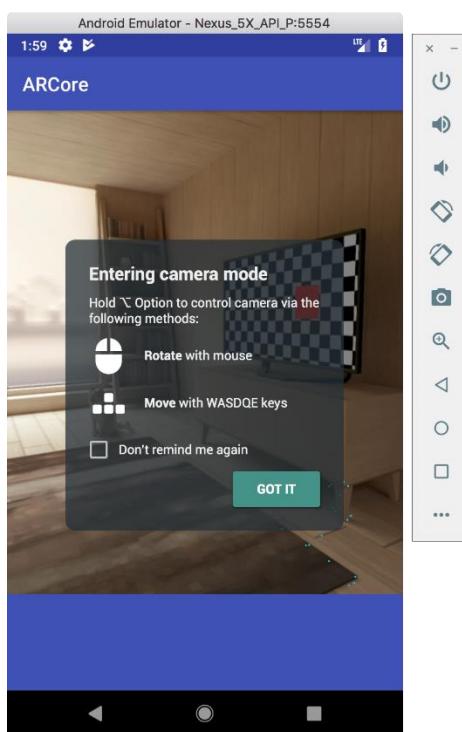
StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

الآن ماذا لو أردنا أن نتعامل مع الكاميرا بشكل مباشر هنا يجب أن نستدعي نفس الصنف والتابع لكن بدلاً من استدعاءه من أجل الاستديو نختار الـ `camera` ويكون بالشكل التالي

```
picker() async {
  File img = await ImagePicker.pickImage(source: ImageSource.camera);
  if (img != null) {
    image = img;
    setState(() {});
  }
}
```

نود أن ننوه أن المحاكي يتيح لنا بيئة تصوير افتراضية فعند تشغيل البرنامج وال التقاط صورة باستخدام الكاميرا ستظهر بالشكل التالي وكأن الكاميرا تعمل:



تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكامل محتويات الكتاب على قناة الـ [Youtube](#)



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

الاشعارات الداخلية

يتتيح flutter إمكانية إشعارات المستخدم بسلسلة كبيرة عن طريق حزمة Flutter-local-notification والتي تقوم عند حدث معين بإشعار المستخدم بالشكل التالي:

مثال تطبيقي:

في مثالنا التالي سنقوم بإنشاء زر عند الضغط عليه سيقوم بإشعارنا بأنه تم الضغط على الزر، كذلك سنقوم بوضع الاستجابة عند الضغط على هذا الإشعار.

التابع التالي هو لتهيئة عمليات الاستشعار لكلا النظمتين Android و IOS

```
FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin;

@Override
void initState() {
    // TODO: implement initState
    super.initState();
    flutterLocalNotificationsPlugin = new
    FlutterLocalNotificationsPlugin();
    var android = new
    AndroidInitializationSettings('@mipmap/ic_launcher');
    var ios = new IOSInitializationSettings();
    var initSettings = new InitializationSettings(android, ios);
    flutterLocalNotificationsPlugin.initialize(
        initSettings, onSelectNotification: clickOnNotification );
}
```

أما التابع showNotification فيمكن بناءه بعدة طرق ولكننا سنبنيه بالشكل التالي:

```
showNotification() async{
    var android = new AndroidNotificationDetails(
        "channelId", "channelName", "channelDescription"
        ,priority: Priority.High,importance: Importance.Max);
    var iOS = new IOSNotificationDetails();

    var platform = new NotificationDetails(android, iOS);
    await flutterLocalNotificationsPlugin.show(
        0, 'Hello World', 'StackOver Flow Arabi', platform,payload:
    Notification);
}
```

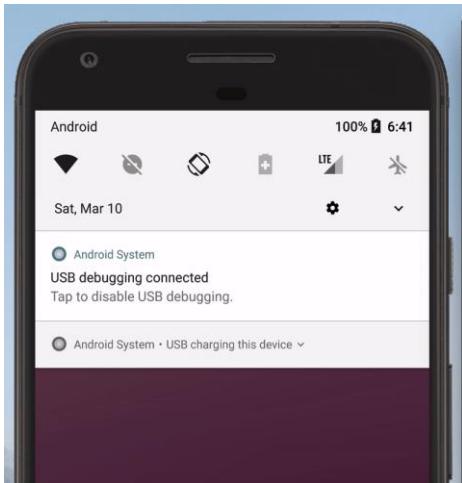
تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com



نلاحظ في الصورة النصوص المكتوبة في الأعلى ومواقعها في الإشعار أما في خصوص العبارة فهي ستظهر عند الضغط على هذا الإشعار ويكون تنفيذ طلب محتوى الإشعار بالتتابع التالي

```
showNotification() async {
    var android = new AndroidNotificationDetails(
        "channelId", "channelName", "channelDescription",
        priority: Priority.High, importance: Importance.Max);
    var iOS = new IOSNotificationDetails();
    print("jhj");
    var platform = new NotificationDetails(android, iOS);
    await flutterLocalNotificationsPlugin.show(
        0, 'Notification 1', 'Notification 2', platform,
        payload: 'Send Messages');
}
```

وهكذا نكون قد فهمنا كل جزء من أجزاء البرنامج المطلوب ويكون برنامجنا كاملاً.

```
import
"package:flutter_local_notifications/flutter_local_notifications.dart";
import 'package:flutter/material.dart';
import 'dart:async';

class MyHomePage extends StatefulWidget {
    @override
    _MyHomePageState createState() => new _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
    FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin;

    @override
    void initState() {
        // TODO: implement initState
        super.initState();
        flutterLocalNotificationsPlugin = new
        FlutterLocalNotificationsPlugin();
        var android = new
        AndroidInitializationSettings('@mipmap/ic_launcher');
        var ios = new IOSInitializationSettings();
        var initSettings = new InitializationSettings(android, ios);
        flutterLocalNotificationsPlugin.initialize(initSettings,
            onSelectNotification: clickOnNotification);
    }
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قناة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```

}

Future clickOnNotification(String payload) {
    debugPrint('print payload : $payload');
    showDialog(
        context: context,
        builder: (_) => AlertDialog(
            title: new Text('Notification'),
            content: new Text('$payload'),
        ),
    );
}

@Override
Widget build(BuildContext context) {
    return new Scaffold(
        appBar: new AppBar(
            title: new Text('Local Notification'),
        ),
        body: new Center(
            child: new Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                    new Text(
                        'Click Button',
                    ),
                    new RaisedButton(
                        child: Text('Click me'), onPressed: showNotification)
                ],
            ),
        ),
    );
}

showNotification() async {
    var android = new AndroidNotificationDetails(
        "channelId", "channelName", "channelDescription",
        priority: Priority.High, importance: Importance.Max);
    var iOS = new IOSNotificationDetails();
    print("jhj");
    var platform = new NotificationDetails(android, iOS);
    await flutterLocalNotificationsPlugin.show(
        0, 'Notification 1', 'Notification 2', platform,
        payload: 'Send Messages');
}
}
}

```



الحركة Animation

مقدمة:

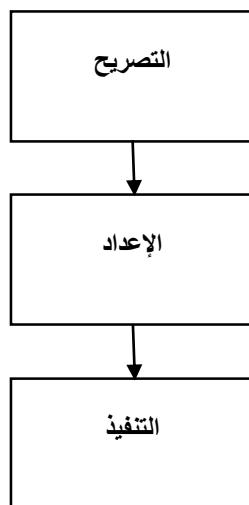
تقدم حزمة Animation Flutter مكتبات جاهزة معها لإنجاز حركات قوية، حيث تعطي الحركة للواجهة قوة وحيوية.

مفهوم الحركات:

معظم الحركات في Flutter تعتمد على الصنف `Animation` الموجود في المكتبة المضمنة افتراضياً:

'package: flutter/animation.dart'
والتي تحوي الكثير من الحركات الجاهزة والتوابع القوية.

إن النمط الهيكلي لأي حركة يتم بتقسيم متطلباتها إلى ثلاثة أجزاء كما في المخطط التالي:



وأيضاً يجب أن نذكر أن لكل حركة متحكم وهو كائن من نوع `AnimationController` والذي يقوم بالتحكم بالحركة من (بدء - توقف - إعادة) سنتعرض الآن هرمية بناء الحركات في Flutter وسننتقل لأمثلة ننفذ فيها عدة حركات. في البداية نصرح عن الكائنات الخاصة بالحركة كما يلي:

```
Animation animation;
AnimationController animationController;
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

الآن سنقوم بتجهيز حركتنا في تابع خاص سنسميـه `initAnimation` وسيتم ضبط فيه كل اعدادات الحركة المطلوبة:

```
void initAnimation() {
    animationController=AnimationController(duration: Duration(seconds:
1),vsync: this);
    flipAnimation=Tween(begin: 0.0,end:
1.0).animate(CurvedAnimation(parent: animationController, curve:
Curves.fastOutSlowIn));
    animationController.forward();
}
```

وسنقوم باستدعاء هذا التابع ضمن التابع الافتراضي `:initState`

```
void initState() {
    super.initState();
    initAnimation();
}
```

السطر الأول:

```
animationController=AnimationController(duration: Duration(seconds:
1),vsync: this);
```

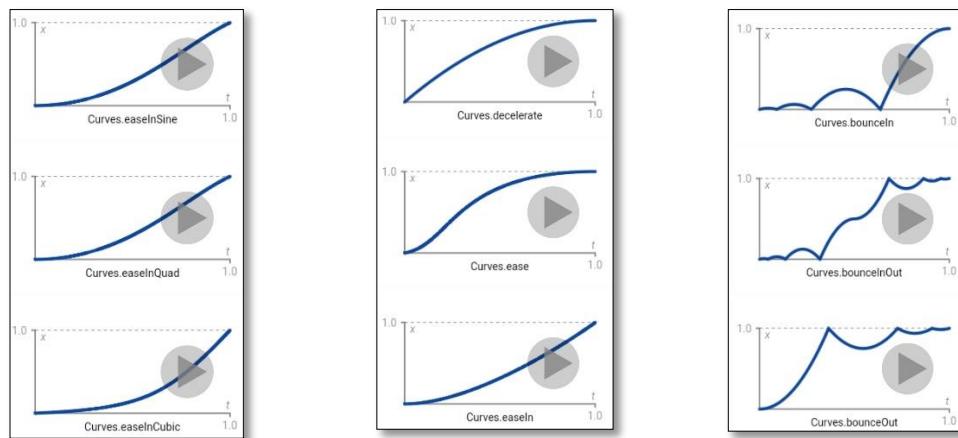
وذلك يقوم بتعيين المدة المطلوبة التي ستستغرقها الحركة وفي حالـنا نحدد ثانية واحدة.

السطر الثاني وهو مهم جدا والـذي يـعرف مكان بدء وانتهـاء الحـركة وكذلك طـريقة دخـولها للشاشة:

```
flipAnimation=Tween(begin: 0.0,end:
1.0).animate(CurvedAnimation(parent: animationController, curve:
Curves.fastOutSlowIn));
```

بدأنا في النقطة .. إلى النقطة إلى .. ودخول نوع `fastOutSlowIn` ويمكنك الاطلاع على الأنـواع

الـآخر بالاعتماد على الجدول.



تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكتاب محتويات الكتاب على قنـاة الـYoutube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

التعليمية الأخيرة هي:

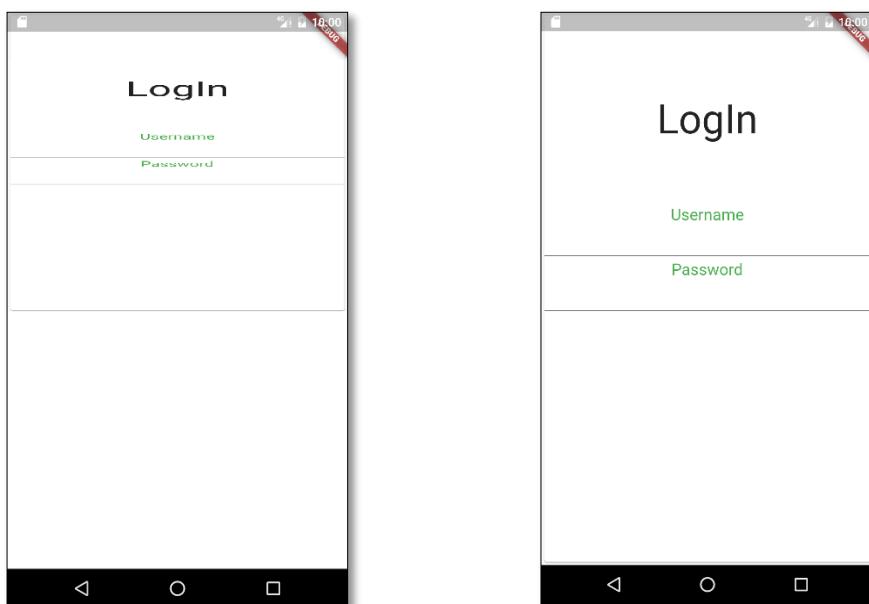
```
AnimationController.forward();
```

والتي تقوم ببدء الحركة (أي تنفيذها).

ويجب أن نعلم أن أي عنصر سيقوم بالحركة سيتم وضعه ضمن Animation Builder الأمثلة التالية ستوضح الحركة بشكل أكثر وضوحا.

مثال تطبيقي:

ليكن لدينا الواجهة التالية ونريد عند فتح التطبيق دخول الصفحة بالشكل التالي:



أي أن تنزل من الأعلى إلى الأسفل

هنا سنبني الصنف Flip و Flip_ والذين سيقومان بتحقيق الواجهة المطلوبة.

الكود التالي لكود كامل الواجهة:

```
import 'package:flutter/material.dart';
import 'package:flutter/animation.dart';
class Flip extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _Flip();
  }
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```

class _Flip extends State<Flip> with SingleTickerProviderStateMixin{
  Animation flipAnimation;
  AnimationController animationController;
  @override
  void initState(){
    super.initState();
    initAnimation();
  }
  void initAnimation(){
    animationController=AnimationController(duration: Duration(seconds: 1),vsync: this);
    flipAnimation=Tween(begin: 0.0,end: 1.0).animate(CurvedAnimation(parent: animationController, curve: Curves.fastOutSlowIn));
    animationController.forward();
  }
  @override
  Widget build(BuildContext context) {
    // TODO: implement build

    return AnimatedBuilder(
      animation: animationController,
      builder: (BuildContext context,Widget chile){
        return UiWithoutAnimation();
      },
    );
  }

  Widget UiWithoutAnimation()
  {
    final double pi=3.14;
    final double width=MediaQuery.of(context).size.width;
    return new Scaffold(
      body: Transform(
        transform:
        Matrix4.identity()..rotateX(2*pi*flipAnimation.value),
        child: new Center(
          child: new Card(
            child: new Column(
              children: <Widget>[
                new Padding(padding: EdgeInsets.only(bottom: 100)),
                new Text("LogIn",style: TextStyle(fontSize: 50),),
                new Padding(padding: EdgeInsets.only(bottom: 75)),
                new Text("Username",style: TextStyle(fontSize: 20,color: Colors.green),),
                new TextField(style: TextStyle(color: Colors.black)),
                new Padding(padding: EdgeInsets.only(bottom: 5)),
                new Text("Password",style: TextStyle(fontSize: 20,color: Colors.green),),
                new TextField(style: TextStyle(color: Colors.black)),
              ],
            ),
          ),
        );
    );
  }
}

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

نلاحظ أننا بنينا الواجهة في تابع منفصل هو `UiWithoutAnimation` كما يجب أن نذكر أن الحركة تمت عن طريق وضع التحويل التالي:

```
transform: Matrix4.identity()..rotateX(2*pi*flipAnimation.value)
```

بتغيير المحاور للدوران مثل `z` سحصل على حركات جديدة.

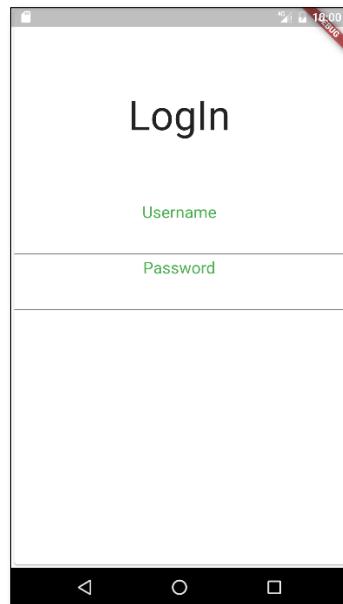
وتم استدعاء الواجهة بالكود الرئيسي `:main`:

```
import 'package:flutter/material.dart';
import 'package:my_project/EnterSlideAnimation.dart';
import 'package:my_project/FlipAnimation.dart';
import 'HideShowAnimation.dart';
void main() => runApp(MaterialApp(home: Flip(),));
```

سنستعرض الآن أمثلة وعلى عجلة مع العلم أن كل واجهة يمكن استدعاءها بالصنف `main` كما شاهدنا لذلك لن نكرر كود التابع `main` كل مرة.

مثال تطبيقي:

اكتب الكود اللازم لبناء الواجهة التالية:



تأليف، فيصل الأسود | مهندس برمجيات

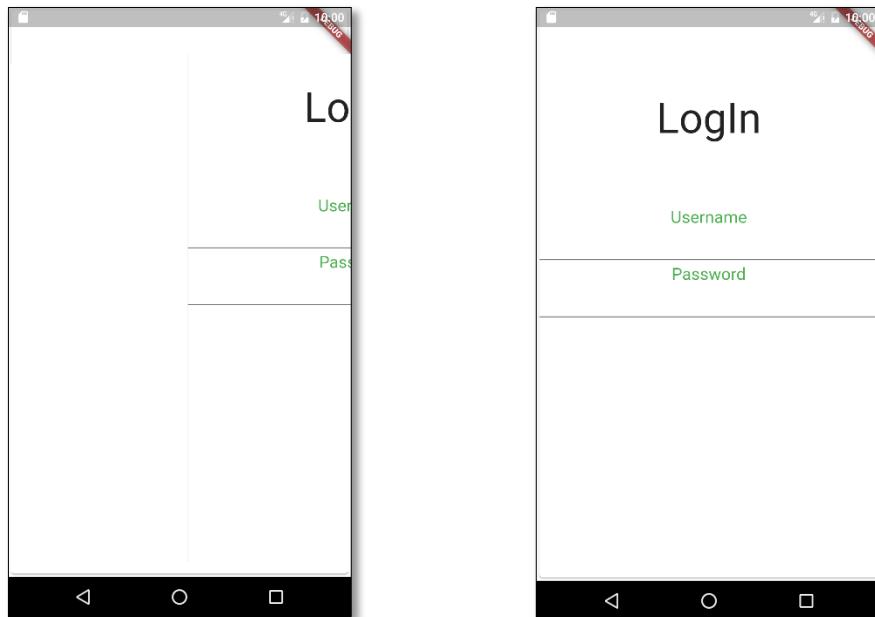
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ [Youtube](#)



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

مع إمكانية تحريكها دخولاً للشاشة كما يلي:



بساطة سوف نستخدم التابع translate والذي يقوم بتحريك الواجهة كما في الكود:

```
transform: Matrix4.translationValues(animation.value*width, 0.0, 0.0),
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكتاب على قناته على Youtube

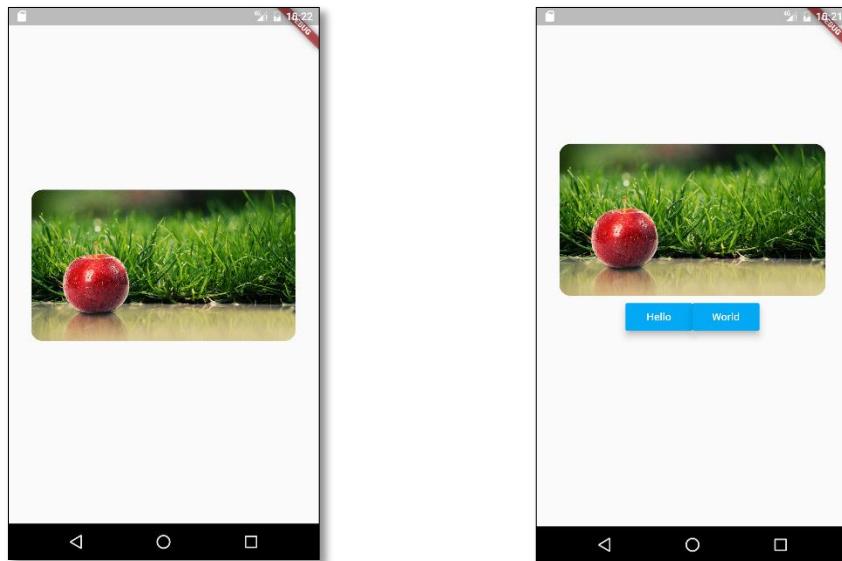


StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

مثال تطبيقي:

اكتب الكود اللازم لبناء الواجهة التالية بحيث عندما نضغط ضغطة واحدة على الصورة سوف تنزل كما في الشكل.



وعندما نضغط ضغطتين عليها مرة أخرى ستعود كما كانت.

وهنا سنستخدم الأحداث ونتحكم بالحركة عن طريق Animation Controller

يجب الذكر أن عملية إضافة الصور في Flutter تتم بالشكل التالي:

- النقر بالزر الأيمن على مجلد المشروع واختيار .show in explorer
- الدخول لمجلد المشروع وإنشاء مجلد اسمه assets ومن ثم ننشأ مجلد آخر اسمه Images
- بداخله ونضع الصور فيه.

نذهب إلى الملف pubspec.yaml في ملفات المشروع.

- نضيف الصورة المطلوبة في الملف تحت كلمة assets كما في الشكل مع مراعاة المسافات هنا:

```
assets:
  - assets/images/apple.jpg
```

والآن نقوم بإنشاء كود الواجهة المطلوبة ويكون بالشكل التالي:

```
import 'package:flutter/material.dart';
import 'package:flutter/animation.dart';
class HideShow extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _HideShow();
  }
}
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```
}

class _HideShow extends State<HideShow> with
SingleTickerProviderStateMixin{
  Animation hideShowAnimation;
  AnimationController animationController;
  @override
  void initState(){
    super.initState();
    animationController=AnimationController(duration: Duration(seconds: 1),vsync: this);
    hideShowAnimation=Tween(begin: 0.0,end: -0.15).animate(CurvedAnimation(parent: animationController, curve: Curves.fastOutSlowIn));
    animationController.forward();
  }
  @override
  Widget build(BuildContext context) {
    // TODO: implement build

    return AnimatedBuilder(
      animation: animationController,
      builder: (BuildContext context,Widget chile){
        return UiWithoutAnimation();
      },
    );
  }

  Widget UiWithoutAnimation()
  {
    final double width=MediaQuery.of(context).size.width;
    return new Scaffold(
body: new Center(
  child: new Stack(
    children: <Widget>[
      new Center(
        child: Container(
          padding: EdgeInsets.all(10.0),
          width: 350.0,
          height: 200,
          decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(15.0)
          ),
        ),
        child: new Row(
          mainAxisAlignment: MainAxisAlignment.end,
          mainAxisSize: MainAxisSize.center,
          children: <Widget>[
            new RaisedButton(
              child: Text("Hello"),
              elevation: 7.0,
              color: Colors.lightBlue,
              textColor: Colors.white,
              onPressed: (){},
            ),
            new RaisedButton(
              child: Text("World"),
              elevation: 7.0,
              color: Colors.lightBlue,
              textColor: Colors.white,
              onPressed: (){},
            ),
          ],
        ),
      ),
    ],
  ),

```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



```

),
new Center(
  child: GestureDetector(
    child: Container(
      alignment: Alignment.bottomCenter,
      width: 350, height: 200,
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(15.0),
        image: DecorationImage(image:
AssetImage('assets/images/apple.jpg'), fit: BoxFit.cover)

      ),
      transform: Matrix4.translationValues(0.0,
hideShowAnimation.value*width, 0.0),
    ),
    onTap: () {animationController.forward();},
    onDoubleTap: () {animationController.reverse();} ,
  ),
),
),
),
),
),
);
}
}

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

رسائل التنبيه السريعة Toast

تعد الرسائل السريعة Toast من أبسط الرسائل التي تقدمها التطبيقات للمستخدم وتكون غالباً على شكل تنبيه يظهر على الشاشة مدة زمنية ويختفي.

فلاتر تقدم لنا في احدى مكتباتها طرق التعامل مع هذا النوع من الرسائل.

لكي نستطيع التعامل مع رسائل Toast علينا أولاً استدعاء المكتبة الخارجية

```
fluttertoast: ^3.0.1
```

إلى الملف Pubsec.yaml ومن ثم get packages بالإضافة إلى استدعاء المكتبة إلى صنف العمل الذي يريد استخدامه كما يلي:

```
import 'package:fluttertoast/fluttertoast.dart';
```

يعد استخدام هذا النوع من الرسائل من أبسط الأشياء حيث يكفي فقط استدعاء الطريقة showToast لإظهار الرسالة كما يمكن ضبط اعدادات خاصة لتلك الرسائل:

```
Fluttertoast.showToast(
    msg: "This is Center Short Toast",
    toastLength: Toast.LENGTH_SHORT,
    gravity: ToastGravity.CENTER,
    timeInSecForIos: 1,
    backgroundColor: Colors.red,
    textColor: Colors.white,
    fontSize: 16.0
);
```

الجدول التالي يظهر أهم الخصائص المستخدمة لإظهار رسائل التنبيه السريعة

property	description
msg	String (Not Null)(required)
toastLength	Toast.LENGTH_SHORT or Toast.LENGTH_LONG (optional)
gravity	ToastGravity.TOP (or) ToastGravity.CENTER (or) ToastGravity.BOTTOM
timeInSecForIos	int (only for ios)
bgcolor	Colors.red
textcolor	Colors.white
fontSize	16.0 (float)

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



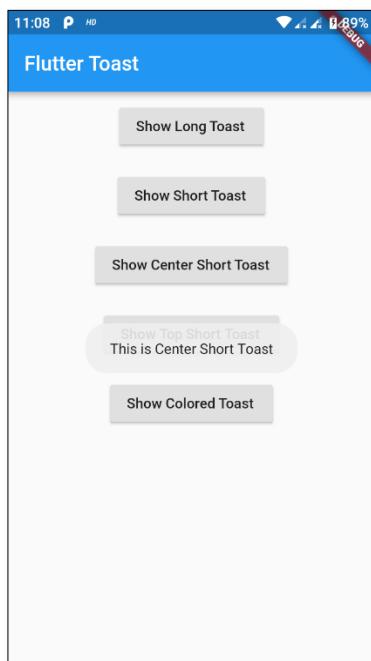
StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

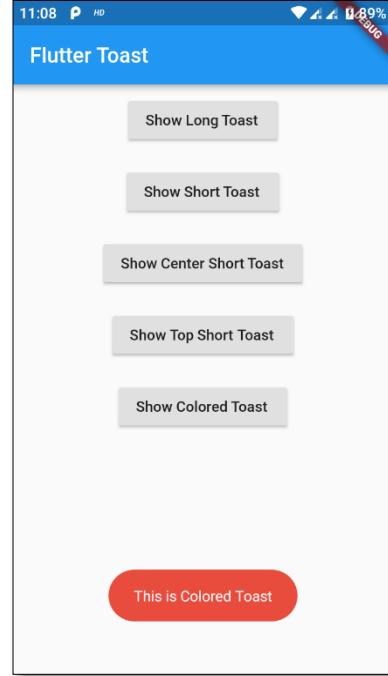
وللإلغاء كافة الرسائل الظاهرة يكون بالتعليمية:

```
Fluttertoast.cancel()
```

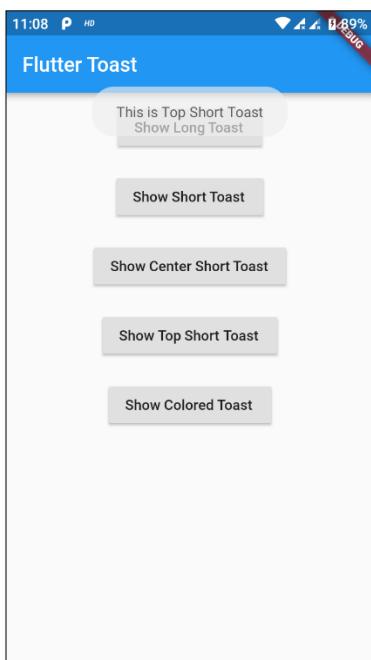
سنستعرض الان بعض رسائل تنبيه سريعة مع خصائص مختلفة:



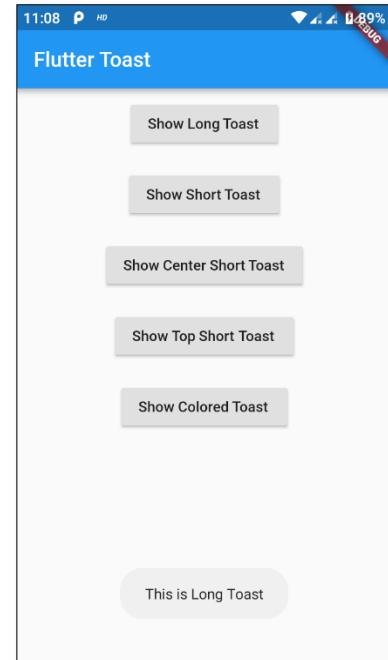
gravity:ToastGravity.CENTER



bgColor:Colors.red



gravity:ToastGravity.TOP



gravity:ToastGravity.BOTTOM

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكتاب على قناته على Youtube

لكلام محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

قواعد البيانات Firebase



Firebase

فاير بيز Firebase: هي خدمة قدمتها Google منذ فترة وقد كانت تقتصر فقط على تخزين البيانات وبعض الأشياء البسيطة، ولكن في 16 Google I/O تم الإعلان عن الكثير من المميزات الجديدة والرائعة وأصبحت حديث الكثير من المطورين

مميزاتها:

- التحقق Authentication : وهي عملية تسجيل الدخول سواء عن طريق حساب Facebook ، وهي نفس الوقت حماية البيانات الموجودة في Database Google, Twitter, Email (بمعنى أنه يمكنه منع أي شخص من استخدام التطبيق دون عملية تسجيل الدخول وهو الوضع الافتراضي)
- قواعد بيانات متزامنة Real-time Database : وهي تفيد في تخزين البيانات على السيرفر وأكثر شيء يميّزها هي أنها Real-time بمعنى أنه أي تغيير يحصل على الداتا بيز سيتغّير فوراً في التطبيق كما سنرى في هذا الشرح.
- التخزين Storage: تخزين الملفات والصور.
- الاستضافة Hosting: لاستضافة موقعك على Firebase.
- الإشعارات Notifications: إرسال إشعارات.

