تعلم Flutter

ملخص مبسط جدا لفهم أساسيات Flutter

تم جمع هذا المعلومات من كورس Flutter للمهندس وائل أبو حمزة من قناته على اليوتيوب

Dart مقدمة :

لغة دارت هي لغة برمجية تم أنشاؤها من قبل شركة جوجل وتستخدم في تطبيقات الويب أو سطح المكتب وتطبيقات الجوال تم ابتكار هذه اللغة من قبل Lars Bak و Kasper Lund وتم إطلاق أول أصدار 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 تمكننا من بناء تطبيقات جوال بواجهات رسومية معتمدين على لغة Dart الذي يميز Flutter أنها تمكننا من بناء تطبيقات لأنظمة مختلفة منها android أو الـ ios الخاص بأجهزة Apple والمذهل أكثر أنه يمكن أيضا استخدامها كاللـغة الأولى لبرمجة تطبيقات نظام جوجل الجديد "فوشيا Fuchsia "والذي قد يزيح android عن مكانه. يجب أن نعلم أيضا أن Flutter تعتمد في تصميميها Material Design التي تم بناؤها من قبل جوجل والتي تساعد في تصميم صفحات الويب.

أكثر الأسئلة الشائعة بين الناس حول تعلم البرمجة..!!

البرمجة : هي عبارة عن لغة تواصل بينك وبين الكمبيوتر الخاص بيك بحيث تطلب منه بمجموعة من الأسطر البرمجية طلبات وهو يقوم بتنفيذها هذا الشرح السهل والأقرب للـفهم .

تختلف لغات البرمجة عن بعضها البعض بطريقة كتابة الكود syntax فقط (اختلاف الشكل وطريقة الكتابة ولكن المضمون نفسه).

- ما هي لغة البرمجة التي يجب أن ابدأ بها أو ما هي لغة البرمجة الأكثر أنتشار وطلب :

1. **يجب تحديد الاختصاص المناسب لك :** مثلا تصميم مواقع الأنترنت أو برمجة تطبيقات هواتف الذكية أو مجال برمجة ذكاء الاصطناعي أو برمجة تطبيقات سطح المكتب أو.. أو... أو... يوجد كتير من اختصاصات ويجب عليك اختيار الاختصاص الذي تحبه ويجب عليك الاختصاص لأن مجال تكنلوجيا المعلومات مجال جدا كبير ويحتاج لأن تختص بأختصاص واحد فقط لتبدع فيه و كل اختصاص له لغات برمجة معينة وما هي الأدوات لازمة لهذا الاختصاص وكل اختصاص داخله اختصاص بمعنى أن مجال برمجة مواقع يوجد فيها عدة اختصاصات فروند أند / وباك أند وكل قسم منهم له لغات البرمجة الخاصة به.

2. **تنفيذ المشاريع والممارسة المستمرة :** يجب عليك أن تضع هدف وخطة معينة ويجب أن تخصص من وقت كل يوم ولو حتى ساعة واحدة لتعلم شيء جديد من هذا الاختصاص 3 ساعات يوميا هي استثمار جيد من وقتك يوميا.

3. **صبر على تعلم :** مجال البرمجة وتكنلوجيا المعلومات يحتاج إلى صبر كبير للـوصول إلى هدفك يجب أن يكون كل يوم لديك شيء جديد تعلمته بطرق وأساليب مختلفة أنا أنصح جدا بلتعلم من الأنترنت يوجد فيه العديد من مواقع المميزة لتعلم مثل موقه يوتيوب أو جوجل أو يوديمي أو كورسيرا وغيرها الكثير من المواقع التعليمية كثير من الناس تدخل إلى هذا المجال في البداية في همة ونشاط كبير ولكن تبدأ تدريجيا بلأنسحاب.

4. **بحث المستمر :** عن كل شيء جديد في مجال اختصاصك البحث المستمر هي احد المهارات المطلوبة التي يجب أن يتمتع بها المبرمج في حال حدوث أخطاء أو مشاكل و لأن هذه المجالات في تطور مستمر.

• ما هي افضل لغة برمجة في الوقت الحالي :

لا يوجد شيء اسمه افضل لغة كل لغة من لغات البرمجة لها تخصص معين وعمل معين نستطيع القيام به عن طريقها على سبيل المثال HTML CSS JAVA SCRIPT هي لغات مختصة بتصميم واجهات المواقع لغة بأيثون على سبيل المثال مختصة في مجال الذكاء الاصطناعي وهذا الأمر ينطبق تماماً على جميع لغات البرمجة لأن كل لغة ولها اختصاص معين.

• هل من الضروري أن أكون جيد في الرياضيات ..؟؟

ليس بلضرورة أن تكون ممتاز في الرياضيات يكفي فقط أن يكون لديك أساسيات.

- ## هل من الضروري أن أكون جيد في لغة الأنكليزية ..؟؟

ستكون لغة الأنكليزية أضافة قوي جدا لك أنت كمبرمج لأن 99 بلمئة من تعاملك سيكون بلغة الأنكليزية سواء خلال الرسمية لاختصاصك لاطلاع على لآخر تحديثات أو البرامج التي تعمل عليها أو في حال حدوث خطأ معين والبحث عن حل مشكلة هذا الخطأ أو أنك تريد متابعة كورس معين بلغة الأنكليزية (المحتوى الأجنبي اقوى بكثير من محتوى العربي) أو من الممكن أن تعمل مع شركة أجنبية عن بعد.

- ## هل من الضروري أن أكون طالب هندسة معلوماتية أو حواسيب حتى استطيع الدخول إلى عالم البرمجة..؟؟

لا ليس من ضروري أن تكون طالب هندسة حواسيب أو معلوماتية أي شخص كأن يستطيع تعلم البرمجة ولكن سيكون الطريق أمامه طويل وأيضا يحتاج إلى الاستفادة من أهل الخبرات في البداية حتى يسير على الطريقة الصحيح ويختار التخصص المناسب له.

- ## هل استطيع تعمل البرمجة من خلال الهاتف ..؟؟

نوعا ما نعم ولكن سيكون الموضوع صعب جدا وتحتاج إلى حاسوب لتعلم والممارسة.

- ## هل البرمجة تحتاج إلى حاسوب بأمكانيات كبيرة..؟؟

مواصفات عادية واقتصادية كافية لتعلم البرمجة والعمل عليه مثلا :

معالج دول كور مثل سيليرون من جيل الحديث مع رأمات 4 جيغا وكرت شاشة مدمج يفي بلغرض فقط يجب عليك أن تبدأ.

كيف تحفز نفسك للــاستمرار بتعلم البرمجة :

1. **لا تقارن نفسك بالآخرين :** أول سبب وضعته هو عدم مقارنة نفسك بالأخرين ،لأن من الطبيعي أن لا يكون مستواك متقدم أو تواجه بعض العقبات بالبداية.. لكن مع الاستمرار والإصرار على التعلم سوف تحقق هدفك!

2. **حدد أهدافك للــتعلم :** لازم تحط لك خطة وأهداف واضحه للــشيء اللــي ترد تعلمه، لأن بدون وضع خطة وأهداف تشعر نفسك ضائع ولا كأنك أنجزت أي شيء ، لذا مهم تحديد هدفك واتباع خطة لتحقيق الهدف

3. **تقبل أخطاءك :** كلنا نخطئ ولا احد معصوم من الخطأ ،لذا من الطبيعي أن تواجه بعض العثرات والأخطاء بالبداية و أثناء تعلمك فلا تجعلها تحبطك وتوقفك عن الاستمرار بالتعلم...

4. **خذ وقت للــراحة :** التعلم المستمر وبشكل متواصل يجعلك تشعر بالملــل بعد فترة ، حتى لو لم تلاحظ ذلك بالبداية ،لكن أجعل لك أوقات للــراحة وأوقات تتعلم فيها ، اخذ بعض من الراحة يصفي ذهنك ويجعلك تركز أكثر ...

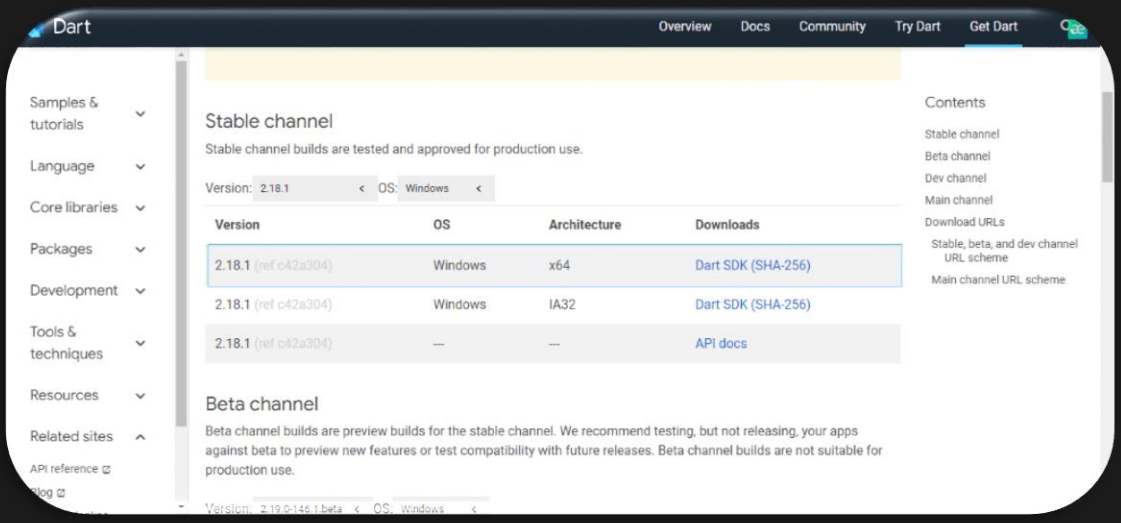
5. **نفذ بعض المشاريع :** متى ما شعرت بالملــل قم بتنفيذ أي مشروع بسيط ، فهو يحمسك للــاستمرار أكثر حتى لو مشروع بسيط جدا.

6. **احتفل بتقدمك :** راقب أداءك وبكل نقطه أو مفاهيم جديده تتعلمها قم بمكافأة نفسك و بالاحتفال بذلك ،حتى لو بشيء بسيط يجعل تشعر بالسعادة..

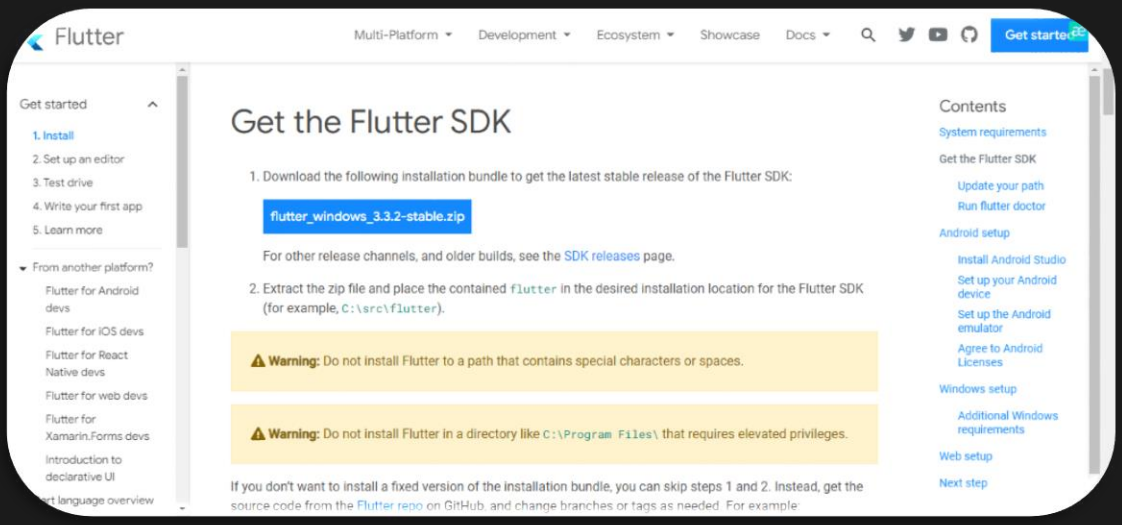
Install Flutter & Dart

تنزيل dart و flutter ومحرر الأكواد المناسب وتجهيز بيئة العمل بشكل كامل

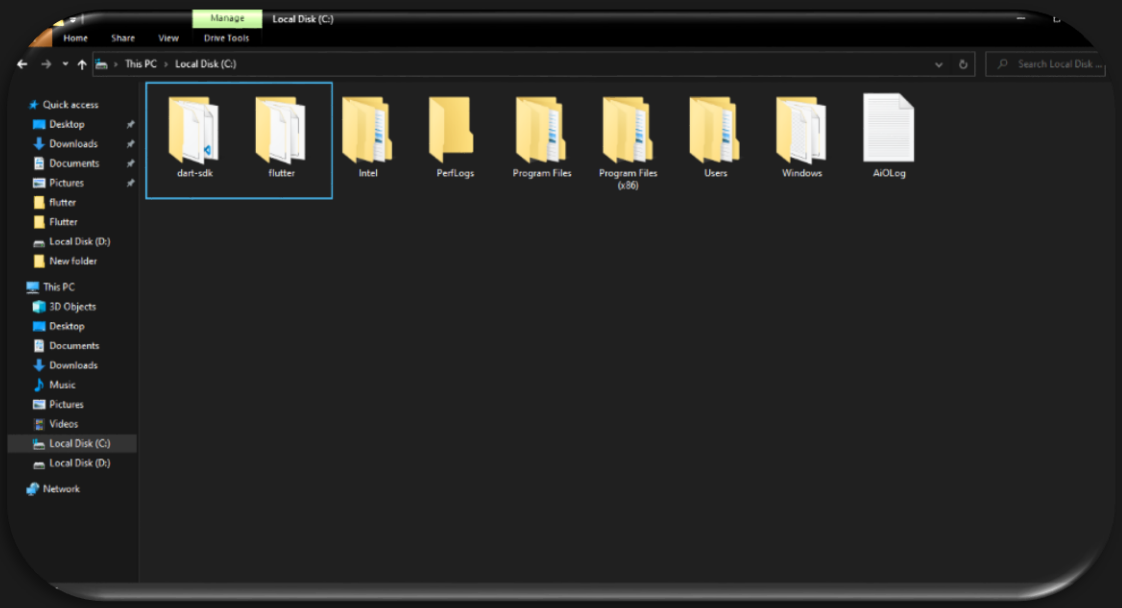
نتوجه بداية إلى الموقع الرسمي dart وتنزيل نسخة Stable channel ونختار النسخة على حسب نظام التشغيل سيتم تنزيلها على هيئة ملف مضغوط لدينا كما هو واضح لدينا في الصورة في الأسفل :



بعد الأنتهاء من Dart نتوجه إلى الموقع الرسمي Flutter ونختار النسخة على حسب نظام التشغيل سنقوم بتنزيل نسخة flutter_windows سيتم تنزيلها على هيئة ملف مضغوط لدينا كما هو واضح لدينا في الصورة في الأسفل :

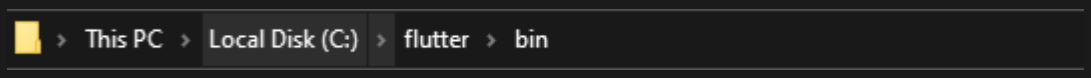


بعد فك الضغط عن ملفين Dart و Flutter نقوم بنقلهما إلى قرص النظام هو غالبا يكون القرص C كما هو واضح لدينا في الأسفل :

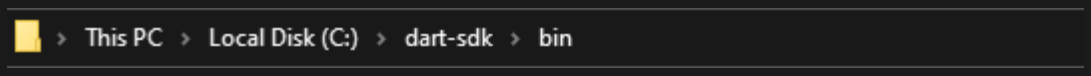


ثم نتوجه إلى مجلد flutter و مجلد Dart ثم إلى مجلد bin ننسخ المسارهما كما هو واضح لدينا في الصورتين في الأسفل :

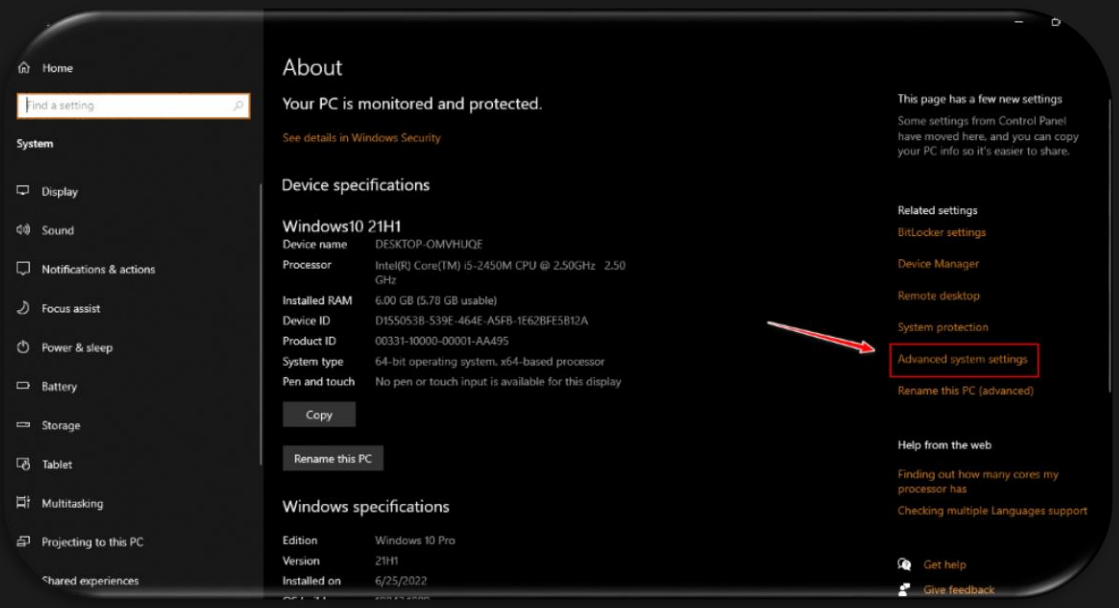
Folder flutter-sdk :



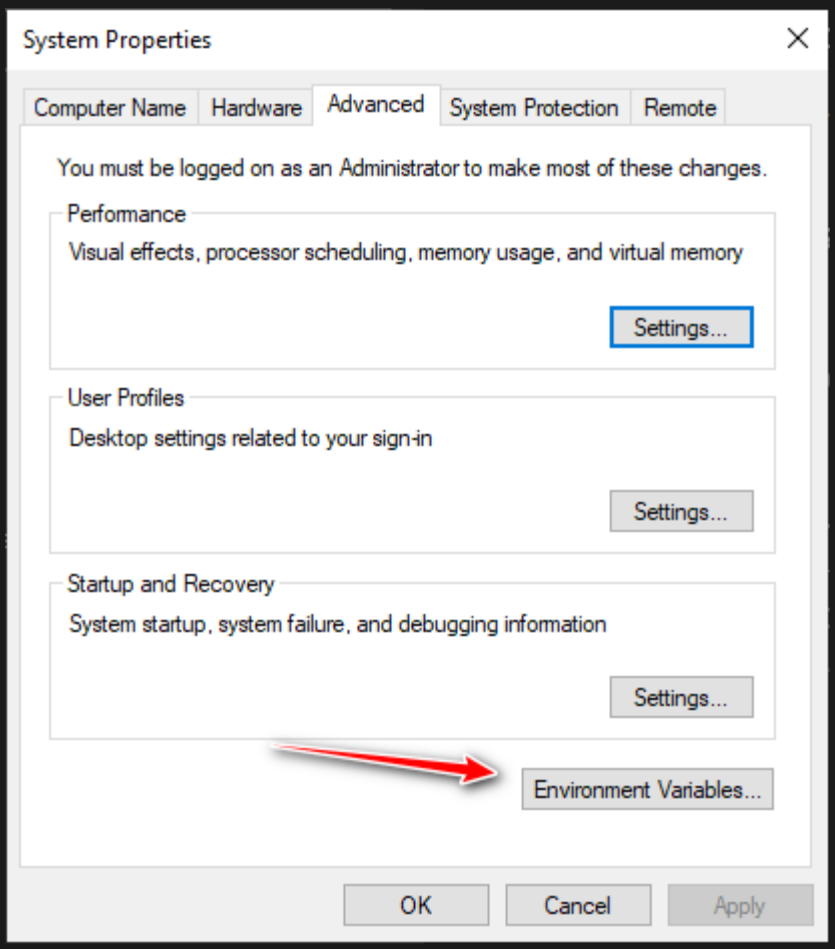
Folder dart-sdk :



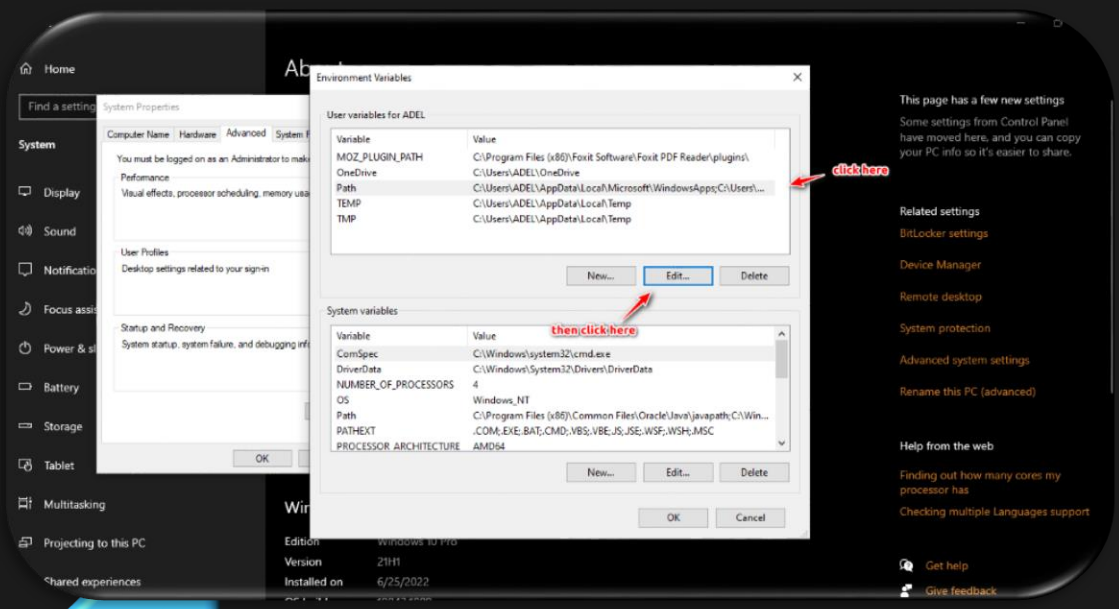
نتوجه الآن إلى خصائص الحاسوب ننقر على Advanced system settings كما هو واضح لدينا في الصورة في الأسفل :



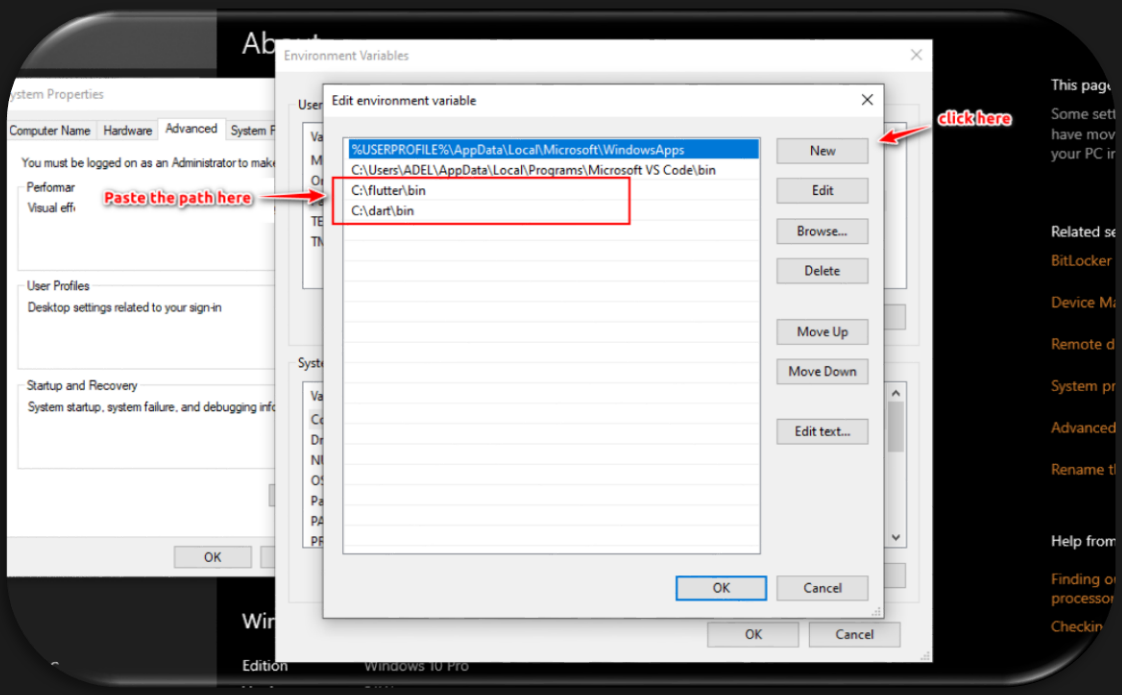
تظهر لنا هذه النافذة نقوم بلضغط على Environment Variables :



تظهر لنا نافذة جديدة نحدد على path ثم نضغط Edit كما هو واضح :

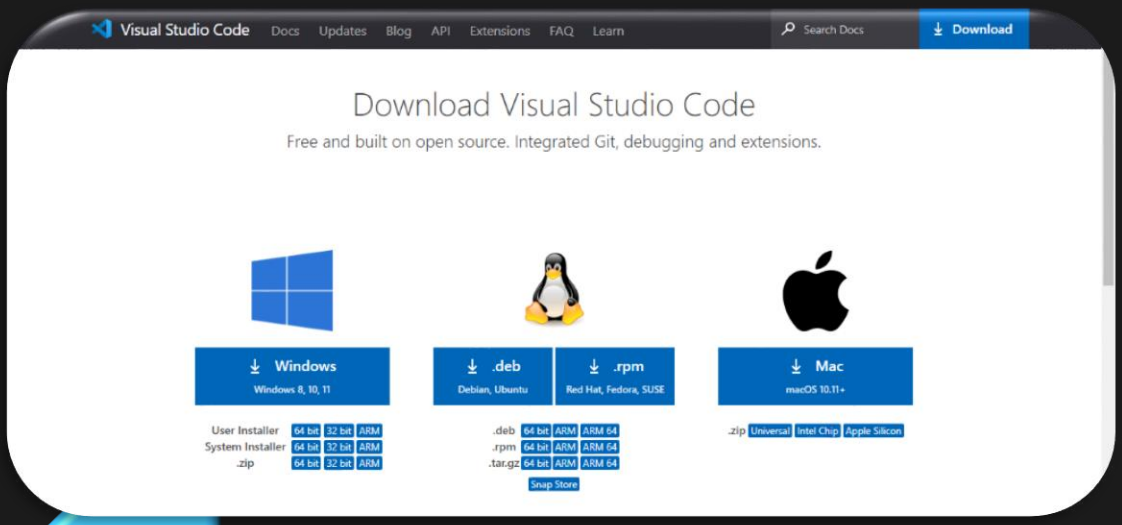


تظهر لنا نافذة جديدة نضغط على New ونقوم بوضع كل مسار بحقل لوحده ونضغط على زر Ok كما هو واضح لدينا في الصورة في الأسفل :

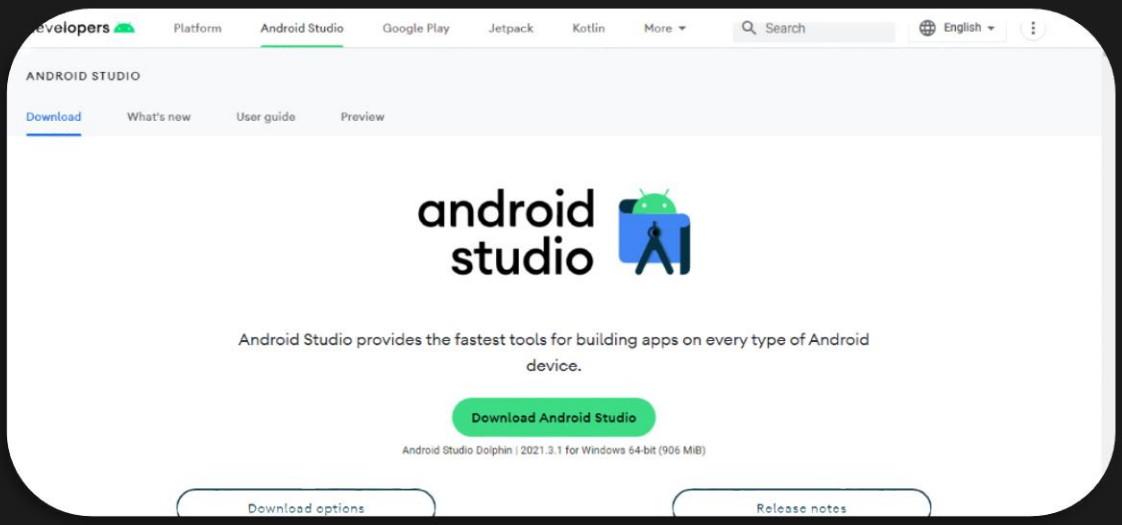


بهذا يكون قد تم تثبيت Dart و Flutter.

في حال كأن حاسوبك الشخصي مواصفاته ضعيفة فأن محرر الأكواد Visual Studio Code جدا مناسب لك نتوجه إلى موقعه الرسمي لتنزيله ونقوم بتحميله كما هو واضح :

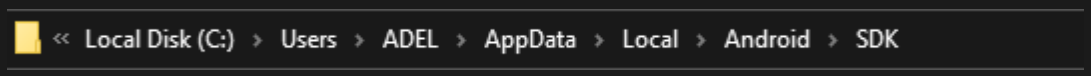


نتوجه الأن إلى الموقع الرسمي Android studio لتنزيله و من خلاله نقوم تنزيل جميع الأدوات وملفات الهامة مثل Android sdk أو محاكي Emulator

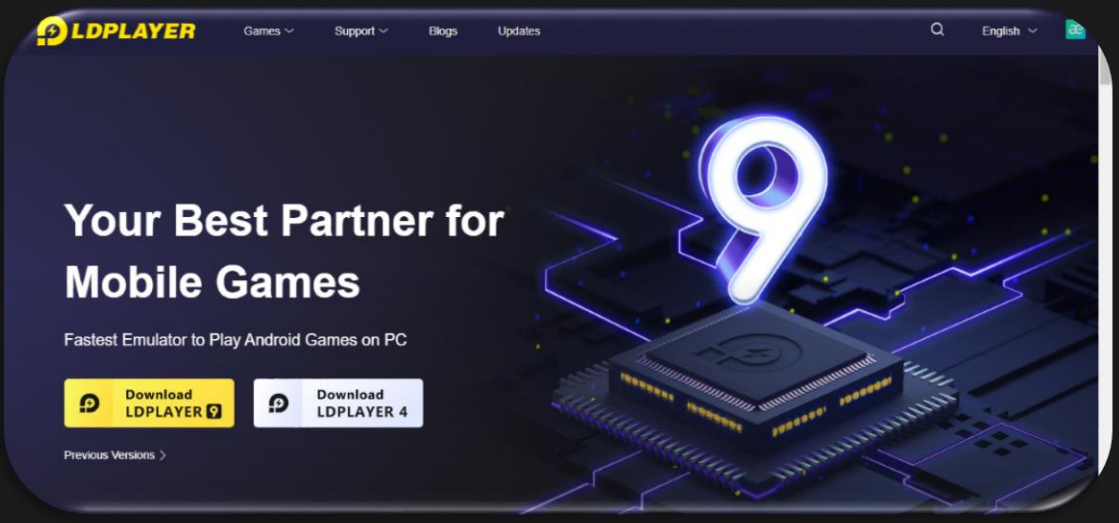


في بعض الأحيان نواجه مشاكل جدا كبيرة أثناء تنزيل ملفات Android studio ربما تكون الدولتك محظور فيها خدمات جوجل بهذا سيتوجب عليك استخدام vpn أو بسبب ضعف الأنترنت لديك (غالبا هذه الملفات يكون حجمها كبير نوعا ما) أو بسبب اسم مستخدم في جهاز الحاسوب لديك يجب أن يكون بلغة الأنكليزية وعبارة عن كلمة واحدة ومن دون فراغات أو رموز والا هذا الأمر سيسبب مشاكل عند تثبيت ملفات Android studio.

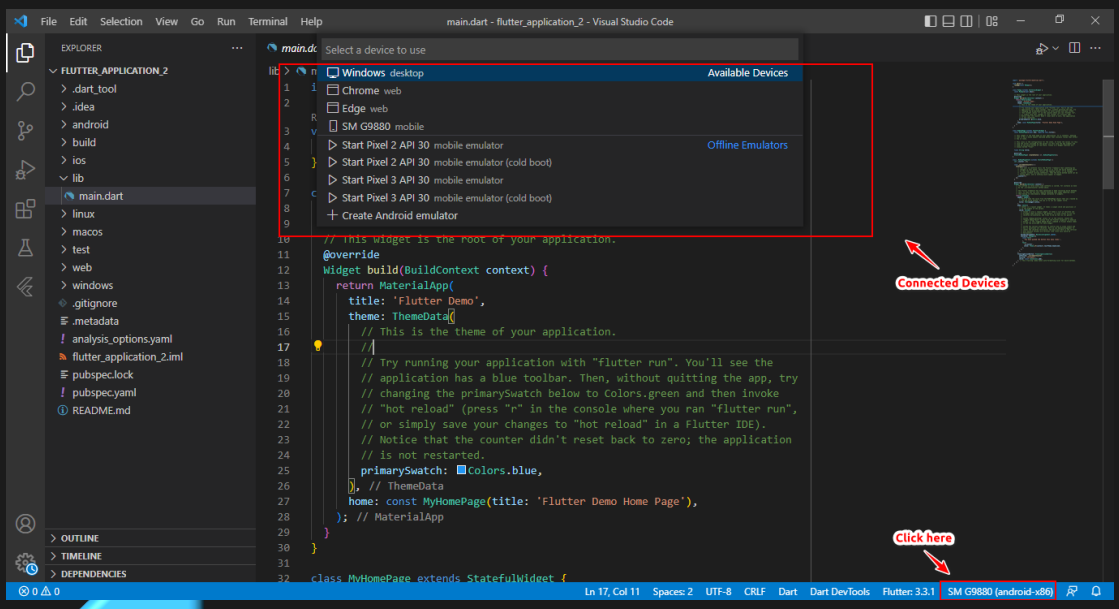
يوجد حل مناسب لحل جميع هذه المشاكل تستطيع الحصول على ملف sdk وتثبيته على حاسوبك بشكل مستقل بهذا المسار طبعا يجب تفعيل ظهور الملفات الخفية لأن مجلد AppData يكون مخفي في مجلد Users.



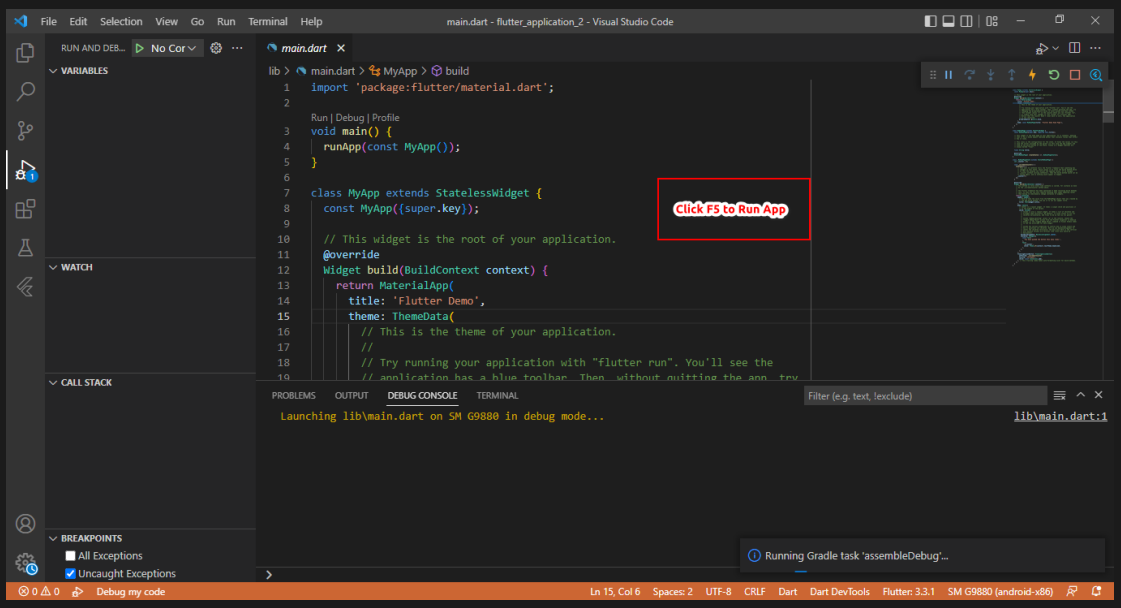
نتوجه الأن إلى موقع الرسمي لمحاكي LD PLAYER ونقوم بتنزيله هو من افضل المحاكيات الخفيفة ومناسب للـحواسيب ذات الإمكانيات البسيطة تتوفر عليه خدمات جوجل بشكل كامل.



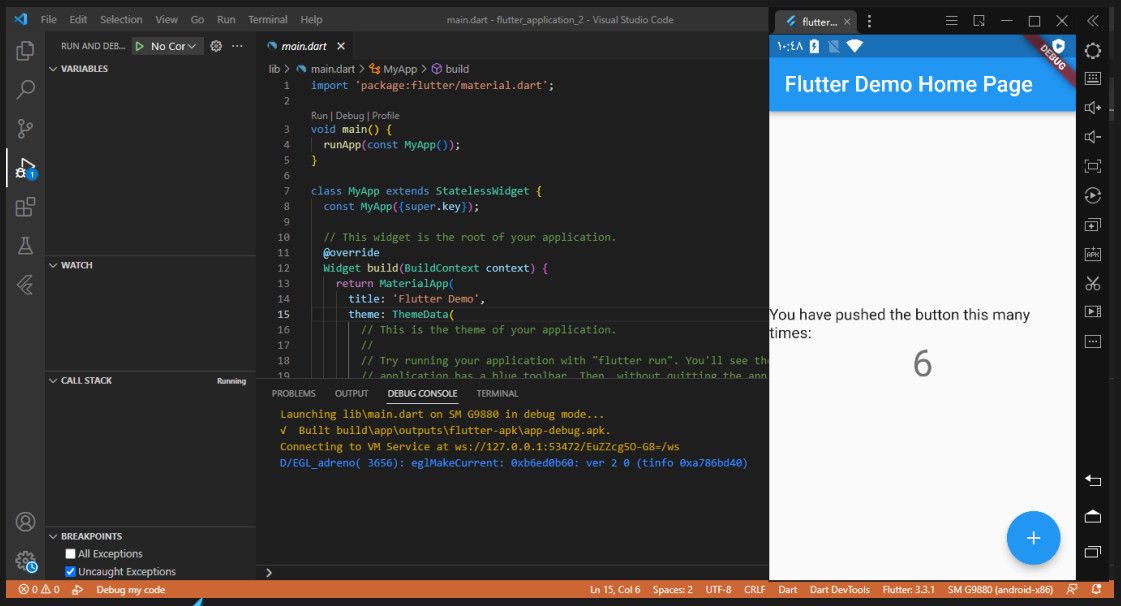
نقوم بأنشاء أول Project على Visual Studio Code تستطيع التعرف على الطريقة في الأسفل Create First Project نقوم بلضغط على connected devices تظهر قائمة بلأجهزة المتصلة التي نستطيع عمل Run App من خلالها كما هو واضح لدينا في المثال في الأسفل :



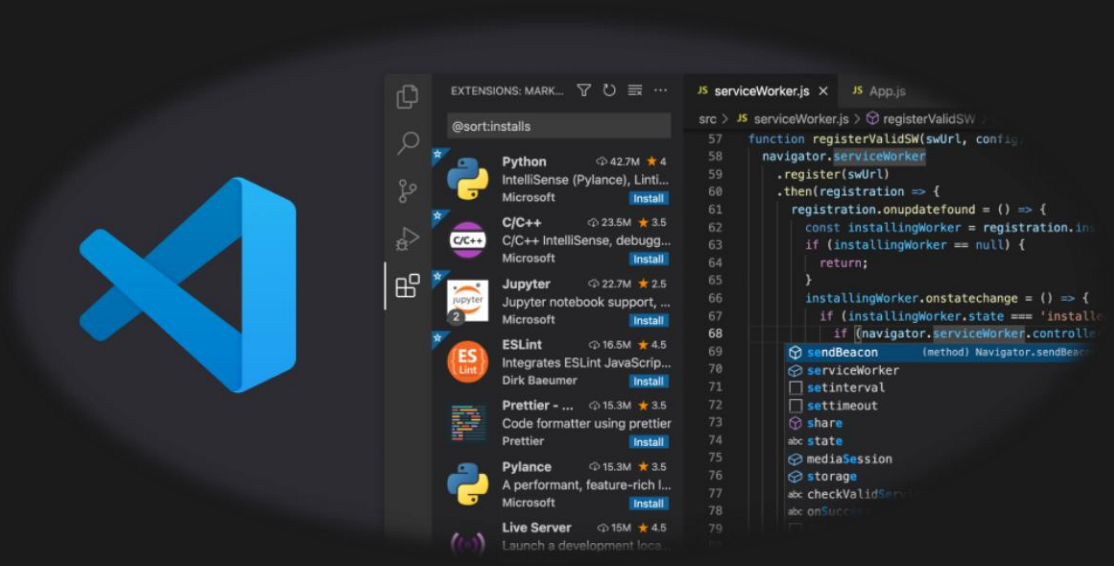
نقوم بلضغط على SM G9800 هو محاكي LD PLAYER نضغط على F5 لعمل Run App للتطبيق ثم يبدأ Visual Studio Code ببناء apk على المحاكي كما هو واضح لدينا في الصورة في الأسفل :



في بدأية أنشاء أول تطبيق ستستغرق عملية البناء بعض الوقت تصل إلى 10 دقائق ثم يعمل التطبيق بهذا الشكل ومن هنا نستطيع بدأ العمل

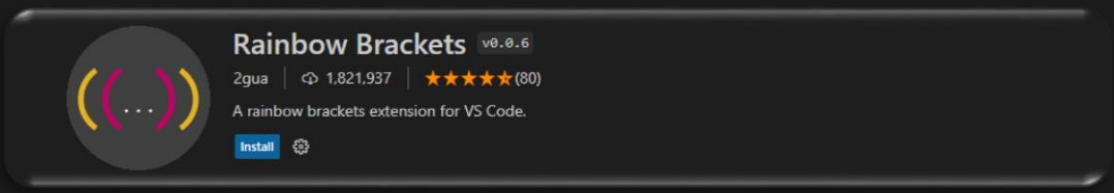


أهم أضافات Visual Studio Code التي يحتاجها مطوري Flutter



Rainbow Brackets .1

تقوم بأضافة نفس اللون في الأقواس المتطابقة مما يجعل الوصول للأقواس أمر سهل كما أنه يجعل الكود يبدو جميل.

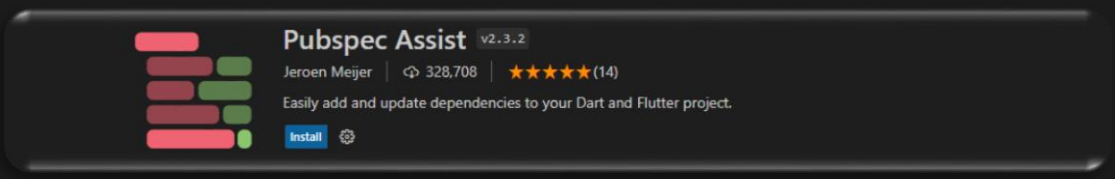


Error Lens .2

تسهل عليك معرفة الخطأ أثناء التكويد حيث أنه يظهر الخطأ بجانب السطر بلون الأحمر يمكن تغيير لون عن طريق ملف settings.json.

Pubspec Assist .3

يمكن أن تكون إضافة مكتبة إلى ملف pubspec.yami مملة ومتعبة يجب عليك زيارة pub.dev والبحث عن المكتبة ثم أضافتها إلى ملف pubspec.yami بشكل يدوي امر متعب جدا أضافة Pubspec Assist تحل لك هذه المشكلة بسهولة لأنه يمكنك البحث المباشر من خلالها وأيضا تقوم بأضافة المكتبة المطلوبة بشكل مباشر داخل ملف pubspec.yami.

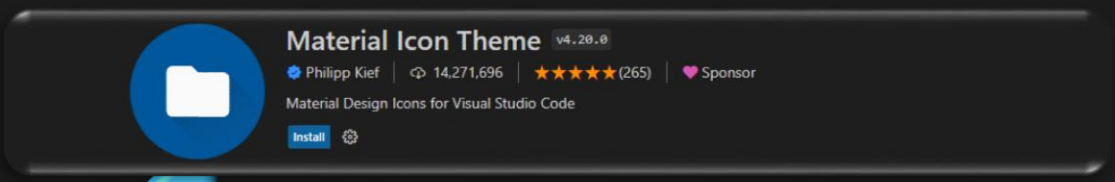


Awesome Flutter Snippets .4

من خلالها تستطيع الحصول على الكثير من الأختصارات فقط نقوم بكتابة اسم Widget وهو يقوم ببناء الكود الخاص بها بكبسة زر.

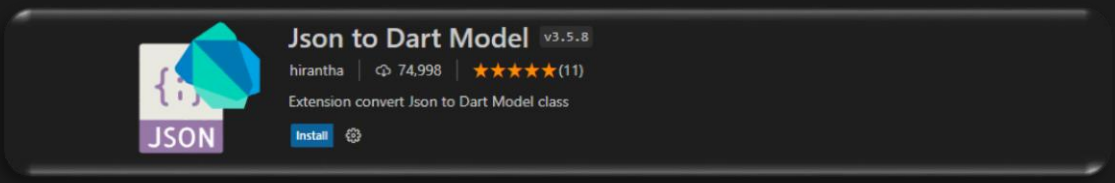
Material Icon Theme .5

أضافة أشكال مختلفة لكل Folder



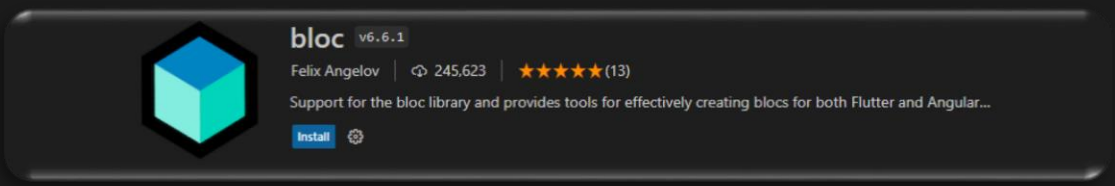
Json to dart model .6

تمكنك من تحويل ملف json إلى model به كل ما تحتاج حيث يتم أنشاء constructor وأيضا دوال toJson() / formJson() فقط اضغط وسيتم تحويل الملف كاملا ويوفر عليك الكثير من الوقت.



Bloc.7

تتيح لك خياريين : أنشاء Cubit وتقوم بأضافة Cubit و State والكود الخاص بهما أو أنشاء Bloc تقوم بأضافة Bloc و Event و State والكود الخاص بهما.



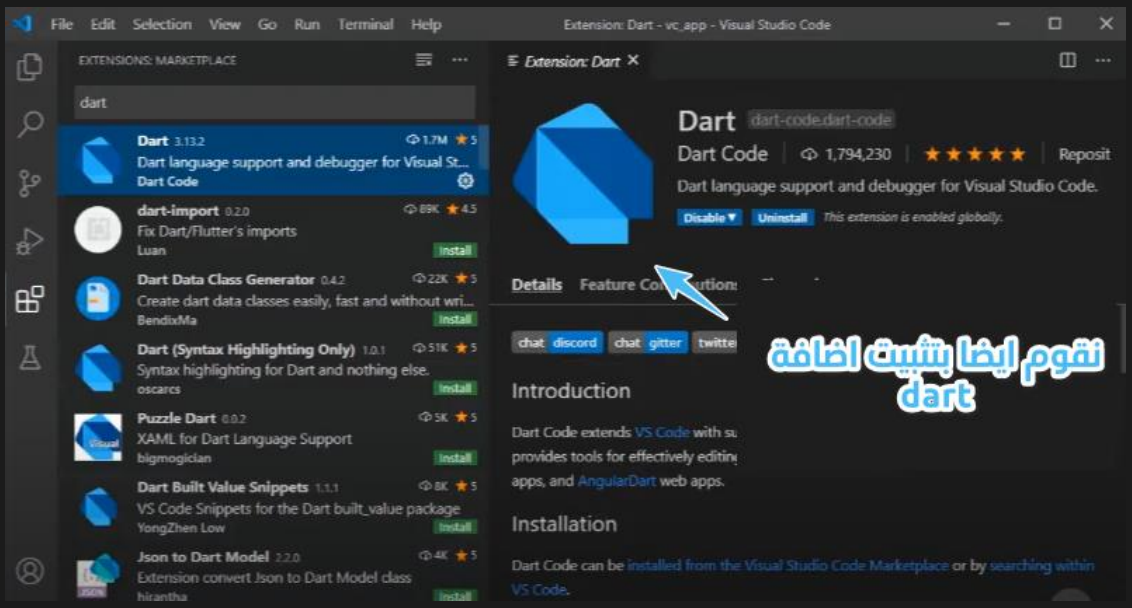
Ayu .8

أضافة تتيح لك تغيير them وتوفر لك عدد من them الحمية جدا لبرنامج Visual Studio Code.

Create First Project

بداية سنقوم بتثبيت الإضافات المهمة في visual studio code

إضافة Dart كما هو واضح في المثال في الأسفل :

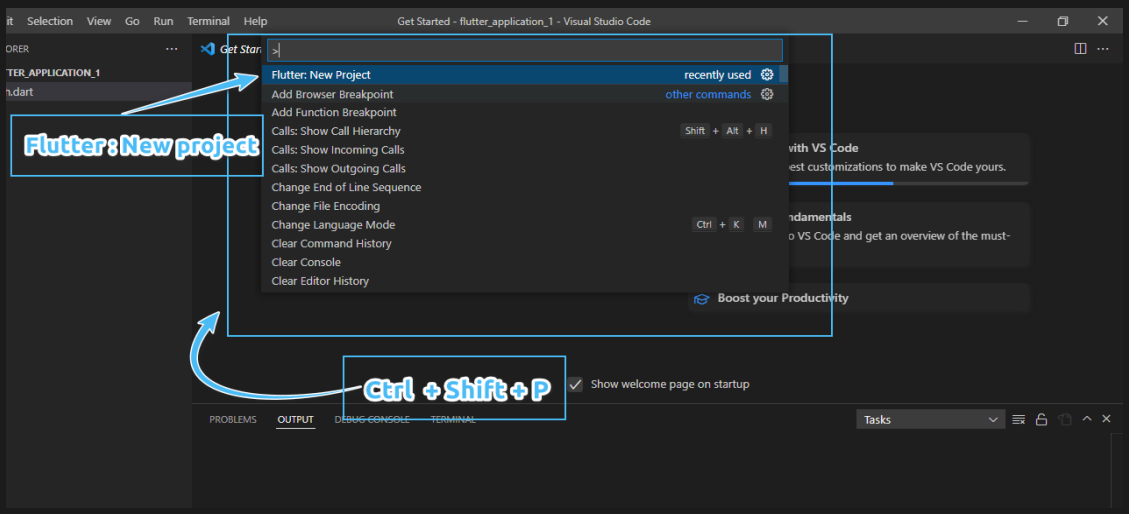


وأضافة Flutter أيضا من قسم Extensions:

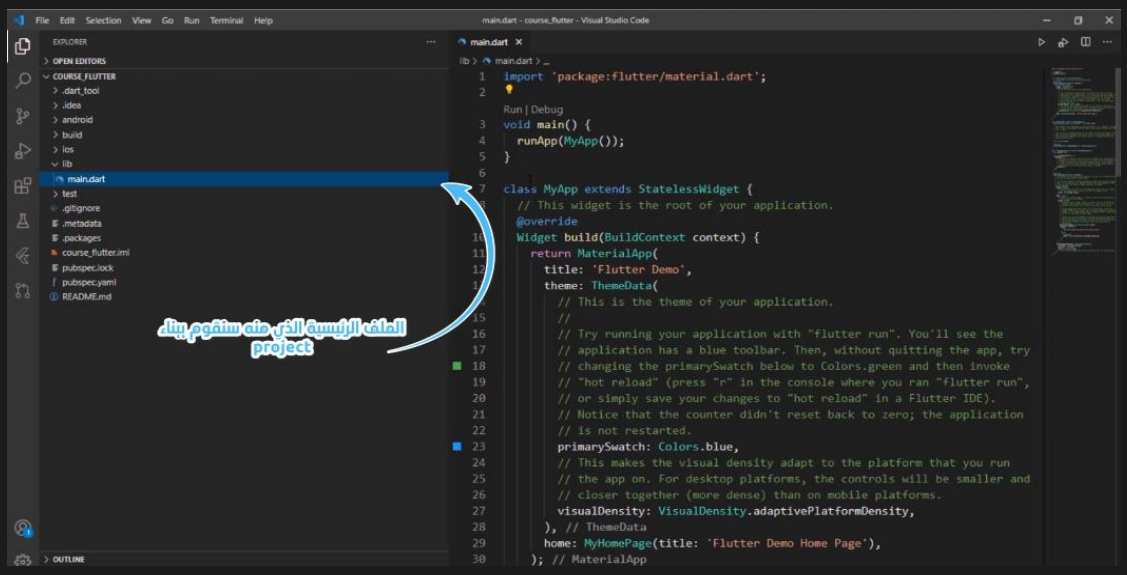
لأنشاء أول project على visual studio code نقوم بلضغط على

Ctrl + Shift + P ومن ثم بكتابة Flutter : New project

ثم يظهر لنا مربع أخر نقوم بكتابة اسم project واختيار مكان الحفظ كما هو واضح لدينا في المثال في الأسفل :

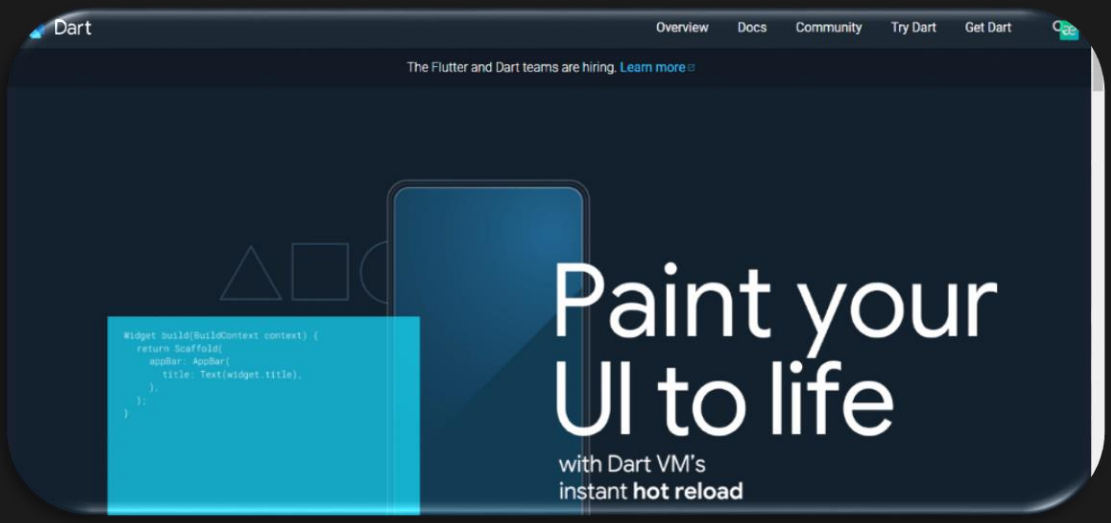


بعد أنتهاء من تجهز بيئة العمل نقوم بلتوجه إلى المجلد lib يحوي بداخله على ملف اسمه main.dart وهو الملف الرئيسي للـ project كما هو واضح لدينا في الصورة في الأسفل :



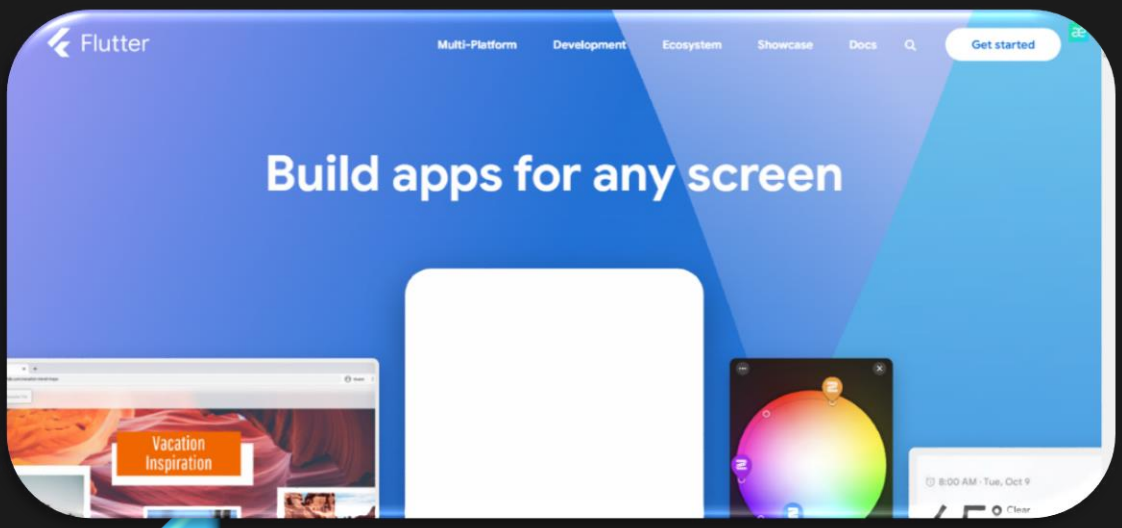
Important sites for every Flutter developer

موقع الرسمي للـغة Dart تستطيع من خلاله الاطلاع على أخر التحديثات شركة :



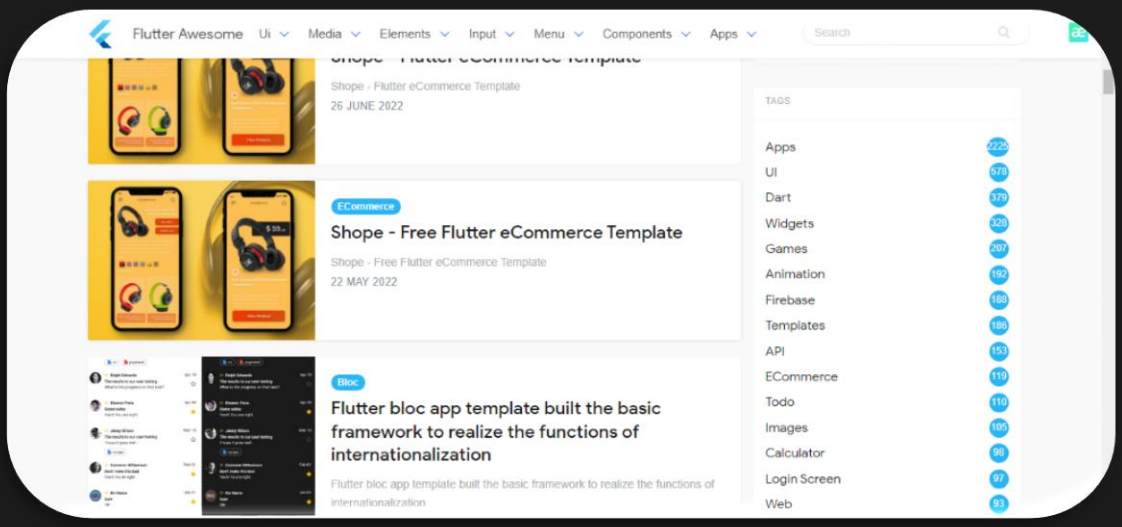
الموقع الرسمي لفلاتر وهو المصدر الرئيسي لتعلم فلاتر Documentation

flutter.dev



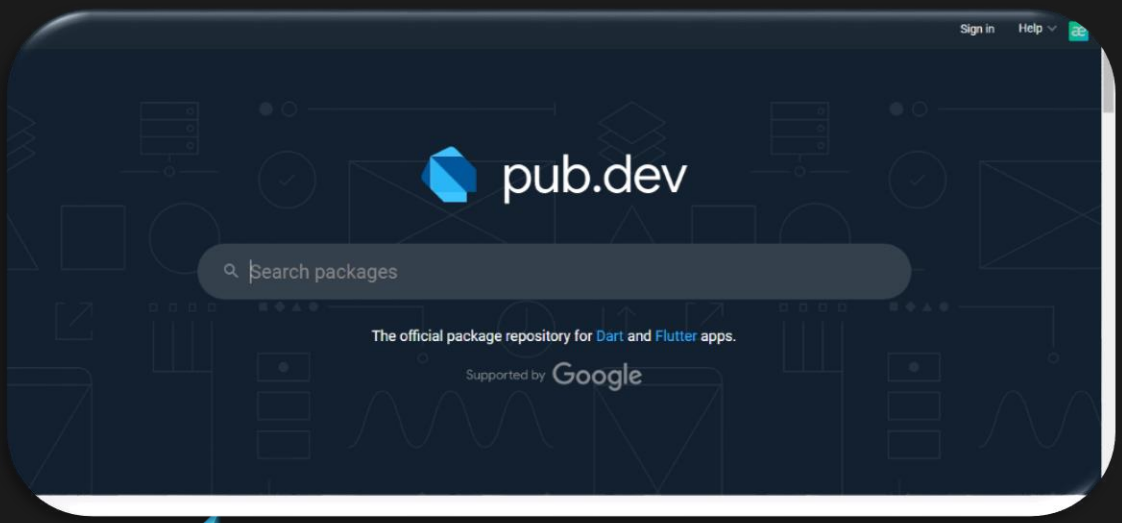
موقع يحتوي على الكثير من التطبيقات مفتوحة المصدر كما يضم عدد كبير من عناصر الشاشة والخدمات التي يمكن اطلاع على أكوادها

flutterawesome.com



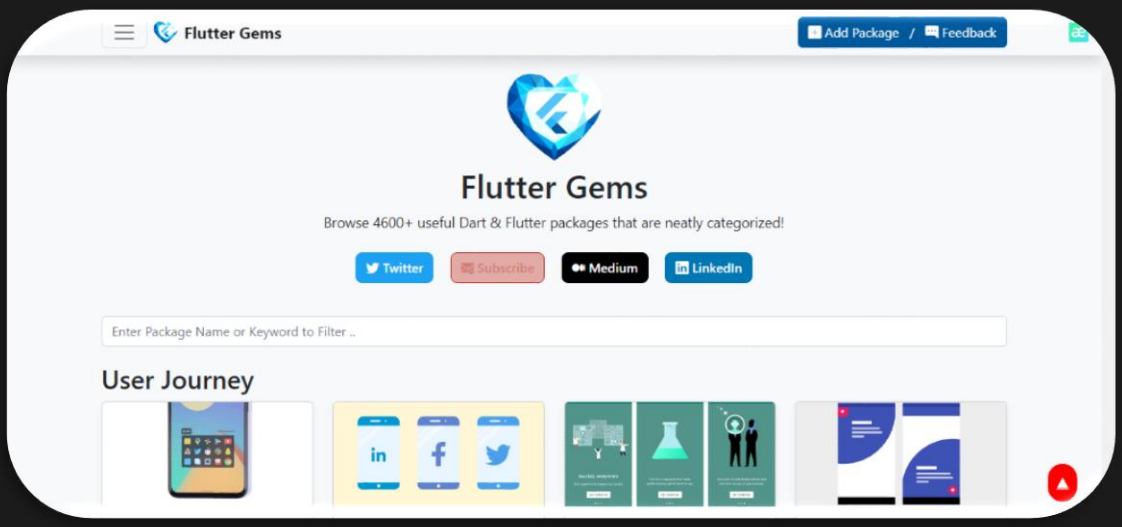
موقع حزم فلاتر الرسمي يحوي على الكثير من Packages التي يمكنك أن تستخدمها وأضافتها في مشروعك فهي تسهل عليك بناء مشروعك

pub.dev

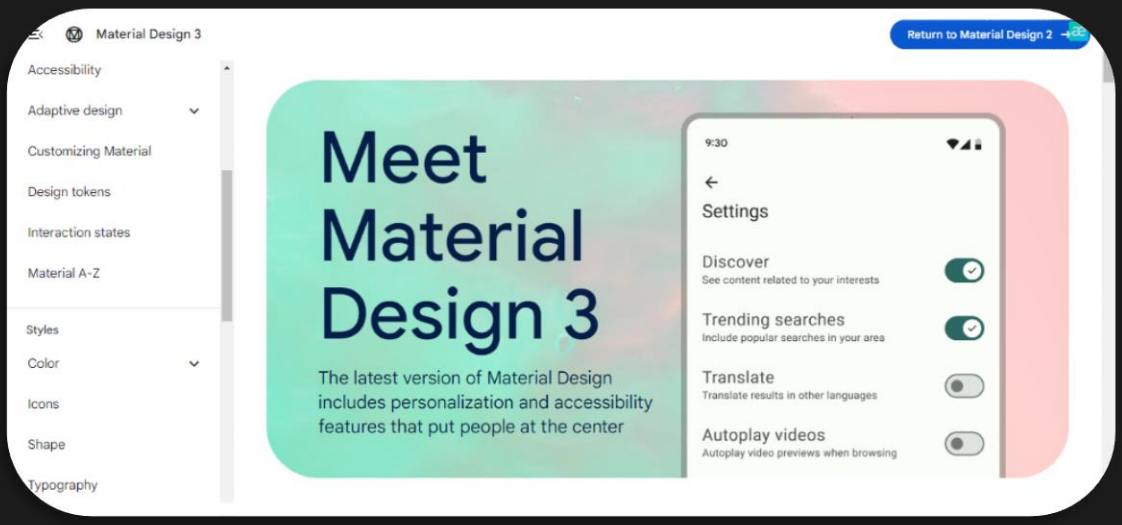


أيضا تحتوي على عدد كبير من Packages معروضة بطريقة مرتبة تسهل عليك البحث من الموقع لرسمي pub.dev

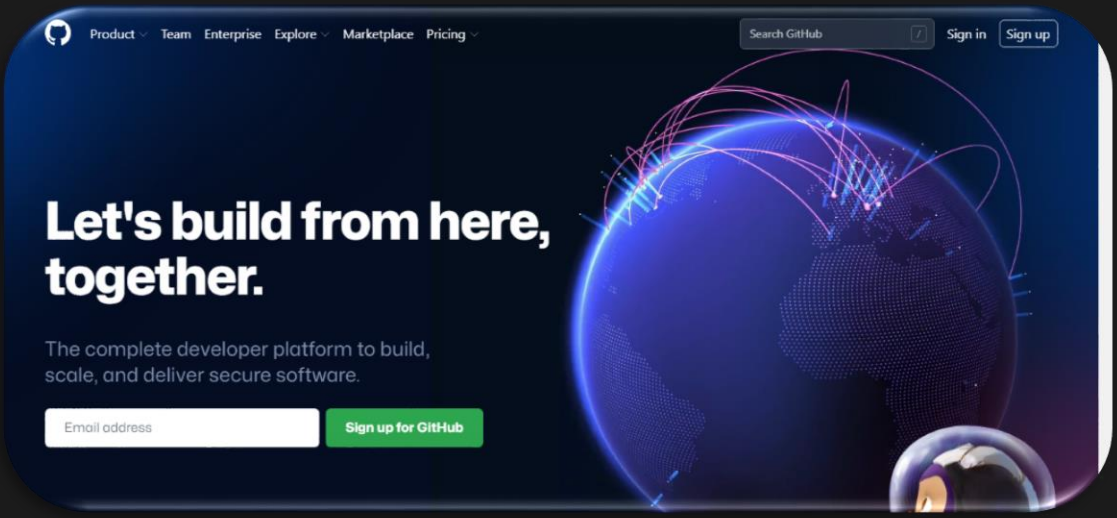
fluttergems.dev



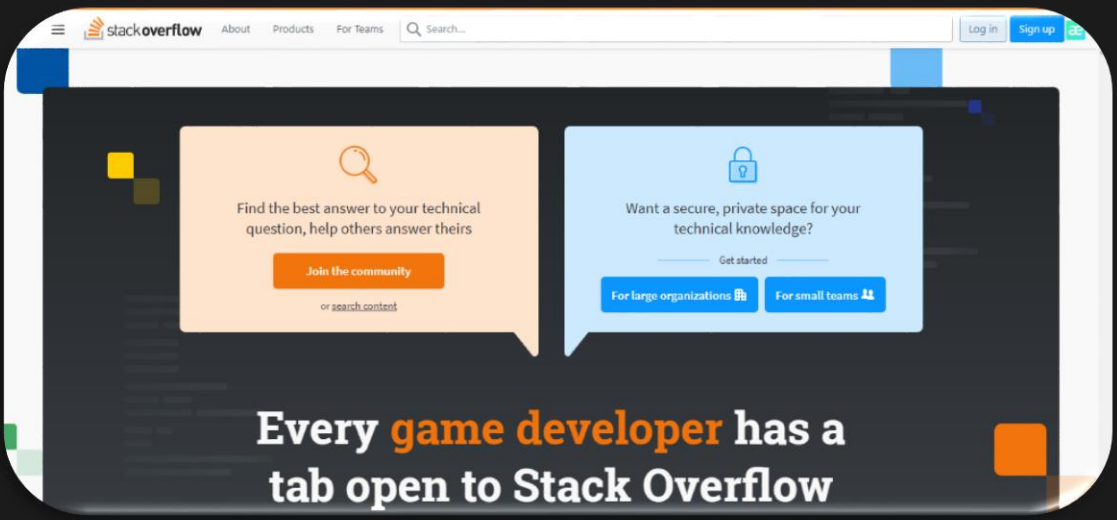
موقع يحتوي على عدد كبير من عناصر الشاشة UI مفتوحة المصدر التي يمكن الاستفادة منها في مشروعك m3.material.io



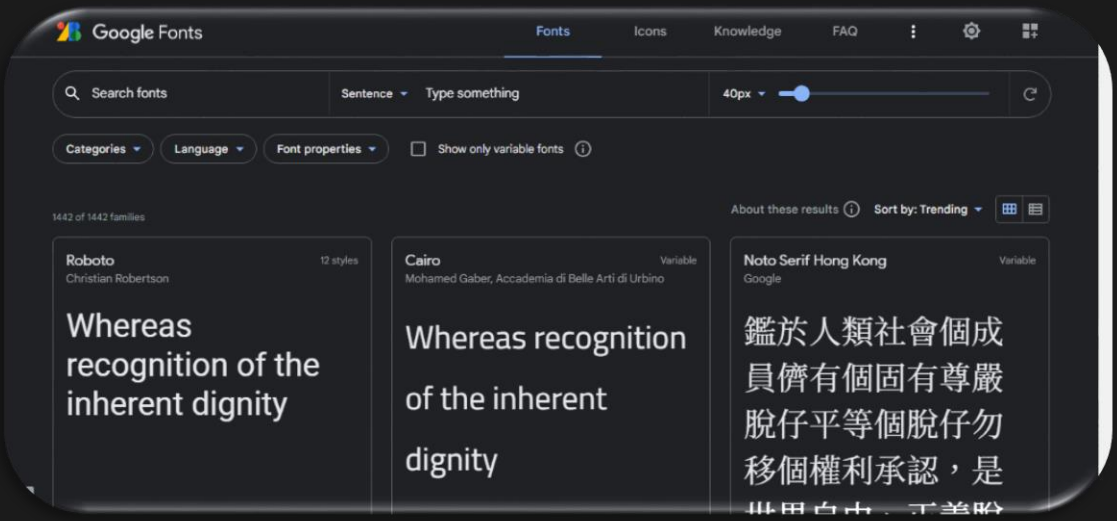
منصة ضخمة ومشهور جدا لدى المبرمجين تستطيع من خلالها استضافة المشاريع الخاصة بك ومشاركتها والتعديل عليها github.com



منصة جدا مشهورة ومفضلة لدى المبرمجين ولمطورين تضم عدد كبير من المشاكل والحلول التي من الممكن أن تواجهك أثناء عملك كما يمكنك طرح مشكلتك هناك stackoverflow.com



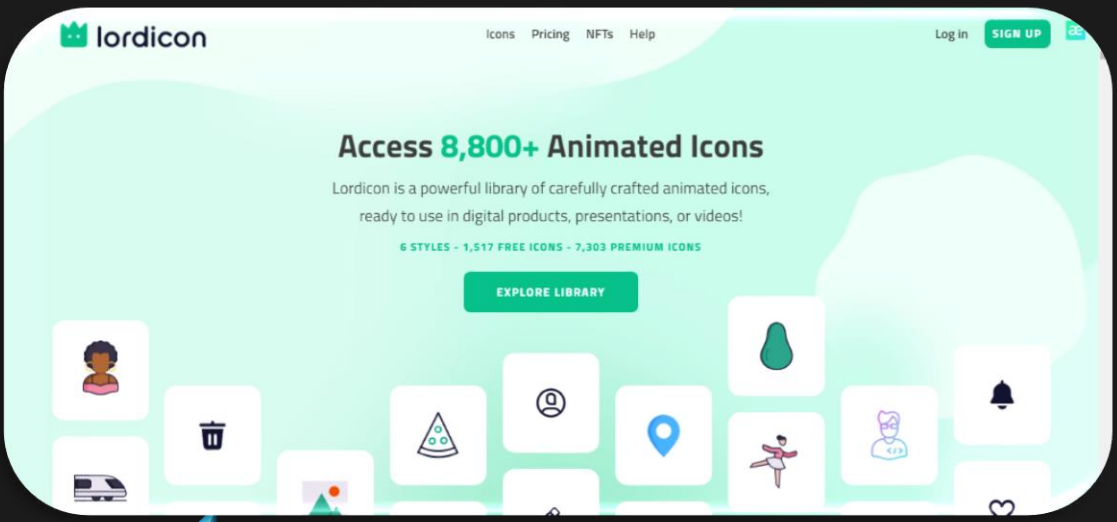
هو موقع من شركة جوجل يحوي على عدد كبير من خطوط المجانية التي تستطيع أن تستخدمها في مشروعك fonts.google.com



موقع يضم أكثر 8800 من رموز المتحرك

تعد lordicon.com مكتبة قوية من الرموز المتحركة المصممة بعناية

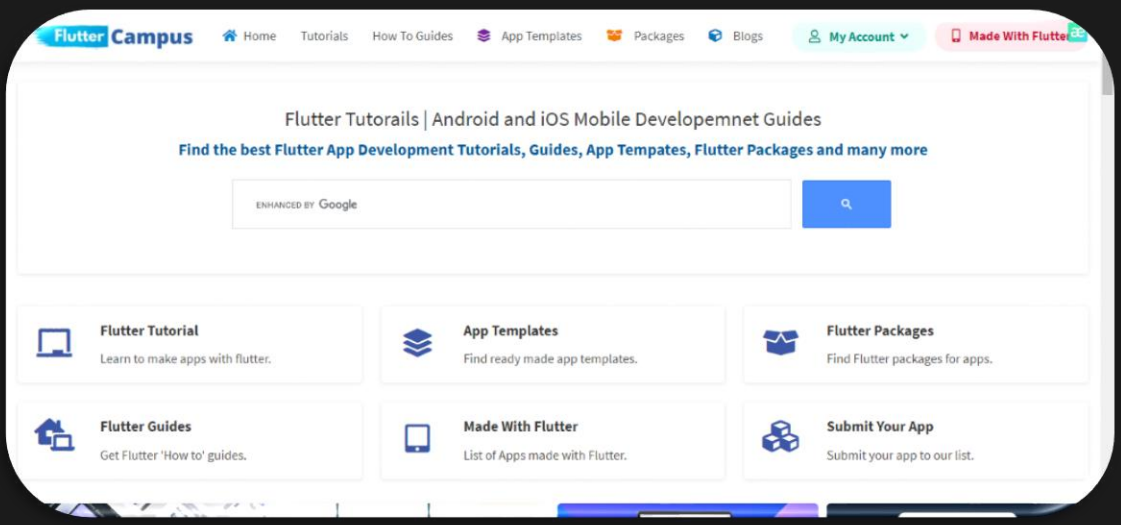
جاهز للأستخدام في المنتجات الرقمية أو العروض التقديمية أو مقاطع الفيديو!



مواقع أكواد وتصاميم Flutter جاهزة ممكن الاطلاع عليها والأستفادة منها في مشروعك الخاص.

موقع Get Widget :

موقع Flutter Campus :



شرح الموقع الرسمي لـ Flutter

(الحصول على المعلومة من المصدر الأساسي والرسمي)

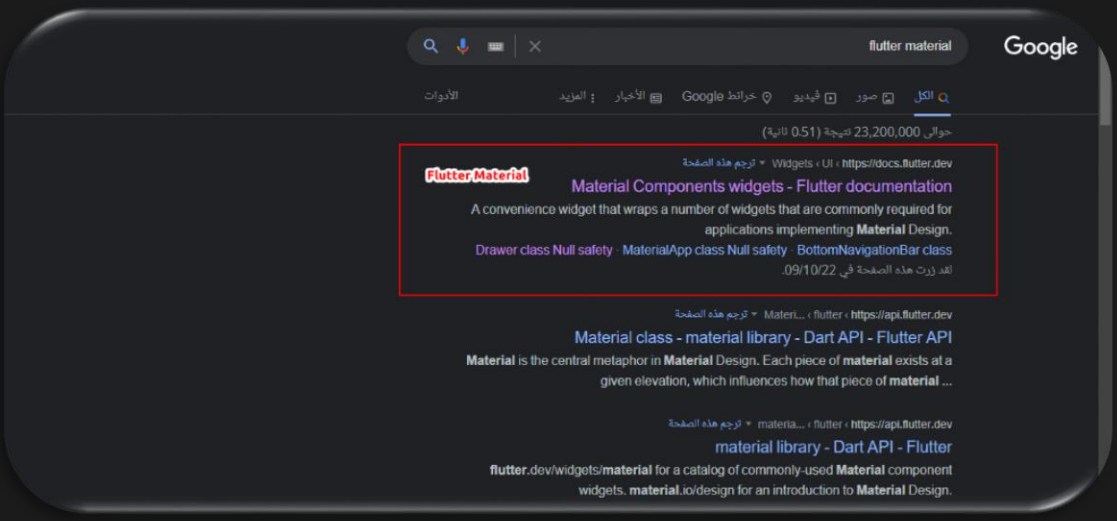
أنت كشخص جديد في البرمجة أو في flutter تقوم في البداية في متابعة كورس معين على اليوتيوب أو من على أحد المواقع التعليمية بلتأكيد يجب عليك في البداية (مبتدأ) تعلم الأساسيات flutter القوية جدا بلأضافة لأهم Widget وأكثرها شيوعا بلأضافة إلى Properties التي تحويها هذه Widget.

- ولكن هذه الكورسات و بكل تأكيد لا يمكن أن تشمل شرح جميع Widget بلأضافة إلى Properties التي تحويها وهذا يحتاج إلى شرح جدا جدا طويل ربما يكون مئات الساعات وهذا غير منطقي لأن flutter جدا ضخمة وتحوي عدد كبير جدا من Widget و Properties.
- أو في حال صدور تحديث جديد لـ flutter يجب عليك معرفة هذه التحديثات بنفسك.
- أو أيضا يمكن أجراء بعض التعديلات من قبل مطورين flutter فيجب عليك أيضا معرفة هذه التعديلات من خلال الموقع الرسمي.

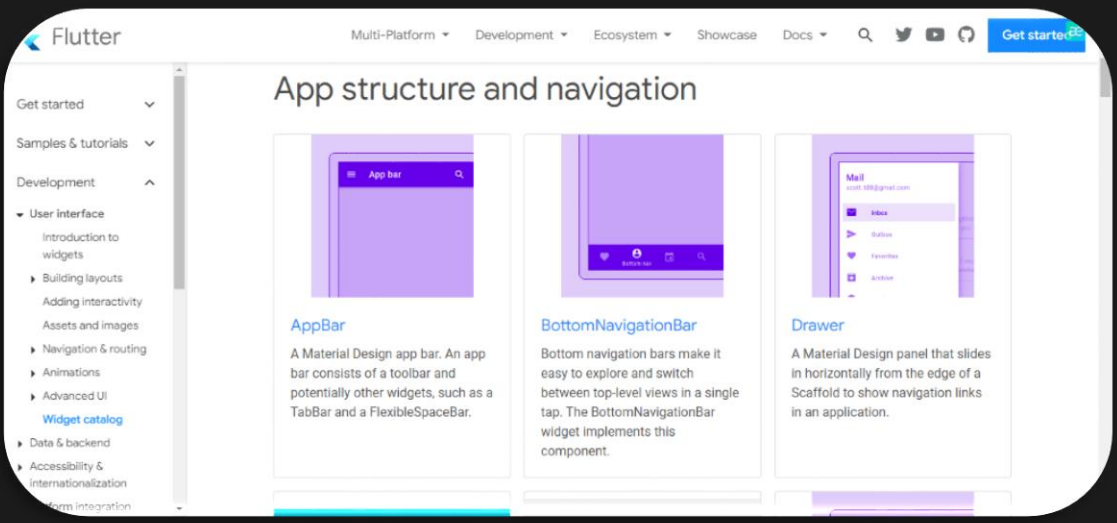
فما هو الحل ..؟؟

من مهمة جدا يجب عليك معرفة طريقة البحث والحصول على المعلومة بنفسك تابع معي....