والعديد من المميزات يمكنها تفدها عند الدخول الى حسابك في Firebase من الجدير بالذكر أن هذه الخدمات مجانية (ولكن بعض الحدود، يمكنك رؤية ماهي الحدود عبر هذا الرابط <https://firebase.google.com>

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

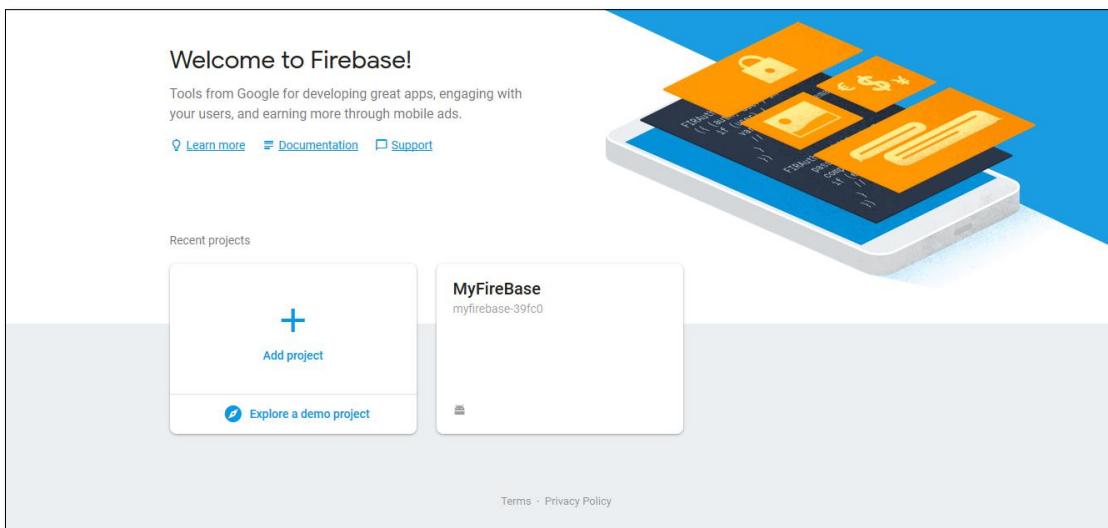


StackOverflow Arabi

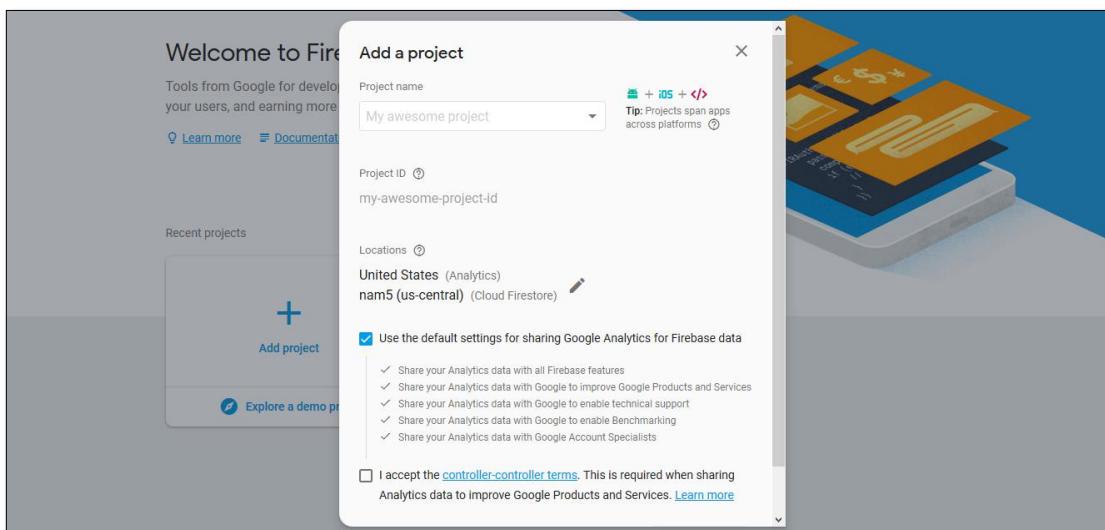
تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

سنبدأ في هذا الدرس شرح عن Firebase Database وسنحاول مستقبلاً بإذن الله شرح باقي الأمور بدايةً يجب أن يكون لديك حساب Google أي حساب Gmail.

نذهب إلى موقع Get Start نضغط على زر Add project ثم نختار Firebase



ثم نضع اسم المشروع الذي نريد ونضع الدولة والموافقة على الشروط ثم إنشاء



تأليف، فيصل الأسود | مهندس برمجيات

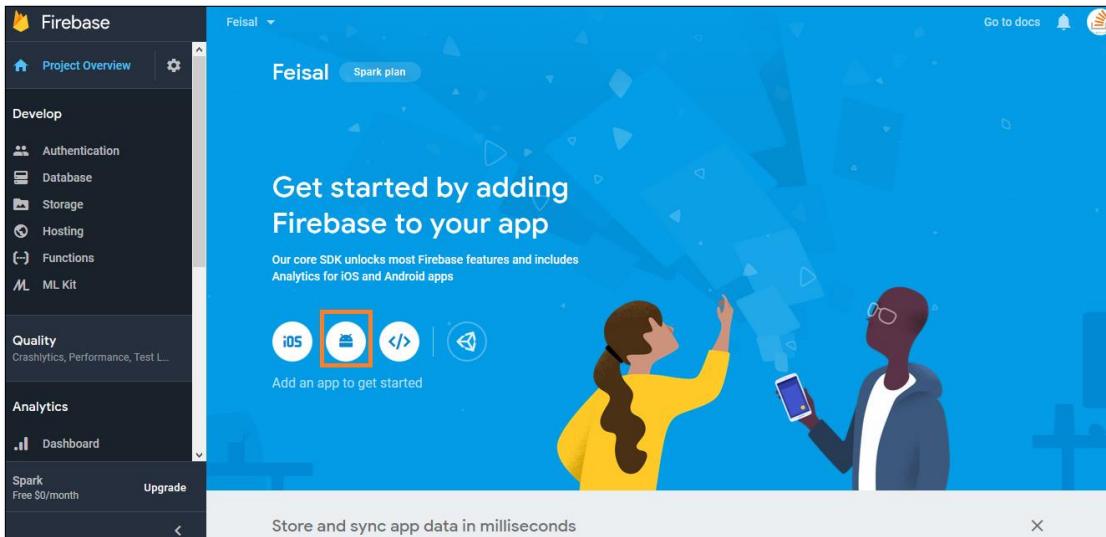
يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



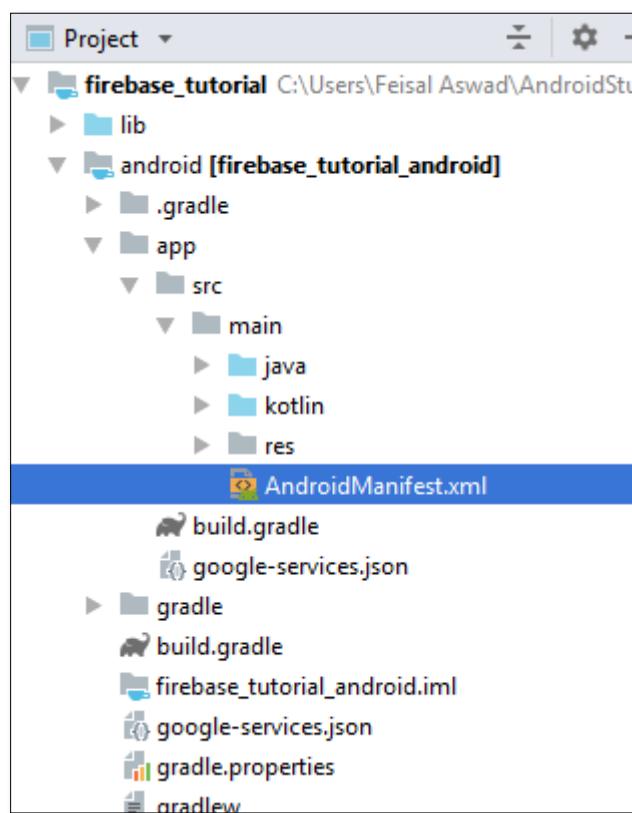
StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