نقوم بفتح أي متصفح ونقوم بكتابة كلمة flutter material في مربع البحث ونقوم بلدخول إلى أول نتيجة بحث تظهر لدينا كما هو واضح لدينا في الصورة في الأسفل :



تظهر لنا هذه النافذة وهذا هو الموقع الرسمي لـ Flutter قسم Widget بلطبع هذا القسم يحوي على جميع Widget في flutter كما هو واضح لدينا على سبيل المثال نقوم بلضغط على Drawer



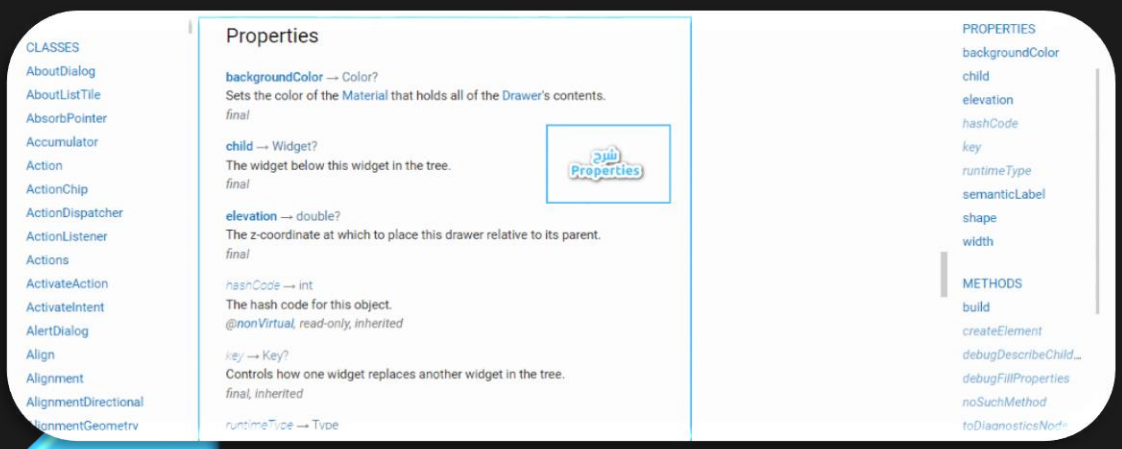
تظهر لنا هذه النافذة تحوي على الكثير من المعلومات عن Drawer (فيديو توضيحي – شرح مفصل عن Widget – كود شرح عن Drawer – جميع Properties التي تحويها بشرح مفصل وما القيم التي تقبلها في داخلها)

القائمة اليسار : قائمة جميع الـ Widget.

القائمة اليمين : قائمة تحوي جميع Properties الخاصة بلـ Widget نفسها.

كما هو واضح لدينا في الصورة في الأسفل :

وعند النزول للـأسفل الصفحة قليلا تظهر لنا Properties التي تحويها Widget مع شرح مفصل عنها بلأضافة إلى القيم التي تقبلها كما هو واضح لدينا في الصورة في الأسفل :



أكثر 10 أوامر أستخداماً في Flutter Terminal

تعرف عليها من خلال هذا الجدول :

flutter pub get	Get packages in a Flutter project
flutter create	Create a new Flutter project.
flutter analyze	Analyze the project's dart code.
flutter clean	Delete the build/ and .dart tool/ directories.
flutter devices	List all connected devices.
flutter test	Run Flutter unit tests for the current project.
flutter run	Run your Flutter app on an attached device.
flutter build apk	Build an Android APK file from your app.
flutter upgrade	Upgrade your copy of Flutter.
flutter doctor	Show information about the installed tooling.

Top 35 Flutter Interview Questions

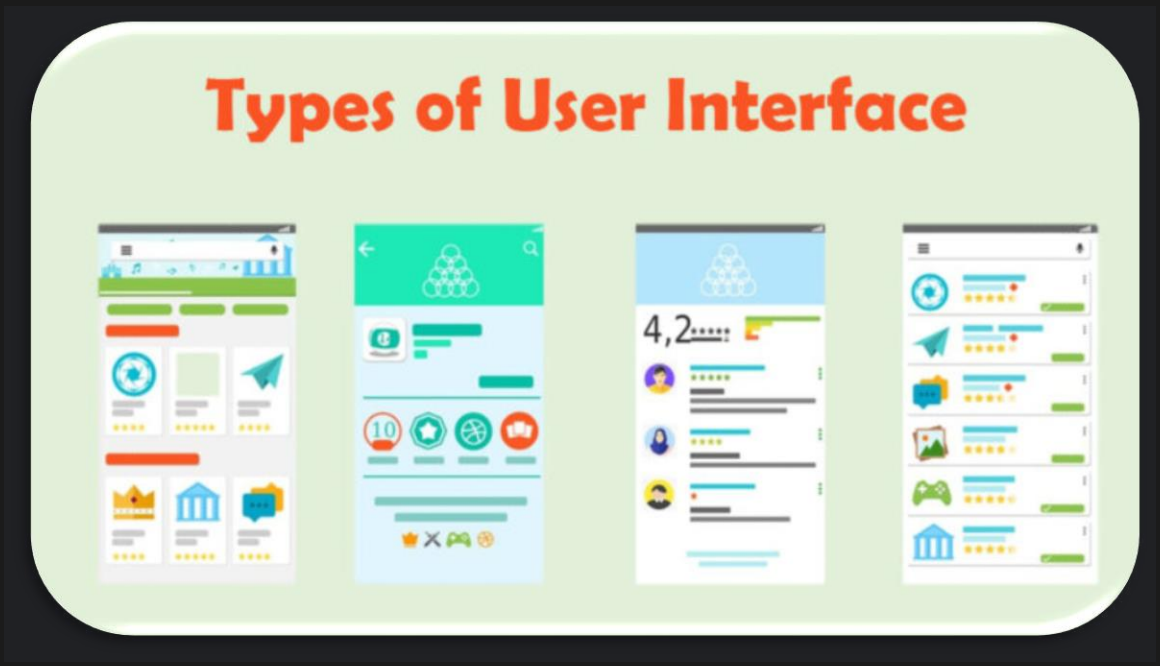
1. What is flutter ?
2. What are the advantages of using Flutter?
3. What are the limitations of Flutter ?
4. Who developed the flutter framework ?
5. What are the resources to learn Flutter ?
6. What type of applications can you develop using Flutter?
7. Is Flutter Open source ?
8. What makes Flutter unique ?
9. Explain Flutter SDK .
10. What do you understand from hot reload and hot restart?
11. How does Flutter run the code on Android ?
12. How does Flutter run the code on ios ?
13. In What technology is Flutter built ?

14. What is the use of the pubspec.yaml file ?
15. Explain stateful widgets and stateless widgets in flutter ?
16. What do you understand from 'State'? What is the use of the setState() method ?
17. Explain the lifecycle of a StatefulWidget?
18. What operating systems flutter support to build the apps ?
19. What is a Cookbook?
20. What is the container class in flutter ?
21. How will you make a HTTP request in flutter ?
22. Can a container have more than one child?
23. What is SafeArea in flutter ?
24. HOW JSON Serialization works in flutter ?
25. How to Parse JSON in flutter ?
26. What do you know about Dart ?
27. What is the use of this keyword while creating constructors in Dart?
28. What are the extension methods in Dart? Why to use it ?
29. In how many ways you can pass the parameters in Dart ?
30. Explain different null operators in Dart .
31. How to access property or method conditionally in Dart ?
32. Explain Spread operator.
33. What is Factory constructor in Dart?
34. How will you create a factory ?
35. How to check for types in Dart? Or What is sound typing in Dart?

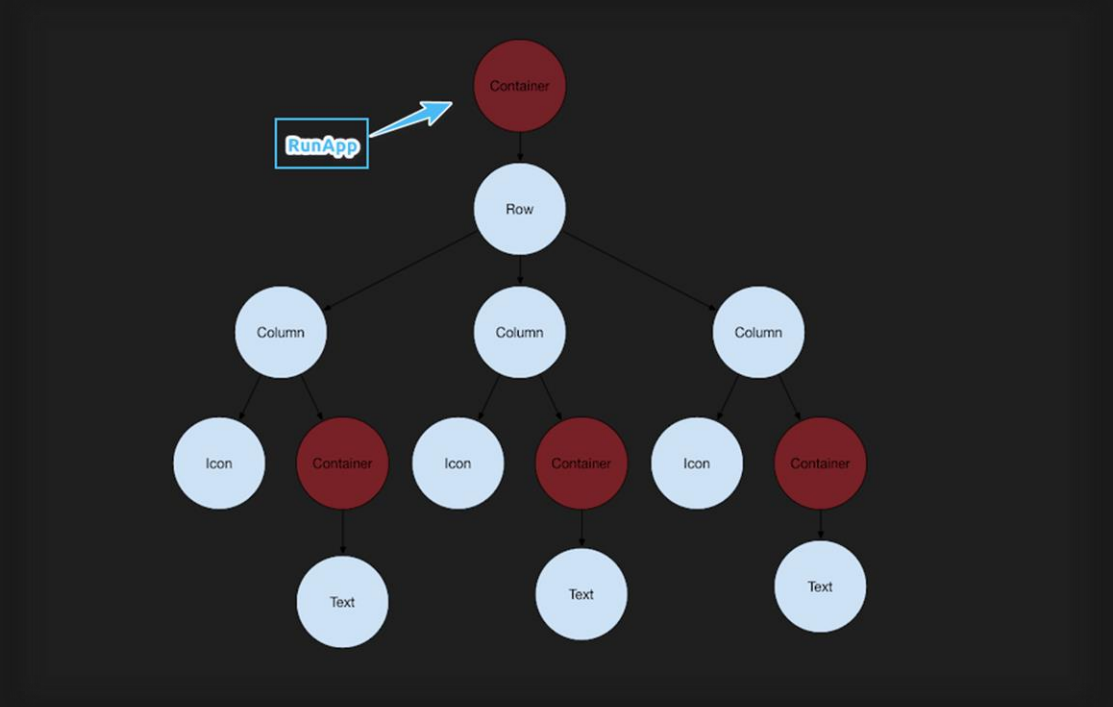
RunApp

Widget لنعتبرها عبارة عن قطعة و MatrialApp (هي مكتبة من شركة جوجل تساعدك برسم على شاشة بطريقة سهلة) ولنعتبرها أيضا هي مجموعة من القطع التي تحوي بداخلها على عدد كبير من Widget التي نستطيع استخدامها لبناء واجهة التطبيق UI.

(UI هي اختصار لكلمة user interface)



RunApp هي عبارة عن Widget مهمتها جعل الـ Widget التي بداخلها هي Root التي يبدأ منها التطبيق كما هو واضح لدينا في الصورة في الأسفل:

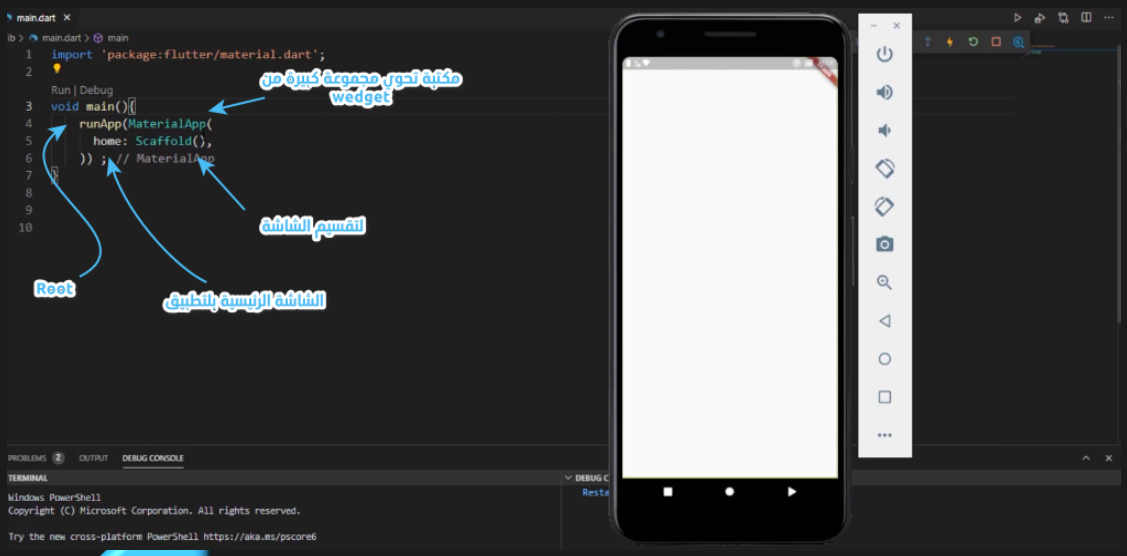


كل Widget هي عبارة عن class يحوي على مجموعة من Properties

Scaffold هي عبارة عن Widget أيضا تحوي بداخلها عدد كبير من Widget

مهمتها تقسيم شاشة التطبيق كما هو واضح لدينا في الأسفل :

Home من خلالها نستطيع اختبار الشاشة الرئيسية للـتطبيق.



Page Components

يتم أنشاء صفحات التطبيق ضمن Class لضمان كتابة الأكواد بشكل منظم أكثر ولسهول التعامل معها عند أجراء تعديل أو إضافة خاصية جديدة على التطبيق.

وتنقسم أنواع الصفحات لنوعين هما :

StatelessWedged صفحة لا يوجد فيها تفاعل.

StatefullWedged صفحة يوجد فيها تفاعل.

(سيتم شرح أنواع الصفحات بلتفصيل الممل في الدروس القادمة).

كما أنه يوجد أيضا نوعين من wedged :

Visible مرئية مثل text و AppBar وغيرها الكثير.

Invisible غير مرئية مثل Scaffold مهمتها تقسيم شاشة وغيرها الكثير.

كما نلاحظ قمنا بأنشاء class ليكون هو الـ class الرئيسي وقمنا بكتابة Widget home لتحديد الصفحة الرئيسية في التطبيق كما هو واضح لدينا في المثال في الأسفل :

```
Run | Debug
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Login(),
    ); // MaterialApp
  }
}
```

Root

صفحة لا يوجد فيها تفاعل

Class Homepage

المبنود التي يتم من خلالها بناء تصميم صفحة

الصفحة الأولى الرئيسية ضمن Class HomePage كما هو واضح لدينا في المثال في الأسفل :

```
class HomePage extends StatelessWidget {


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      drawer: Drawer(),
      body: Text("Homepage"),
    ); // Scaffold
  }
}
```

page one

الصفحة الثانية صفحة تسجيل الدخول ضمن Class Login كما هو واضح لدينا في المثال في الأسفل :

```
class Login extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      drawer: Drawer(),
      body: Text("Login"),
    ); // Scaffold
  }
}
```

page two

من الممكن أن البعض لم يفهم الكثير من هذا الدرس ولكن جميع الأمور التي ذكرت هنا هي أمور تنظيمية فقط (تنظيم الكود) وستتوضح هذه الأمور أكثر في الدروس القادمة.

text

تتمتع Widget text بلعديد من الخصائص التي تستطيع من خلالها أجراء جميع التغييرات التي تريدها على الخط بمشروعك وطبعا تقبل بداخلها String

ولاستعراض هذه الخصائص أما نقوم بلوقوف عند اسم class text كما هو واضح لدينا في المثال في الأسفل :

```
body: Text("How Are You"  ,    ),
(new) Text Text(String data, {Key key, TextStyle style, StrutStyle
strutStyle, TextAlign textAlign, TextDirection textDirection, Locale
locale, bool softWrap, TextOverflow overflow, double textScaleFactor,
int maxLines, String semanticsLabel, TextWidthBasis textWidthBasis,
TextHeightBehavior textHeightBehavior})
package:flutter/src/widgets/text.dart
Creates a text widget.
If the [style] argument is null, the text will use the style from
the closest enclosing [DefaultTextStyle].
textScaleFactor:
```

Properties

أو نقوم بوضع إشارة كومة (,) بعد النص وضغط على زر

Ctrl + Space فيقوم بعرض جميع الخصائص :

```
drawer: Drawer(),
body: Text("How Are You"  ,    ),
Key
}
key:
locale:
maxLines:
overflow:
semanticsLabel:
softWrap:
strutStyle:
style:
textAlign:
textDirection:
textHeightBehavior:
textScaleFactor:
```

Ctrl + Speas

Properties

أول خصائص text هي Style تقبل بداخلها Text Style وهي من Widget invisible غير مرئية تحتوي بداخلها على مجموعة مهمة جدا من Properties

ملاحظة : يجب وضع كومة (,) بين Properties

جدو لأهم TextStyle Properties :

Widget	عملها
FontSize	تقبل بداخلها قيمة من نوع double وهي مسؤولة عن التحكم بحجم الخط .
color	تغيير لون الخط ويتم ادخلها بطريقتين الأولى : كتابة اسم لون : Color : colors.red أو بكتابة Hex Code : Color : colors(0xff3498db)
fontweight	يتم من خلاله تحدد عرض الخط Fontweight : fontweight.bold
LetterSpacing	مسؤول عن تباعد بين الاحرف في الكلمة وتقبل قيمة من نوع double
WordSpacing	مسؤول عن تباعد بين الكلمات في وتقبل قيمة من نوع double
decoration	تقبل Textdecoration لها ثلاث حالات Linethrough خط وسط الكلمة. Overline خط فوق الكلمة. Underline خط تحت الكلمة. None لا يوجد خط ابدا

backgroundColor	تقبل color وخصائصها نفس خصائص color
shadwos	تقبل list من نوع shadwos تتمتع بثلاث خصائص رئيسية : Color تحديد لون shadwos blurRadius ظل الكلمة ويقبل قيمة من نوع double. Offset : لتحديد اتجاه shadwos وموضعه تقبل بداخلها كلاس من نوع offset وتأخد قيمتأن من نوع double : قيمة على محور العرضي x قيمة على محور الطولي y
textAlign	تستخدم لتحديد موضع النص تقبل بداخلها textAlign ولها عدد حالات : textAlign : textAlign.left يسار textAlign : textAlign.right يمين textAlign : textAlign.center منتصف textAlign : textAlign.start على حسب لغة التطبيق يكون الاتجاه textAlign : textAlign.end على حسب لغة التطبيق يكون الاتجاه

ملاحظة مهمة :

ما الفرق بين Start و left ..؟؟

تعتمد اتجاه Start اعتماد كلي على لغة التطبيق فأذا كأن التطبيق بلغة الأنكليزية فأن start هي يسار لأن اتجاه كتابة بلغة الأنكليزية من يسار إلى يمين.

أما في حال كأن لغة التطبيق عربية فأن start من يمين إلى يسار لأن اتجاه الكتابة بلغة العربية من اليمين إلى اليسار.

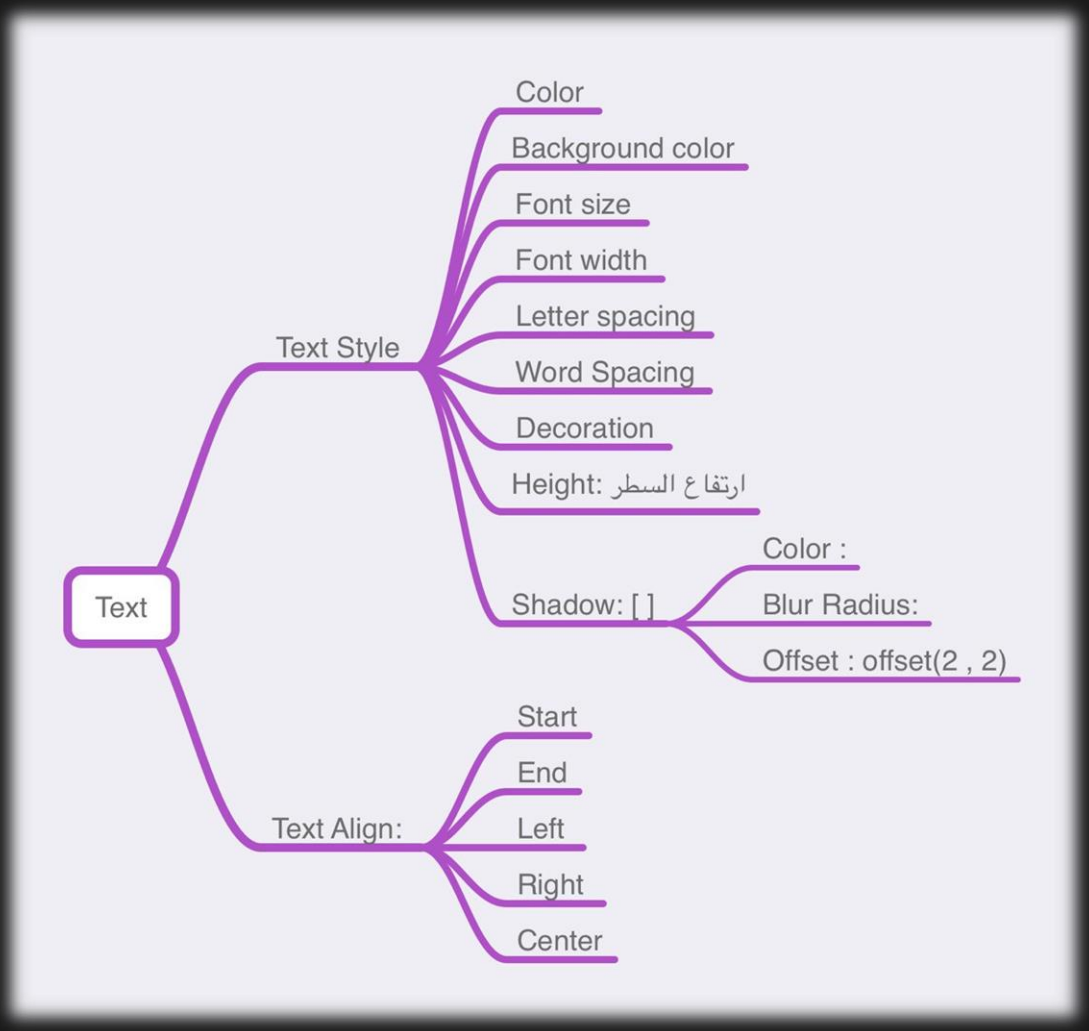
أما left فهي اتجاه محدد لا تعتمد على لغة الطبيق ابدا.

بلطبع يوجد الكثير الكثير من Properties الخاصة بل style text ولكن قمنا بذكر أهمها وأكثر استخدأما مثال على shadwos كما هو ظاهر لدينا في المثال في الأسفل :

```
36      child: Text("How Are You",
            style: TextStyle(
38              fontSize: 30,
39              color: Colors.black,
40              decoration: TextDecoration.none,
41              shadows: [
                    Shadow(color: Colors.red , blurRadius: 3.0 , offset: Offset(2.0 , 100.0) )
43              ])), // TextStyle // Text
44  ), // Container
45  ; // Scaffold
46
47
48
```

حجم الخط
لون الخط
خط على النص
محور x
محور y
shadwos

Widget text مخطط



Adel Abo baker

Container

هي واحدة من اهم Widget في flutter وتعني (وعاء أو حاوية أو صندوق) تستطيع وضع Widget أخرى داخلها وأضافة بعض الخصائص الإضافية على Widget التي تحويها سنتعرف على هذه الخصائص بلتفصيل.

داخل container اهم Properties فيها هي Child تقبل بداخلها أي Widget في flutter.

لنتعرف الأن على بعض خواص Container من خلال هذا الجدول :

Widget	عملها
color	تغير لون Container وخصائص color التي ذكرناها في السابق تنطبق على Container .
width	من خلالها تستطيع تغيير عرض Container وتقبل قيمة double يوجد أيضا خاصية : Width : double.infinity بهذه الحالة يقوم بأخذ عرض شاشة (بلكامل).
height	من خلالها تستطيع تغيير طول Container وتقبل قيمة double.

Margin

الهوامش الخارجية تتعامل مع pixels ولها عدة حالات سنتطرق لذكرها

<<<

Margin: EdgeInsets.all(10)

يقوم بأخذ هوامش خارجية للـ Container من جميع الاتجاهات.

Margin: EdgeInsets.only( )

وتقبل اربع اتجاهات

Top , right , left , bottom

تستطيع تحديد جهة واحد أو جهتان أو اربع جهات معا وإعطاء كل جهة قيمة مختلفة عن الأخرى.

Margin: EdgeInsets.symetric( )

تقبل بداخلها قيمتان أما :

Vertical عامودي

(من الأعلى و الأسفل).

أو Horizontal أفقي

(من اليمين و اليسار).

.FromLTRB( )

ويتم التعامل معها وكتابتها بترتيب من Left ثم Top ثم Right و bottom ولا نستطيع الاستغناء عن أي جهة من هذه الجهات.

padding	الهوامش الداخلية للـ Container وتنطبق خصائص Margin نفسها على padding.
Alignment	موضع العنصر داخل Container وتقبل بداخلها Alignment ولها عدة حالات ومواضع : Center منتصف Center right منتصف يمين Center left منتصف يسار Top right اعلى يمين Top left اعلى يسار Top center اعلى منتصف Bottom right أسفل يمين Bottom left أسفل يسار Bottom center أسفل منتصف

decoration	تقبل بداخلها Box decoration هي عبارة عن خاصية تقبل بداخلها عدة خصائص Properties. >> منها Color المسؤولة عن لون Container. ملاحظة هامة جدا : عند استخدام decoration داخل Container ممنوع استخدم Widget Color الا داخل Container حصرا . Border وهي خاصية تسطيع من خلالها إضافة اطار حول Container وتحديد لونه Color و سماكة width الاطار و تقبل بداخلها border ولها عدة حالات : Border.all(color , width) إضافة اطار من كل اتجاهات Container مع تحديد لون و السماكة الاطار.

: Decoration **(Border)**	.Border(all directions) Left , right , top , bottom وكل جهة تقبل بداخلها BorderSide و BorderSide تقبل بداخلها تحديد لون وسماكة كل جهة على حدى. ملاحظة مهمة جدا : (Borderالاطار) و (padding الهامش الداخلي) يتم اقتطاع مساحتهما من مساحة Container نفسو.
: Decoration **(border Radius)**	borderRadius لو اردنا عمل اطار Border ولكن بحواف دائرية الشكل كيف ذلك؟؟ عن طريق خاصية borderRadius تقبل بداخلها BorderRadius ولها أكثر من حالة : BorderRadius.circular( ) عمل أطار بحواف دائرية من جميع الاتجاهات.

: Decoration
(border Radius)

BorderRadius.only( )

تقبل بداخلها كل زوأية Container

Bottom Left

Bottom Right

Top Left

Top Right

وكل زأوية تقبل بداخلها

Radius.circular(Double)

BorderRadius.all( )

تقبل بداخلها Radius.circular عمل

اطار بحواف دائرية من جميع

الاتجاهات.

BorderRadius.Horizontal( )

تقبل بداخلها جهتان

Left , Right

وكل جهة تقبل

Radius.circular( )

BorderRadius.vertical( )

تقبل بداخلها جهتان

Top , Bottom

وكل جهة تقبل

Radius.circular( )

Decoration Image
(background)

وضع صورة لخلفية Container تقبل بداخلها Decoration Image وداخل Decoration Image تقبل Image.

<<<

وداخل Image تقبل نوعين :

NetWork Image
تقبل بداخلها URL يتم تحميل الصورة من شبكة الأنترنت بشكل مباشر.

Asset Image
سيتم شرحها بدرس مفصل

Fit
لو فرضنا أن أبعاد الصورة تختلف اختلاف كلي عن أبعاد Container ولا سيما أننا لا نستطيع وضع أبعاد ثابتة Container و صورة نفسها قد يكون لدينا شاشة هاتف بأبعاد مختلفة تمام مثلا أن يكون هاتف بشاشة اصغر من شاشة التي اعتمدنا عليها أبعاد Container فأن ذلك سيسبب مشاكل في أبعاد الصورة. لهذا فأن خاصية Fit تعالج هذه المشكلة بشكل كامل تقبل بداخلها BoxFit ولها عدة حالات :

Decoration Image (Fit) <<<	Fit : BoxFit.fill تقوم بتمديد الصورة على كامل مساحة Container بغض النظر عن تلف دقة الصورة بعد التمديد. Fit : BoxFit.contain هي الوضع الافتراضي فقط يقوم بوضع الصورة داخل Container (أي أنه لا يقوم بأي عملية معالجة لأبعاد الصورة بلنسبة لأبعاد Container. Fit : BoxFit.cover يقوم "بتكبير" الصورة داخل Container ولكن مع اقتطاع أجزاء من الصورة للـحفاظ على دقتها من التلف. (أي أنه على عكس fill لا يقوم بعملية تمديد للـصورة أنما تكبير واقتطاع الأجزاء زائدة).
Decoration Image (Repeat) <<<	Repeat تكرار الصورة داخل Container تقبل بداخلها imageRepeat ولها حالتين : Repeat : imageRepeat.repeatX تكرار الصورة على محور أفقي. Repeat : imageRepeat.repeatY تكرار الصورة على محور عامودي.

Decoration
(Box shadow)

نستطيع من خلاله عمل ظل للـ Container من خلال Decoration يقبل داخله list من نوع Box shadow ويقبل عدة خواص :

Color تحديد لون الظل.

blurRadius ظل Container ويقبل قيمة من نوع double.

Offset : لتحديد اتجاه shadwos وموضعه تقبل بداخلها كلاس من نوع offset وتأخد قيمتأن من نوع double :

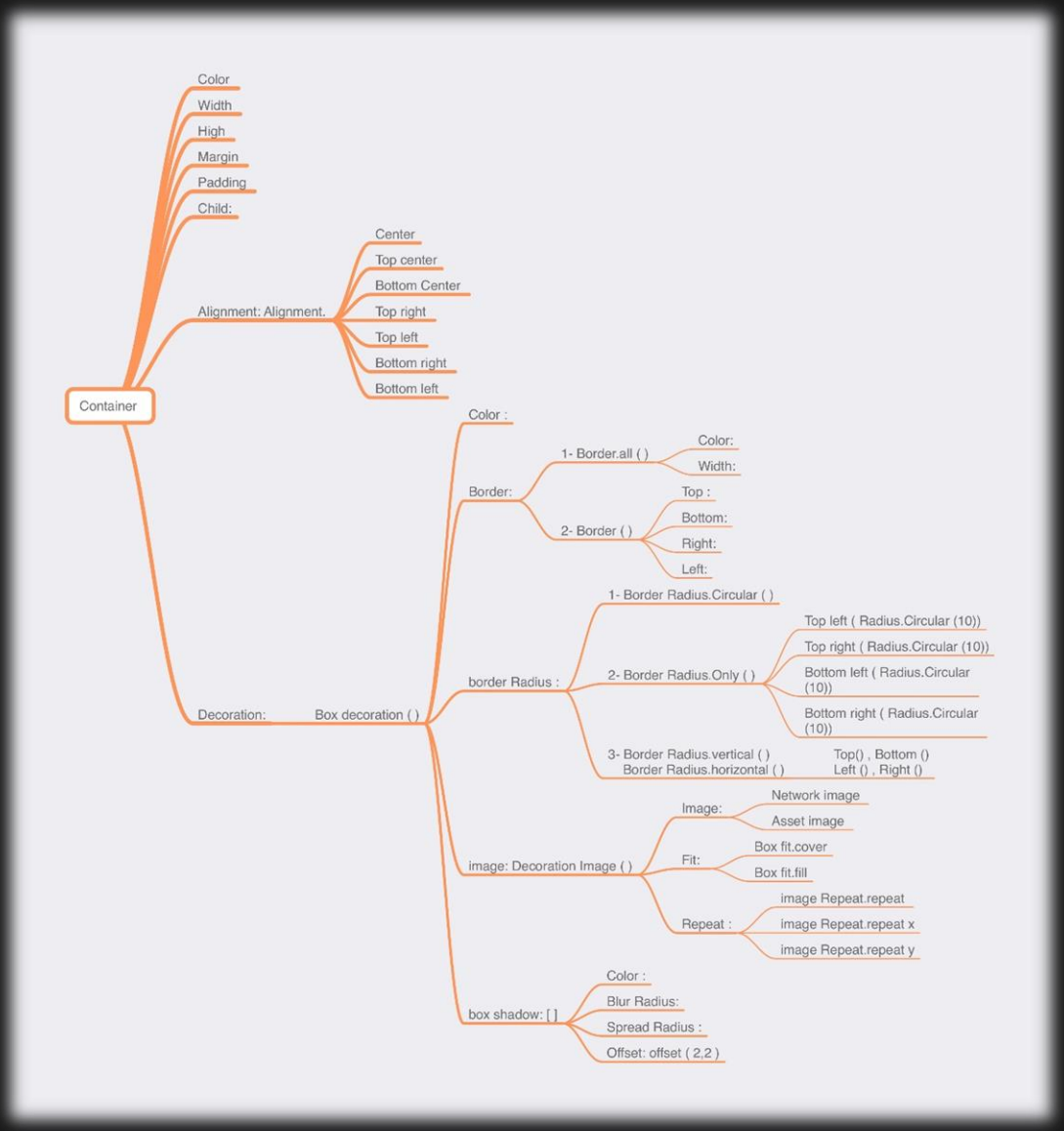
قيمة على محور العرضي x

قيمة على محور الطولي y

Spread Radius

المقصود فيها تمدد ظل Container وتقبل قيمة عشرية.

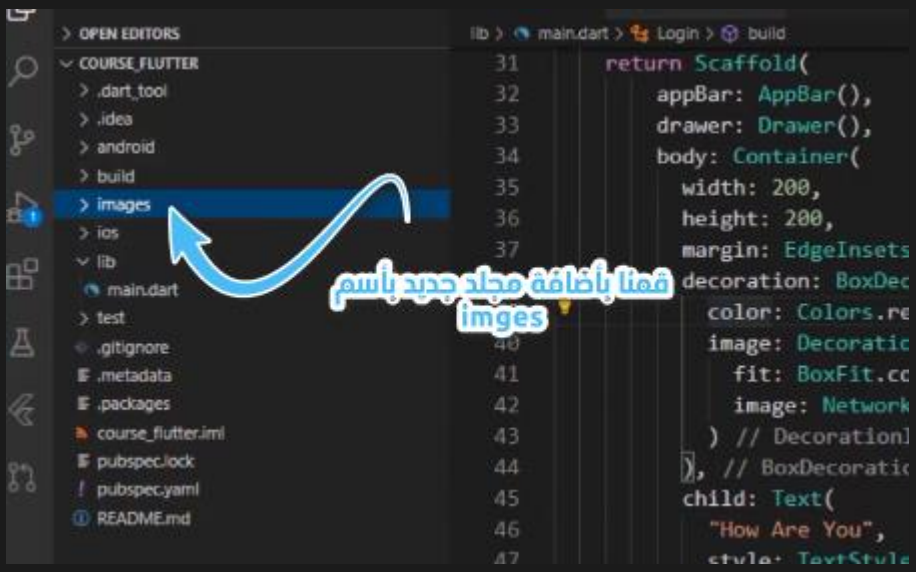
Widget container مخطط



Asset image

نستطيع من خلالها استخدام صورة من تطبيق نفسو ويكون أدائها بسرعة التحميل أفضل بلأضافة إلى أنه ثابتة مثال على ذلك (شعار التطبيق أو الأيقونة) على عكس NetWork التي يتم جلبها من الأنترنت وقد تستغرق وقت للـتحميل على حسب سرعة الأنترنت.

لأضافة صورة محلية تحتاج إلى عدة خطوات :



ونضع الصورة المراد تحميلها بلتطبيق داخل هذا المجلد الذي قمنا بأنشائه

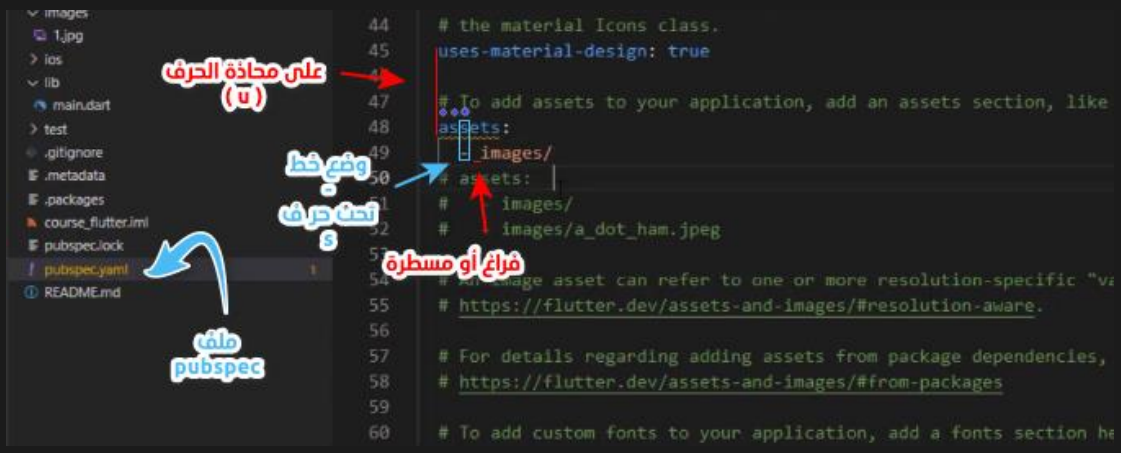
ثم نتوجه إلى ملف موجود في ملف project اسمه pubspec.yami وهو ملف حساس جدا .

(أي خطأ صغير فيه كفيل بأن يحدث خطأ ومشاكل كثير في project)

له عدة وظائف : مثل إضافة package إضافة New Folder

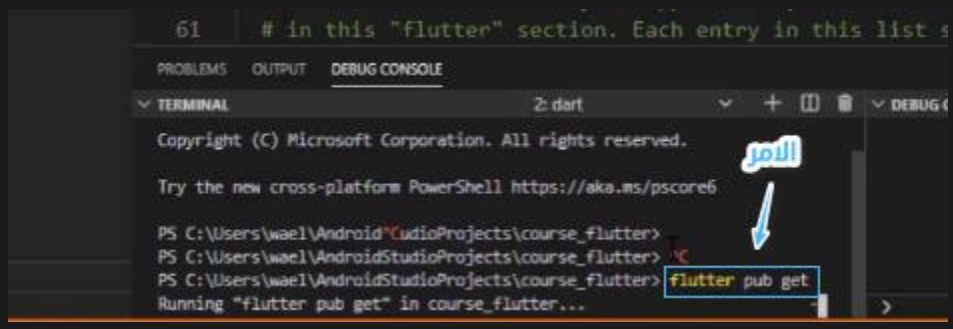
إضافة خطوط Fonts.

كما هو واضح لدينا في الصورة في الأسفل خطوات إضافة مجلد جديد ضمن ملفات project :

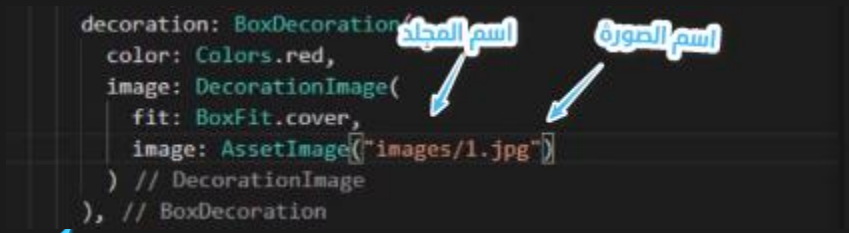


بعد الأنتهاء من ملف pubspec نتوجه إلى Debug Console ونقوم بتنفيذ الأمر التالي Flutter pub get

كما هو واضح لدينا في الصورة في الأسفل :



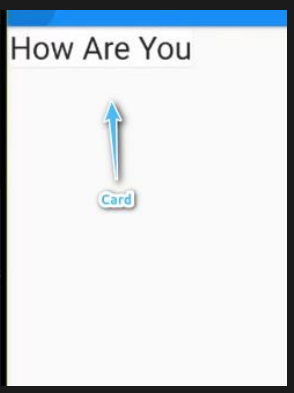
بعد الأنتهاء نقوم بكتابة مسار الصورة داخل Asset image كما هو واضح في المثال في الأسفل :



Card

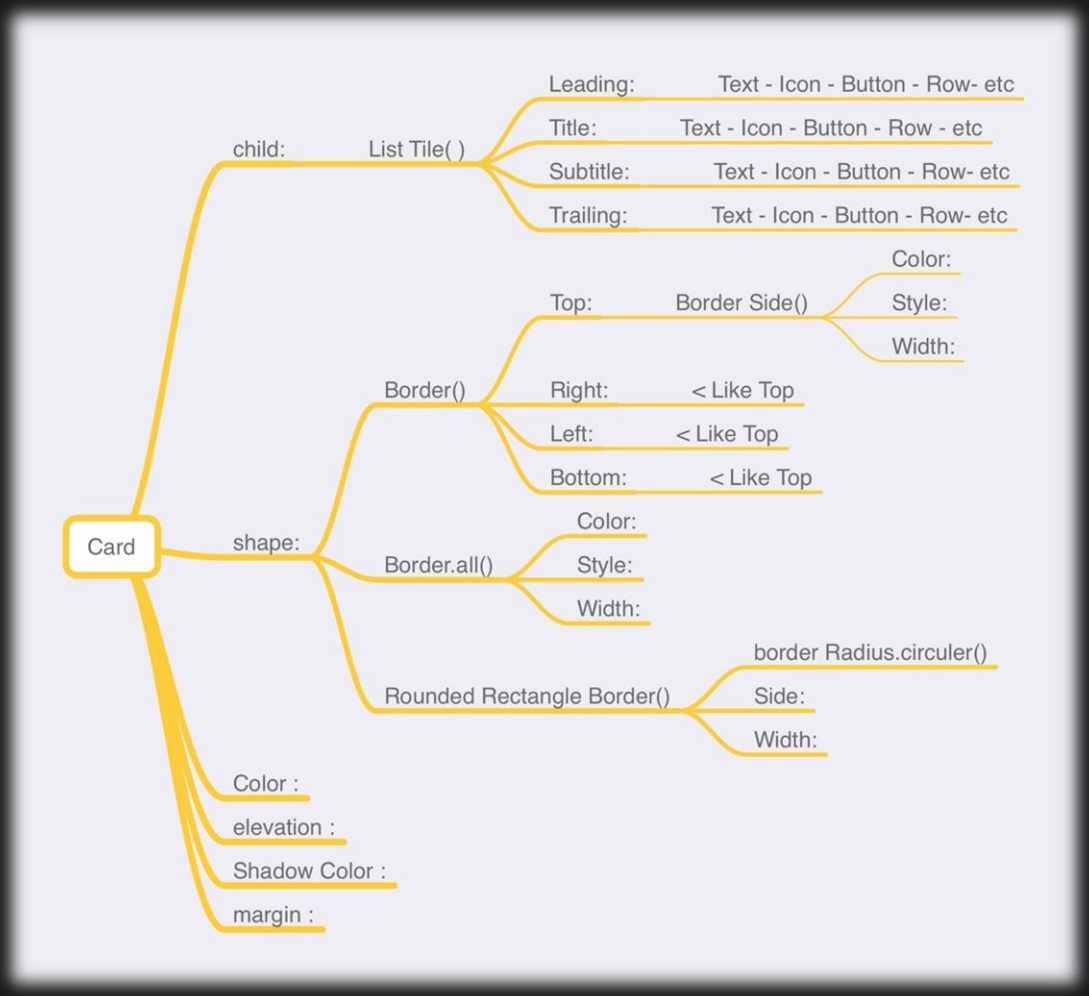
هي من Widget الحاوية تقبل بداخلها child مثلها مثل Container ولكن مع بعض الخصائص الإضافية والجاهزة مثل الظل الرمادي الخفيف الذي يحيط بل card من كل جوانبه كما هو واضح في المثال في الأسفل :

```
      body: Text("Homepage"),
    ); // Scaffold
  }
}

class Login extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(),
        drawer: Drawer(),
        body: Card(child:  Text("How Are You" , style: TextStyle(fontSize: 40),),
        ); // Scaffold
  }
}

//
```



Widget	عملها
Card ملاحظة : من الممكن عمل جميع هذه الخصائص على Container <<<	Color إعطاء لون للـــ Card.
	Elevation تقبل بداخلها قيمة عشرية مهمتها إعطاء ظل للـــ Card.
	Shdow color إعطاء لون للـظل.
	Child تقبل بداخلها أي Widget.

Widget Card مخطط



Widget master

لنبسط بعض الأمور التي تخص Widget ولنقوم بتقسيمها لأربع أنواع لتسهيل دراستها على المبتدئين في هذا الجدول :

نوع Widget	شرح عملها
Widget specified المخصصة أو المحددة.	نضع بداخلها شيء محدد. مثل : text نضع بداخله String أو image نضع بداخلها صورة أي أنك.
Widget parent الحاوية	تقبل بداخلها child (أبن) أي أنها تحوي Widget واحدة. مثل : Card أو container أو SizdBox.
Widget Master (أب لأكثر من ولد)	تقبل بداخلها children أي أنها تقبل بداخلها أكثر من Widget. (عدد لا نهائي من Widget) مثل : (Column أو Row أو stack أو listview أو Gridview)
Widget invisible الغير مرئية	إعطاء الخصائص الأساسية للـ project مثل : Scaffold , MatrialApp

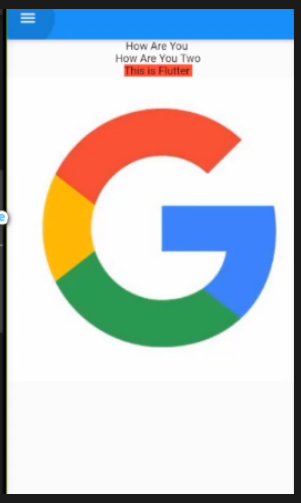
Column

هي من Widget master تقبل بداخلها children و children تقبل list من نوع Widget تستطيع وضع عدد لا نهائي من Widget المخصصة أو الحاوية.

وتمتاز أنها تقوم بترتيب Widget التي تحويها بشكل عامودي كما هو واضح لدينا في المثال في الأسفل :

```
}

class Login extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(),
        drawer: Drawer(),
        body:Column(children: [
          Text("How Are You ") ,
          Text("How Are You Two") ,
            Container(color: Colors.red,child: Text("This is Flutter "),) ,
            Image.asset("images/1.jpg")  ,

        ],) // Column
        ); // Scaffold
  }
}

//
// 1 - NetworkImage  load image from  Network internt
// 2 - AssetImage  load from my app
```

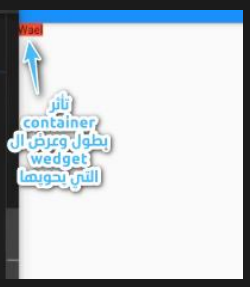
wedget one
wedget two
wedget three
wedget master
wedget four



ملاحظة : أي Widget حاوية تتأثر بطول وعرض Widget التي تحويها (ما لم يتم تحديد طول أو عرض) للـ Widget نفسها كما هو واضح لدينا في المثال في الأسفل :

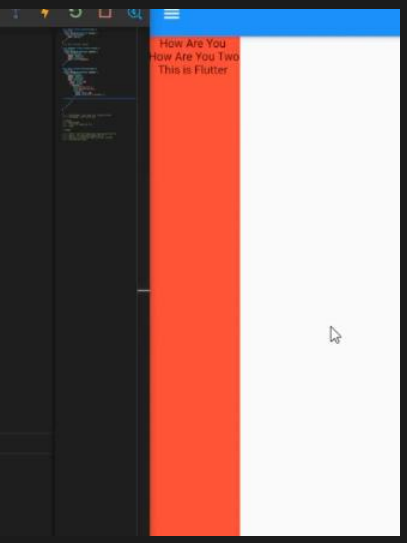
```
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      drawer: Drawer(),
      body: Container(

        color: Colors.red,
        child:Text("Wael "), // Container
    ); // Scaffold
  }
}
```

wedget حاوية
container color
text



ملاحظة مهمة : Column تقوم بأخذ طول الـ page بشكل كامل والعرض على حسب عرض Widget التي تحويها (ما لم يتم تحديد العرض) كما هو واضح لدينا في المثال في الأسفل :

```
}

class Login extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      drawer: Drawer(),
      body: Container(
        color: Colors.red,
        child: Column(
          children: [
            Text("How Are You "),   wedget one
            Text("How Are You Two"),   wedget two
            Container(   wedget three
              color: Colors.red,
              child: Text("This is Flutter "),
            ), // Container

          ],
        ), // Column
      ), // Container
    ); // Scaffold
```



Widget	عملها
Main Axiss Size <<<	تقبل بداخلها MainAxissSize تقبل قيمتان : MainAxissSize.Min يقوم Column بأخذ اقل طول ممكن (طول الذي يكفي لأحتواء Widget الذي يحويها). MainAxissSize.Max وهي الحالة الافتراضية يقوم بأخذ طول page كاملة.
	تقبل بداخلها MainAxissAlignment وتقبل عدة قيم :

Main Axiss alignment محاذاة محور الحقيقي بلنسبة للـ **Column** <<<	MainAxissAlignment.center محاذاة Widget التي بداخل Column إلى منتصفه.
	MainAxissAlignment.start محاذاة Widget التي بداخل Column إلى الأعلى.
	MainAxissAlignment.end محاذاة Widget التي بداخل Column إلى الأسفل.
	MainAxissAlignment.Around تقسيم مسافة (حسب طول) بين الـ Widget بشكل متساوي تماماً. ولكن تفصل مسافة قبل Widget الأولى و ومسافة بعد Widget الأخيرة مسافة يسن رأس الصفحة و Widget هي نص المسافة بين Widget و Widget.

Main Axiss alignment	MainAxissAlignment.Between تقسيم مسافة (حسب طول) بين ال Widget بشكل متساوي تماماً من دون ترك مسافة من الأعلى و الأسفل.
	MainAxissAlignment.Evenly تقسيم مسافة (حسب طول) بين ال Widget بشكل متساوي تماماً
cross Axiss alignment محور وهمي بلنسبة للـ **Column** **<<<**	تقبل بداخلها CrossAxissAlignment ولها ثلاث حالات :
	CrossAxissAlignment.start محاذاة محور الأفقي إلى بداية.
	CrossAxissAlignment.center محاذاة محور الأفقي إلى منتصف.
	CrossAxissAlignment.end محاذاة محور الأفقي إلى نهاية.

Widget Column مخطط

Row

هي من Widget master تقبل بداخلها children و children تقبل list من نوع Widget تستطيع وضع عدد لا نهائي من Widget المخصصة أو الحاوية.

وتمتاز أنها تقوم بترتيب Widget التي تحويها بشكل.

ملاحظة مهمة :

1- Row تقوم بأخذ عرض الـ page بشكل كامل.

2- تتشابه خواص Row مع خواص Column ولكن مع اختلاف بسيط

حيث أن المحور الأساسي هنا **Main Axiss alignment** وتأخذ ثلاث قيم

(start , center , end).

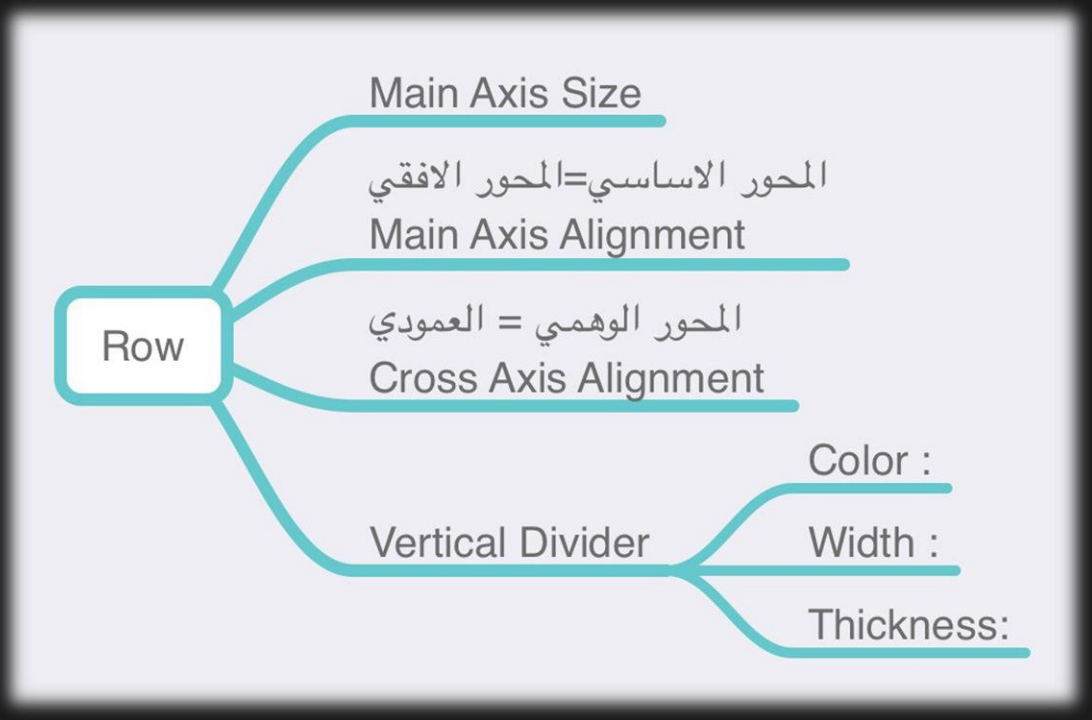
هو المحور الأفقي والمحور الوهمي **cross Axiss alignment**

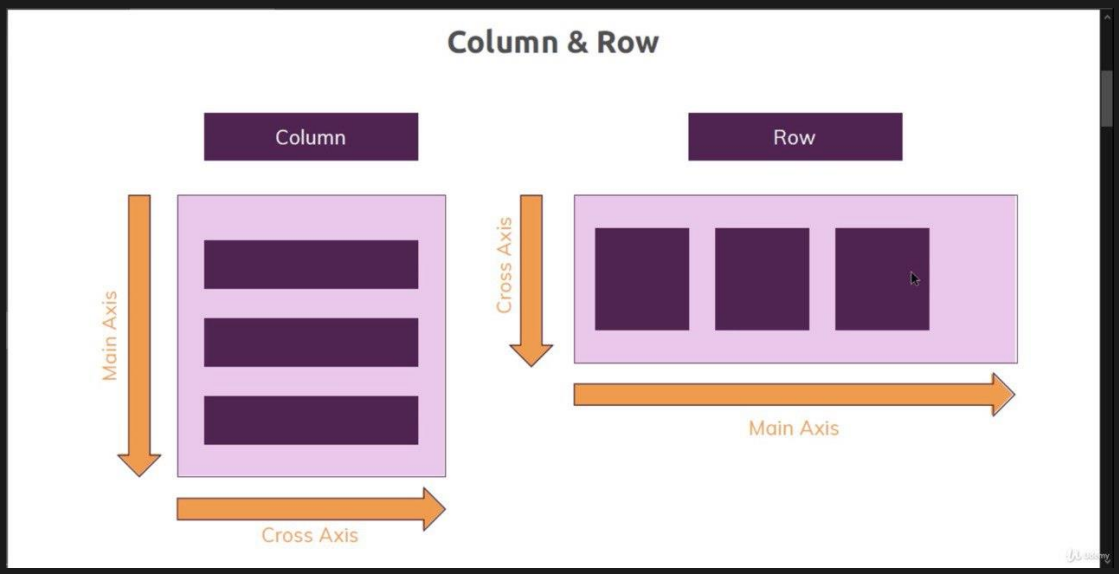
هو محور العامودي ويأخذ ثلاث قيم (start , center , end).

على عكس Column الذي يكون فيه المحور العامودي هو محور الأساسي

والمحور الأفقي هو المحور الوهمي.

Widget Row مخطط





Stack

هي من Widget master لأنها تقبل بداخلها children الموضوع مشابه للـ Column و Row من ناحية Properties.

ولكن تختلف بشكل احتواء Widget أذن أنها تقوم بترتيبها على شكل طبقات فوق بعضها البعض.

كما هو واضح لدينا في الأمثلة في الأسفل :

```
25    }
26  }
27
28  class Login extends StatelessWidget {
29    @override
30    Widget build(BuildContext context) {
31      return Scaffold(appBar: AppBar(), drawer: Drawer(),
32
33        body:Stack(children: [
34
35          Container(color:Colors.red , width: 400,height: 400,) ,
36          Container(color:Colors.green , width: 300,height: 300,) ,
37          Container(color:Colors.blue , width: 200,height: 200,) ,
38
39        ],) // Stack
40
41        ); // Scaffold
42    }
```

wedget one
wedget two
wedget three

تحكم بمحاذاة stack ضمن container :

```
return Scaffold(
    appBar: AppBar(),
    drawer: Drawer(),
    body: Container(
      width: 400,
      height: 400,
      color: Colors.black,
      child: Stack(
        alignment: Alignment.center,
        children: [
          Container(
            color: Colors.red,
            width: 200,
            height: 200,
          ), // Container
          Container(
            color: Colors.green,
            width: 100,
            height: 100,
            child: Text("Widget TWo"),
          ), // Container
```

محاذاة stack ضمن container
wedget one
wedget two

Widget TWo
wedget two
wedget one
container

(Postioned - OverFlow)

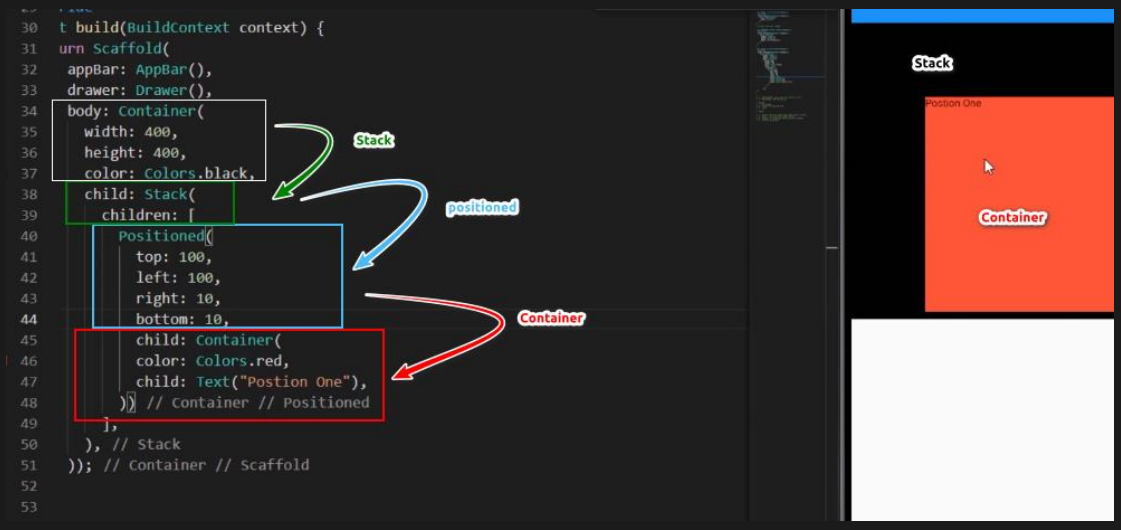
postioned يتم استخدامها داخل stack فقط ويتم أستخدامها لتحديد موضع Widget داخل stack تقبل بداخلها child و child كما نعمل أنه يقبل أي Widget وتقبل عدة اتجاهات :

.(Left , right , top , bottom)

بلأضافة لتحديد الطول وعرض Widget ضمنها :

.(height , width)

على سبيل المثال في حال تم أستخدام left , right (الاتجاهان متعاكسان) في نفس الوقت في postioed فأن الـ Widget سيتمدد على حسب القيمة التي نسندها للـ left أو right ونفس الأمر بلنسبة للـ top , bottom كما هو واضح لدينا في المثال في الأسفل :



ملاحظة : نستطيع عمل أكثر من postioned داخل Stack.

في حال قمنا بأستعمال height فلا نستطيع استعمال top , bottom في نفس الوقت من الممكن أن يحدث تمدد ويحدث خطأ.

السبب : لأننا قمنا بتحديد طول ولا يمكن الخروج عن الطول التي قمت بتحديده مع height.

كما هو واضح لدينا في المثال في الأسفل :

```
t build(BuildContext context) {
urn Scaffold(
 appBar: AppBar(),
 drawer: Drawer(),
 body: Container(
   width: 400,
   height: 400,
   color: Colors.black,
   child: Stack(
     children: [
       Positioned(
         width: 200,
         height: 200,
         top: 10,
         bottom: 10,
         child: Container(
           color: Colors.red,
           child: Text("Postion One"),
         )) // Container // Positioned
```

Error

```
'package:flutter/src/widg
basic.dart': Failed asser
line 3488 pos 15: 'top ==
|| bottom == null || heig
null': is not true.
See also:
https://flutter.dev/docs
/testing/errors
```

Error

نفس الأمر بلنسبة للـ width مع Left , right في نفس الوقت من الممكن أن يحدث تمدد ويحدث خطأ.

نفس السبب : لأننا قمنا بتحديد طول ولا يمكن الخروج عن الطول التي قمت بتحديده مع width.

كما هو واضح لدينا المثال خرج postioned عن قيود Stack

أذ قمنا بأعطاء Widget من الأعلى ارتفاع top : 300 ولأن طول و عرض = 400 Stack بناء على container مما أدى لخروج postioned عن قيود Stack مما أدى لأقتطاع جزء من container كما هو واضح لدينا في المثال في الأسفل :

```
30  t build(BuildContext context) {
31  urn Scaffold(
32   appBar: AppBar(),
33   drawer: Drawer(),
34   body: Container(
35     width: 400,
36     height: 400,
37     color: Colors.black,
38     child: Stack(
39       children: [
40         Positioned(
41           width: 200,
42           height: 200,
43           top: 300,
44
45           child: Container(
46             color: Colors.red,
47             child: Text("Postion One"),
48           )) // Container // Positioned
49       ],
50     ), // Stack
51   )); // Container // Scaffold
```

الطول والعرض

موضع Wedget داخل Stack

أختفاء جزء من wedget

ولحل في ذلك نستطيع أستخدام overflow وتقبل بداخلها overflow ولها حالتان :

Overflow : overflow.clip

سيتم الأقتطاع وهي الحالة الأفتراضية للـ stack.

Overflow : overflow. Visible

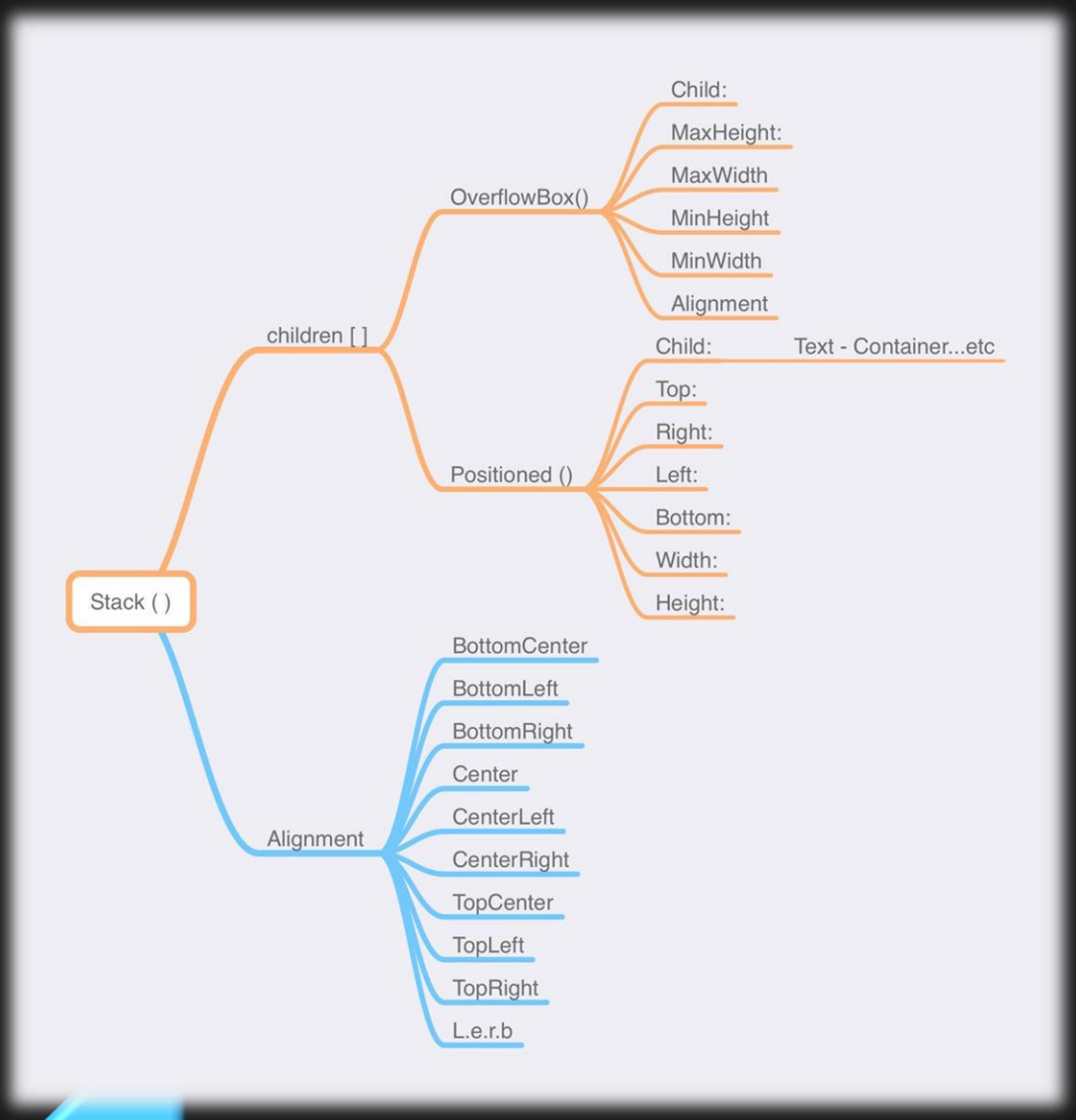
الحالة Visible وهي المرئية هي يقوم بأظهار Widget حتى ولو خرجت خارج قيود stack كما هو واضح لدينا في المثال في الأسفل :

```
drawer: Drawer(),
body: Container(
  width: 400,
  height: 400,
  color: Colors.black,
  child: Stack(
    overflow: Overflow.visible,
    children: [
      Positioned(
        width: 200,
        height: 200,
        top: 340,
        child: Container(
          color: Colors.red,
          child: Text("Postion One"),
        )) // Container // Positioned
    ],
  ), // Stack
)); // Container // Scaffold
```

Overflow

Visible

Widget Stack مخطط



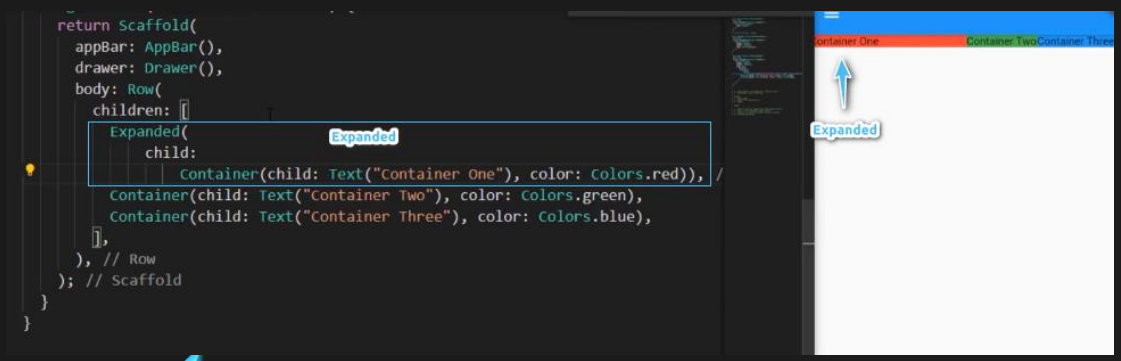
Expanded (Row)

من المعلوم أن أحجام شاشات الهواتف الذكية تختلف من جهاز لأخر ومن شركة لأخر

بفرض أن لدينا Row يحوي بداخله على ثلاثة container وقمنا بتحديد عرض width : 130 كما هو واضح لدينا في المثال في الأسفل :

لكي يغطي كل عرض الشاشة ولكن قد يكون لدينا هاتف بشاشة أكبر هنا سيصبح لدينا مساحة فارغة ونحن نريد التعامل مع هذه الحالة بشكل اتوماتيكي (نريد أن يكون العرض بناء على عرض الشاشة).

هنا تأتي مهمة Expanded حيث يقوم بتقسيم عرض الشاشة على حسب المساحة المتوفرة والفارغة من عرض الشاشة وتقبل بداخلها child كما هو واضح لدينا في المثال في الأسفل :



قام بتقسيم عرض الشاشة على ثلاثة container بتساوي بعد وضعها ضمن Expanded كما هو واضح لدينا في المثال في الأسفل :

```
31    return Scaffold(
32      appBar: AppBar(),
33      drawer: Drawer(),
34      body: Row(
35        children: [
36          Expanded(                    Expanded one
37            child:
38              Container(child: Text("One"), color: Colors.red)), // Expanded
39          Expanded(                    Expanded two
40            child:
41              Container(child: Text("Two"), color: Colors.green)), // Expand
42          Expanded(                    Expanded three
43            child: Container(
44              child: Text("Three"), color: Colors.blue)), // Container // Ex
45        ],
46      ), // Row
```

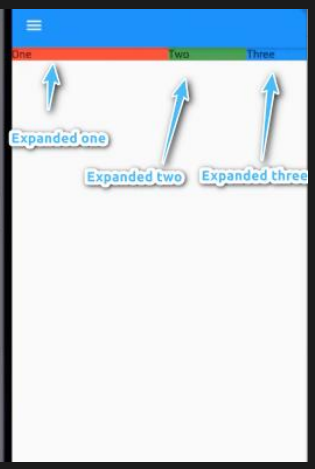
فلنتعرف الأن على خاصية flex وهي خاصية تمنحك تحكم كبير بتقسيم عرض الشاشة على هيئة أجزاء وقيمتها الأفتراضية هي 1.

كما هو واضح لدينا في المثال في الأسفل قمنا بأستخدام flex داخل Expanded وقمنا بتغير قيمة flex في Expanded one وأعطائه قيمة 2 بهذا سيصبح لدينا مجموع flex هو 4 أجزاء

(Flex one = 2 , flex two = 1 , flex three = 1)

سيتم إعطاء نص عرض الشاشة للـ Expanded one كما هو واضح لدينا في المثال في الأسفل :

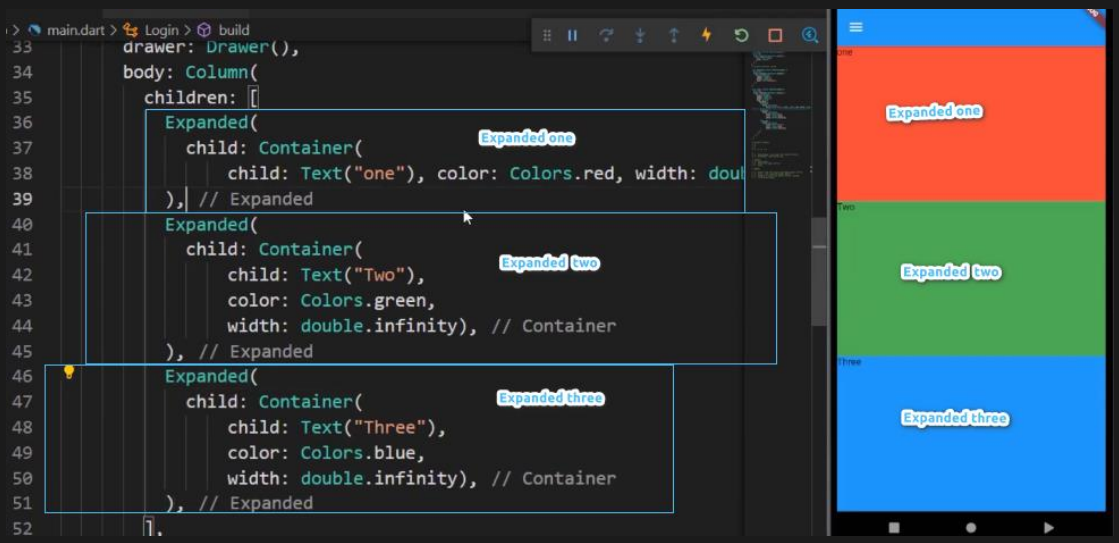
```
return Scaffold(
  appBar: AppBar(),
  drawer: Drawer(),
  body: Row(
    children: [
      Expanded(                    Expanded one
        flex: 2,
          child:
            Container(child: Text("One"), color: Colors.red)), // Expanded
      Expanded(
        flex: 1,                   Expanded two
          child:
            Container(child: Text("Two"), color: Colors.green)), // Expand
      Expanded(                    Expanded three
        flex: 1,
          child: Container(
            child: Text("Three"), color: Colors.blue)), // Container // Ex
    ],
  ), // Row
); // Scaffold
```



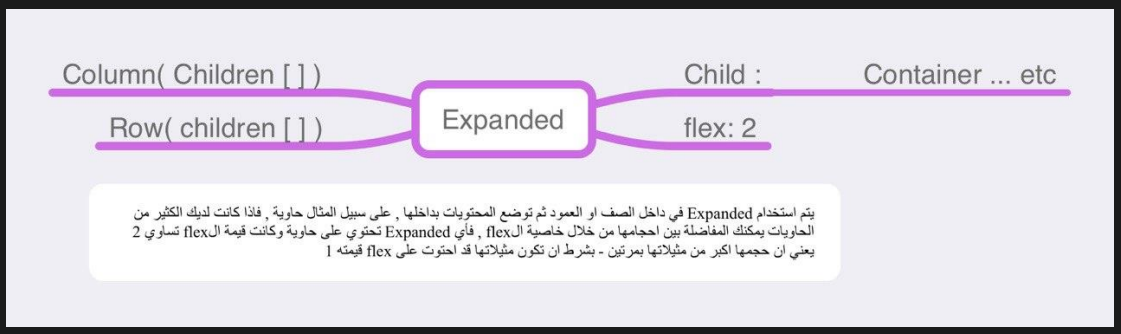
Expanded (Column)

نفس خواص Expanded التي تنطبق على Row تنطبق على Column ولكن بدال عرض الشاشة هنا نتحدث على طول الشاشة وسنوضح ذلك مع الأمثلة.

قمنا بعمل ثلاثة Expanded داخل Column وقام بتقسيم الشاشة إلى ثلاثة أجزاء متساوية وتغطية جميع مساحات الفارغة بشكل طولي بلنسبة للشاشة حيث أن قيمة flex = 1 كما هو واضح لدينا المثال في الأسفل :



مخطط Widget Expanded

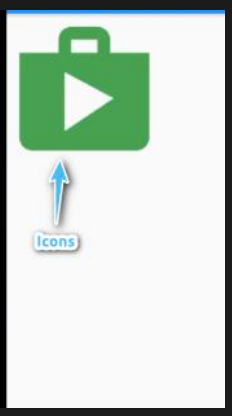


Icons

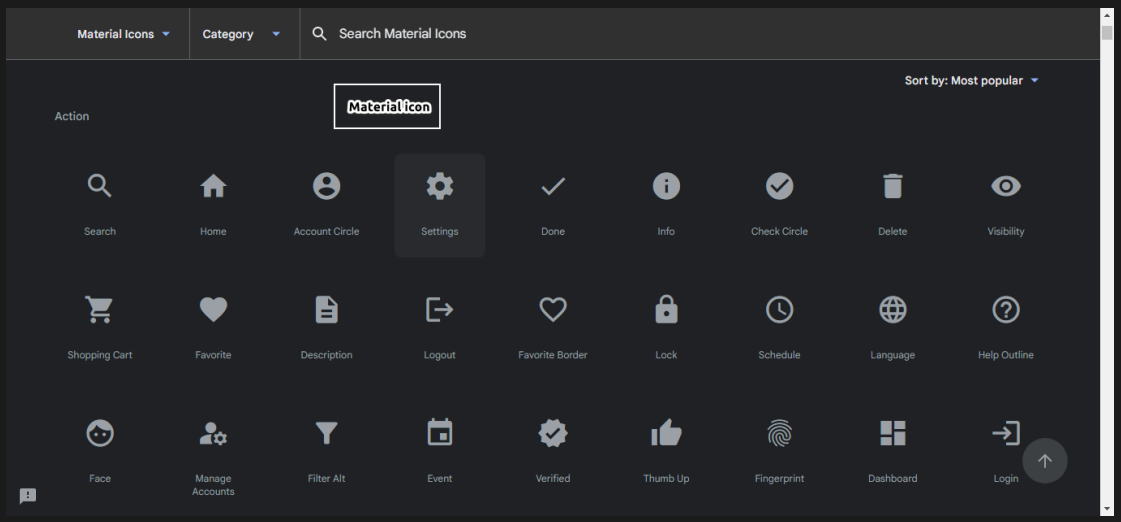
يوجد عدد كبير من icons ضمن مكتبة material design يمكن الاستفادة منها تقبل بداخلها عدة Properties مثل color و size كما هو واضح لدينا في المثال في الأسفل :

```
class Login extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      drawer: Drawer(),
      body:Icon(Icons.shop , size: 200,color: Colors.green,),
    ); // Scaffol
  }
}
```

size color
name icon
icon color



لسهولة البحث عن icons المطلوبة نستطيع من خلال مكتبة Material Icons فقط نقوم بجلب اسم Icons



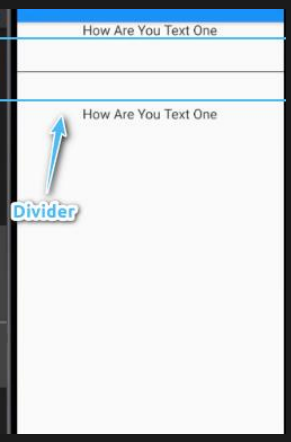
Divider

هو عبارة عن خط يقبل عدة Properties مهمته فصل Widget عن بعضها البعض بشكل أفقي متساوي يقوم بأخذ عرض الشاشة بشكل كامل ويتميز أنه يكون له بشكل تلقائي margin بسيط من الأعلى ومن الأسفل.

Widget	عملها
Divider أفقي	Color تغيير لون Divider.
	Height تغيير ارتفاع Divider.
	Thickness تغيير سماكة الخط وتقبل قيمة double.

مثال عن Divider كما هو واضح لدينا المثال في الأسفل :

```
27
28  class Login extends StatelessWidget {
29    @override
30    Widget build(BuildContext context) {
31      return Scaffold(
32        appBar: AppBar(),
33        drawer: Drawer(),
34        body:Container(
35          // width: 300,
36          // color: Colors.blue,
37          child: Column(children: [
38 wedget one   Text("How Are You Text One" , style: TextStyle(fontSize:
39 Divider      Divider(color: Colors.black,height: 100,thickness: 1.2,
40 wedget two   Text("How Are You Text One" , style: TextStyle(fontSize:
41          ],), // Column
42        ), // Container
```

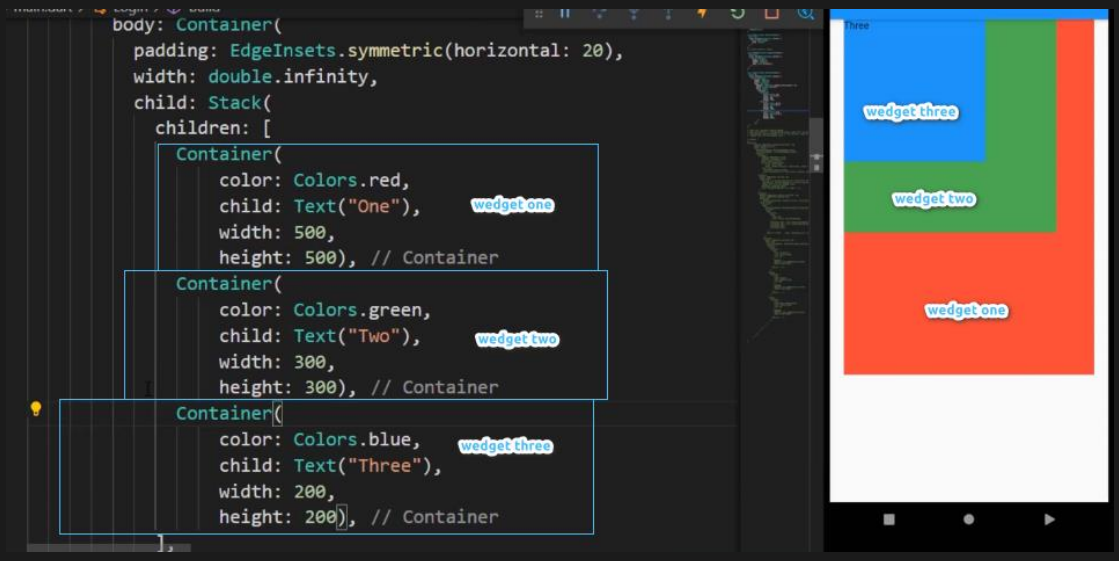


كما أيضا يوجد Divider بشكل عامودي ويأخذ Properties التالية :

Widget	عملها
Divider Vertical عامودي	Color تغيير لون Divider.
	width تغيير عرض Divider.
	Thickness تغيير سماكة الخط وتقبل قيمة double.

indexed Stack

كما تعلمنا من الدروس السابقة أن stack هي من Widget master تقبل بداخلها children وعرض Widget داخلها على هيئة طبقات فوق بعضها البعض فلطبقة السفلية هي Widget الأولى تليها الطبقة التي تكون فوقها تكون Widget ثانية وهكذا على حسب عدد Widget التي بداخلها.



indexed Stack لا تشكل فارق الكبير عن stack الأ في امر واحد فقط أذ تقوم بعرض Widget واحدة فقط وعلى حسب قيمة index نختار Widget المراد عرضها مثلا :

اذا كأنت index = 0 يقوم بعرض Widget الأولى.