بعد ذلك سيتم إنشاء المشروع ، نضغط على Add Firebase to Your Android App



بعد ذلك سيطلب منا اسم ال package الخاص بتطبيق الاندرويد



تأليف فيصل الأسود | مهندس برمجيات

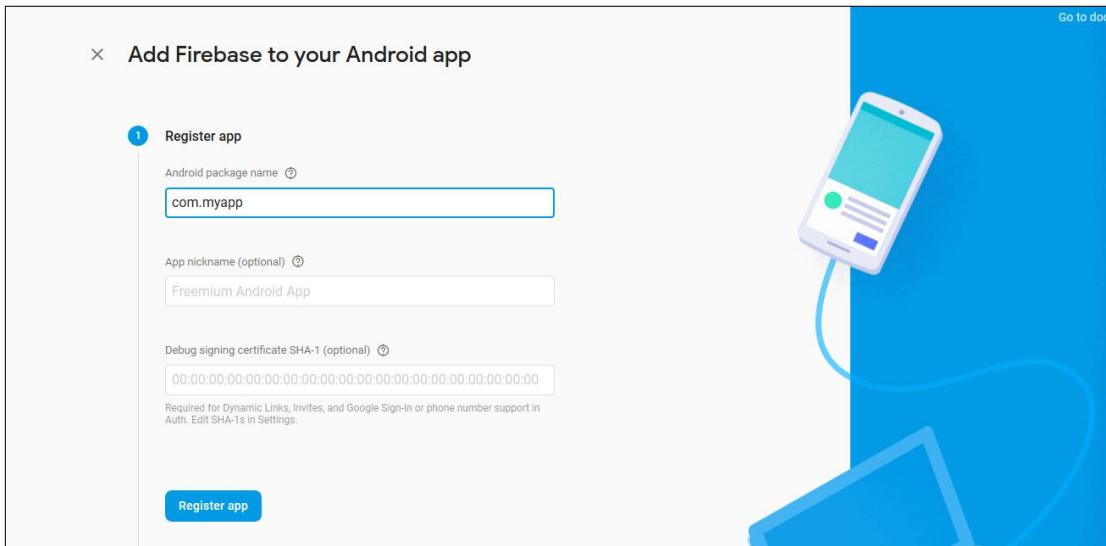
يمكنك الحصول على شرح للكامل محتويات الكتاب على قناته على Youtube



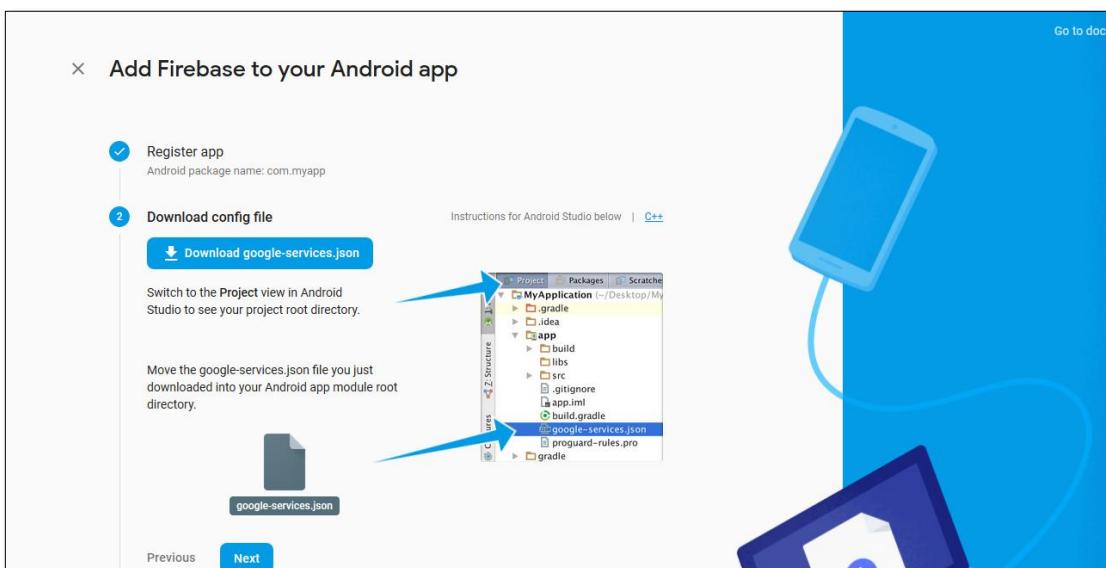
StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

نقوم بإنشاء تطبيق اندرويد جديد ثم نتوجه الى `AndroidManifest.xml` للتقاط اسم الحزمة ونضعها كما يظهر في الصورة ثم نضغط `:Register App`:



بعد ذلك نضغط على `Download google-services.json` وهو ملف إعدادات الذي يربط مشروعنا على Firebase بمشروعنا في Android Studio.



تأليف، فيصل الأسود | مهندس برمجيات

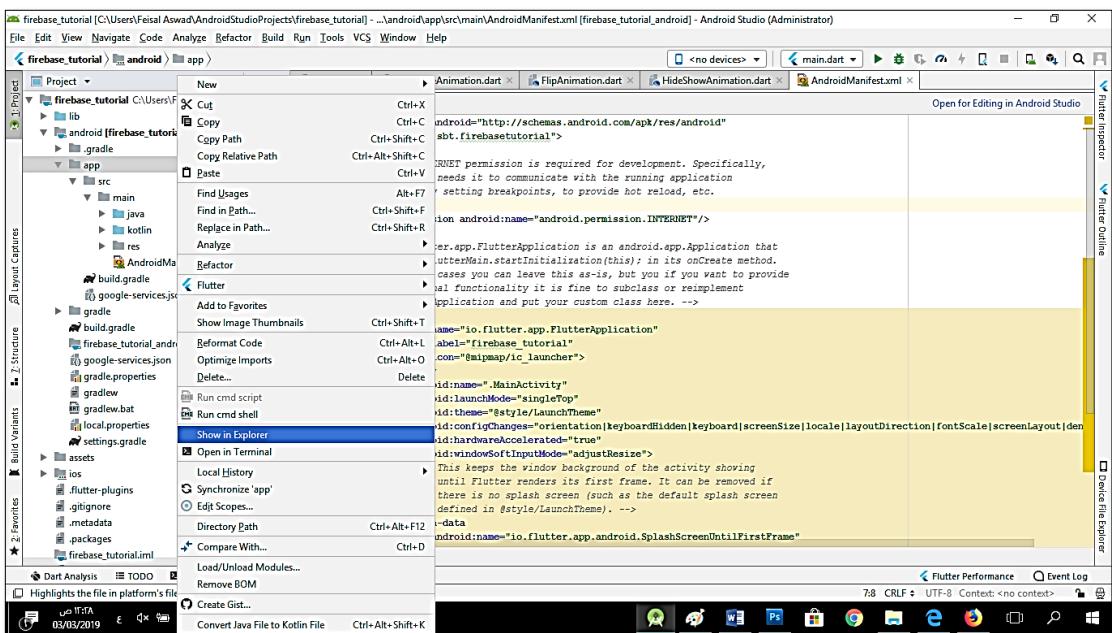
يمكنك الحصول على شرح لـكامل محتويات الكتاب على قناة الـ [Youtube](#)



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

سيتم تحميل هذا الملف قم بنسخه ثم نتوجه الى Android Studio ونضغط بالزر الأيمن على مجلد app ونختار Show in Explorer



سيتم فتح مجلد المشروع في حاسوبك نقوم بلصق هذا الملف الذي حملناه في مجلد app

الآن يجب علينا إضافة مكتبات Firebase الى مشروعنا في Android Studio



ثم نضغط get packages

وفي ملف الصنف الذي سنبني به واجه التخاطب مع Firebase نضيف المكتبات كما يلي:

```
import 'package:firebase_database.firebaseio_database.dart';
```

تأليف فيصل الأسود | مهندس برمجيات

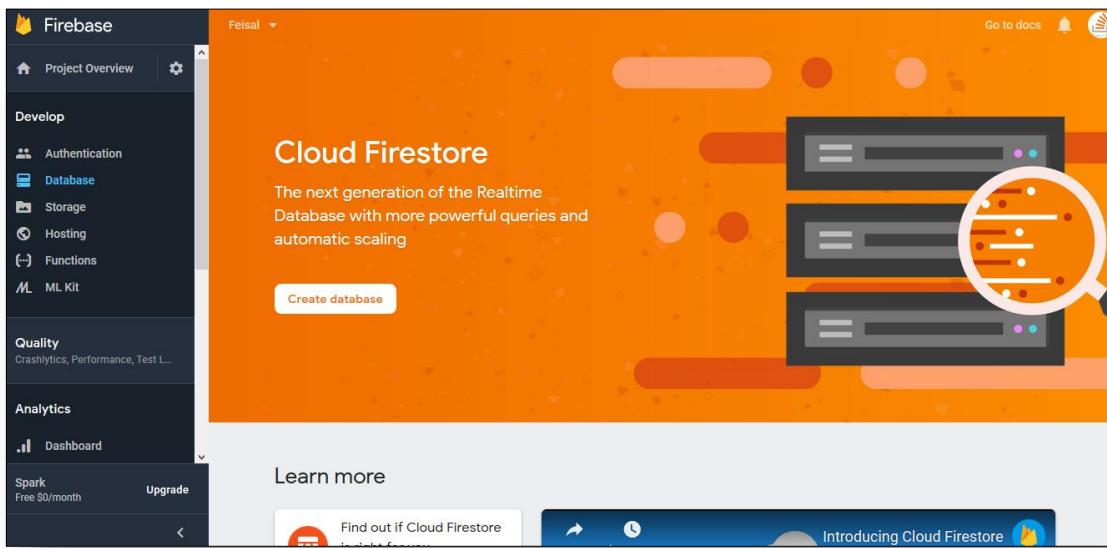
يمكنك الحصول على شرح لـكامل محتويات الكتاب على قناة الـ Youtube



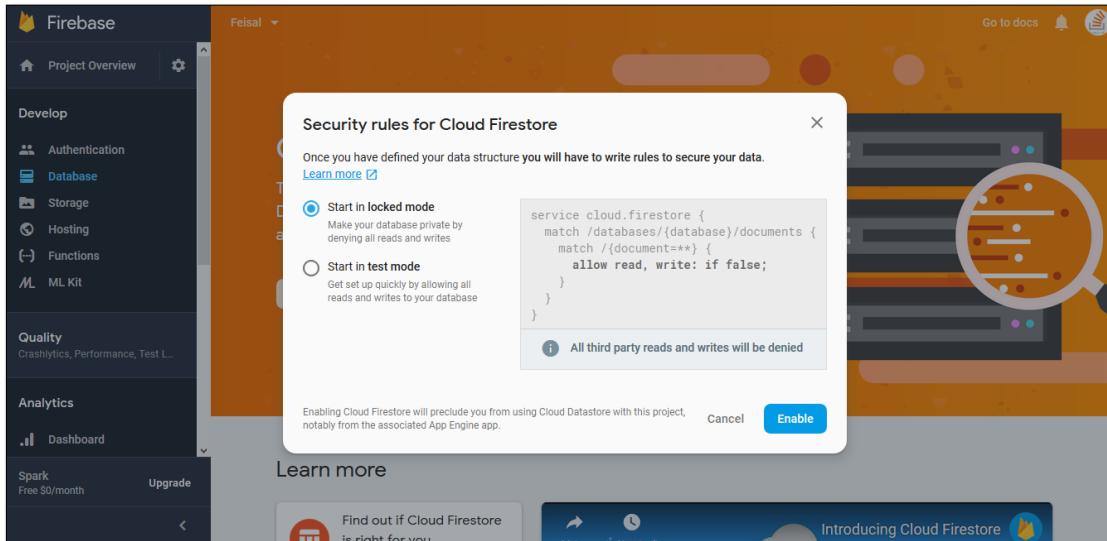
StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