أو اذا كأنت index = 1 يقوم بعرض Widget الثانية.

وهكذا على حسب عدد Widget التي تحويها indexed Stack.

سنوضح الأمر أكثر عن طريق هذا المثال الذي في الأسفل :

```
ain.dart > Login > build
body: Container(
  padding: EdgeInsets.symmetric(horizontal: 20),
  width: double.infinity,
  child: IndexedStack(
    index: 1,                    Item number in List
    children: [                                          List
      Container(
          color: Colors.red,
          child: Text("One"),                 index 0
          width: 500,
          height: 500), // Container
      Container(
          color: Colors.green,                index 1
          child: Text("Two"),
          width: 300,
          height: 300), // Container
      Container(
          color: Colors.blue,
          child: Text("Three"),               index 2
          width: 200,
          height: 200), // Container
```

Wrap

هي من Widget master تقبل بداخلها Children ولكن ما هو الفرق بينها وبين Widget master أخرى سنتعرف على ذلك بعد قليل..!!

على سبيل المثال لو قمنا بعمل Row ووضعنا داخله عدة container كما هو واضح لدينا في المثال في الأسفل :

```
> main.dart > Login > build
43            height: 100,
44          ), // Container
45          Container(
46            margin: EdgeInsets.all(10),
47            color: Colors.red,
48            width: 90,
49            height: 100,
50          ), // Container
51          Container(
52            margin: EdgeInsets.all(10),
53            color: Colors.red,
54            width: 90,
55            height: 100,
56          ) ,  // Container
57           Container(
58                margin: EdgeInsets.all(10),
59                color: Colors.red,
60                width: 90,
61                height: 100,
```

عند وضع أكثر من ثلاثة container نلاحظ ظهور خطأ في البرنامج بسبب عدم توفر مساحة الكافة من container الرابع من عرض الشاشة ومن المعروف أن Row يجبر Widget داخله بترتيب بشكل أفقي.

و لحل هذه المشكلة هنا يأتي دور Wrap طالما أن مساحة غير كافية سيقوم بعمل إزاحة Widget إلى الأسفل كما هو واضح لدينا في المثال في الأسفل :

```
main.dart > Login > build
drawer: Drawer(),
body: Wrap(
  children: [
    Container(
      margin: EdgeInsets.all(10),
      color: Colors.red,
      width: 90,
      height: 100,
    ), // Container
    Container(
      margin: EdgeInsets.all(10),
      color: Colors.red,
      width: 90,
      height: 100
```

wedget master

حل مشكلة بعد استخدام Wrap

نفس الأمر بلنسبة column كما هو واضح لدينا في المثال في الأسفل :

```
height: 100,
) , // Container
  Container(
margin: EdgeInsets.all(10),
color: Colors.red,
width: 90,
height: 100,
) , // Container
  Container(
margin: EdgeInsets.all(10),
color: Colors.red,
width: 90,
height: 100,
) , // Container
  Container(
margin: EdgeInsets.all(10),
color: Colors.red,
width: 90,
height: 100,
) , // Container
```

ظهور خطأ

Wrap تقبل بداخلها direction وظيفتها تغيير الأتجاه direction يقبل حالتين

Horizontal وهو الحالة الأفتراضية الأفقي والذي هو Row

Vertical العامودي الذي هو Column.

كما هو واضح لدينا في المثال في الأسفل :

```dart
class Login extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      drawer: Drawer(),
      body: Wrap(
        direction: Axis.vertical ,
        children: [
          Container(
            margin: EdgeInsets.all(10),
            color: Colors.red,
            width: 90,
            height: 100,
          ), // Container
          Container(
            margin: EdgeInsets.all(10),
            color: Colors.red,
            width: 90,
```


Sized Box

هي من Widget الحاوية تقبل بداخلها child لا يوجد فرق كبيرة بينها وبين container الا أنها لا تقبل color بداخلها وأستخداماتها قليل جدا وعدد قليل من مبرمجين يلجؤون لهذه Widget.

يمكن استعمالها على سبيل المثال لأنشاء فراغ بين اثنان Widget كما هو واضح لدينا في المثال في الأسفل :

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(),
      drawer: Drawer(),
      body: Column(
        children: [
          Text("wael"),
          SizedBox(height: 300),
          Text("Mohammad")
        ],
      )); // Column // Scaffold
}
```


أو تستخدم لتحجيم Widget داخلها أي أنه SizedBox يأخذ حجم Widget التي يحويها بداخله.

Single Child Scroll View

تقبل يداخلها child من خلالها نستطيع تحريك محتويات ما داخل Row أو Column مع تحديد الاتجاه عن طريق خاصية Scroll direction.

كما هو واضح لدينا في المثال في الأسفل مع column :

```
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(),
      drawer: Drawer(),
      body: SingleChildScrollView(
        scrollDirection: Axis.vertical,
        child: Column(
          children: [
            Container(
              margin: EdgeInsets.all(10),
              color: Colors.blue,
              width: 90,
              height: 100,
            ), // Container
            Container(
              margin: EdgeInsets.all(10),
```

الاتجاه

تحرك للأعلى وللأسفل

Column

كما هو واضح لدينا في المثال في الأسفل مع Row :

```
class Login extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(),
        drawer: Drawer(),
        body: SingleChildScrollView(
          scrollDirection: Axis.horizontal,
          child: Row(
            children: [
              Container(
                margin: EdgeInsets.all(10),
                color: Colors.blue,
                width: 90,
                height: 100,
              ), // Container
              Container(
                margin: EdgeInsets.all(10),
                color: Colors.blue,
```

الاتجاه أفقي

تحريك لليمين و لليسار

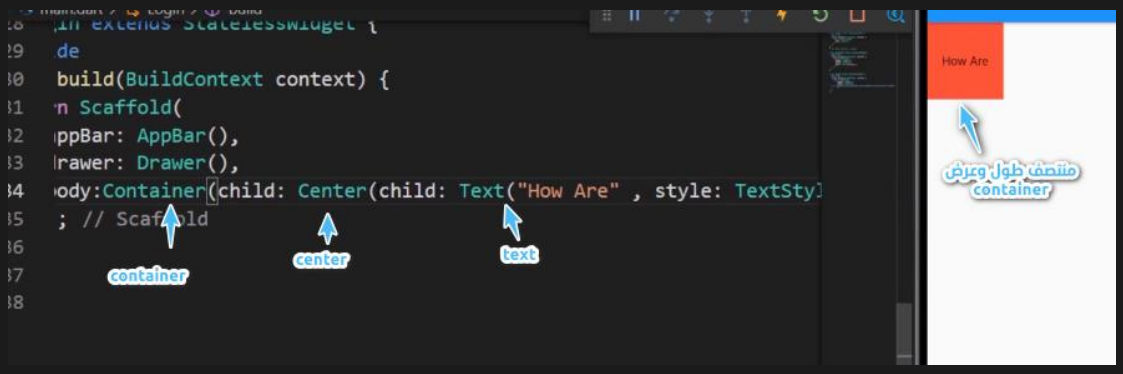
ملاحظة : تستطيع التحريك محتويات ما داخل Row أو Column فقط عند اللـمس فوقها وتحريك بحسب الاتجاه الموضوع.

Center

مهمته توسيط Widget ضمن شاشة الهاتف أو ضمن Widget الحاوية (container على سبيل المثال) لهذه Widget وتقبل بداخلها child.

كما هو واضح لدينا في المثال في الأسفل توسيط text في منتصف الشاشة :

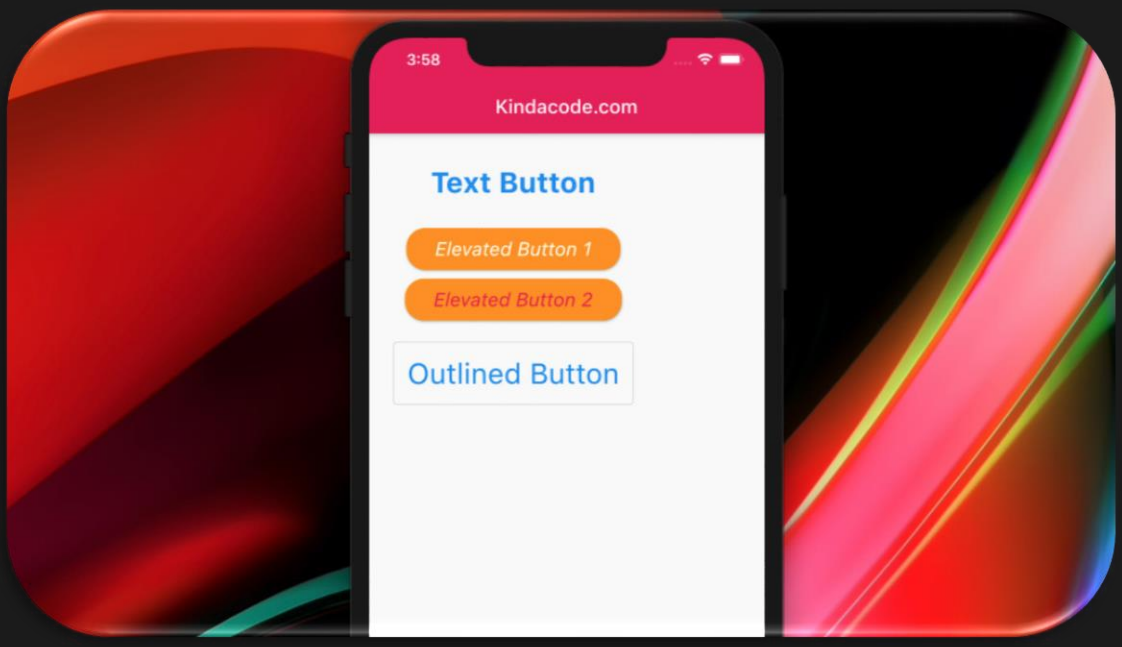
توسيط Widget ضمن Widget حاوية (container) كما هو واضح لدينا في المثال في الأسفل :



Button

Elevated Button

مهمته تنفيذ حدث أو امر معين عند الضغط عليها ويوجد أنواع كثيرة من button منها Elevated Button تقبل بداخلها عدد كبير جدا من Properties.



سوف نقوم بأستعراض الأهم منها في هذا الجدول الذي في الأسفل :

Widget	عملها
Elevated Button	Child تقبل بداخلها أي Widget وهي required.
	On pressed مهمتها تنفيذ امر معين عند ضغط على زر بلغة Dart تقبل بداخلها method أو function.
	Color لتغيير لون الزر.

Elevated Button	Text color لتغيير لون النص داخل Button.
	On long pressed تنفيذ امر معين method أو function ولكن بلضغط على button بشكل مطول.
	Splash color يتغير لون button أثناء الضغط عليه فقط.
	Disabled color يمكن تحكم بلون الزر الغير فعال عن طريق هذه Propertie.
	Elevation عمل ظل للـ button وتقبل قيمة من نوع double.
	Disabled text color يمكن تحكم بلون النص داخل الزر الغير فعال عن طريق هذه Propertie.
	Disable elevation يمكن تحكم بظل button الغير فعال عن طريق هذه Propertie.
	Style تقبل بداخلها style button يوجد لدينا style جاهز اسمه Elevated Button.stylefrom()

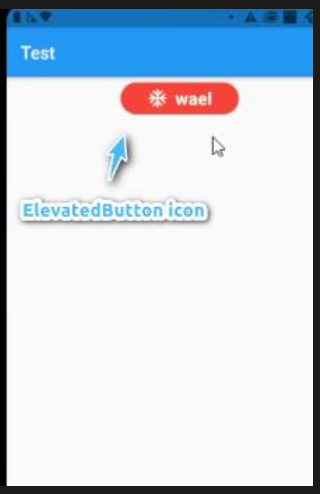
Elevated Button icon

Properties	Widget
Icon تقبل بداخلها icon ثم نختار icon الذي نريد وهي من Widget مطلوبة Elevated Button icon	**Elevated Button icon** تشبه ElevatedButton مع اختلاف بسيط في Properties <<<
Label تقبل بداخلها Widget من نوع text.	

مثال عن Elevated Button icon كما هو واضح لدينا في المثال في الأسفل :

```
children: [
  Center(
    child: ElevatedButton.icon(
      onPressed: () {
        print("Click");
      },
      icon: Icon(Icons.ac_unit),
      label: Text("wael"),
      style: ElevatedButton.styleFrom(
          primary: Colors.red,
          padding: EdgeInsets.symmetric(horizontal: 30),
          shape: RoundedRectangleBorder(borderRadius: BorderRadius
          textStyle: TextStyle(fontSize: 20 , fontWeight: FontWeig
      ), // ElevatedButton.icon
    ) // Center
  ],
)); // Column // Scaffold
```

Properties required in ElevatedButton icon



ملاحظة : لماذا يوجد عدد كبير من أنواع button مع أن اغلبها يشترك بنفس Properties الذي يحويها ؟

سبب توفر لك Flutter العديد من الخيارات التي تساعدك على بناء أفضل كود بأقل مجهود وأفضل أداء ممكن.

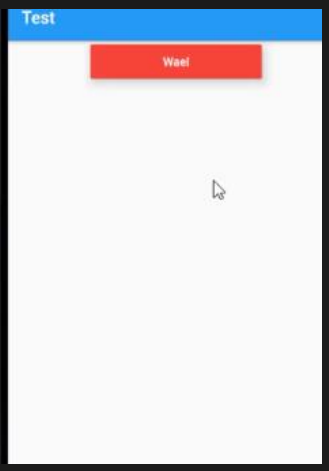
Material Button

هو يعد من أنواع button الشامل السبب أنه يحوي على اغلب Properties التي يتمتع بها باقي الأنواع كما هو واضح لدينا في المثال في الأسفل :

```
body: Column(
  children: [
    Center(
      child: MaterialButton(
        onPressed: () {},
        child: Text("Wael"),
        elevation: 10,
        splashColor: Colors.black,
        color: Colors.red,
        height: 40,
        minWidth: 200,
        textColor: Colors.white,
      ) // MaterialButton // Center
    ),
  ],
), // Column // Scaffold
```

Material Button

Properties Material Button



Text Button

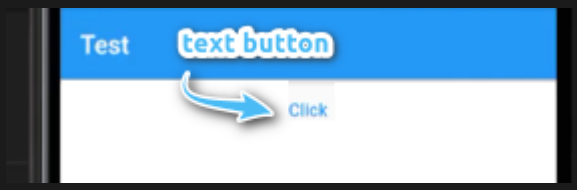
هو من ابسط أنوع button على الأطلاق أذ يمتاز بمجموعة من Properties البسيطة جدا كما هو واضح لدينا في المثال في الأسفل :

```
child: TextButton(
  onPressed: () {},
  child: Text("Click"),
  style: TextButton.styleFrom(primary: Colors.white , backgroundColor: Colors.red , ),
) // TextButton // Center
],
), // Column // Scaffold
```

text button

color text button

background color button



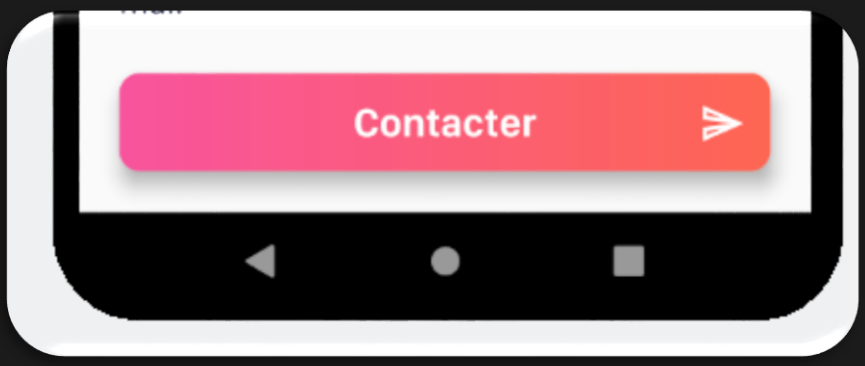
Icon Button

هو عبارة عن أيقونة مهمته تنفيذ حدث أو امر معين عند الضغط عليها ببساطة مثلها مثل أي نوع من أنواع button المختلفة كما هو واضح لدينا في المثال في الأسفل :

```
        child: IconButton(

          onPressed: (){
            print("ahmed") ;
          },
         icon: Icon(Icons.person),
        ), // IconButton
      ))); // SizedBox // Center // Scaffold
  }
}
```

InkWell Button

هو نوع من أنواع Button في flutter بعطيك أمكانية بناء button custom أي أنه مخصص تستطيع بنائه من الصفر من دون أي خواص جاهزة وعمل button الذي تريد ويقبل بداخله onTap بدلا من on pressed من ممكن إضافة صورة عادية بداخل inkwell button وعند الضغط عليه سيقوم بعمل مهمة معينة والكثير الكثير من الخصائص الرائعة التي يتمتع بها هذا نوع.



Gesture Detector

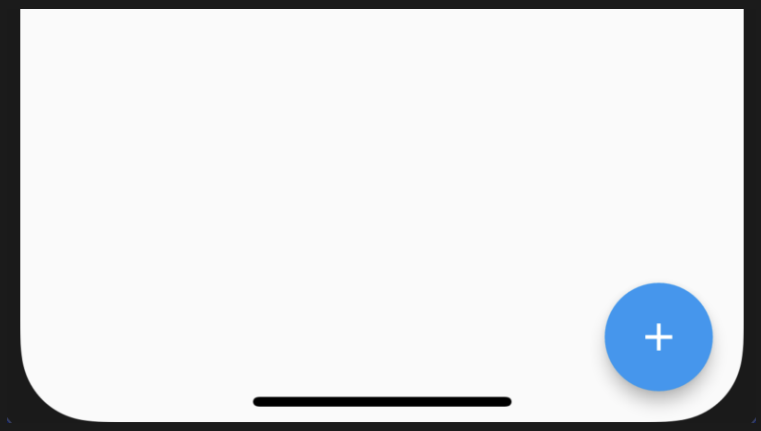
هو نوع ن أنواع button يشبه بشكل كبير inkwell button ولكن ما يميزه عنه أنه يحتوي على عدد كبير من function التي سوف نتعرف على أهمها من خلال هذا الجدول الذي في الأسفل :

function	Widget
onTapUp : (need parameters) { } عند رفع الضغط من على الزر سيقوم بتنفيذ function.	Gesture Detector <<<
onTapDown : (need parameters) عند الضغط على الزر سيتم تنفيذ function التي بداخله.	
onTapCancel : (Does not need parameters) { } تعمل عند الضغط على الزر ثم إزاحة المؤشر خارج الزر سيقوم بتنفيذ function التي بداخله.	
onLongPress : ( ) { } سيتم تنفيذ function عند الضغط المطول على الزر.	

onLongPressStart : (need parameters) { } عند الضغط المطول على الزر سيتم تنفيذ function التي بداخله.	**Gesture Detector <<<**
onLongPressEnd: (need parameters) { } عند رفع الضغط من على الزر سيقوم بتنفيذ function.	
onDoubleTap : (Does not need parameters) { } عند الضغط مرتين بشكل سريع على الزر سيتم تنفيذ function.	

Floating Action Button

هو عبارة عن button يتمتع بنفس خصائص button العادي ولكن ما يميزه أنه زر عائم في واجهة التطبيق يبقى ظاهر فوق كل مكونات الصفحة كما هو واضح لدينا في المثال في الأسفل :

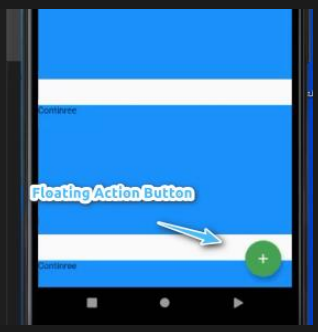


مثال عن Floating Action Button يحوي خصائص button التي تعلمناها سابقا كما هو واضح لدينا في المثال في الأسفل :

```
floatingActionButton: FloatingActionButton(
  backgroundColor: Colors.green,
  onPressed: (){

},
child: Icon(Icons.add),
), // FloatingActionButton
body: ListView(
  children: [
    Container(
      child: Text("Continree"),
      color: Colors.blue,
      height: 200,
```

لون الزر

يقبل child



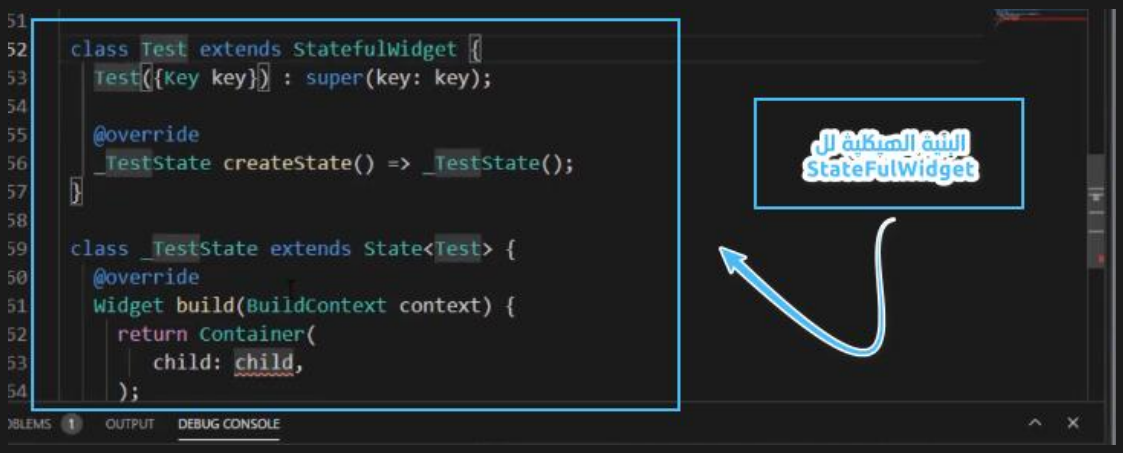
StateFulWidget

ما الفرق بين Statefulwidget و Statelesswidget :

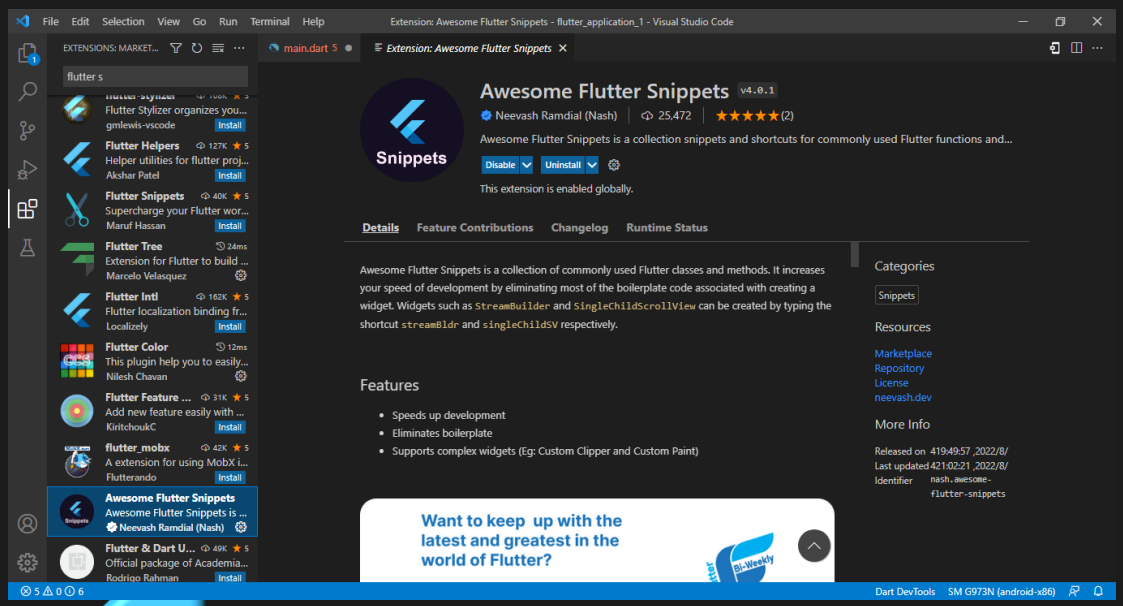
StateLessWidget عندما يكون لدينا تفاعل بلصفحة.

StateFulWidget عندما لا يكون لدينا تفاعل بلصفحة.

ولديها عدة خواص أهمها Set state سنتعرف لذلك من خلال الأمثلة

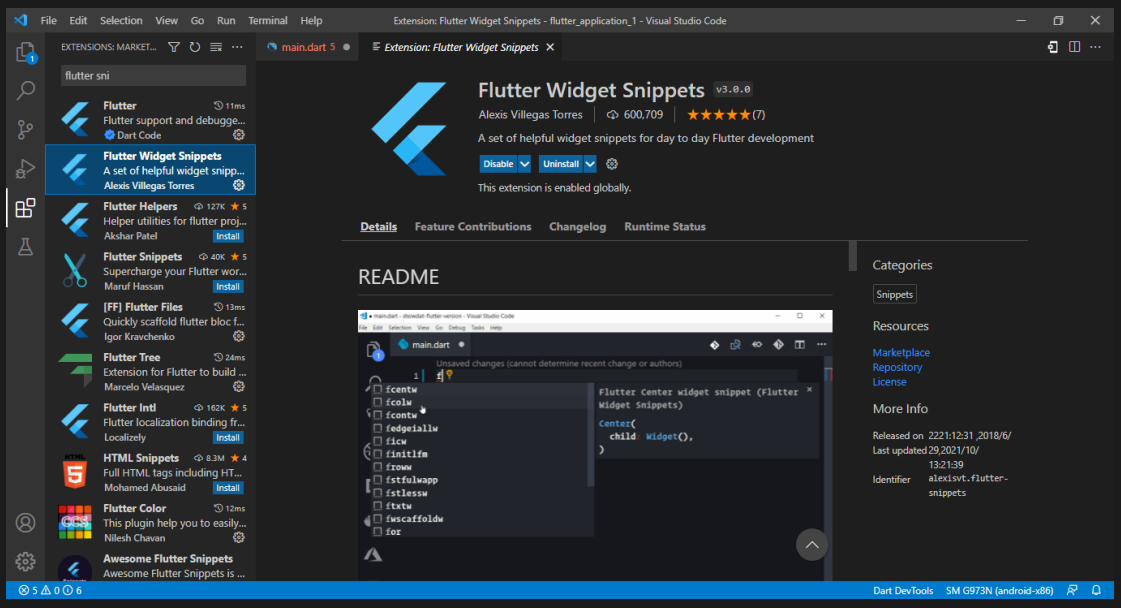


نستطيع كتابتها يدويا أو الاستعانة بأضافة تسمى Awesome Flutter Snippets كما هو واضح في الصورة في الأسفل :

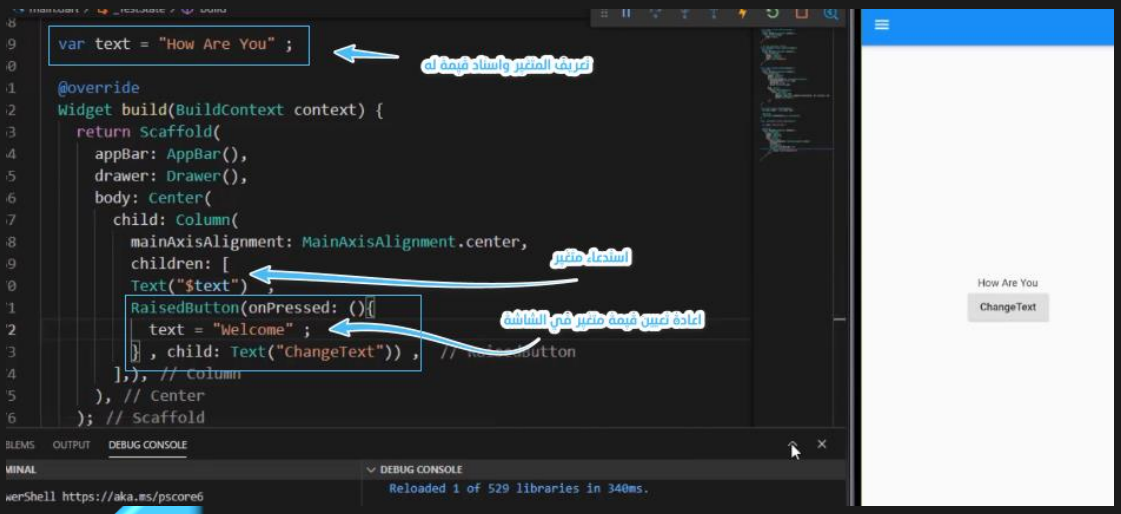


و إضافة أخرى تسمى Flutter Widget Snippets مهمة هذه الأضافات الاستكمال التلقائي مثلا عند كتابة StateFulWidget يقوم بأنشاء البنية البرمجية الأساسية للـ StateFulWidget

كما هو واضح لدينا في الصورة في الأسفل :



لنقوم بشرح هذا المثال البسيط عن StateFulWidget قمنا بأنشاء زر داخل الصفحة مهمته عند الضغط عليه يقوم بتغير جملة How are you في الشاشة إلى welcome كما هو واضح في المثال في الأسفل :



وعند الضغط على الزر فعلا يقوم بأعادة تعيين قيمة المتغير بلقيمة الجديدة التي قمنا بأسنادها له ولكن فقط Debug Console لم يقوم بتغييرها بشاشة كما هو واضح لدينا في المثال في الأسفل :

```
body: Center(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
    Text("$text")  ,
    RaisedButton(onPressed: (){
      text = "Welcome" ;
      print(text) ;
    } , child: Text("ChangeText")) ,  // RaisedButton
  ],), // Column
 ), // Center
); // Scaffold
}
```

خرج البرنامج

```
OUTPUT   DEBUG CONSOLE
DEBUG CONSOLE
I/flutter ( 4487): Welcome
```

Now Are You
ChangeText

( ) Set State حيث تقوم هذه method بعملية الهدم والبناء السريع للـ Widget وتستخدم فقط في StateFulWidget نقوم بتمرير قيمة إعادة التعيين متغير داخلها وعند إعادة تشغيل التطبيق والضغط على الزر change text نلاحظ أن قيمة تتغير في كل من شاشة و Debug Console كما هو واضح لدينا في المثال في الأسفل :

```
body: Center(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text("$text"),
      RaisedButton(
        onPressed: () {
          setState(() {
            text = "Welcome";
          });
          print(text);
        },
        child: Text("ChangeText")), // RaisedButton
    ],
```

Set State in StateFulWidget

تغير القيمة المسندة بعد الضغط

Welcome
ChangeText

عند الضغط

```
OUTPUT   DEBUG CONSOLE
DEBUG CONSOLE
Restarted application in 796ms.
I/flutter ( 4487): Welcome
```

خرج البرنامج في Debug Console

Drop Down Button

القائمة المنسدلة يوجد عدة أنواع للـقوائم المنسدلة سنتعرف الأن على DropDownButton لنتعرف أولا على اهم الـ Properties المطلوبة داخل هذه Widget من خلال هذا الجدول :

Properties	Widget
items تقبل بداخلها list من نوع Drop Down Menu Item من نوع string وتقبل هذا النوع حصرا . Drop Down Menu Item تقبل بداخلها two Properties وهما : Child تقبل بداخلها Widget وهي القيمة الظاهرة للـمستخدم. Value وهي القيمة الحقيقية التي تخزين بداخلها جميع عناصر items.	**Drop Down Button** **<<<**
On change تقبل بداخلها method وهذه ال method تقبل بداخلها parameter.	
OnTap عند الضغط على Drop Down Button يتم تنفيذ method التي بداخلها.	

شرح طريقة بناء Widget DropDownButton بلتفصيل من خلال عدة خطوات:

1- Hint الملاحظة التي تظهر على DropDownButton قبل الاختيار.

2- items تقبل بداخلها list من نوع Drop Down Menu Item من نوع string وتقبل هذا النوع حصرا.

3- تحويل من list من نوع string إلى list ترجع قيمة من نوع Drop Down Menu Item من نوع string.

4- Child تقبل بداخلها Widget وهي القيمة الظاهرة للـمستخدم.

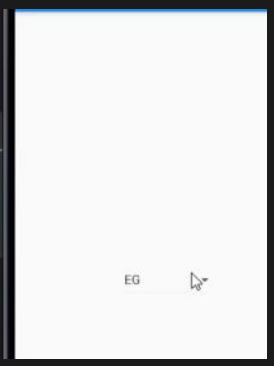
5- Value وهي القيمة الحقيقية التي تخزين بداخلها جميع عناصر items.

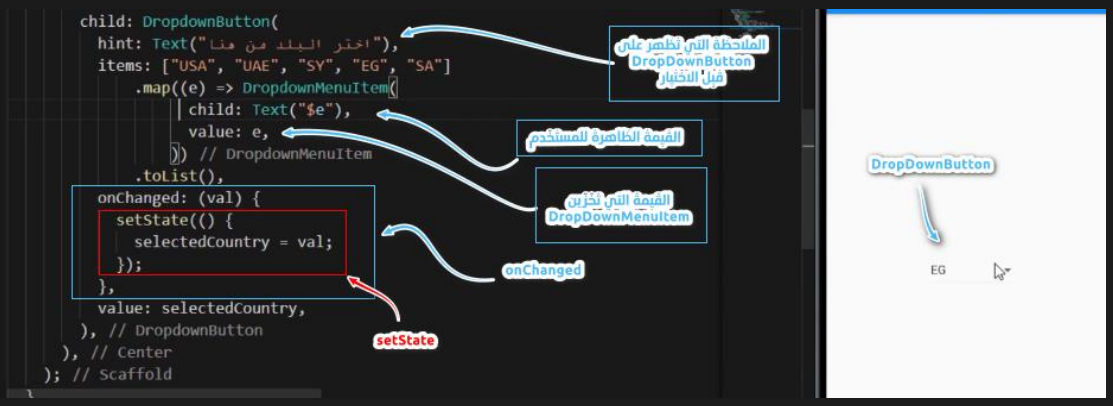
6- Map تقوم بأرجاع Iterable يجب تحويلها إلى list لأن items تحتاج لأن القيمة التي ترجع لها من نوع list.

7- تمرير القيمة المختارة داخل On change إلى المتغير select country داخل Set State.

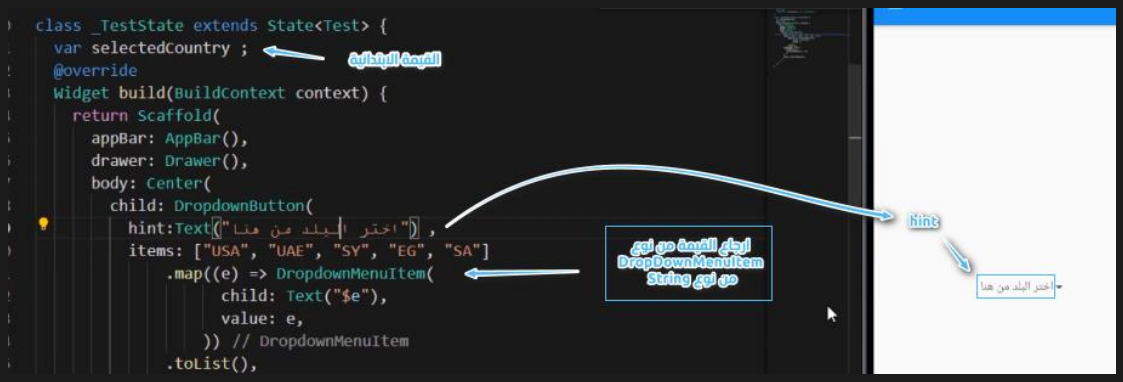
8- القيمة النهائية للـقيمة المختارة.

```
child: DropdownButton(
  hint: Text("اختر البلد من هنا"),  1
  items: ["USA", "UAE", "SY", "EG", "SA"]  2
      .map((e) => DropdownMenuItem(  3
          child: Text("$e"),  4
          value: e,  5
          )) // DropdownMenuItem
      .toList(),  6
  onChanged: (val) {
    setState(() {
      selectedCountry = val;  7
    });
  },
  value: selectedCountry,  8
), // DropdownButton
```

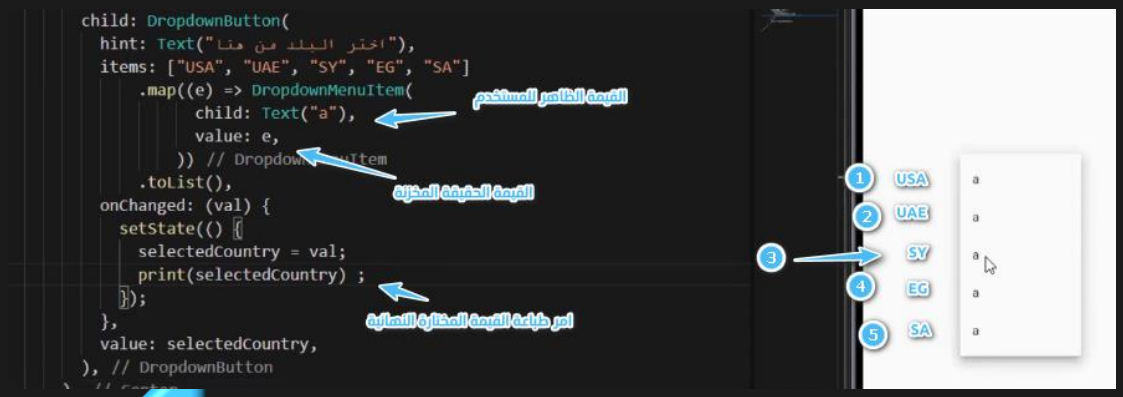




نستطيع التعرف على طريقة كتابة Properties وطريقة بناء Widget DropDownButton من خلال هذه الصور التي في الأعلى و الأسفل :



يتغير القيمة الظاهرة إلى الحرف A كما هو ظاهر أن جميع عناصر القائمة المنسدلة أصبحت حرف A ولكن عند اختيار العنصر الثلاث على سبيل المثال من قائمة نلاحظ أن من خلال امر طباعة قيمة المختارة النهائية قام بطباعة SY في Dubg console.



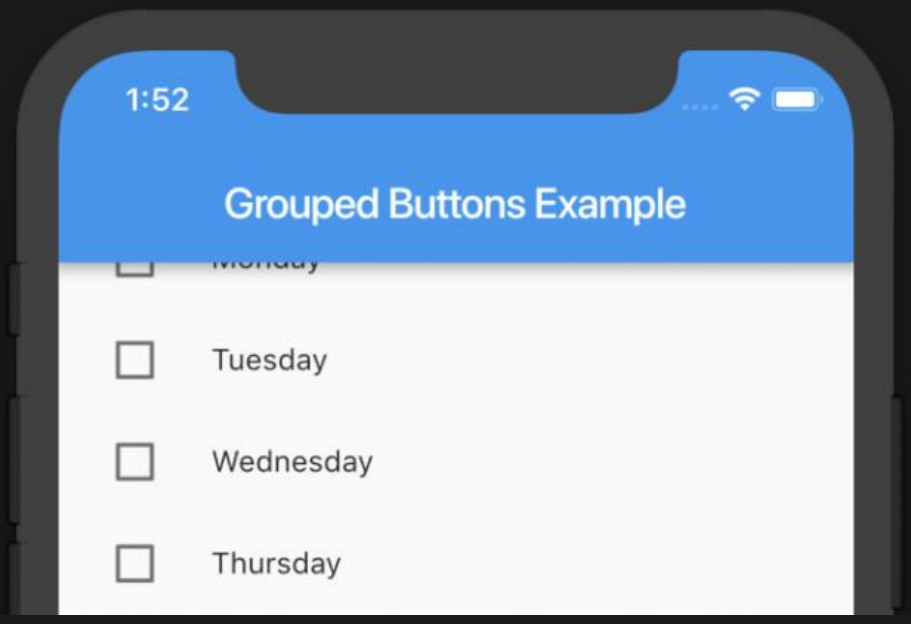
يوجد لدينا مجموعة من Properties يتمتع بها Drop Down Button

سنستعرضها في هذا الجدول :

Properties	Widget
ItemHeight إمكانية تغير ارتفاع Drop Down Button وتقبل قيمة double.	Drop Down Button <<<
Underline بشكل تلقائي يكون أسفل Drop Down Button خط خفيف نستطيع التحكم به عن طريق Divider. أو يوجد طريقة أخرى بوضع Drop Down Button ضمن Widget تسمى : DropDownButton HideUnderLine	
IsExpanded تعطي Drop Down Button كامل عرض شاشة الهاتف و تأخذ قيمة true أو false.	
IconDisabledColor تكون فعالة عندما تكون Drop Down Button معطلة وتكون onChange تأخذ قيمة null.	
Hint الملاحظة التي تظهر على DropDownButton قبل الاختيار.	

CheckBox

تستطيع من خلاله تحديد عدة اختيارات في نفس الوقت من مجموعة من الاختيارات سنتعرف الآن على مربع الاختيار وأهم Properties التي يحويها :



Properties	Widget
OnChange وهي من Properties المطلوبة مهمتها تبديل الحالة من True سيكون الحقل مختار إلى false سيكون الحقل غير مختار و بلعكس وتحتاج إلى parameter.	**CheckBox** <<<
Value من Properties المطلوبة أيضا مهمتها إعطاء قيمة ابتدائية للـ CheckBox لوحده.	

AvtiveColor لون مربع الاختيار. CheckColor لون icon داخل مربع الاختيار CheckBox.	**Check Box** **<<<**

مثال عن Check Box بشكل عملي كما هو واضح لدينا في المثال في الأسفل :

```
class _TestState extends State<Test> {
  bool usa = false ;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      drawer: Drawer(),
      body: Container(padding: EdgeInsets.all(10),child: Column(children: [

          Text("Choose Country" , style: TextStyle(fontSize: 30)) ,

          Row(children: [
            Text("USA") ,
            Checkbox(value: usa, onChanged: (val){
              setState(() {
                usa = val ;
              });
            }) // Checkbox
          ]) // Row
```

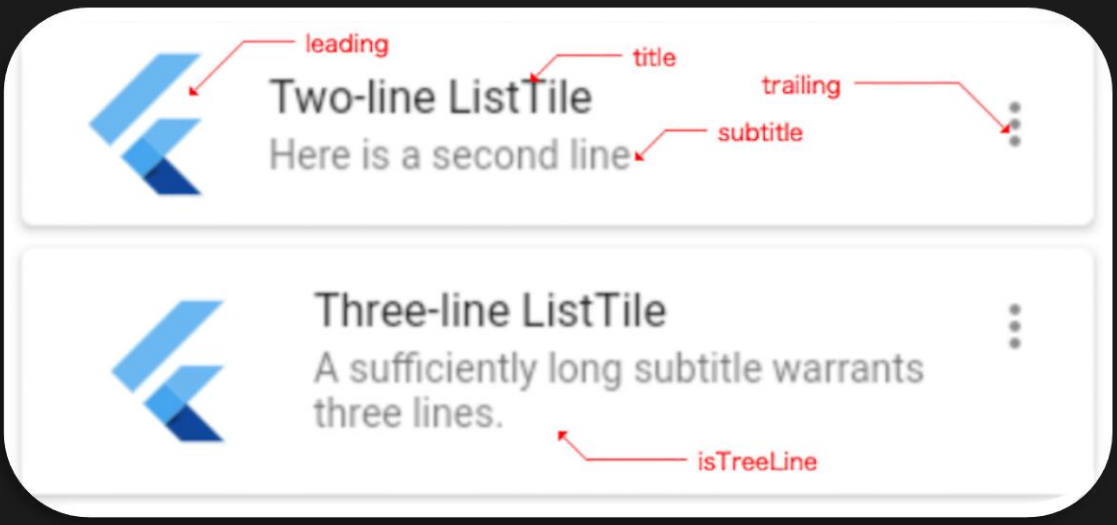
Check Box ListTile

تشبه CheckBox العادية ولكن بلأضافة لبعض Properties التي تمتع بها بلأضافة لأنها تأخذ عرض شاشة الهاتف المحمول بشكل كامل أي أنها تكون على هيئة صف Row.

Properties	Widget
title هو العنوان الرئيسي للـ Check Box List Tile	**Check Box ListTile** <<<
Subtitle هو العنوان الفرعي ويأتي غالبا أسفل title	
Secondary تقبل Widget وتظهر قبل title و Subtitle.	
isThreeLine تقبل أما true أو false مهمتها محاذاة جميع عناصر Check Box ListTile.	
Selected تقبل قيمة true أو false تأخذ لون activecolor عندما تكون true وعندما تكون القيمة false تأخذ اللـون الافتراضي أو بأمكاننا كتابة قيمة value داخل Selectes وسيتم تطبيق الحالة على حسب القيمة الابتدائية.	

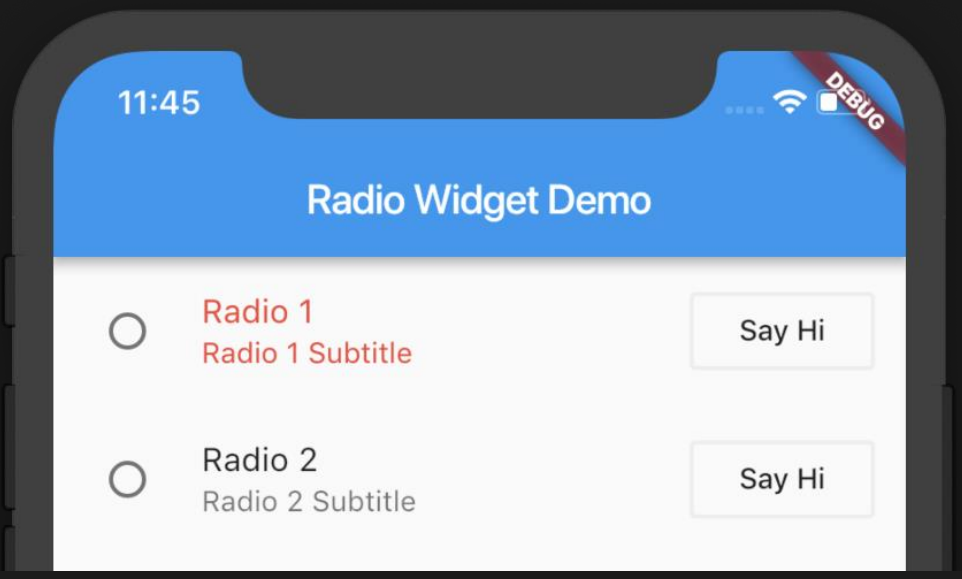
Control Affinity تقبل بداخلها List Tile Control Affinity ولها عدة حالات أهمها : ListTileControlAffinity.trailing وهي الحالة الافتراضية للـ Check Box ListTile ListTileControlAffinity.leading بعكس أماكن title و subtitle و جميع عناصر من اليمين إلى اليسار.	**Check Box ListTile <<<**

صورة توضيحية عن أهم Properties ل Check Box ListTile كما هو واضح لدينا في الصورة في الأسفل :



Radio

تستطيع من خلاله تحديد اختيار واحد فقط من مجموعة من الاختيارات.



في هذا الجدول سنقوم بشرح اهم Properties الموجودة في radio :

Properties	Widget
Value من نوع dynamic أي أنها تقبل أي قيمة من أي نوع string أو int أو	Radio <<<
OnChange تقوم بأخذ قيمة value وتمررها كـ parameter للـ function.	
groupValue يجب أن تكون هذه Properties تحمل داخلها نفس القيمة (أي أنها من مجموعة واحدة) حتى نستطيع تحديد اختيار واحد من عدة اختيارات.	

مثال عن radio تعريف groupValue كما هو واضح لدينا في الصورة في الأسفل :

```
class  TestState extends State<Test> {
  String country ;
  String car ;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(),
        drawer: Drawer(),
        body: Container(
            padding: EdgeInsets.all(10),
            child: Column(children: [
              Text("Choose Country", style: TextStyle(fontSize: 30)),
              Row(children: [
                Text("USA") ,
                Radio(value: "usa", groupValue:country , onChanged: (val){
                    setState(() {
                        country = val ;
```

تعريف groupValue

Radio One

هنا نستطيع اختيار واحد فقط لأن groupValue تحمل نفس القيمة ومن نفس المجموعة كما هو ظاهر لدينا في المثال في الأسفل :

```
],)  , // Row
  Row(children: [
  Text("egypt") ,
  Radio(value: "eg", groupValue:country , onChanged: (val){
      setState(() {
          country = val ;
          print(country) ;
      });
  }) // Radio
],) ,  // Row
  Row(children: [
  Text("sadui") ,
  Radio(value: "sa", groupValue:country , onChanged: (val){
      setState(() {
          country = val ;
          print(country) ;
      });
  }) // Radio
```

radio twe

radio three

هنا نستطيع تحديد أكثر من خيار groupValue لا تحمل نفس القيمة وليست من نفس المجموعة كما هو ظاهر لدينا في المثال في الأسفل :

```
Text("Choose Country", style: TextStyle(fontSize: 30)),
Row(children: [
  Text("USA") ,
  Radio(value: "usa", groupValue:country , onChanged: (val){
      setState(() {
          country = val ;
          print(country) ;
      });
  }) // Radio
],)  , // Row
  Row(children: [
  Text("egypt") ,
  Radio(value: "eg", groupValue:car , onChanged: (val){
      setState(() {
          car = val ;
          print(country) ;
```

Radio One

GroupValue Different

Radio Twe

Radio List Tile

تتشابه خصائص RadioListTile مع خصائص Check Box ListTile

كما هو موضح في الجدول في الأسفل :

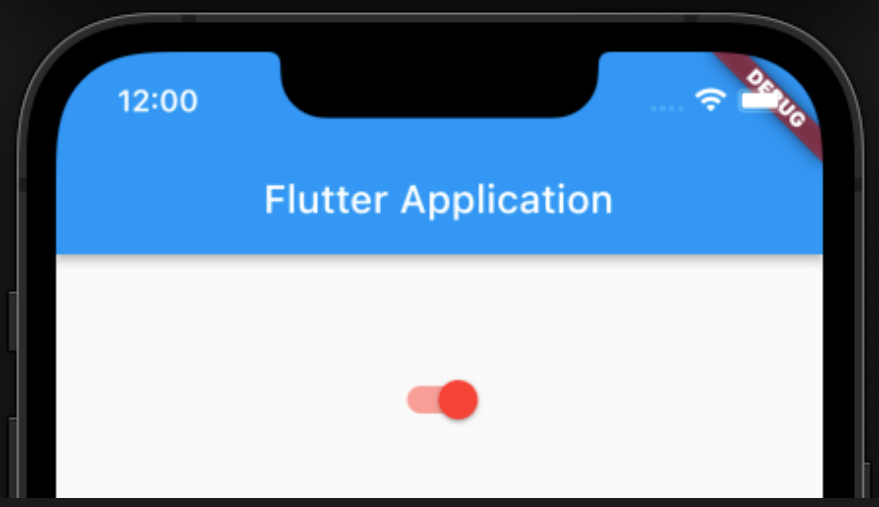
Properties	Widget
title هو العنوأن الرئيسي للـ Radio ListTile	Radio ListTile <<<
Subtitle هو العنوان الفرعي ويأتي غالبا أسفل title	
Secondary تقبل Widget وتظهر قبل title و Subtitle.	
isThreeLine تقبل أما true أو false مهمتها محاذاة جميع عناصر Check Box ListTile.	
Selected تقبل قيمة true أو false تأخذ لون activecolor عندما تكون true وعندما تكون القيمة false تأخذ اللـون الافتراضي أو بأمكاننا كتابة قيمة value داخل Selectes وسيتم تطبيق الحالة على حسب القيمة الابتدائية.	

Radio ListTile
<<<

Control Affinity تقبل بداخلها
List Tile Control Affinity ولها عدة
حالات أهمها :
ListTileControlAffinity.trailing
وهي الحالة الافتراضية للـ Radio ListTile
ListTileControlAffinity.leading
بعكس أماكن title و subtitle و جميع
عناصر من اليمين إلى اليسار.

في النهاية لن يبقى معك إلا نفسك .. فاعتن بها جيدا

Switch



Switch تعني مفتاح أي أن له حالتان أما on أو off في حالة on تكون قيمة val تساوي on في حالة off تكون قيمة val تساوي false اهم Properties سنستعرضها في هذا الجدول :

Properties	Widget
Value تقبل قيمة من نوع bool أي أن قيمتها تقبل أما true أو false.	Switch <<<
OnChange تقوم بأخذ قيمة value وتمررها كـ parameter للـ function.	
AvtiveColor تغير لون switch كامل في في حالة ON (أي أنه فعال). (يكون اللـون الافتراضي هو الأزرق)	
AvtiveTrackColor تغيير لون مسار switch فقط في حالة ON (أي أنه فعال).	

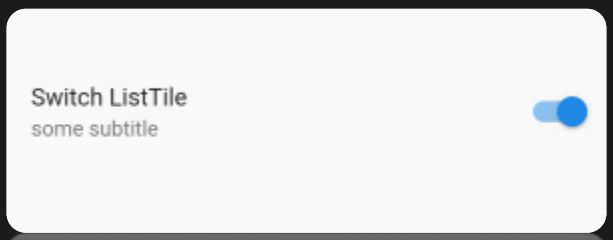
In AvtiveTrackColor تغيير لون مسارة switch فقط في حالة OFF (أي أنه غير فعال).	
In AvtiveThumbColor تغيير لون Thumb (دائرة switch) في حالة OFF (أي أنه غير فعال).	

مثال بسيط عن switch كما هو واضح لدينا في المثال في الأسفل :

```
class _TestState extends State<Test> {
  bool notify = false ;        قيمة البداية
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(),
        drawer: Drawer(),
        body:    Center(child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
          Text("هل تريد تشغيل الاشعارات")  ,
          Switch(value: notify, onChanged: (val){
            setState(() {
              notify = val ;
              print(notify) ;          Switch
            });
```

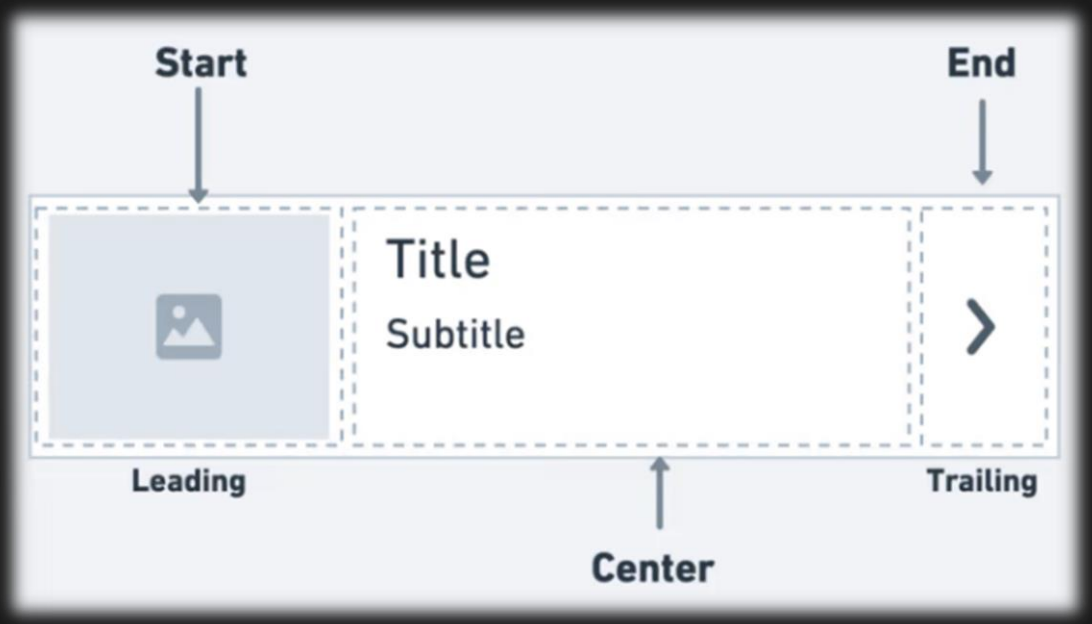
Switch ListTile

تشبه Switch العادية ولكن بلأضافة لبعض Properties التي تمتع بها بلأضافة لأنها تأخذ عرض شاشة الهاتف المحمول بشكل كامل أي أنها تكون على هيئة صف Row وتتشابه Properties الخاصة بها مع كل من check ListTile و Radio ListTile التي تعلمنها في الجداول في الأعلى.



List Tile

سنتعرف على Widget هي ليست جديدة علينا و أنما مألوفة نوعا ما بلأضافة لبعض Properties التي تمتع بها و أضف إلى ذلك أنها تأخذ عرض شاشة الهاتف المحمول بشكل كامل أي أنها تكون على هيئة صف Row وتتشابه Properties الخاصة بها مع كل من check ListTile و Radio ListTile وغيرها من Widget.



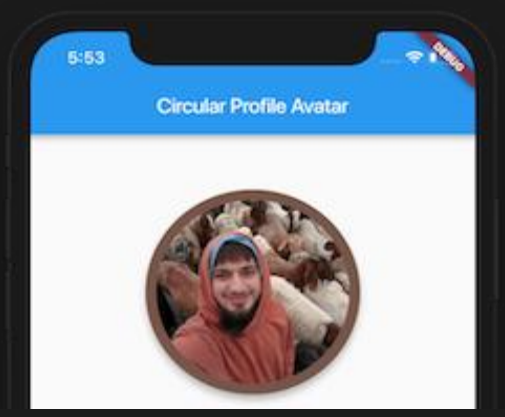
ملاحظة هامة جدا : flutter تتيح لك التحكم في Widget بعدة طرق على سبيل المثال لو قمنا بوضع List Tile ضمن Container وقمنا بتغيير لونها عن طريق container إلى لون الأحمر ثم قمنا بأستخدام خاصية TileColor لتغير لونها في نفس الوقت إلى اللـون الأخضر فأن للـون الأخضر هو الذي سيطبق في شاشة الهاتف ما هو السبب ..؟؟

السبب : الخاصية التي تكون موجودة ضمن Widget نفسها (List Tile) تكون اقوى من الخاصية التي ترثها من لآب (Container).

(هذه الحالة تنطبق على جميع Widget في flutter).

Circle Avatar

كمعنى حرفي تعني : (الصورة الرمزية الدائرة) ببساطة هي Widget عبارة عن دائرة نستطيع استخدامها داخل تطبيقنا وتحوي على عدة Properties .

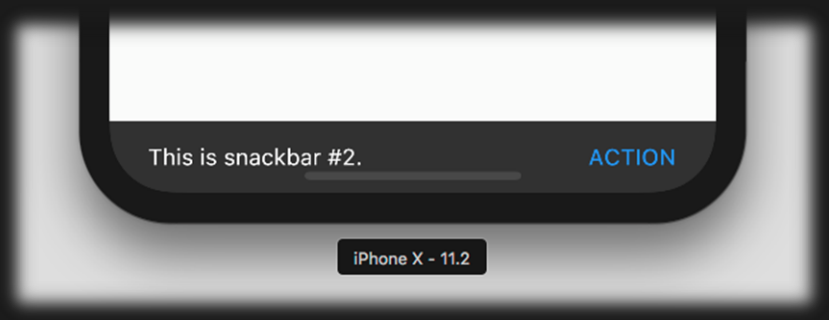


اهم هذه Properties هي child تقبل أي Widget و Radius لتغير حجم الدائرة وتكون قيمتها الافتراضية 20 بلأضافة لخاصية backgroundImage لوضع صورة خليفة للـدائرة و backgroundColor لتغيير لون خلفية الدائرة في الأسفل مثال بسيط عن كل الخصائص التي ذكرنها كما هو واضح :

```
get build(BuildContext context) {
eturn Scaffold(
  appBar: AppBar(),
  drawer: Drawer(),
  body: Container(
    color: Colors.red,
    child: Center(
      child: CircleAvatar(
        backgroundImage: AssetImage("images/1.jpg"),
        radius: 40,
        child: Text("wa" , style: TextStyle(fontSize: 30 , col
      ), // CircleAvatar
    ), // Center
  ); // Container // Scaffold
```

SnackBar

هي عبارة عن رسالة تظهر للـمستخدم لزمن معين أثناء تنفيذ حدث محدد.



1- عندما أرى key في أي Widget يجب أن نعلم أنها عبارة عن مفتاح ما مهمة هذا المفتاح..؟؟ نستطيع من خلاله الوصول لمزايا Widget يقبل متغير Global Key.

2- نقوم بأستدعاء instance من class Globalkey سيكون من نوع scaffoldState طالما أنه سيكون للـ scaffold الهدف من كل هذا هو الوصول لمزايا scaffold.

3- نقوم بأنشاء button مهمته عند الضغط عليه يقوم بأظهار SnackBar.

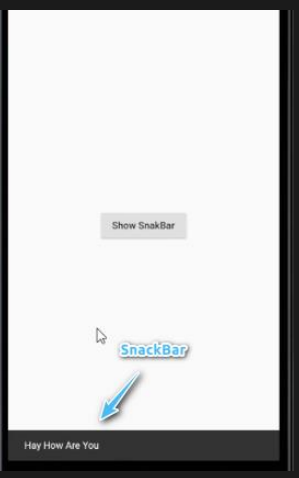
4- داخل onPressed نقوم بأستدعاء متغير الذي قمنا بأنشائه:

scaffoldkey.currentState.ShowsnakBar(snackBar)

5- نقوم بوضع snackBar ضمن المتغير Var وهذا المتغير يقبل Widget من نوع snackBar وتقبل content تقبل بداخلها قيمة من نوع Widget.

6- صورة توضيحية لجميع الشرح المذكور في الأعلى :

```
3  class Test extends StatefulWidget {
4    Test({Key key}) : super(key: key);
5    @override
6    _TestState createState() => _TestState();
7  }
8
9  class _TestState extends State<Test> {
10   GlobalKey<ScaffoldState> scaffoldkey = new GlobalKey<ScaffoldState>(); (2)
11   @override
12   Widget build(BuildContext context) {
13     return Scaffold(
14       key: scaffoldkey, (1)
15       appBar: AppBar(),
16       drawer: Drawer(),
17       body: Center(
18 (3)   child: RaisedButton(
19         onPressed: () {
20           var snackbar = SnackBar(content: Text("Hay How Are You")) ; (5)
21           scaffoldkey.currentState.showSnackBar(snackbar) ; (4)
22         },
23         child: Text("Show SnakBar"),
24       ), // RaisedButton
```



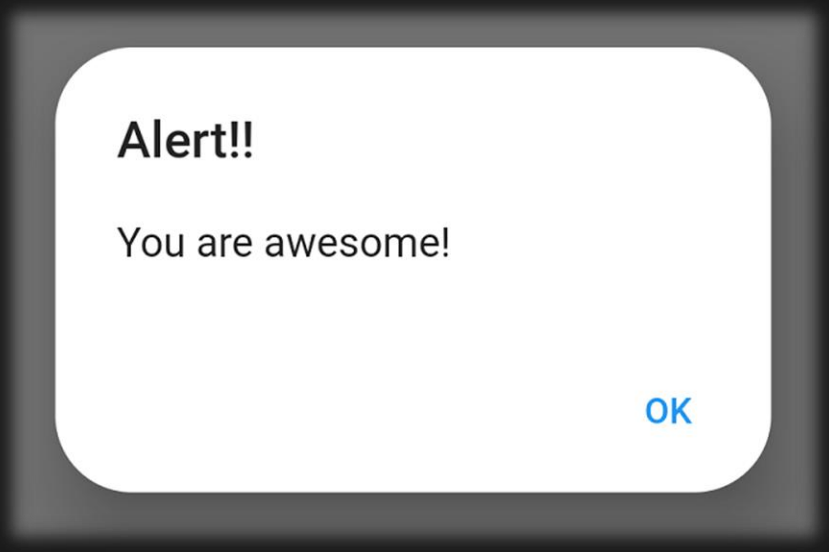
من خلال هذا الجدول سنقوم بتوضيح أهم Properties في SnackBar :

Properties	Widget
content وهي Widget أساسية في SnackBar تقبل بداخلها قيمة من نوع Widget.	SnackBar <<<
Duration مهمتها تحديد زمن لظهور SnackBar تقبل بداخلها Duration ولها عدة حالات : Days أيام. Hours ساعات. Minutes دقائق. Seconds ثوأني. (Microseconds – Milliseconds) أجزاء من ثأنية.	
backgroundColor لون خلفية SnackBar.	
Padding الهوامش الداخلية.	
Behavior كيفية ظهورSnackBar وتقبل بداخلها SnackBarBehavior ولها حالاتأن : SnackBarBehavior.fixed تأخذ عرض الشاشة بشكل كامل وهي الحالة الافتراضية للـ SnackBar ولا يمكن استخدام margin في هذا الحالة. SnackBarBehavior.floating لا تأخذ عرض الشاشة بشكل كامل (تكون بشكل عائم فوق الشاشة) و يمكن استخدام margin في هذا الحالة.	

onVisible تعمل هذه function في فترة ظهور SnackBar.
Action تقبل SnackBarAction مهمته هو عبارة عن زر يظهر SnackBar مهمته تكون بشكل افتراضي اغلاق SnackBar في فترة ظهوره بغض النظر عن زمن ظهوره يقبل عدة Properties أهمها : 1- Label اسم SnackBarAction. 2- OnPressed لعمل حدث معين عند ظهور SnackBar والضغط على زر SnackBarAction. 3- textColor لتغير لون النص في SnackBarAction.
ملاحظة : يوجد أيضا الكثير من Widget التي تحويها SnackBar ولكن اغلب هذه Properties قمنا بأخذها في الدروس المسبقة يمكنك مراجعتها وتذكرها مجدد.

Alert Dialog

هي عبارة عن نافذة تظهر عند تنفيذ حدث معين.



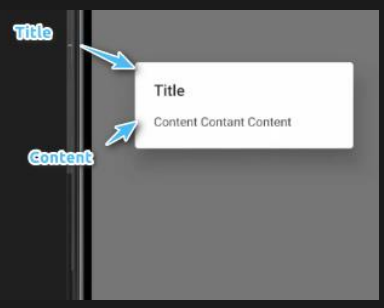
لنتعرف الأن على بعض الخصائص Alert Dialog من خلال هذا الجدول :

Properties	Widget
Context يقبل متغير من نوع build context وهو من Properties الأساسية في AlertDialog.	AlertDialog <<<
Builder هو بارامتر من نوع : Function (build context) ويقوم بأرجاع Widget من نوع : AlertDialog وهو أيضا من Properties الأساسية في AlertDialog.	
Title تقبل بداخلها Widget وغالبا يكون من نوع text.	
Content والمقصود فيه هو محتوى نافذة ويقبل بداخله Widget وغالبا يكون من نوع text أيضا.	

Titlepadding إضافة هوامش داخليه لـ title. Contentpagging إضافة هوامش داخليه لـ content.	**AlertDialog** **<<<**
TitleTextStyle إضافة Style للـنص في الـ title. ContentTextStyle إضافة Style للـنص في الـ content. Backgroundcolor لتغيير لون خلفية AlertDialog.	
Actions هي عبارة Widget تقبل بداخلها list من نوع Widget وغالبا تكون هذه الـ Widget من نوع button في اغلب التطبيقات. Actions padding أضافة هوامش داخلية للـــ actions داخل AlertDialog.	

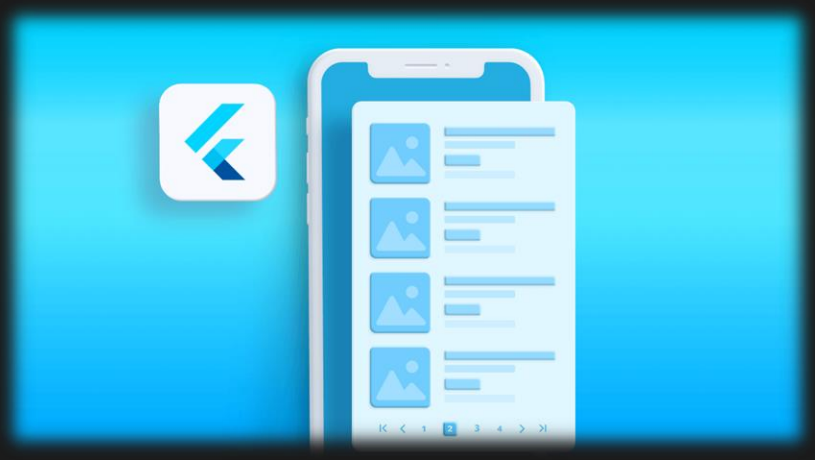
مثال توضيحية عن Alert Dialog كما هو واضح لدينا :

```
body: Center(
  child: RaisedButton(
    onPressed: () {
        showDialog(context: context ,builder: (context) {
                return AlertDialog(
                  title: Text("Title"),
                  content: Text("Content Contant Content"),
                ) ;  // AlertDialog
        }) ;
    },
    child: Text("Show Alert"),
  ), // RaisedButton
)); // Center // Scaffold
```



ListView

هي Widget تقبل بداخلها children من نوع list و list تقبل بداخلها Widget مع أمكانية التحريك scroll سواء كأن بشكل Horizontal or Vertical.

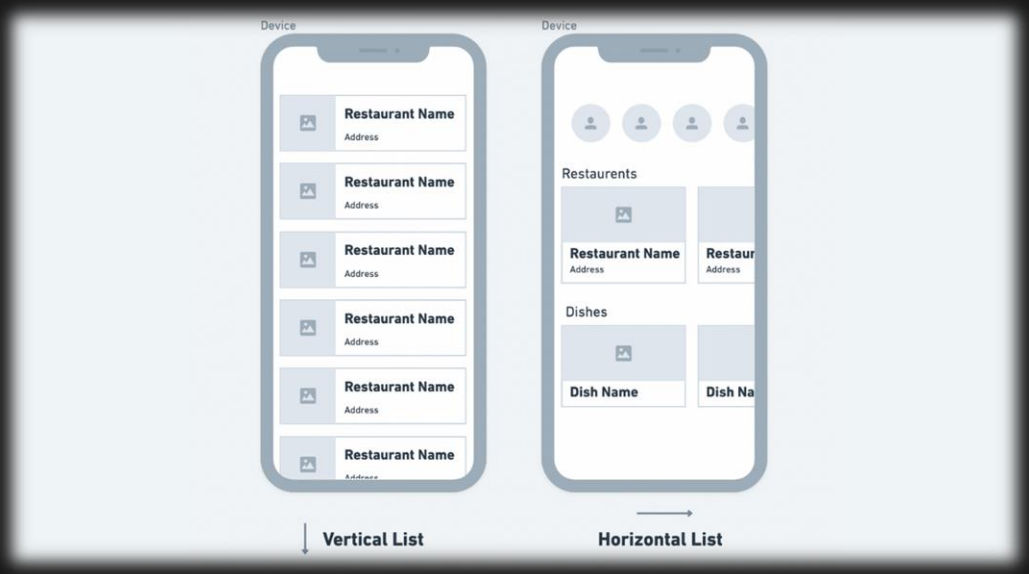


لنتعرف الأن على أهم خصائص Listview من خلال هذا الجدول :

Widget	Properties
Listview <<<	Scroll Direction اتجاه التحريك وتقبل قيمتين : ScrollDirection : Axis.Horizontal تحريك بشكل أفقي. ScrollDirection : Axis.vertical تحريك بشكل عامودي وهي الحالة الأفتراضية. ملاحظة مهمة : في الوضع الأفقي horizontal تأخذ الـ Widget طول page بشكل كامل حتى لو كأن الطول محدد. أما في الوضع العامودي vertical تأخذ الـ Widget عرض page بشكل كامل حتى لو كأن العرض محدد.

Reverse تقبل بداخلها قيمة من نوع bool أي أنها تقبل أما true أو false مهمتها: ترتيب بشكل تصاعدي مع scroll وتأخذ قيمة true(أي أن جميع Widget تنعكس ترتيبها داخل Listview من الأعلى إلى الأسفل مع scroll أيضا). أو ترتيب بشكل تنازلي مع scroll وتأخذ قيمة false وهي الحالة الافتراضية.	**Listview <<<**
Physics مقصود فيه نوع scroll ولها عدة حالات : Physics : Clamping Scroll Physics() هي الحالة الأفتراضية للـــ scroll. Physics : Bouncing Scroll Physics() عمل قفزة Bouncing للـــ scroll. Physics : Never Scrollable Scroll Physics() أيقاف scroll في Listview بشكل نهائي.	

ملاحظات مهمة حول Listview : لا يمكننا أنشاء Listview داخل Listview من دون تحديد ارتفاع Listview الداخلية ما هو السبب ..؟؟



كما نعلم أن الاتجاه الافتراضي للـــ Listview هو عامودي vertical ويأخذ طول الشاشة بشكل كامل عند وضع Listview داخل Listview سيصبح لدينا مشكلة كبير الا وهي أن الأثنان Listview سيأخذان ارتفاع الشاشة بشكل كامل وهذا غير منطقي وهنا يوجد لدينا طريقتين لحل هذا المشكلة :

- أما عن طريق Container : بوضع Listview داخل container وتحديد ارتفاع للـــ Listview عن طريقه.

- أو بأستخدام خاصية ShrinkWrap : وهي احد أهم مزايا Listview تستخدم داخلها تأخذ أما قيمة bool أما true أو false في حال تفعيلها واعطائها قيمة true هنا تقوم Listview تأخذ الأرتفاعها على حسب ارتفاع Widget التي تحويها بهذه الطريقة نستطيع أنشاء Listview داخل Listview أخرى ولكل Listview منها scroll خاص بها و Properties أيضا منفصلة عن الأخرى.

Listview Builder

هي عبارة عن Listview العادية مع جميع خصائصها بلأضافةللـــ Loop من خلالها نستطيع تكرار Widget أكثر من مرة.

Listview Builder = Listview + loop

حيث تعد من أهم Widget في flutter والسبب لكثر استخدامها بين المبرمجين 90 بلمئة من التطبيقات نستخدم فيها Listview Builder.

أهم خاصية فيها من نوع function هو itemBuilder يقبل بارمترين الأول هو من نوع context (تستطيع اختيار الاسم الذي تريد) و الثاني هو من نوع int (الرقم التكرار) وتقوم بأرجاع Widget بلأضافة للـــ itemCount لتحديد عدد مرات تكرار Widget داخل Listview Builder (في حال لم يتم استخدام itemCount سيتم تكرار Widget إلى ما لا نهاية).

مثال بسيط على Listview Builder :

```
  Test({Key key}) : super(key: key);
  @override
  _TestState createState() => _TestState();
}

class _TestState extends State<Test> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(),
        drawer: Drawer(),
        body: Container(
          child: ListView.builder(
            itemCount: 2,
            itemBuilder:  ( context , i ){
                    return Container(child: Text("Container : $i")) ;
          }) // ListView.builder
        )); // Container // Scaffold
  }
```

itemCount

Container : 0
Container : 1

1. قمنا بأنشاء list mobile ونقوم بأنشاء Map داخل list mobile.

2. ثم قمنا بأسناد itemCount قيمة length لأرجاع عدد العناصر داخل list mobile.

3. وبما أن Listview Builder ترجع قيمة من نوع Widget قمنا بأستخدام List Tile والاستفادة من جميع خصائصها لعرض عناصر List Mobile فيها كما هو واضح لدينا في المثال في الأسفل :

```
class _TestState extends State<Test> {
  List mobile = [
    {"name": "s20 ultra", "screen": "6.4", "cpu": "8 core"},
    {"name": "s21 ultra", "screen": "6.5", "cpu": "8 core"},
    {"name": "s10", "screen": "6.2", "cpu": "8 core"},
    {"name": "s9", "screen": "6.7", "cpu": "12 core"},
    {"name": "iphone 12 pro max", "screen": "6.1", "cpu": "8 core"},
  ];
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(),
        drawer: Drawer(),
        body: Container(
            child: ListView.builder(
                itemCount: mobile.length,
                itemBuilder: (context, i) {
                  return ListTile(
                    title: Text("${mobile[i]['name']}"),
                    subtitle: Text("Screen :   ${mobile[i]['screen']}"),
                    trailing: Text("cpu : ${mobile[i]['cpu']}"),
                  ); // ListTile
                })); // ListView.builder // Container // Scaffold
```

List Mobile

Length List

عدد مرات التكرار

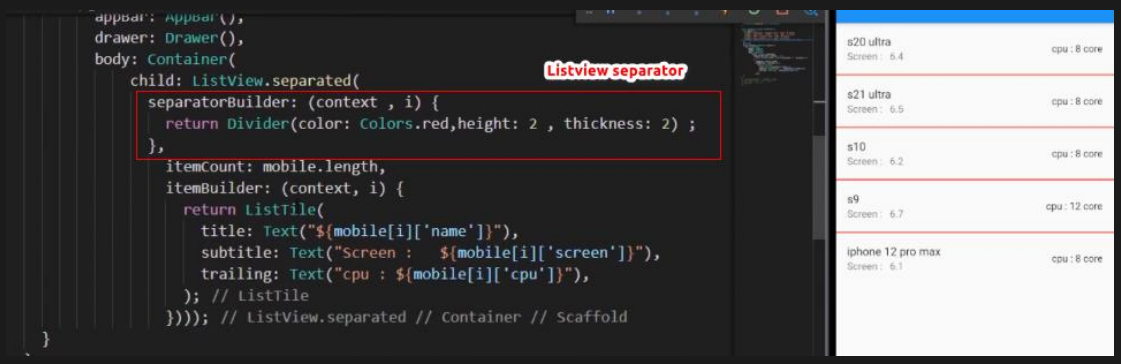
بأختصار الاستخدام الشائعة لـ Listview Builder : تأتينا المعلومات من database على هيئة List ونقوم بعرض عناصرها من خلال Listview Builder بلـ Widget التي تناسبنا .

Listview separator

تتشابه خواص Listview separator مع Listview Builder بشكل كبير مع اختلاف بسيط الأ وهو عمل فواص بين Widget و المقصود بكلمة separator هي الفاصل تقبل بداخلها separator Builder أهم خاصية فيها من نوع function هو itemBuilder يقبل بارمترين الأول هو من نوع context

(تستطيع اختيار الاسم الذي تريد) و الثاني هو من نوع int (الرقم التكرار) وتقوم بأرجاع Widget.

الشيء الوحيد الذي يساهم بتحديد قيمة الـ (i) في separator Builder بشكل اتوماتيكي وعلى حسب عدد عناصر List هو itemCount داخل Listview Builder كما هو واضح لدينا في المثال في الأسفل :



GridView Builder

من خلال دراستنا السابقة للـــ Listview كل عنصر من عناصر Listview يأخذ سطر كامل لوحده لو اردنا على سبيل المثال أن نعرض أكثر من عنصر في نفس الصف هنا تأتي مهمة GridView بلأضافة إلى أن جميع Properties الموجودة في Listview موجودة أيضا في GridView.

Widget	Properties
GridView Builder <<< المقصود بكلمة Builder أي أنها تحتوي على loop.	ولكن تختلف فقط بخاصية واحدة التي GridDelegate تقبل بداخلها :
	SliverGridDelegateWithMaxCrossAxisExtent( )
	وتحوي على عدة Properties أهمها :
	CrossAxisCount وتقبل بداخلها عدد صحيح لتحديد عدد عناصر GridView في كل صف.
	CrossAxisSpacing
	تقبل قيمة من نوع double وهو المحور الحقيقي.
	MainAxisSpacing
	تقبل قيمة من نوع double وهو المحور الوهمي.
	ChildAspectRatio
	تقبل قيمة Double تقوم بأعطاء طول عناصر GridView بحسب طول وعرض شاشة الهاتف وتكون اقرب للـمربع.

- ملاحظة : نفس الخواص التي ذكرنها في ListView نفساها في GridView العادية.

Gridview Count

تعمل نفس عمل Gridview العادية ولكن الاختلاف الوحيد أنها تحوي بداخلها جميع خصائص GridDelegate مثل : crossAxisCount - crossAxisSpacing mainAxisSpacing ولكن دون الحاجة لكتابتها.

وهذا مثال يوضح هذا الامر :