الآن نعود الى واجهة السيرفر ونضغط على زر لإنشاء قاعدة جديدة Database



هنا عليك الاختيار بين قاعدة بيانات تدريبية وهي الخيار الثاني او قاعدة بيانات لمشروع حقيقي
ويفضل في حال كانت تجربتك الأولى مع فايربزيز ان تبدأ بقاعدة بيانات تدريبية مع العلم انها غير
امنة



تأليف فيصل الأسود | مهندس برمجيات

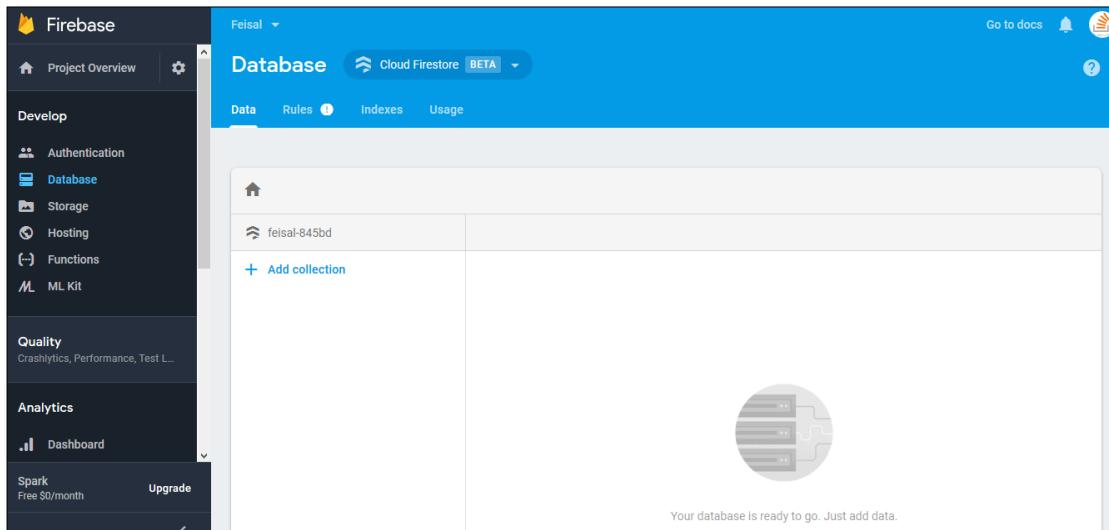
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

كما يمكننا ان نرى البيانات التي نملكونها في قاعدة بياناتنا من التبويب Database في واجهة Firebase



الآن يجب ان نبدأ برمجياً بعمليات الاتصال بالـ Firebase الخاصة بنا.
ما سنقوم به بالبداية هو تعريف كائن مهمته الاتصال مع قاعدة البيانات الخاصة بنا وذلك كما يلي:

```
final
studentReference=FirebaseDatabase.instance.reference().child("student");
```

هذا التابع اتصل مع قاعدة البيانات الخاصة بنا وانشاء حقل هو student يمثل جدول الطلاب.

الخطوة التالية هي ان ننشئ احداث وتتابع تستجيب لها لكي يكون الاتصال مع القاعدة متزامناً بحيث أي تغير يحصل في القاعدة سوف ينعكس اثره على تطبيقنا مباشرة.

```
StreamSubscription<Event> _onStudentAddedSubscription;
StreamSubscription<Event> _onStudentChangedSubscription;

_onStudentAddedSubscription=
studentReference.onchildAdded.listen(_onStudentAdded);

_onStudentChangedSubscription=
studentReference.onchildChanged.listen(_onStudentUpdated);
```

في الكود السابق قمنا بإنشاء مراقبين لحدثين لمراقبة الإضافة والتعديل وعند حدوث أي تغيير بهما سوف يستدعي التابعين الخاصين بالإضافة والتعديل كما يلي على الترتيب:

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```
void _onStudentAdded(Event event) {
  setState(() {
    items.add(new Student.fromSnapshot(event.snapshot));
  });
}
```

```
void _onStudentChanged(Event event) {
  var oldStudentValue=items.singleWhere((student)=>student.id==event.snapshot.key);
  setState(() {
    items[items.indexOf(oldStudentValue)]=new Student.fromSnapshot(event.snapshot);
  });
}
```

اما بخصوص تابع الحذف فيكون بالشكل التالي:

```
void _deleteStudent(BuildContext context, Student student, int position) async{
  await studentReference.child(student.id).remove().then((_) {
    setState(() {
      items.removeAt(position);
    });
  });
}
```

ويكون تابع الواجهة الكلي لاظهار والتعامل مع البيانات كما يلي:

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:firebase_demo/model/Student.dart';
import 'package:firebase_demo/ui/Student_Screen.dart';
class ListViewStudent extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _ListViewStudent();
  }
}
final
studentReference=FirebaseDatabase.instance.reference().child("student");
;
class _ListViewStudent extends State<ListViewStudent>{

  List<Student> items;
  StreamSubscription<Event> _onStudentAddedSubscription;
  StreamSubscription<Event> _onStudentChangedSubscription;
  @override
  initState(){
    super.initState();
    items=new List();
    _onStudentAddedSubscription=studentReference.onchildAdded.listen(_onStudentAdded);
  }
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ [Youtube](#)



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```

_onStudentChangedSubscription=studentReference.onchildChanged.listen(_o
nStudentUpdated);
}
@Override
void dispose() {
    super.dispose();
    _onStudentAddedSubscription.cancel();
    _onStudentChangedSubscription.cancel();
}
@Override
Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(
        appBar: new AppBar(title: new Text("Firebase"),
            centerTitle: true,
            backgroundColor: Colors.greenAccent
        ),
        body: Center(
            child: ListView.builder(itemCount: itmes.length,
                padding: EdgeInsets.only(top: 12.0),
                itemBuilder: (context,position){
                    return new Column(
                        children: <Widget>[
                            Divider(height: 6.5, ),
                            ListTile(title: new Text("${itmes[position].name}"),
                                style: TextStyle(color: Colors.red,fontSize: 25.0),
                            ),
                            subtitle:new Text("${itmes[position].name}"),
                            style: TextStyle(color: Colors.red,fontSize: 11.0),
                            ),
                            leading: Column(
                                children: <Widget>[
                                    CircleAvatar(backgroundColor: Colors.redAccent,
                                        radius: 16.0,
                                        child: new Text("${itmes[position].name}"),
                                    )
                                ],
                                IconButton(icon: Icon(Icons.delete),color:
Colors.lightBlueAccent, onPressed:
()
=>_deleteStudent(context,itmes[position],position),),
                                IconButton(icon: Icon(Icons.delete),color:
Colors.lightBlueAccent, onPressed:
()
=>_onStudentChanged())
                            ],
                            onTap:
()
=>_NavigateToStudent(context,itmes[position],position),
                        )
                    ],
                );
            }
        ),
        floatingActionButton: FloatingActionButton(onPressed:
()
{ _createNewStudent();},
        child: new Icon(Icons.add),),
    );
}
void _onStudentAdded(Event event){
setState(() {
    itmes.add(new Student.fromSnapShot(event.snapshot));
});
}
void _onStudentChanged(Event event) {
    var
}

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```

oldStudentValue=items.singleWhere((student)=>student.id==event.snapshot.key);
    setState(() {
        items[items.indexOf(oldStudentValue)]=new Student.fromSnapshot(event.snapshot);
    });
}
void _deleteStudent(BuildContext context,Student student,int position) async{
await studentReference.child(student.id).remove().then((_) {
setState(() {
    items.removeAt(position);
});
});
}
void _NavigateToStudent(BuildContext context,Student student,int position){
    await Navigator.push(context,MaterialPageRoute(builder: (context)
=>StudentScreen(student))
);
}
void _createNewStudent(BuildContext context){
    await Navigator.push(context,MaterialPageRoute(builder: (context)
=>StudentScreen(new Student(null, "", "", "", "", "")))
);
}
}

```

التابع `_createNewStudent` مهمته هي الانتقال للشاشة التالية والتي تقوم بعرض مكان حقول ادخال مستخدم جديد او العديل عليه.

```

void _createNewStudent(BuildContext context){
    await Navigator.push(context,MaterialPageRoute(builder: (context)
=>StudentScreen(new Student(null, "", "", "", "", "")))
);
}

```

اما الشاشة التالية فهي بتصميم بسيط لاظهار حقول الادخال ويكون كما يلي:

```

import 'dart:async';
import 'package:flutter/material.dart';
import 'package:firebase_demo/model/Student.dart';

class StudentScreen extends StatefulWidget{
    final Student student;
    StudentScreen(this.student);
    @override
    State<StatefulWidget> createState() {
        // TODO: implement createState
        return new _StudentScreen(student);
    }
}
final
studentReference=FirebaseDatabase.instance.reference().child("student")
;
class _StudentScreen extends State<StudentScreen>{
    TextEditingController _nameController;

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ [Youtube](#)



```

    TextEditingController _ageController;
    TextEditingController _departmentController;
    TextEditingController _descriptionController;
    TextEditingController _cityController;
    final Student student;
    _StudentScreen(this.student);
    @override
    void initState() {
        _nameController=new TextEditingController(text:
    widget.student.name);
        _ageController=new TextEditingController(text: widget.student.age);
        _departmentController=new TextEditingController(text:
    widget.student.department);
        _descriptionController=new TextEditingController(text:
    widget.student.description);
        _cityController=new TextEditingController(text:
    widget.student.city);
    }