```
Widget build(BuildContext context) {
  List user = ["wael", "basel", "mohmmad" , "majed"];
  return Scaffold(
      appBar: AppBar(),
      drawer: Drawer(),
      body: Container(
          padding: EdgeInsets.all(10),
          child: GridView.count(
            crossAxisCount: 3,
            children: [
                        Container(child: Text("Container One"),color: Colors.r
                        Container(child: Text("Container two"),color: Colors.g
                        Container(child: Text("Container three"),color: Colors
                        Container(child: Text("Container four"),color: Colors.
                        Container(child: Text("Container five"),color: Colors.
            ]
            ), // GridView.count
          )); // Container // Scaffold
}
```

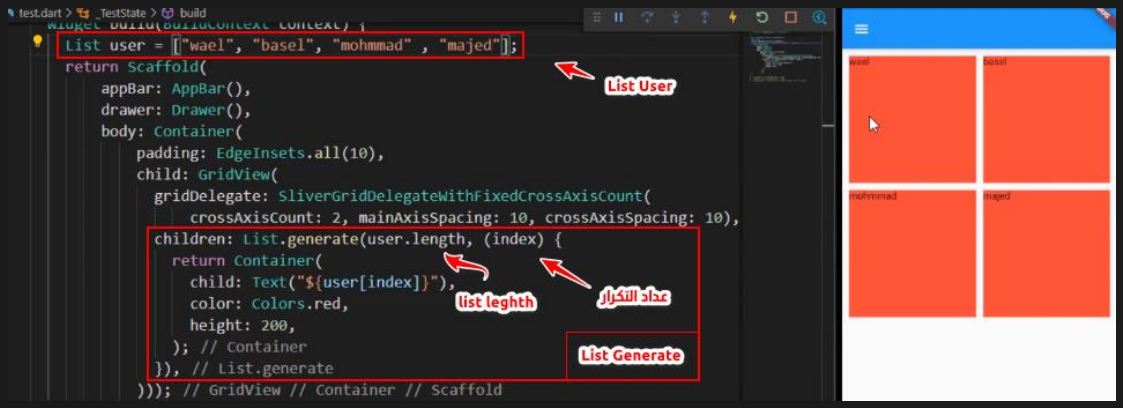
List Generate

تستطيع في flutter كتابة الكود بأكثر من شكل وجميع هذا الأشكال ممكن في النهاية أن تعطيك النتيجة ذاتها ومثال على ذلك هي List Generate.

List Generate : هي عبارة عن Widget تقوم بنفس مهمة GridView Builder و listView Builder الأ وهو (Loop) ولكن بطريقة كتابة مختلفة تعد List Generate من Widget البديل لكل منهما.

تقبل بداخلها بارمترين الأول هو من نوع Length (هو طول List) و الثاني هو من نوع int (الرقم التكرار) وتقوم بأرجاع Widget.

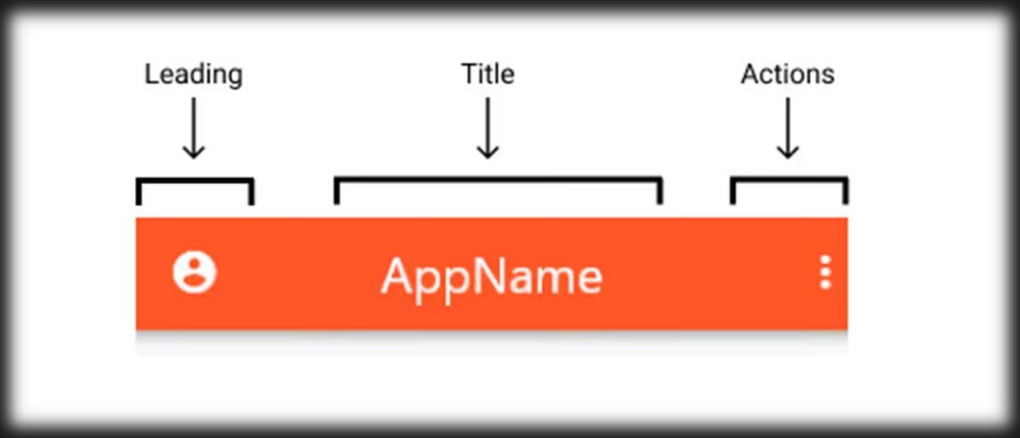
لنأخذ مثال بيسط عن هذه List Generate كما هو واضح لدينا في الأسفل :



- ملاحظة : ليس من الضروري التركيز في هذه Widget أنما عليك فقط فهمها لأنه هي عبارة عن طريقة بديلة لكتابة الكود ليس الأ.

AppBar

هو عبارة عن الشريط العلوي للـتطبيق يحوي على عدة أقسام رئيسية مثل (Leading – Title – Action).



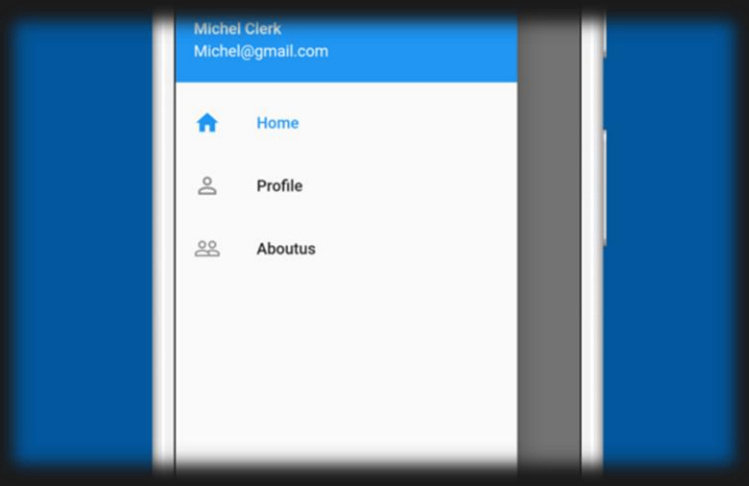
Properties	Widget
Title تقبل بداخلها Widget وبأغلب التطبيقات يتم استخدامها لوضع text بداخلها.	AppBar <<<
Leadind تظهر قبل title و تقبل بداخلها Widget وبأغلب التطبيقات يتم استخدامها لوضع icon button بداخلها.	
Actions تقبل بداخلها list من نوع Widget وبأغلب التطبيقات يتم استخدامها لوضع icon button بداخله.	
Elevation ظل AppBar يقبل بداخله قيمة من نوع double. Shadow Color لون الظل.	

AppBar <<<	Leadind width تحكم بعرض leading ضمن AppBar وتقبل قيمة من نوع double.
	Background Color تغيير لون AppBar.
	Brightness تحكم بلون الأيقونات في شريط الأشعارات notification bar وتقبل Brightness ولها حالتان dark و light.
	Center Title وضع title في منتصف AppBar وتقبل قيمة من نوع bool أما true أو false.

لا أحد يعلم كم كلفك هذا النجاح

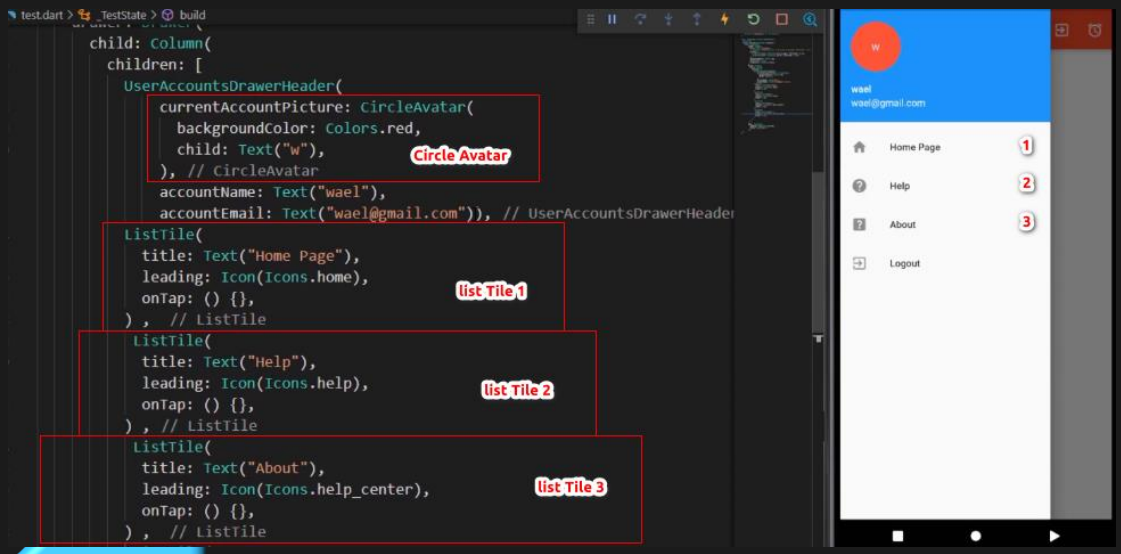
Drawer - EndDrawer

هي عبارة عن قائمة جانبية يمكن تخصيصها على حسب احتياج المستخدم تظهر عند السحب على حسب الاتجاه من اليسار إلى اليمين Drawer أو بعكس EndDrawer.



عند استعمال leading في AppBar وفي حال وجود Drawer في نفس الوقت هذا الأمر يؤدي إلى اختفاء أيقونة Drawer وظهور أيقونة leading ونفس الامر بلنسبة للـــ EndDrawer و Actions في AppBar.

مثال بسيط عن Drawer قمنا بأستخدام UserAccount ضمن Drawer لتنفيذ التصميم بسهولة وسرعة اكبر كما هو وضاح ليدنا في المثال في الأسفل :



يمكننا أيضا أنشاء button ومن خلاله نستطيع فتح Drawer كيف ذلك من خلال key التي سبق وتحدثنا عنها في الدروس السابقة وأن مهمته الوصول إلى مزايا Widget كما هو واضح لدينا في المثال في الأسفل :

```
class _TestState extends State<Test> {
  GlobalKey<ScaffoldState> scaffoldkey = new GlobalKey<ScaffoldState>() ;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        key: scaffoldkey,
```

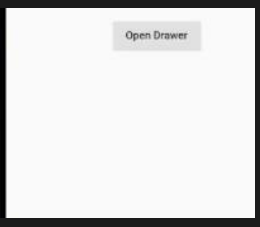
instance Global Key

Key

وقمنا بأنشاء button من خلال الضغط عليه يقوم بفتح Drawer كما هو واضح لدينا في المثال في الأسفل :

```
body: Center(
  child: Container(
    padding: EdgeInsets.all(10),
    child: RaisedButton(onPressed:(){
      scaffoldkey.currentState.openDrawer()
    }, child: Text("Open Drawer"),), // RaisedButton
  ), // Container
)); // Center // Scaffold
```

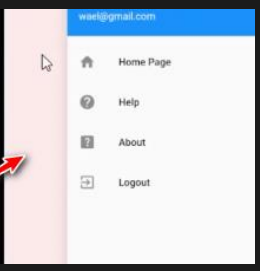
Scaffold Key



توجد خاصية في Scaffold من خلالها نستطيع التحكم بلون في Drawer Scrim Color وأيضا تحكم في شفافية من خلال withOpacity وتقبل قيمة double من 0 إلى 1 كما هو واضح لدينا في المثال في الأسفل :

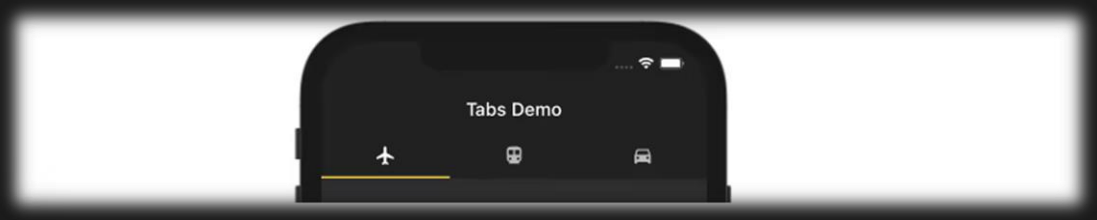
```
    ],
    backgroundColor: Colors.red,
    centerTitle: true,
    brightness: Brightness.dark,
  ), // AppBar
  drawerScrimColor: Colors.red.withOpacity(0.1),
  endDrawer: Drawer(
    child: Column(
      children: [
        UserAccountsDrawerHeader(
          currentAccountPicture: CircleAvatar(
            backgroundColor: Colors.red,
```

DrawerScrimColor



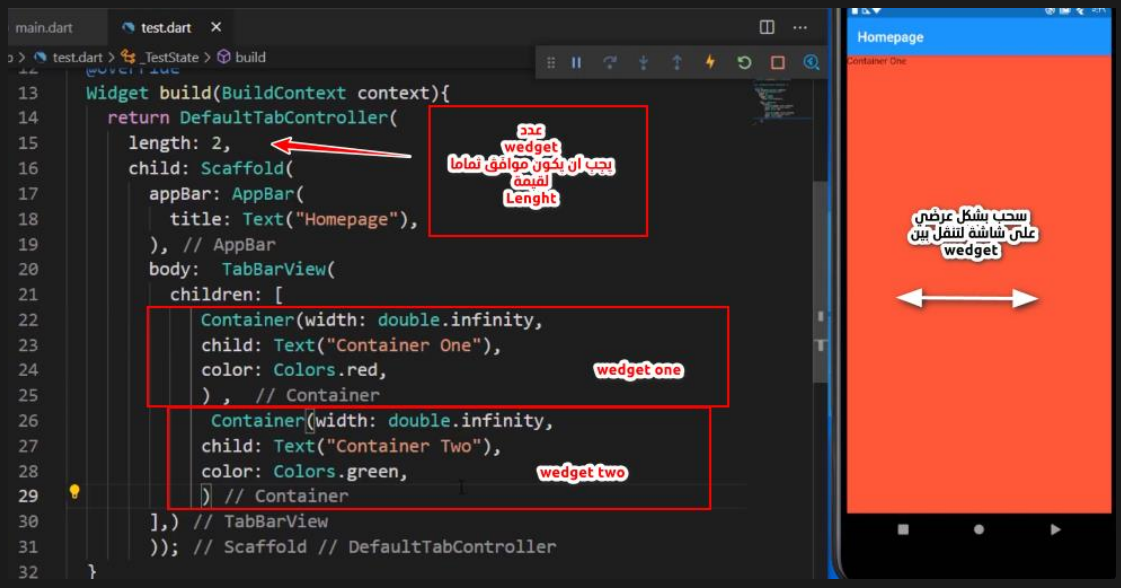
TabBar View

هي من Widget master تقبل لداخلها children أي أنها تقبل list من نوع Widget يتم التنقل بين Widget عن طريقها بسحب بشكل عرضي على شاشة.



معلومة جدا مهمة عن TabBarView :

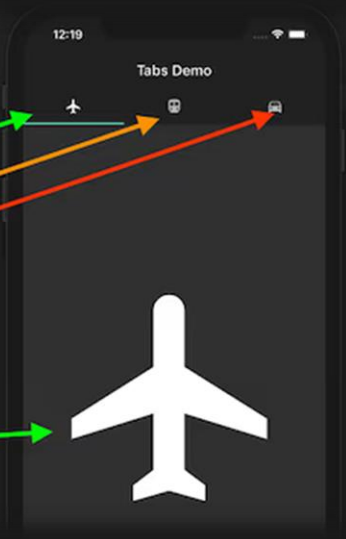
يجب أن يكون قيمة Length متوافقة تماماً مع عدد Widget (الابن المباشر داخل TabBarView) داخل TabBarView في حال كأنت اكبر أو اقل من قيمة length سيظهر خطأ في التطبيق



TabBar And Tabs

تحكم بلـ Widget الموجودة ضمن TabBarView من خلال Taps.

```
class _TabBarDemoState extends State<TabBarDemo> {
  @override
  Widget build(BuildContext context) {
    return DefaultTabController(
      length: 3,
      child: Scaffold(
        appBar: AppBar(
          bottom: TabBar(
            tabs: [
              Tab(icon: Icon(Icons.flight)),
              Tab(icon: Icon(Icons.directions_transit)),
              Tab(icon: Icon(Icons.directions_car)),
            ],
          ), // TabBar
          title: Text('Tabs Demo'),
        ), // AppBar
        body: TabBarView(
          children: [
            Icon(Icons.flight, size: 350),
            Icon(Icons.directions_transit, size: 350),
            Icon(Icons.directions_car, size: 350),
```

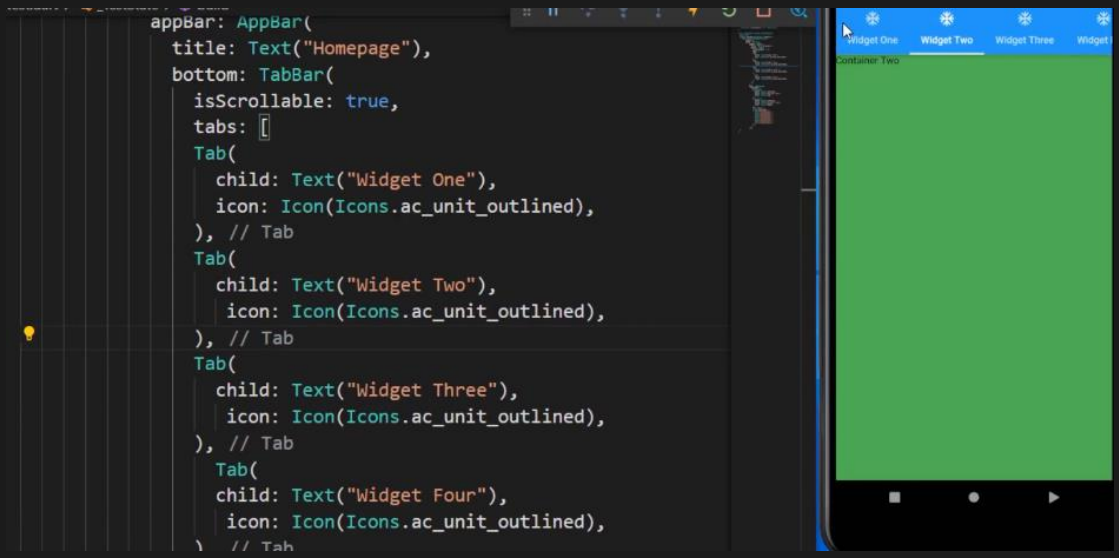


من خلال الجدول هذا سنتعرف على أهم Properties الشائعة في TabBar :

Properties	Widget
Taps تقبل بداخلها List من نوع Tap و Tap تحوي بداخلها على أهم Property شائعة ومستخدمة في TapBAR وهي : child وتقبل Widget كما نعلم Icon تقبل بداخلها أيقون.	**TabBar And Tabs** توجد خاصية في AppBar تسمى bottom تقبل بداخلها Property من نوع TabBar و TabBar تقبل بداخلها taps . <<<
isScrollable من خلالها تستطيع تحريك TabBar بشكل عرضي تقبل قيمة من نوع bool أما true أو false.	
Label color تغيير لون النص والأيقونة داخل TabBar.	

Indicater color تغيير لون الخط الذي يوجد أسفل TabBar.	
Indicater weight تغيير سمك الخط الذي يوجد أسفل TabBar.	
Indicater padding إضافة هامش داخلي للـخط الذي يوجد أسفل TabBar.	

ملاحظة مهمة : أيضا في **TabBar** يجب توفق عدد Widget داخل Taps مع كل من قيمة Legth داخل Default Tap Controller وعدد Widget داخل TabBarView كما هو واضح لدينا في المثال في الأسفل :



initState( );

initState( ) ببساطة عندما يتم الدخول إلى صفحة معينة في التطبيق يتم استدعائها بشكل مباشر عند الدخول إلى الصفحة من دون تنفيذ أي حدث من قبل المستخدم مثلها مثل SetState( ) يتم استدعاها فقط في StateFulWidget.

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

  noteControllerYT =
```

Tab Controller

TabController مرتبطة ارتباط وثيق من initState( ) في المثال السابق في TabBar قمنا بأستخدام Default Tap Controller حيث يقوم بأنشاء Controller بشكل تلقائيا سنأخذ مثال عن هذا الامر عن ريقة أنشاء Controller بشكل يدوي بعدة خطوات :

نقوم بأنشاء Controller نسميه myController في TabBar كما هو واضح لدينا في المثال في الأسفل :

```
Widget build(BuildContext context) {
  return Scaffold(
        appBar: AppBar(
          title: Text("Homepage"),
          bottom: TabBar(
              controller: mycontroller,
              isScrollable: true,
              indicatorColor: Colors.red,
              indicatorWeight: 2,
              labelColor: Colors.white,
```

Controller

نقوم بأنشاء Controller نسميه myController في TapBarView كما هو واضح لدينا في المثال في الأسفل :

```
body: TabBarView(
  controller: mycontroller,
  children: [
    Container(
      width: double.infinity,
      child: Text("Container One"),
      color: Colors.red,
    ), // Container
    Container(
      width: double.infinity,
```

Controller

نقوم أيضا بأنشاء Controller في initState( ) كما هو واضح لدينا في المثال في الأسفل :

```
class _TestState extends State<Test> with SingleTickerProviderSta

  TabController mycontroller ;

  @override
  void initState() {
    mycontroller = new TabController(length: 4, vsync: this)

    super.initState();
```

Controller

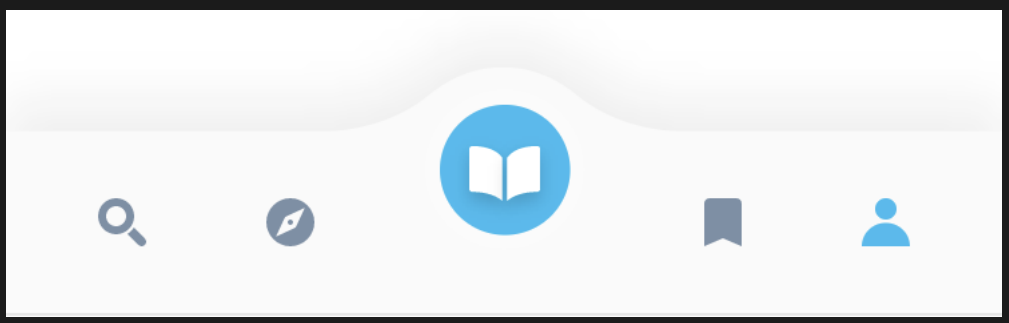
عدد wedget

نستطيع تحديد الصفحة أو page التي نريد عرضها أولا عند بدأ التطبيق أو عند الدخول إلى التطبيق من خلال initial index في initState( ) كما هو واضح لدينا في المثال في الأسفل :

```
w TabController(length: 4, vsync: this  , initialIndex: 2) ;

;
```

Bottom Navigation Bar

هي البار السفلية في أي تطبيق تحوي بداخلها على icon و lable نستطيع من خلالها تنقل بين صفحات التطبيق.



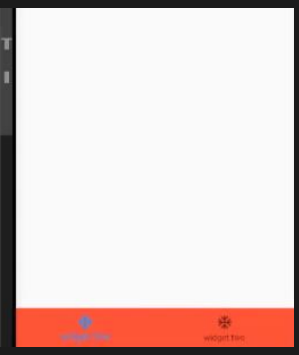
لنستعرض الأن أهم Properties في BottomNavigationBar من خلال هذا الجدول الذي في الأسفل :

Widget	Properties
Bottom Navigation Bar تقبل بداخلها BottomNavigationBar <<<	BottomNavigationBar وتحوي على items تقبل list من نوع BottomNavigationBarItem وتحوي بداخلها على عدة Properties مهمة مثل : label وهو نص item و Icon الخاصة بلـ item.
	backgroundColor تغيير لون BottomNavigationBar.
	Current index تحديد item الفعالة في BottomNavigationBar وتقبل قيمة من نوع int وهو رقم العنصر داخل list.

Selected item color تغيير لون item الفعالة بشكل كامل. Un Selected item color تغيير لون item الغير الفعالة بشكل كامل.
Selected lable style تقبل textStyle و textStyle تعطينا مجموعة من الخصائص لتحكم بلنص مثل اللــون الحجم ونوع وغيرهاا.
onTap تنفيذ حدث معين عن اختيار item من BottomNavigationBar.

مثال بسيط على طريقة بناء BottomNavigationBar في flutter كما هو واضح لدينا في المثال في الأسفل :

```
 _TestState extends State<Test> {
erride
get build(BuildContext context) {
eturn Scaffold(
  appBar: AppBar(
    title: Text("Homepage"),
  ), // AppBar
  bottomNavigationBar: BottomNavigationBar(
    backgroundColor: Colors.red,
    items: [
    BottomNavigationBarItem(label: "widget One" , icon: Icon(Icons.ac_unit)),
    BottomNavigationBarItem(label: "widget two" , icon: Icon(Icons.ac_unit)),
  ],), // BottomNavigationBar
  body: Text("Wae")); // Scaffold
```



Page View

هي من تعتبر من Widget master بحيث تقبل داخلها children و children تقبل List من نوع Widget مهمتها الأنتقال بين Widget التي تحويها من خلال اللـمس سواء كأن بشكل عامودي أو أفقي.

```
class _TestState extends State<Test> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Text("Homepage"),
        ), // AppBar
        body: Column(children: [
          Container(
              height: 300,
              child: PageView(
                children: [
                  Image.asset("images/1.jpg" , fit: BoxFit.fill) ,
                  Image.asset("images/2.png" , fit: BoxFit.fill) ,
                  Image.asset("images/3.png" , fit: BoxFit.fill) ,
                  Image.asset("images/4.jpg" , fit: BoxFit.fill) ,
                ],
              )) // PageView // Container
        ])); // Column // Scaffold
```

لنتعرف الأن على بعض الخواص PageView من خلال هذا الجدول :

Properties	Widget
Reverse وتقبل قيمة من نوع bool أما true أو false مهمتها عكس اتجاه التنقل بين Widget داخل PageView.	PageView <<<
scrollDirection تقبل قيمتأن : scrollDirection : Axis.vertical اتجاه التنقل بشكل عامودي . scrollDirection : Axis. Horizontal اتجاه التنقل بشكل أفقي.	
onPageChanged تعطي حالة الأنتقال وتقبل بارامتر من نوع int على سبيل المثال نسمه index ويقوم بأرجاع رقم Widget داخل list ضمن PageView.	

Controller وهو المتحكم يوجد في كل Widget تحوي على المتحكم يتم التحكم بها عن بعد من خلاله

PageController يتم استخدامها بعد أنشاء controller في PageView ويقبل عدة Properties:

initialPage : تحديد page التي سيتم عرضها عند بدأ تشغيل التطبيق وتقبل قيمة من نوع int وترجع رقم Widget داخل PageView.

View port fraction تقبل قيمة من 0 إلى 1 من نوع double والقيمة الافتراضية لها هي 1 حيث أنها تأخذ كل Widget في PageView كامل عرض الشاشة.

مثال بسيط عن page controller كما هو واضح لدينا في الأسفل :

```
PageController pc ;

@override
void initState() {
  pc = new PageController(initialPage: 1 , viewportFraction: 0) ;
  super.initState();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(
        title: Text("Homepage"),
```

إمكأنية التحكم بلـ Widget الداخلية من خلال controller خارجي

في هذا المثال نريد الأنتقال إلى الصورة الثالثة في pageView من خلال button عند الضغط عليه.

كأول خطوة قمنا بأنشاء button ومن خلال onPressed قمنا بأستدعاء المتحكم pc ثم قمنا بأستدعاء method خاصة بلأنيميشن تقبل بداخلها قيمة من نوع int وهو رقم العنصر داخل list في pageView كما هو واضح لدينا في المثال في الأسفل :

pageView Builder

هي عبارة عن pageView العادية مع جميع خصائصها بلأضافة للـــ Loop من خلالها نستطيع تكرار Widget أكثر من مرة.

pageView Builder = pageView + loop

حيث تعد من أهم Widget في flutter والسبب لكثر استخدامها بين المبرمجين 90 بلمئة من التطبيقات نستخدم فيها pageView Builder.

أهم خاصية فيها من نوع function هو itemBuilder يقبل بارمترين الأول هو من نوع context (تستطيع اختيار الاسم الذي تريد) و الثأني هو من نوع int (الرقم التكرار) وتقوم بأرجاع Widget بلأضافةللـــ itemCount لتحديد عدد مرات تكرار Widget داخل Listview Builder (في حال لم يتم استخدام itemCount سيتم تكرار Widget إلى ما لا نهأية).

مثال بسيط على pageView Builder قمنا بأنشاء list images كما هو واضح:

```
17
18  List images = [
19    {"url" : "images/1.jpg" } ,
20    {"url" : "images/2.png" } ,
21    {"url" : "images/3.png" } ,
22    {"url" : "images/4.jpg" } ,
23  ] ;
```

ثم قمنا بأستدعاء list images في pageView Builder كما هو واضح لدينا :

```
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(
        title: Text("Homepage"),
      ), // AppBar
      body: Column(children: [
        Container(
            height: 300,
            child: PageView.builder(
              controller: pc,
              itemCount: images.length,
              itemBuilder: (context , i){
                return Image.asset(images[i]['url']) ;
              },
            ) ,  // PageView.builder // Container
```

Text Form Field

حقول الأدخال : يمكنك استخدام حقل الإدخال النصي في تطبيق حيث أنه من الأكثر الأمور المستخدمة في التطبيقات والمواقع بحيث تريد من المستخدم إدخال بيانات معينة مثل بيانات تسجيل (اسم المستخدم _ كلمة المرور) أو تسجيل بيانات شخصية خاصة في المستخدم.

لنتعرف الأن على بعض الخواص TextFormField من خلال هذا الجدول :

Widget	Properties
TextFormField تقبل بداخلها Decoration و تقبل Decoration وتقبل Input Decoration خاصة بلـ Properties TextFormField <<<	Hint Text تقبل بداخلها String مهمتها تظهر للـمستخدم داخل حقل الأدخال تعرفه ماذا يجب أن يدخل وعند كتابة أول حرف فيه تختفي hint.
	Hint Style تقبل بداخلها Text Style مهمتها إضافة مؤثرات نصية على Hint text.
	Hint Max Line تقبل قيمة من نوع int مهمتها تحديد عدد الأسطر الـ Hint Text.
	Prefix والمقصود بها السابقة لها عدة أنواع وتقبل بداخلها Widget.
	Prefix icon وهو نوع من أنوع prefix وتقبل بداخلها أيقونة تأتي في بداية الحقل ولا تختفي عند بدأ أدخال البيانات.

Prefix text وهو نزع من أنوع prefix وتقبل بداخلها text ويكون على هيئة نص ثابت في بداية الحقل لا يختفي و لا يمكن ازالته.	
Sufix نفس خواص prefix ولكن تأتي في نهاية TextFormField وله أيضا عدة أنواع مثل : Sufix text – Sufix icon	
fillColor إعطاء لون لحقل الأدخال يجب تفعيل filled قبل استخدام fillColor حيث تقبل قيمة من نوع bool أما true أو false.	
Label Text تقبل بداخلها text تشبه hint من حيث المبدأ اذ أنها عبارة عن ملاحظة تظهر على الحقل لتعرف المستخدم م الذي يجب أن يدخله في الحقل الا أنها لا تختفي مثل hint عند بدأ الكتابة أنما ترتفع إلى الأعلى قليلا ويصغر حجمها ولها خاصية أخرى وهي lable Style مهمتها إضافة مؤثرات نصية على lable text.	

يوجد في TextFormField عدة حالات للأطار المحيط أو الخارجي (border) ويأخذ شكلين :

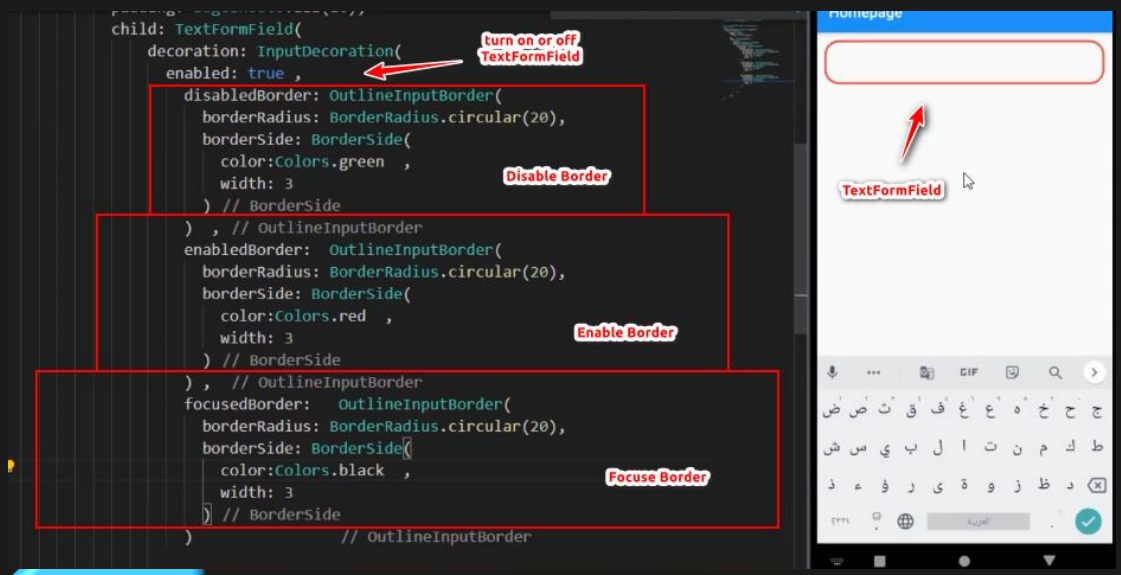
- OutLineInputBorder أطار خارجي من جميع اتجاهات TextFormField.
- UnderLineInputBorder أطار سفلي للـــ TextFormField.

1. **أول حالة وهي disableBorder :** نستطيع التحكم بخصائص الأطار الخارجي (border) للـــ TextFormField في حال كأن TextFormField في حالة false أي أنه غير مفعل من خلال enabled وتقبل قيمة bool أما true أو false.

2. **الحالة الثانية وهي enabledBorder :** نستطيع التحكم بخصائص الأطار الخارجي (border) للـــ TextFormField في حال كأن TextFormField في حالة true أي أنه مفعل من خلال enabled.

3. **الحالة الثالثة وهي FocuseBorder :** هذه الحالة تعمل عند وضع المؤشر داخل TextFormField أو اللـمس داخله وتقبل Properties مثل borderRadius و borderSide.

مثال عن حالات الثلاث التي تم ذكرها في الأعلى كما هو واضح لدينا :



TextFormField Properties	
Icon تظهر خارج TextFormField على عكس preFixIcon التي تظهر بداخل TextFormField.	cursorColor وهو لون مؤشر الكتابة داخل حقل الادخال ويكون لونه الافتراضي هو الأزرق.
cursonWidth تحكم بعرض المؤشر الكتابة داخل حقل الأدخال ويقبل قيمة من نوع double.	cursonHeight تحكم بأرتفاع المؤشر الكتابة داخل حقل الأدخال ويقبل قيمة من نوع double.
keyboardType وتقبل بداخلها textInputType من خلالها نستطيع أختيار نوع keyboard الذي نريد أن يظهر للـمستخدم عند نقر فوق حقل الأدخال ويوجد عدة أنواع : keyboardType : textInputType.datatime keyboardType : textInputType.phone keyboardType : textInputType.url keyboardType : textInputType.number	
maxLenght تقبل قيمة من نوع int وتستخدم لتحديد الحد الاقصى لعدد الحروف داخل TextFormField.	maxLiens تقبل قيمة من نوع int وتستخدم لتحديد الحد الأقصى لعدد سطور الكتابة داخل حقل الأدخال (لا يوجد حد معين لعدد الحروف عند استخدام maxLines داخل حقل الأدخال يجب أن نفرق).
Minliens تقبل قيمة من نوع int وتستخدم لتحديد الحد الأدنى لعدد سطور الكتابة داخل حقل الأدخال	obscureText تقبل قيمة من نوع bool أما true أو false تستخدم مع كلمات المرور لأخفاء مدخلات المستخدم وتحويلها إلى نقاط.

readOnly تقبل قيمة من نوع bool مهمتها جعل حقل الأدخال فقط قابل للـقراءة أو النسخ.	Style تغيير تنسيق الخط داخل حقل الأدخال تقبل بداخلها textStyle.

textInputAction مهمتها تغيير نوع وشكل زر الأدخال داخل keyboard تقبل بداخلها textInputAction ولها عدة أنواع :

textInputAction : textInputAction.done

textInputAction : textInputAction.go

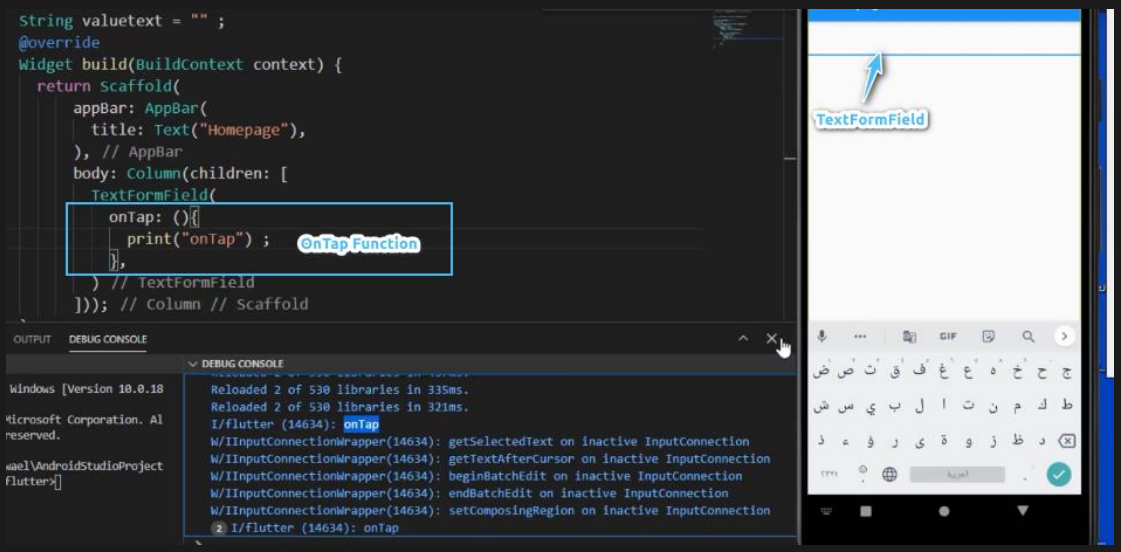
textInputAction : textInputAction.search

textInputAction : textInputAction.join

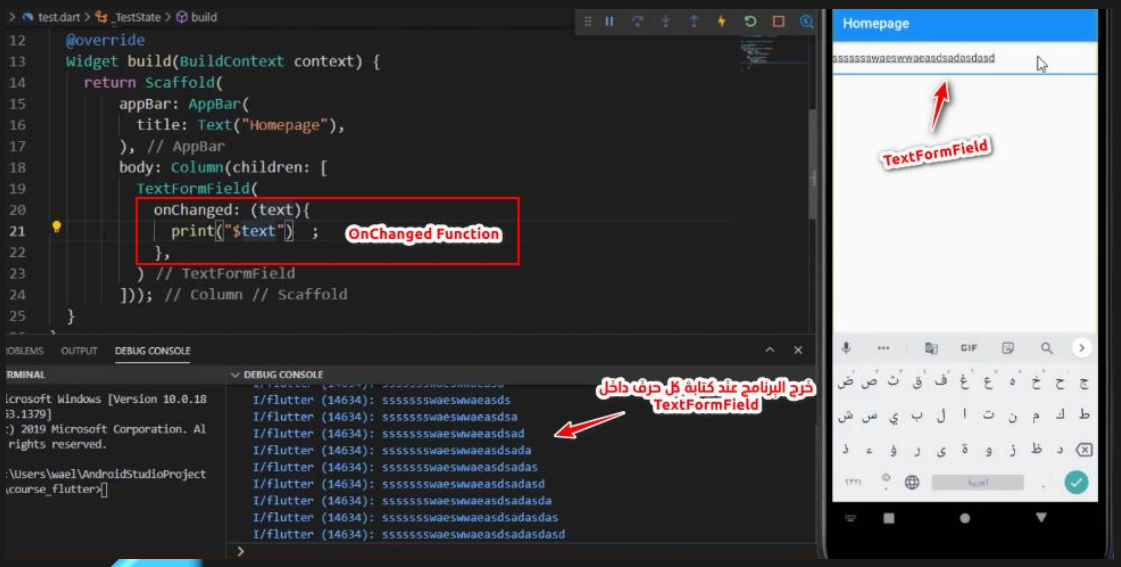
TextFormField (function)

توجد في TextFormField عدة أنواع من function سنتعرف اليها.

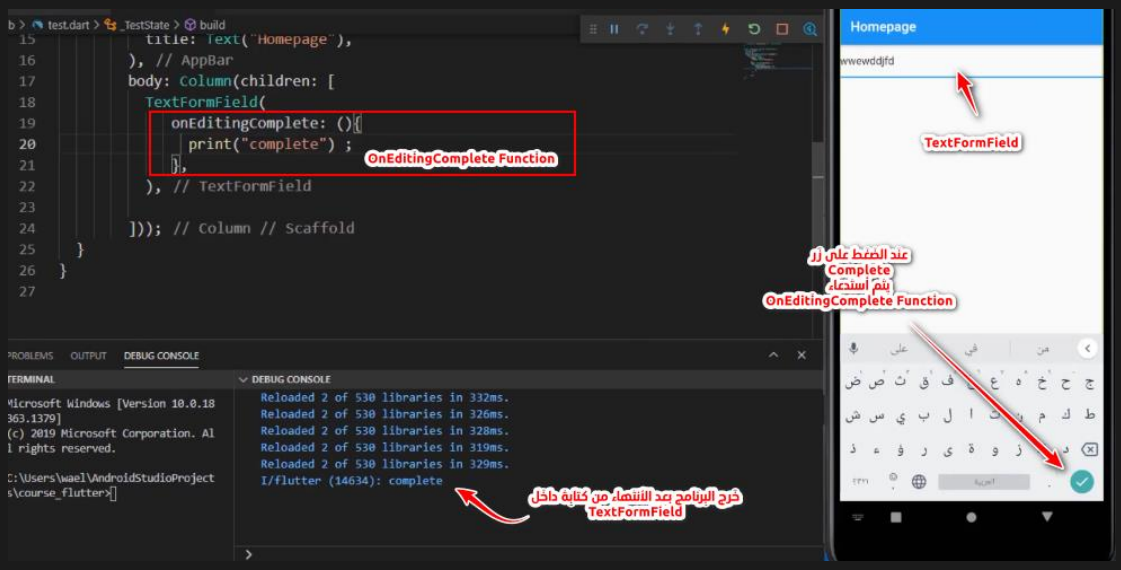
OnTap Function : تعمل عند الضغط داخل الحقل TextFormField ولا تحوي بداخلها على بارامتر كما هو واضح لدينا في المثال في الأسفل :



OnChanged Function : يتم أستدعائها عن كتابة كل حرف داخل الحقل TextFormField وتقبل داخلها بارامتر من نوع string كما هو واضح لدينا في الصورة في الأسفل :



OnEditingComplete Function : يتم أستدعائها بعد الأنتهاء من الكتابة داخل حقل TextFormField وعند الضغط على زر Complete ولا تحوي داخلها أي بارامتر كما هو واضح لدينا في المثال في الأسفل :



TextFormField (Validator)

Validator : هي عبارة عن Function تقبل بداخلها بارامتر من نوع string مهمتها التحقق من المدخلات داخل الحقل TextFormField على حسب الشرط الذي في داخلها.

ولعمل هذا الـ Function في عدة خطوات :

1. نقوم بأنشاء key نسميه formstate على سبيل المثال داخل TextFormField للـوصول إلى الخصائص

ملاحظة : نستطيع أنشاء أكثر من form من خلال أنشاء Form يقبل داخله child وداخله Column وداخل Column يقبل Children نستطيع كتابة أكثر من TextFormField كما هو واضح لدينا في المثال في الأسفل :

```
), // AppBar
body: Form(
  key:formstate ,
  child: Column(children: [
    TextFormField(
      validator: (text){
        if (text.length < 4){
          return "لا يمكن ان يكون النص اقل من اربع احرف او ارقام" ;
        }
        return null ;
      },
    ), // TextFormField
```

Key

Validator

2. نقوم بأنشاء GlobalKey من نوع FormState كما هو واضح لدينا في المثال في الأسفل :

```
class _TestState extends State<Test> {

  @override
  Widget build(BuildContext context) {

    GlobalKey<FormState> formstate = new GlobalKey<FormState>();
```

Global Key

3. ثم نقوم بأنشاء Validator داخل TextFormField من أجل التحقق من المدخلات داخل الحقل ونقوم بكتابة الشرط كما هو واضح لدينا في المثال في الأسفل :

```
TextFormField(
    validator: (text){
      if (text.length < 4){
        return "لا يمكن ان يكون النص اقل من اربع احرف او ارقام" ;
      }
      return null ;
    },
), // TextFormField
```

4. نقوم بأنشاء زر نسميه send ونقوم بداخله بأنشاء متغير من نوع var لنقوم بتخزين بيأنات FormState داخل كما هو واضح لدينا في المثال في الأسفل :

```
RaisedButton(
  onPressed:  send,
  child: Text("Send"),
) // RaisedButton
```

Function داخل button send :

```
send(){
  var formdata = formstate.currentState ;

}
```

5. الأ في حال كان الشرط محقق وقمنا بلضغط على زر send لن يقوم البرنامج بأظهار رسالة خطأ للـمستخدم كما هو واضح لدينا في المثال في الأسفل :

```
return Scaffold(
    appBar: AppBar(
      title: Text("Homepage"),
    ), // AppBar
    body: Form(
      key:formstate ,
      child: Column(children: [
        TextFormField(
          validator: (text){
            if (text.length < 4){
              return "لا يمكن ان يكون النص اقل من اربع احرف او ارقام" ;
            }
            return null ;
          },
        ), // TextFormField
        RaisedButton(
          onPressed:  send,
          child: Text("Send"),
        ) // RaisedButton
      ]), // Column
    ), // Form // Scaffold
}
```

الكود بشكل كامل

الشرط محقق عدد الحروف اكبر من 4

عند الضغط على الزر

6. في حال كأن الشرط غير محقق سيقوم بأظهار رسالة خطأ بلون الأحمر أسفل TextFormField متضمن الرسالة التي قمنا بأنشائها للـمستخدم تعرفه ما الخطأ الذي قام بأرتكابه عند أدخال البيانات داخل الحقل كما هو واضح لدينا في المثال في الأسفل :

```
return Scaffold(
    appBar: AppBar(
      title: Text("Homepage"),
    ), // AppBar
    body: Form(
      key:formstate ,
      child: Column(children: [
        TextFormField(
          validator: (text){
            if (text.length < 4){
              return "لا يمكن ان يكون النص اقل من اربع احرف او ارقام" ;
            }
            return null ;
          },
        ), // TextFormField
        RaisedButton(
          onPressed:  send,
          child: Text("Send"),
        ) // RaisedButton
      ]), // Column
    ), // Form // Scaffold
```

رسالة الخطأ لان الشرط غير محقق وعدد الحروف أقل من 4

Auto validate mode : التحقق من الشرط عند أدخال كل حرف بشكل أتوماتيكي حتى تحقق الشرط بمعنى ليس بضرورة ضغط عل الزر للـتحقق أو حتى ليس من الضرورة وجود زر لتحقق من المدخلات داخل الحقل في حال وجود Auto validate mode داخل TextFormField وتختفي رسالة الخطأ مباشرة عند تحقق الشرط كما هو واضح لدينا في المثال في الأسفل :

```
    }else {
      print("Not Valid") ;
    }
  }

  return Scaffold(
      appBar: AppBar(
        title: Text("Homepage"),
      ), // AppBar
      body: Form(
        autovalidateMode: AutovalidateMode.always,
        key:formstate ,
        child: Column(children: [
          TextFormField(
            validator: (text){
              if (text.length < 4){
                return "لا يمكن ان يكون النص اقل من اربع احرف او ارقام" ;
              }
```

التحقق بشكل اتوماتيكي

ملاحظة : نستطيع أستخدام Auto validate mode داخل كل TextFormField أو أستخدامه بجميع الحقول عند وضعه داخل Form.

TextFormField(OnSaved)

OnSaved : ببساطة هي Function تقبل بداخلها بارامتر من نوع String مهمتها تخزين القيم المدخلات (البيانات التي يقوم بأدخالها المستخدم) داخل المتغيرات.

سنقوم بأنشاء function بعدة خطوات :

أنشاء متغيرين من نوع var أول نسميه username والثاني phone كما هو واضح لدينا في المثال في الأسفل :

```
8
9   class _TestState extends State<Test> {
10
11    var username ;
12    var phone  ;
13
```

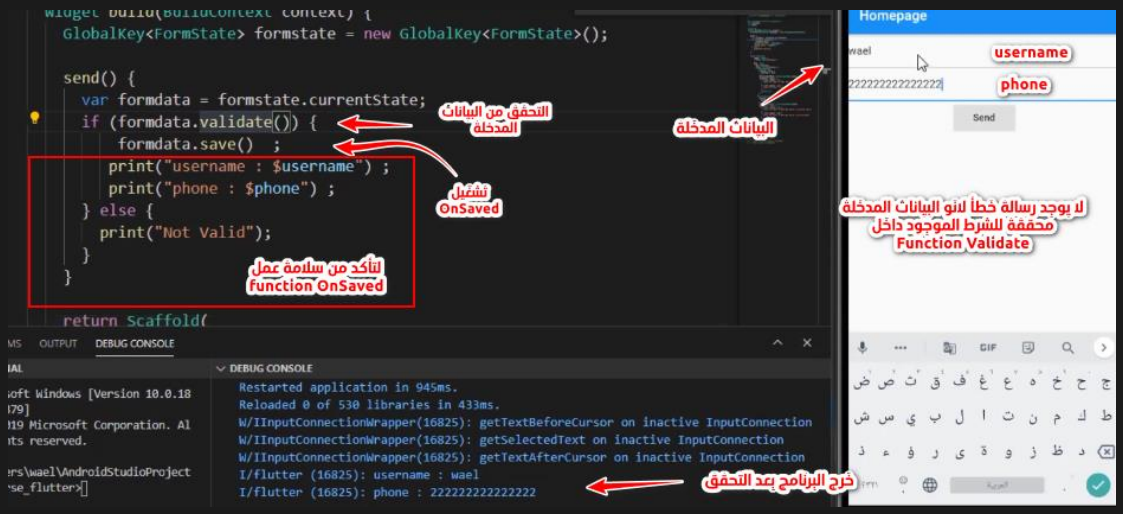
ثم نقوم بأنشاء OnSaved داخل كل TextFormField موجودة لدينا في التطبيق ونقوم بأستدعاء المتغير الذي قمنا بأنشائه وسيقوم بأخذ قيمته من text كما هو واضح لدينا في المثال في الأسفل :

```
TextFormField(
  onSaved: (text) {
    username = text;
  },
```

نفس الأمر بلنسبة لحقل phone كما هو واضح لدينا في المثال في الأسفل :

```
TextFormField(
  onSaved: (text) {
    phone = text;
  },
```

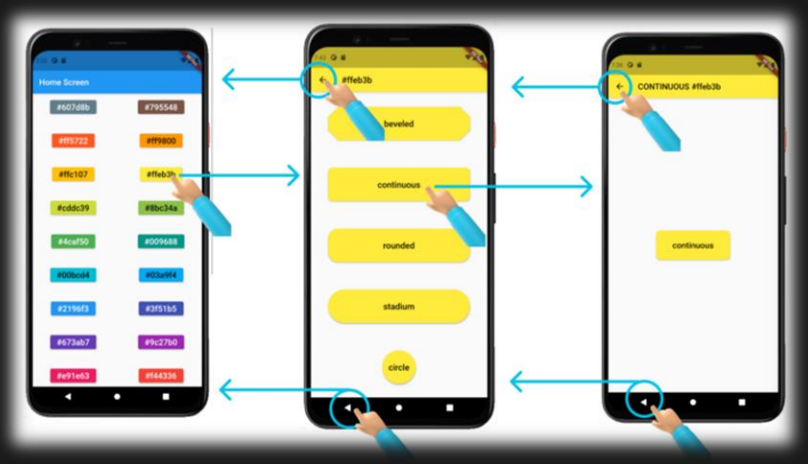
التحقق من الشرط عند ادخال كل حرف بشكل آلي حتى تحقق الشرط لا يوجد رسالة خطأ لأن البيانات المدخلة محققة للـشرط الموجود داخل Function Validate مهمة formdate.Save() هي بمثابة زر تشغيل للـــ function OnSave داخل كل TextFormField بعد التحقق من البيانات المدخلة من قبل function Validate كما هو واضح لدينا في المثال في الأسفل :



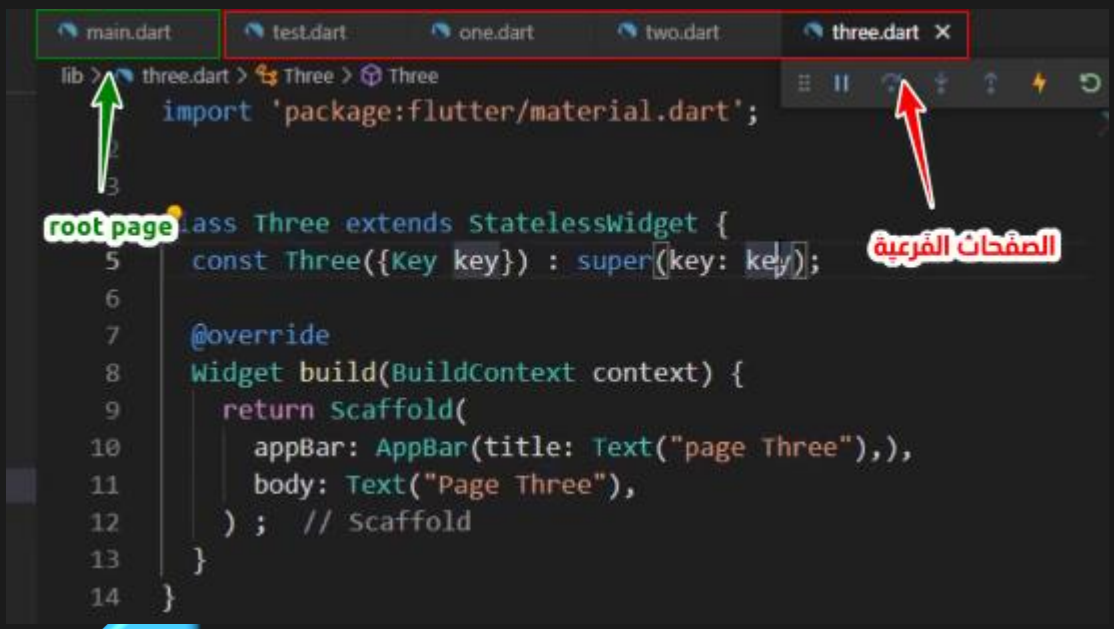
Navigator

(Push And PushNamed And Route)

من خلاله تستطيع التنقل بين الصفحات التطبيق على فرض أن لدينا تطبيق معين يحوي على مجوعة من الصفحات ونريد للـمستخدم التنقل بينها هنا نقوم بأستخدام Navigator ويتم كتابة الكود الخاص بها في Function.



لفهم الموضوع بشكل أكبر سنقوم بأنشاء هذا المثال : ليكن لدينا هذا التطبيق نقوم بأنشاء مجموعة من الصفحات كما هو واضح لدينا في المثال في الأسفل :



نريد الأنتقال من page test إلى page one في صفحة test نقوم بأنشاء button وداخله نستخدم Function OnPressed وداخل OnPressed نقوم بأنشاء Navigator ونستخدم Push وهو نوع من أنوع Navigator يستخدم في حال لم يكن لدينا root نقوم بأنشاء root من خلال MaterialPageRoute لتفعيل الأنتقال كما هو واضح لدينا في المثال في الأسفل :

```
Navigator.of(context).push(MaterialPageRoute(builder: context)
  return One() ;
})) ;
```

page one

Root

بفرض أننا نريد أنشاء root بشكل يدوي نقوم بلتوجه إلى الصفحة الرئيسية في التطبيق page root ومن هنا نقوم بأستخدام خاصية routes وتقبل بداخلها map و الـ key الخاص بها يقبل string حصرا و value تقبل function من نوع Widget وتقبل بداخلها بارامتر من نوع context وتقوم بأرجاع الأسم الصفحة المراد الأنتقال اليها كما هو واضح لدينا في المثال في الأسفل :

```
9
10  class MyApp extends StatelessWidget {
11    @override
12    Widget build(BuildContext context) {
13      return MaterialApp(
14        debugShowCheckedModeBanner: false ,
15        home: Test(),
16        routes: {
17          "one" : (context) => One()
18        },
19      ); // MaterialApp
20    }
21  }
22
```

Root

وبعد الأنتهاء من أنشاء root في الصفحة الرئيسية نقوم بأنشاء Navigator ولكن من نوع PushNamed لأننا قمنا بأنشاء root بشكل يدوي كما هو واضح لدينا في المثال في الأسفل :

```
return Scaffold(
    appBar: AppBar(
      title: Text("Test Page"),
    ), // AppBar
    body: Column(
      children: [
        Center(
          child: RaisedButton(
            onPressed: () {
              Navigator.of(context).pushNamed("one") ;
            },
            child: Text("Go To Page One"),
          ) // RaisedButton // Center
      ],
    ); // Column // Scaffold
  }
}
```

Navigator

ملاحظة توضيحية : ما الفرق بين push و pushNamed ..؟؟

- Push : نستخدمها عندما لا يكون لدينا Root.
- PushNamed : نستخدمها عندما يكون لدينا root حيث نقوم بأنشاء root بشكل يدوي في الصفحة الرئيسية للـتطبيق Page root.

Navigator

(Pop And CanPop)

Pop : مهمتها العودة إلى الصفحة السابقة بأختصار (أرجاع خطوة واحدة إلى الخلف) وهذا مثال توضيحي عن Navigator Pob كما هو واضح لدينا في المثال في الأسفل :

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

class One extends StatelessWidget {
  const One({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("page one"),),
      body: Column(children: [
        Text("Page One"),
        RaisedButton(onPressed: (){
          Navigator.of(context).pop() ;
        },
        child: Text("Back"),
        ) // RaisedButton
      ],) // Column
    ) ;  // Scaffold
```

Navigator Pop

page one

Page One

Back

CanPop : تحقق من أمكانية الرجوع إلى الخلف وترجع قيمة أما true أو false ولكن كيف يمكن ذلك وما فائدة التحقق في الأساس على فرض أننا في الصفحة الرئيسة في التطبيق وهي الصفحة الأولى أي أننا لا نستطيع الرجوع إلى الخلف ونحن في الأساس في الصفحة الأساسية وعلى العكس لو كنا في الصفحة الثانية الفرعية من التطبيق وقمنا بأستخدام Can Pop نستطيع الرجوع إلى الخلف وتكون القيمة المرجعة من Can Pop هي true ونستطيع التحقق منها من خلال terminal كما هو واضح لدينا في المثال في الأسفل :

```
main.dart   test.dart   one.dart   two.dart   three.dart
lib > test.dart > Test > createState
                onPressed: () {
                  //  Navigator.of(context).pushNamed("one") ;
              print(Navigator.of(context).canPop() ) ;
                },
                child: Text("Go To Page One"),
              )) // RaisedButton // Center
            ],
          )); // Column // Scaffold
        }
      }

PROBLEMS 1   OUTPUT   DEBUG CONSOLE
TERMINAL
Microsoft Windows [Version 10.0.18363.1379]
(c) 2019 Microsoft Corporation. All rights reser
DEBUG CONSOLE
Restarted application in 946ms.
I/flutter (18475): false
```

root page

الصفحة الاولى في التطبيق

Can Pop

خرج لبرنامج وأرجاع قيمة false

Test Page

Go To Page One

عند الضغط

وعلى فرض أننا قمنا باستخدام Pop العادية داخل الصفحة الرئيسية وقمنا بلضغط على زر الرجوع ما هو الحدث المتوقع أن يحدث ..؟؟

بلطبع سيتم الخروج من التطبيق بشكل كامل كما هو واضح لدينا في المثال في الأسفل :

```
        Navigator.of(context).pop() ;

      },
      child: Text("Go To Page One"),
    )) // RaisedButton // Center
  ],
)); // Column // Scaffold
```

والحالة الأصح لأستخدام Can Pop هو وضعها ضمن دالة شرطية if في حال قامت بأرجاع قيمة true سيتحقق الشرط وسيقوم البرنامج في الرجوع خطوة إلى الخلف في حال قامت بأرجاع قيمة false لم يتحقق الشرط ولن يقوم البرنامج بلرجوع خطوة إلى الخلف في حال كنا في الصفحة الأولى للـتطبيق كما هو واضح لدينا في المثال في الأسفل :

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

Navigator

(pushReplacementNamed And pushReplacement)

ببساطة وأختصار وما هو الفرق ..!!

pushReplacementNamed : يقوم بلأنتقال إلى صفحة معينة ولكن مع عم أمكأنية الرجوع إلى الصفحة السابقة أي أنه يقوم بحذف مسار الأنتقال ونستخدم pushReplacementNamed عندما نكون قد بأنشاء root بشكل يدوي.

pushReplacement : يقوم بلأنتقال إلى صفحة معينة ولكن مع عم أمكأنية الرجوع إلى الصفحة السابقة أي أنه يقوم بحذف مسار الأنتقال ببساطة ونستخدم pushReplacement في حال لم نقوم بأنشاء root.

ملخص Navigator ببساطة..!!

ما الفرق بين الأنتقال Psuh والعودة Pop بين الصفحات ..؟؟

الأنتقال Psuh : يقوم بلأنتقال بين الصفحات بشكل عشوائي وعلى حسب الصفحة بلأضافةلأنه يقوم (بمسح) جميع المدخلات عند الأنتقال من صفحة إلى أخرى ببساطة.

أما العودة Pop : يقوم بلرجوع خطوة إلى الصفحة السابقة وبشكل تسلسلي بلأضافةلأنه يقوم (بلحفاظ) على جميع المدخلات عند الرجوع خطوة إلى الوراء على سبيل المثال لو كأن لدينا مجموعة من صفحات تحوي على حقول أدخال وأراد المستخدم أن يرجع إلى الخلف أو إلى الصفحة السابقة لتعديل بيانات حقل من حقول الأدخال سيرجع إلى الخلف وسيجد جميع المدخلات حيث يستطيع التعديل عليها على عكس Push الأنتقال سيقوم بحذف جميع مدخلات المستخدم.

Slider

هو عبارة عن شريط تستطيع من خلاله تحديد قيمة معينة عن طريق السحب ويكون له مجال محدد يأخذ قيمة Min و Max من نوع Double.

أهم Properties للـــ Slider :

Properties	Widget
Min القيمة الدنيا للـــ slider ويقبل قيمة من نوع double.	Slider
Max القيمة العظمى للـــ slider و يقبل قيمة من نوع double.	
activeColor لون slider.	
inActiveColor لون مسار slider.	
Value القيمة الابتدائية للـــ slider.	
OnChanged هي function تقبل بارامتر من نوع double.	

مثال عملي عن Slider كما هو واضح لدينا في المثال في الأسفل :

```
test.dart > _TestState > build
13    Widget build(BuildContext context) {
14      return Scaffold(
15        appBar: AppBar(
16          title: Text("Test Page"),
17        ), // AppBar
18        body: Column(
19          children: [
20            Slider(
21              min: 0.0,
22              max: 200.0,
23              activeColor: Colors.red,
24              inactiveColor: Colors.black,
25              value: _valslider,
26              onChanged: (val) {
27                setState(() {
28              _valslider =  val ;
29                  print(_valslider);
30                });
31              }) // Slider
32          ],
33        ); // Column // Scaffold
```

Slider

Scroll Controller

التنقل بين عناصر الصفحة بأستخدام السحب على الشاشة وعلى حسب الاتجاه من بشكل عامودي أو أفقي.

سنتعرف اليوم على بعض الخصائص الأضافية للـــ Scroll أهمها Controller

من خلال هذا المثال العملي في البداية نقوم بعمل listView وبداخلها list generate وبداخل listGenerate وبداخلها container ونضيف بعض الخصائص فيه كما هو واضح لدينا في المثال في الأسفل :

```
class _TestState extends State<Test> {


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Test Page"),
      ), // AppBar
      body: ListView(
        children: [
          ...List.generate(
              20,
              (index) => Container(
                  margin: EdgeInsets.all(10),
                  child: Text("Container $index "),
                  color: index.isEven ? Colors.green : Colors.red,
                  height: 100,
                  width: double.infinity)) // Container // List.generate
        ],
      )); // ListView // Scaffold
```

ثم نقوم بأنشاء Controller داخل ListView نسميه على سبيل المثال sc ونقوم بأنشاء initState و ننشأ Controller و addListener هو متتبع حالة وقيمة scroll كما هو واضح لدينا في المثال في الأسفل :

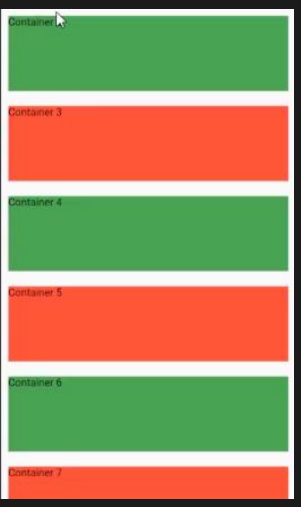
```
11    ScrollController sc ;
12
13    @override
14   void initState() {
15      sc = new ScrollController() ;
16      sc.addListener(() {
17          print(sc.offset) ;
18      }) ;
19      super.initState();
20   }
21
```

Controller

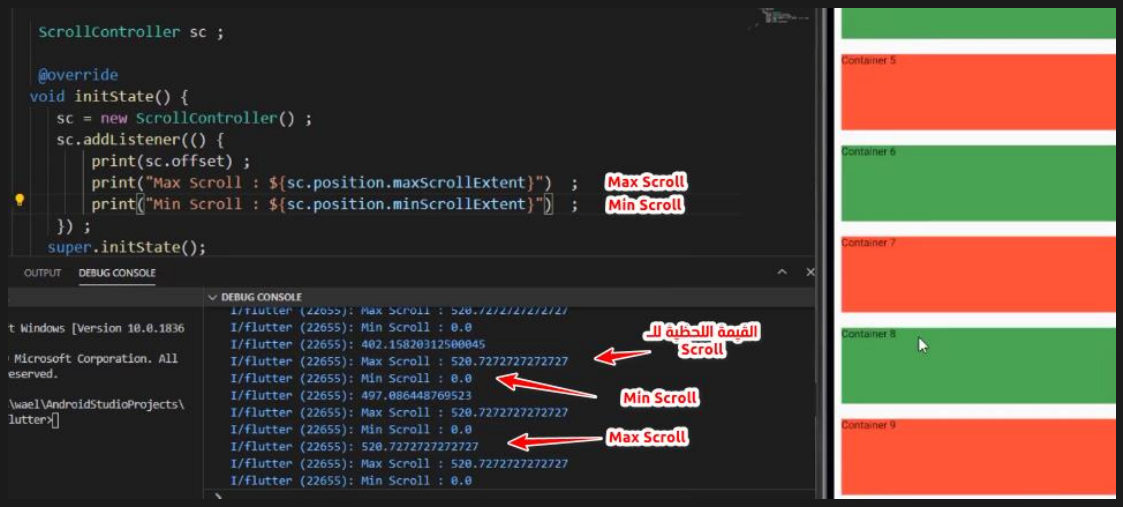
تحرك للأعلى وللأسفل

```
DEBUG CONSOLE
I/flutter (22655): 203.27170632182278
I/flutter (22655): 204.72523082386368
I/flutter (22655): 206.9178355823864
I/flutter (22655): 208.0018199573864
I/flutter (22655): 209.4553444602273
I/flutter (22655): 210.5393288352273
I/flutter (22655): 211.64794921875003
I/flutter (22655): 212.73193359375003
I/flutter (22655): 214.18545809659093
I/flutter (22655): 215.26944247159093
```

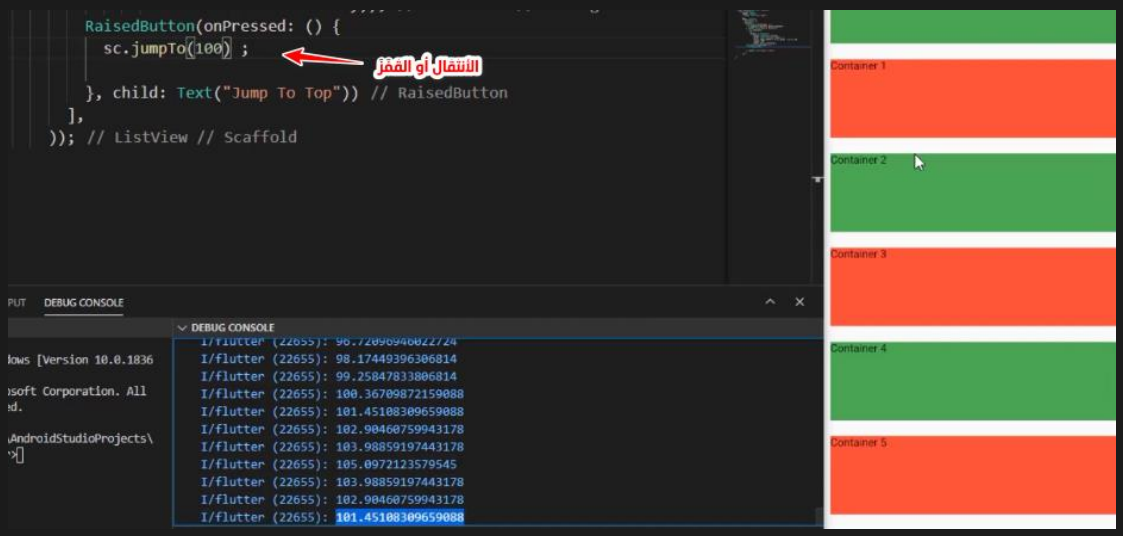
قيمة الـ Scroll عند التحريك



نستطيع طباعة اعلى قيمة للـــ Scroll وأدنى قيمة للـــ Scroll كما هو واضح لدينا في المثال في الأسفل :



أو أنشاء button واستخدام خاصية JumpTo وهي خاصية للأنتقال أو القفز من خلال Scroll وتقبل قيمة من نوع double كما هو واضح لدينا في المثال في الأسفل :

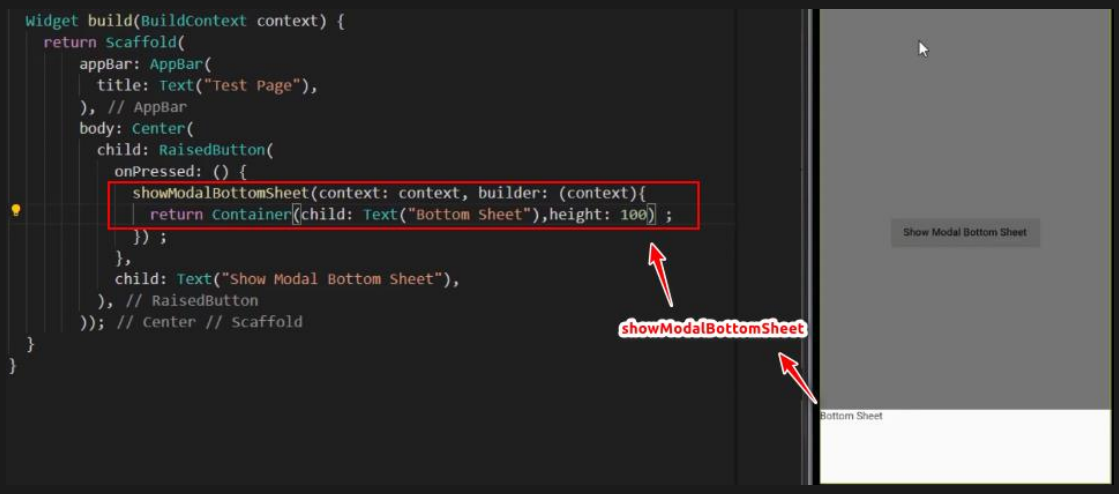


Show Modal Bottom Sheet

هي عبارة عن Widget تشبه AlertDialog بشكل جدا كبيرة من حيث البنية البرمجية ولكن تختلف بطريقة الاستخدام فقط والخصائص الأساسية هي :

Context يقبل متغير من نوع build context وهو من Properties الأساسية في AlertDialog.
Builder هو بارامتر من نوع : Function (build context) ويقوم بأرجاع Widget من نوع : AlertDialog وهو أيضا من Properties الأساسية في AlertDialog.

مثال عملي عن showModalBottomSheet كما هو واضح لدينا في المثال في الأسفل :



Search Delegate

هو عبارة عن مربع بحث تستطيع من خلاله البحث عن عنصر معين في التطبيق وسنقوم ببناء الـ class الخاص وشرحه بلتفصيل.

بدأية نقوم بأنشاء زر داخل actions في appBar وهو عبارة عن icon ثم نقوم بأنشاء class نسميه Data Search (اسم الكلاس اختياري) ونجعله extends من Search Delegate كما هو واضح لدينا في المثال في الأسفل :

```
appBar: AppBar(
  actions: [
    IconButton(icon: Icon(Icons.search), onPressed: (){

    }) // IconButton
  ],
  title: Text("Test Page"),
), // AppBar
body: Center(
  child:  Text("wael")
)); // Center // Scaffold
}
}

class DataSearch extends SearchDelegate {
```

داخل onPressed نقوم بأستدعاء SearchDelegate

name class : DataSearch

Class SearchDelegats

ثم نقوم بلوقوف فوق اسم class تظهر لنا نافذة صغيرة نقوم بلضغط على Quick Fix كما هو واضح لدينا في المثال في الأسفل :

```
class DataSearch extends SearchDelegate<dynamic>
package:course_flutter/test.dart
Missing concrete implementations of
'SearchDelegate.buildActions', 'SearchDelegate.buildLeading',
'SearchDelegate.buildResults', and
'SearchDelegate.buildSuggestions'.
Try implementing the missing methods, or make the class
abstract. dart(non_abstract_class_inherits_abstract_member)
Peek Problem (Alt+F8)  Quick Fix... (Ctrl+.)
class DataSearch extends SearchDelegate {

class DataSearch exte
Create 4 missing override(s)
Make class 'DataSearch' abstract
Create 'noSuchMethod' method
}
```

ثم نقوم بلضغط على Creat 4 missing override(s) ليقوم بعدها البرنامج بأستدعاء 4 ميثود مطلوبة كما هو واضح لدينا في المثال في الأسفل :

```
class DataSearch extends SearchDelegate {
  @override
  List<Widget> buildActions(BuildContext context) {
    // TODO: implement buildActions
    throw UnimplementedError();
  }

  @override
  Widget buildLeading(BuildContext context) {
    // TODO: implement buildLeading
    throw UnimplementedError();
  }

  @override
  Widget buildResults(BuildContext context) {
    // TODO: implement buildResults
    throw UnimplementedError();
  }

  @override
  Widget buildSuggestions(BuildContext context) {
  // TODO: implement buildSuggestions
  throw UnimplementedError();
```

1. buildActions تشبه إلى حد كبير Action في AppBar وترجع قيمة من نوع Widget وغالبا تكون button icon.
2. buildLeading تشبه إلى حد كبير Leading في AppBar وترجع قيمة من نوع Widget وغالبا تكون button icon أيضا.
3. BuildResults وهي نتيجة البحث.
4. buildSuggestions هي محتوى البحث في search Delegate وترجع أيضا قيمة من نوع Widget.

كما هو اضح لدينا في المثال في الأسفل :

```
class DataSearch extends SearchDelegate {
  @override
  List<Widget> buildActions(BuildContext context) {
      return [
          IconButton(icon: Icon(Icons.close), onPressed: (){})
      ] ;
  }

  @override
  Widget buildLeading(BuildContext context) {

    return  IconButton(icon: Icon(Icons.arrow_back), onPressed: (){}) ;

  }

  @override
  Widget buildResults(BuildContext context) {
          return null ;
  }

  @override
  Widget buildSuggestions(BuildContext context) {

    return Center(child: Text("محتوى البحث")) ;
```

تشبه Action في appBar (1)

تشبه Leading في appBar (2)

(3)

محتوى البحث (4)

بعد القيام بأستدعاء جميع ميثود والأنتهاء من كتابة الميثود المطلوبة في search delegate نقوم بتوجه إلى iconButon في actions في appBar نقوم بأستدعاء search delegate داخل onPressed وتقبل show Search وتقبل بارامترين context ويقبل context و delegate وتقبل اسم class الخاص بلـ search delegate الذي قمنا بأنشائه في الأسفل :

```
appBar: AppBar(
  actions: [
    IconButton(icon: Icon(Icons.search), onPressed: (){
            showSearch(context: context, delegate: DataSearch()) ;
    }) // IconButton
  ],
```

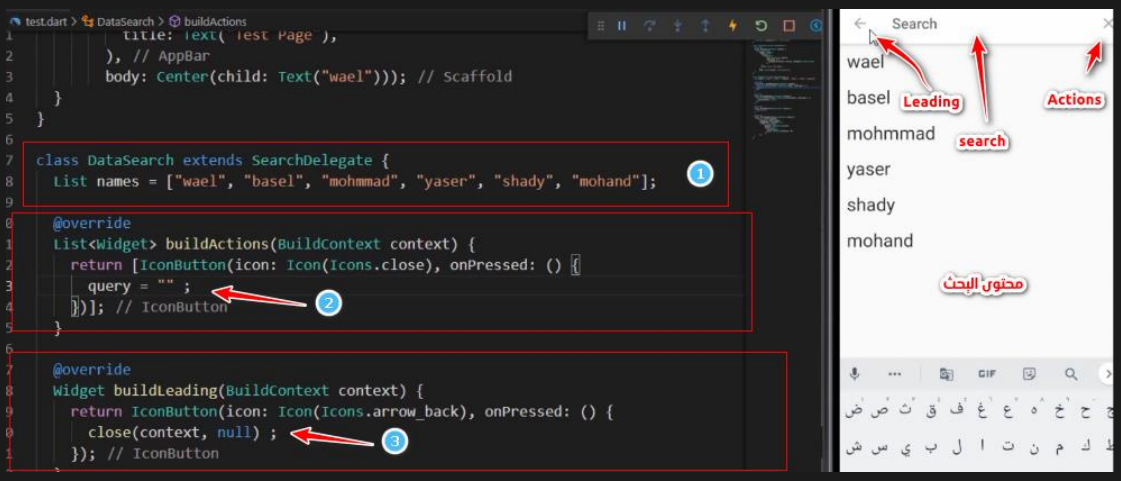
ملاحظة : بعد القيام بأنشاء class search delegate تستطيع استدعائه داخل أي حدث تريديه على سبيل المثال onTap - onLongTap .....

1. لنقوم بشرح هذا المثال العملي على search delegate قمنا بأنشاء list اسمها names تحوي على عناصر من نوع list كما هو واضح في الرقم 1.

2. Query في search delegate هو متغير معرف مسبقا مهمته تخزين القيمة الموجودة داخل مربع البحث في buildActions داخل iconButton في onPressed قمنا بتعريف الـ query وجعل قيمتها فارغة وذلك عند الضغط على زر close يقوم بمسح محتوى مربع البحث كما هو واضح في الرقم 2.

3. وفي bulidLeading قمنا بأنشاء زر للـرجوع إلى الخلف وداخل onPressed قمنا بأستدعاء close كما هو واضح لدينا في الرقم 3.

ملاحظة : نستطيع أستخدام Navigator Pop ولكن نحن نريد أستخدام الطريقة الأصح في برنامجنا.



داخل buildsuggestions كأول خطوة قمنا بأنشاء listView builder من اجل استدعاء عناصر list التي قمنا بأنشائها في الأعلى كما هو واضح في الرقم 1.

- وقمنا بأنشاء list أخرى من اجل فلترة العناصر والهدف من البرنامج عندما يقوم المستخدم بكتابة داخل مربع البحث يقوم البرنامج بفتلرة العناصر داخل محتوى البحث ويعرضها للـمستخدم قمنا بتسمية هذه list بأسم filterNames ثم قمنا بأستدعاء دالة where وهي دالة في لغة دارت تقوم بفلترة العناصر وقمنا بأرجاع قيمة query داخل element للـعناصر الذي تبدأ بحرف معين من خلال دالة startWith (على حسب الحرف الذي يقوم المستخدم بأدخاله) كما هو واضح في الرقم 2.

- داخل itemCount في قمنا بأستخدام دالة شرطية مختصرة في listview في حال كأنت قيمة query فراغة يقوم البرنامج بعرض جميع محتويات list names والا يقوم بعرض عناصر list filter names.

ومن أجل عرض المتغيرات على الشاشة قمنا بأنشاء container داخل listview وقمنا بتطبيق نفس الدالة الشرطية في itemCount كما هو واضح لدينا في المثال في الأسفل:

```
ib > test.dart > DataSearch
45    Widget buildResults(BuildContext context) {
46      return null;
47    }
48
49    @override
50    Widget buildSuggestions(BuildContext context) {
51
52      List filternames  = names.where((element) =>   element.startsWith(query) ).toL
53
54      return ListView.builder(
55          itemCount: query == "" ? names.length : filternames.length,
56          itemBuilder: (context, i) {
57            return Container(
58                padding: EdgeInsets.all(10),
59                child: query == "" ? Text(
60                  "${names[i]}",
61                  style: TextStyle(fontSize: 25),
62                ) :  // Text
63                Text(
64                  "${filternames[i]}",
65                  style: TextStyle(fontSize: 25),
66                ) // Text
67                ); // Container
68          }); // ListView.builder
69    }
70 }
```

else