    @override
    Widget build(BuildContext context) {
        // TODO: implement build
        return new Scaffold(
            appBar: AppBar(title: Text("Student")),
            ),
            body: new Container(
                margin: EdgeInsets.only(top: 12.0),
                alignment: Alignment.center,
                child: new Column(
                    children: <Widget>[
                        TextField(
                            controller: _nameController,
                            decoration: InputDecoration(icon:
Icon(Icons.person),labelText: "name"),
                        ),
                        Padding(padding: EdgeInsets.only(top: 7.0)),
                        TextField(
                            controller: _cityController,
                            decoration: InputDecoration(icon:
Icon(Icons.location_city),labelText: "name"),
                        ),
                        Padding(padding: EdgeInsets.only(top: 7.0)),
                        TextField(
                            controller: _departmentController,
                            decoration: InputDecoration(icon:
Icon(Icons.departure_board),labelText: "name"),
                        ),
                        Padding(padding: EdgeInsets.only(top: 7.0)),
                        TextField(
                            controller: _descriptionController,
                            decoration: InputDecoration(icon:
Icon(Icons.description),labelText: "name"),
                        ),
                        Padding(padding: EdgeInsets.only(top: 7.0)),
                        TextField(
                            controller: _ageController,
                            decoration: InputDecoration(icon:
Icon(Icons.data_usage),labelText: "name"),
                        ),
                        Padding(padding: EdgeInsets.only(top: 7.0)),
                        FlatButton(
                            onPressed: (){
                                if(widget.student.id!=0){
                                    studentReference.child(widget.student.id).set({

```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قناة الـYoutube



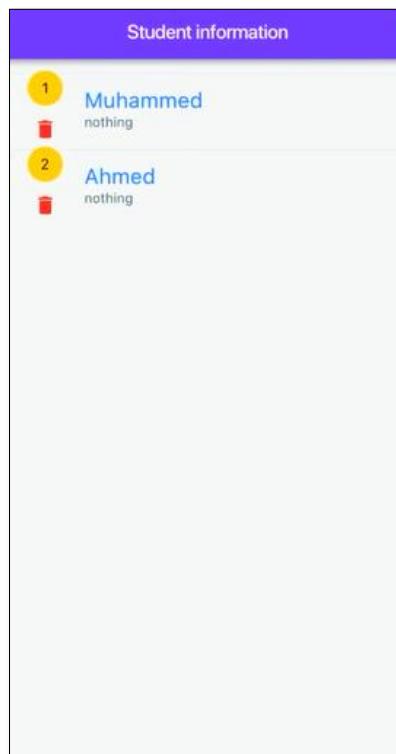
StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```

        'name': _nameController,
        'age': _ageController,
        'city': _cityController,
        'department': _departmentController,
        'description': _descriptionController
    }).then((_) {
        Navigator.pop(context);
    }));
}
else
{
    if(widget.student.id!=0) {
        studentReference.push.set({
            'name': _nameController,
            'age': _ageController,
            'city': _cityController,
            'department': _departmentController,
            'description': _descriptionController,
        }).then((_) {
            Navigator.pop(context);
        });
    }
},
child:
(widget.student.id!=null) ?Text ("Update") :Text ("Add") ,
)
],
),
),
);
}
}

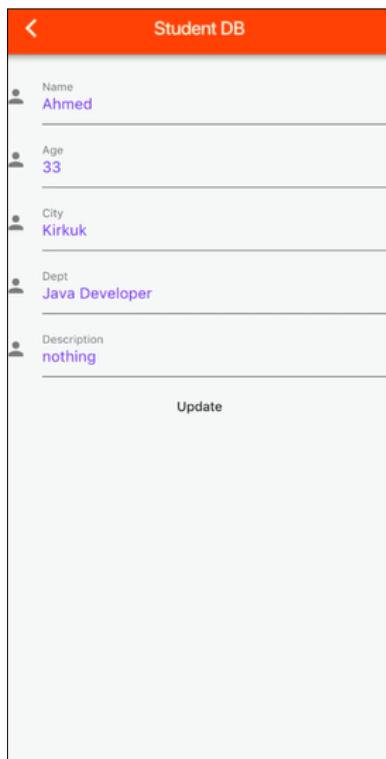
```



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube





كذلك يوجد الكثير من الأشياء التي يمكن ان نقوم بها باستخدام قواعد البيانات فايربيز مثل الدخول باستخدام الایمیل او حساب الفیس بوک او حتى التعامل مع الصور ضمن قواعد البيانات والصوتیات ولكن سنكتفي في كتابنا الى هذا الحد ولعل باقي الشروحات ستكون على القناة على اليوتيوب.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكامل محتويات الكتاب على قناة الـ [Youtube](#)



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

النشر والربح من التطبيقات

Publish Applications



WE CAN ONLY SEE A SHORT DISTANCE AHEAD ,BUT WE CAN
SEE PLENTY THERE THAT NEEDS TO BE DONE

Alan Turing

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

إضافة إعلانات Admob

إعلانات Admob التي تقدمها جوجل تتيح لك عرض إعلانات على تطبيقك والحصول على مبالغ مقابل تلك العروض.

بالبداية يجب أن تملأ حساب Ad mob لكي تستطيع بناء وحدتك الاعلانية ومن ثم الحصول على كود تلك الوحدة لكي تعرضها على تطبيقك.

المكتبة الأساسية التي سنستخدمها في هذا القسم هي مكتبة firebase_admob والتي يجب إضافتها كما فعلنا مع المكتبات السابقة، كذلك يجب استدعاء المكتبات الأساسية للـ .firebase

لنبدأ بشرح إضافة أنواع الإعلانات المختلفة لتطبيق فلاتر ، بالبداية علينا تحديد معلومات التطبيق من نوع الجهاز الهدف ، الكلمات المفتاحية المتعلقة بفكرة التطبيق، تاريخ النشر ، وإمكانية الاستخدام من قبل الأطفال والعديد.... نعرف كائن معلومات تطبيق كما يلي:

```
static final MobileAdTargetingInfo targetInfo
=new MobileAdTargetingInfo(
    textDevices:<String>[],  

    keywords:<String>[ 'music' , 'video' , 'picture' ],  

    birthday:<new DateTime.now(),  

    childDirected:<true,  

);
```

الآن سننشئ وحدتين اعلانيتين مختلفتين ، الأولى من نوع BannerAd والتي تظهر كما في الشكل التالي:



تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

يمكن تعريفها بالشكل:

```
BannerAd createBannarAd() {
    return new BannerAd(
        adUnitId:BannerAd.testAdUnitId,
        size:AdSize.bannar,
        targetingInfo:targeInfo,
        listener:(MobileAdEvent event)
    {
        print("Bannar event :$event");
    }
);
}
```

كما يمكن تغيير موضعها عن طريق الخاصية anchor

الثانية من نوع InterstitialAd والتي تملأ الشاشة كما في الشكل التالي:



يمكن تعريفها بالشكل:

```
InterstitialAd createInterstitialAd() {
    return new InterstitialAd(
        adUnitId:InterstitialAd.testAdUnitId,
        targetingInfo:targeInfo,
        listener:(MobileAdEvent event)
    {
        print("Interstitial event :$event");
    }
);
}
```

الوحدة الاعلانية الأولى يمكن استدعائهما عن طريق الطريقة الموروثة initState() بحيث نكتب ضمنها:

```
_bannerAd=createBannarAd()..show();
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

ونغلقها في الطريقة dispose كما يلي:

```
_bannerAd.dispose();
```

اما الوحدة الاعلانية الثانية يمكن استدعائها عن طريق أحد الازار لظهور كما يلي:

```
child: InkWell(
  onTap: () {
    createInterstitialAd()
      ..load()
      ..show();
  },
},
```

كذلك لا ننسى ان نضعها في الطريقة dispose كما يلي:

```
_bannerAd.dispose();
```

والان يكون الكود الكامل لكلا الوحدتين كما يلي:

```
import 'package:flutter/material.dart';

void main() => runApp(new MaterialApp(
));

class MyAppScreen extends StatefulWidget
{
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return new _MyAppScreen();
  }
}
class _MyAppScreen extends State<MyAppScreen>
{
  static final MobileAdTargetingInfo targeInfo=new
MobileAdTargetingInfo(
  textDevices:<String>[],
  keywords:<String>['music','video','picture'],
  birthday:new DateTime.now(),
  childDirected:true,
);
@override

  BannerAd _bannerAd;
  InterstitialAd _interstitialAd;

  BannerAd createBannarAd(){
    return new BannerAd(
      adUnitId:BannerAd.testAdUnitId,
      size:AdSize.bannar,
      targetingInfo:targeInfo,
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

```

        listener:(MobileAdEvent event)
        {
            print("Bannar event :$event");
        }
    );
}

InterstitialAd createInterstitialAd(){
    return new InterstitialAd(
        adUnitId:InterstitialAd.testAdUnitId,
        targetingInfo:targeInfo,
        listener:(MobileAdEvent event)
    {
        print("Interstitial event :$event");
    }
);
}

void initState(){
_bannerAd=createBannarAd()..show();
}
@Override
void dispose()
{
    _bannerAd.dispose();
    _interstitialAd.dispose();
}
@Override
Widget build(BuildContext context) {
    // TODO: implement build
    return new Scaffold(body: new Container(
        child: InkWell(
            onTap: (){
                createInterstitialAd()
                    ..load()
                    ..show();
            },
        ),
    );
}
}

```



ولكن يجب ان نعلم ان كل ما بنيناه هو لوحدات إعلانية تجريبية فيجب ان نضع رقم وحدتنا الصحيح التي انشأناها في حسابنا في Admob وتوضع كما يلي:

adUnitId: "ca-app-pub-4245565566455613/641455455455",

الى هنا تكون قد اضفنا الإعلانات المطلوبة تبقى لدينا ان ننشر التطبيق في احد المتاجر.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflowArabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

نشر تطبيقات اندرويد

متجر جوجل (Google Play Store)

متجر جوجل (Google Play Store) يمثل الكمية الرسمية لنشر تطبيقات الاندرويد . إن نشر تطبيقك على متجر جوجل يجعل من التحميل قابل للتحميل والاستخدام من قبل المستخدمين في جميع أنحاء العالم . يمكن أيضاً للمستخدمين إضافة تعليقات أو إعطاء تقييم للتطبيق بما يمكن من اكتشاف الأخطاء أو تحسين التطبيق لاحقاً . كما يوفر متجر جوجل بعض الإحصائيات التي يمكن الاستفادة منها لقياس نجاح أي تطبيق .

في هذه القسم سيتم شرح خطوات نشر التطبيق على متجر جوجل (Google Play Store) يمكن تلخيص خطوات نشر التطبيق على متجر جوجل بما يلي :

- تصدير التطبيق (Export) كملف من نوع (.APK) (Android Package)
- عمل توقيع رقمي (Digital Signature) للتطبيق باستخدام شهادة (Certificate)
- توقيع التطبيق رقمياً.

يساعد نظام اندرويد في تحديد هوية مالك التطبيق
رفع التطبيق على المتجر.

- استخدام سوق اندرويد (Android Market) لاستضافة وبيع التطبيق.

في ما يلي سنقوم بشرح خطوات إعداد التطبيق للتواقيع ومن ثم سنقوم بشرح طريقة نشر التطبيق:

• مراجعة ملف ال Manifest.xml

راجع ملف بيان التطبيقات الافتراضي AndroidManifest.xml الموجود في

`<app dir> / android / app / src / main`

وتحقق من صحة القيم ، وخصوصاً:

- (a) في الوسم application عدل الخاصية label لتعكس الاسم النهائي للتطبيق.
- (b) إزالة إذن android.permission.INTERNET إذا كان التطبيق الخاص بك لا يحتاج إلى الوصول إلى الإنترنت . يتضمن القالب القياسي هذه العلامة لتمكين الاتصال بين أدوات flutter والتطبيق الجاري تشغيله.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

- **مراجعة ملف التكوين Build:**

راجع ملف إنشاء ملف Gradle الافتراضي الموجود في

`<app dir>/android/app`

وتحقق من صحة القيم، وخصوصاً:

في القسم defaultConfig حدد معرف التطبيق ApplicationId واسم وكود النسخة
 كذلك حدد اصدارات النظام الأدنى المقبول او اصدارات
 جهاز الهدف للتطبيق .minSdkVersion & targetSdkVersion

- **أضف ايقونة خاصة لتطبيقك:**

أي تطبيق جديد له ايقونة افتراضية وهي رمز فلتر يمكنك تغييرها بأيقونتك الخاصة كما
 يلي:

بعد اعداد ايقونتك بما تتناسب مع احجام الايقونات الخاصة بالاندرويد افتح المجلد

`<app dir>/android/app/src/main/res/mipmap-`

ثم ضع ايقونتك الخاصة.

في الملف Manifest.xml حدث القيمة حسب اسم ملف الايقونة الخاص بك كما يلي:

`<application android:icon="@mipmap/my_icon"`

اعد تشغيل تطبيقك لتحقق من حقيقة استخدام ايقونتك الخاصة عوضاً عن تلك
الافتراضية.

- **إنشاء المفتاح الخاص بتطبيقك Keystore:**

في حال عدم تملكك لمفتاح خاص بتطبيقك عليك ان تولد واحداً بالتعليمات التالية في
الكونسول الخاص به flutter.

```
keytool -genkey -v -keystore ~/key.jks -keyalg RSA -  
keysize 2048 -validity 10000 -alias key
```

- **ربط المفتاح من التطبيق:**

أنشئ ملفاً اسمه Key.properties في المسار التالي:

`<app dir>/android/key.properties`

```
storePassword=<password from previous step>  
keyPassword=<password from previous step>  
keyAlias=key  
storeFile=<location of the key store file, e.g. /Users/<user  
name>/key.jks>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



مع العلم ان هذا الملف خاص بك وحدك ولا يجب ان يتطلع عليه الاخرون.

• عمليات ضبط التسجيل:

سنقوم الان في تعديلات شاملة في بعض الخصائص ، لنفتح الملف build.gradle الموجود في المجلد:

<app dir>/android/app/build.gradle

حيث نستبدل بعض التعليمات بتعليمات أخرى كما يلي:

الأساس:	البديل:
android {	
	def keystoreProperties = new Properties()
	def keystorePropertiesFile =
	rootProject.file('key.properties')
	if (keystorePropertiesFile.exists()) {
	keystoreProperties.load(new
	FileInputStream(keystorePropertiesFile))
	}
android {	

الأساس:	البديل:
buildTypes {	
release {	
// TODO: Add your own signing config for the	
// release build.	
// Signing with the debug keys for now, so	
`flutter run --release` works.	
signingConfig signingConfigs.debug	
}	
}	
signingConfigs {	
release {	
keyAlias keystoreProperties['keyAlias']	
keyPassword keystoreProperties['keyPassword']	
storeFile file(keystoreProperties['storeFile'])	
storePassword keystoreProperties['storePassword']	
}	
}	
buildTypes {	
release {	
signingConfig signingConfigs.release	
}	
}	

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

• **تفعيل الـ Proguard**

بالبداية ننشئ ملفاً باسم proguard-rules.pro في المسار التالي:

`/android/app/proguard-rules.pro`

نضع فيه ما يلي:

```
#Flutter Wrapper
-keep class io.flutter.app.** { *; }
-keep class io.flutter.plugin.** { *; }
-keep class io.flutter.util.** { *; }
-keep class io.flutter.view.** { *; }
-keep class io.flutter.** { *; }
-keep class io.flutter.plugins.** { *; }
```

الاعدادات السابقة هي اعدادات أساسية لحماية مكتبات فلاتر الأساسية وأي مكتبة خارجية

اضافة عليك إضافة قاعدتها الخاصة ضمن الملف ايضاً.

ثانياً علينا تفعيل الـ Proguard في ملف ال build.gradle بحيث نضيف ونعدل بعض

الاعدادات:

```
android {
    ...
    buildTypes {
        release {
            signingConfig signingConfigs.release
            minifyEnabled true
            useProguard true
            proguardFiles
                getDefaultProguardFile('proguard-android.txt'), 'proguard-
                rules.pro'
        }
    }
}
```

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

- تحرير التطبيق للنشر:

استخدم سطر الأوامر كونسول الخاص بـ flutter لكتابة التعليمات التالية:

1. cd <app dir> (replace <app dir> with your application's directory).
2. Run flutter build apk (flutter build defaults to --release).

التطبيق سيكون جاهزا في المسار:

`<app dir>/build/app/outputs/apk/release/app-release.apk`

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناته على Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

رفع التطبيق:

لكي تتمكن من رفع تطبيقك على متجر Google Store يجب ان تملك حساب مطور وان يكون لديك جاهزاً ما يلي:

- ملف APK الخاص بالتطبيق والذي تم توقيعه رقمياً.

- صور (screenshots) توضح واجهات التطبيق (صورتان على الأقل)

- وصف مختصر للتطبيق ووظائفه.

من الصفحة الرئيسية للمطور قم بالنقر على الرابط Upload Application حيث ستظهر الصفحة في شكل والتي من خلالها يتم رفع ملف APK وكذلك ملفات الصور .يجب ايضاً إدخال بعض البيانات الخاصة بالتطبيق مثل عنوان التطبيق، وصف له، آخر التعديلات على التطبيق نوع التطبيق وتصنيفه.

بعد نشر التطبيق على متجر جوجل، يمكن تتبع أي تعليقات يرسلها المستخدمون وكذلك أي أخطاء محتملة في التطبيق وعدد مرات تزيله.

تأليف، فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لـكامل محتويات الكتاب على قناة الـ Youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

المراجع

Eric Windmill , 2018. "Flutter In Action" USA

Macro L. Napoli, 2018. "Beginning Flutter A Hands On Guide To App Development"

Katby Walratb & Setb Ladd , 2009. "What is Dart"

Moises Belchin & Patricia Juberias 2014 ."Web Programming With Dart"

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح للكتاب على قناتي على youtube

يمكنك الحصول على شرح لكتاب كامل على قناتي على youtube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com

النحو

تأليف، فيصل الأسود | مهندس برمجيات
يمكنك الحصول على شرح لـكامل محتويات الكتاب على قنـاة الـYoutube



StackOverflow Arabi

تعلم المزيد عن برمجة التطبيقات : www.Learn-Barmaga.com



لغة دارت هي لغة برمجية تم إنشاؤها من قبل شركة جوجل وتنستخدم في تطبيقات الويب أو سطح المكتب وتطبيقات الجوال. تم ابتكار هذه اللغة من قبل Kasper Lund و Lars Bak وتم إطلاق أول إصدار منها في عام 2011.

من المهم أن نعلم أن Dart هي لغة Cross-Platform أي أنها تعامل على مختلف المنصات، كما أنها Native Flutter أي تتعامل مع العتاد وبشارة بدون مفسرات وسيطة وهذا يعطيها سرعة عالية جداً. أنها فلاتر Language فهي منصة تمكنتا من بناء تطبيقات جوال بواجهات رسومية معتمدين على لغة Dart. الذي يميز Flutter أنها تمكنتا من بناء تطبيقات لنظام مختلفة منها الاندرويد أو ال IOS الخاص بأجهزة Apple والمدهل أكثر أنه يمكن أيضاً استخدامها كاللغة الأولى لبرمجة تطبيقات نظام جوجل الجديد "فوشيا" Fuchsia والذي قد يزيل الاندرويد عن مكانه. يجب أن نعلم أيضاً أن Flutter تعتمد في تصمييمها Material Design التي تم بناؤها من قبل جوجل والتي تساعد في تصميم صفحات الويب.



فيصل الاسود | مهندس برمجيات
درست الاجازة في هندسة الحاسوب في جامعة حلب
العريقة واكملاً دراستي الدكتوراه مروراً بالماستر
في جامعة SRM الدولية للعلوم والتكنولوجيا.
اتمنى لكم التوفيق