🤫

Summary Level(1)🔥

تحفيز: أول كود تكتبه هو بداية رحلتك

✅ `MaterialApp`

- الأساس اللي بيبدأ بيه أي تطبيق Flutter
- بيحدد:
  - الشاشة الرئيسية (`home`)
  - اسم التطبيق (`title`)
  - الستايل العام (`theme`)
  - التنقل بين الصفحات (`routes`)

🪣 `Scaffold`

- بيوفر الهيكل العام للتطبيق:
  - `AppBar` (الشريط العلوي)
  - `Body` (المحتوى)
  - `Drawer` , `BottomNavigationBar` لو حبيت

- أساسي في أي شاشة فيها واجهة كاملة

🔵 AppBar

- الشريط اللي بيكون في أعلى التطبيق
- تقدر تحط فيه عنوان، أيقونات، زرار رجوع... إلخ

📦 Container

- ويدجت مرنة جدًا تقدر:
  - تضبط حجمها (`height`, `width`)
  - تغير لون الخلفية (`color`)
  - تحط هوامش (`margin`) أو مسافات داخلية (`padding`)
  - تزود ديكورات زي حدود وظلال (`decoration`)

🔲 Padding و Margin

- `Padding`: مسافة **داخل** العنصر (جوه الكونتينر مثلًا)
- `Margin`: مسافة **بين** العنصر والعناصر اللي حواليه

🎨 Decoration

- `Container` تقدر تعمل ديزاين جميل للـ:
  - خلفية ملونة أو متدرجة
  - زوايا دائرية (`borderRadius`)
  - ظل (`boxShadow`)
  - حدود (`border`)

🎯 نصائح سريعة:

- دايمًا خليك منظم، وحاول تلف كل شاشة بـ `MaterialApp` ➜ `Scaffold` ➜ جواها الباقي.
- `Container` هو البطل في تنسيق العناصر.

😉 **تحفيز:** الكمبايلر ما بيعضش، خليه يعلمك

✅ Row و Column

- `Row`:
  - `children`: قائمة العناصر اللي هتكون جنب بعض أفقيًا.
  - `mainAxisAlignment`: تحكم في المحاذاة الأفقيّة للعناصر (مثلًا: `center`, `start`, `end`).
  - `crossAxisAlignment`: تحكم في المحاذاة العموديّة للعناصر.
- `Column`:
  - `children`: قائمة العناصر اللي هتكون فوق بعض عموديًا.
  - `mainAxisAlignment`: تحكم في المحاذاة العموديّة للعناصر.
  - `crossAxisAlignment`: تحكم في المحاذاة الأفقيّة للعناصر.

🪣 Expanded و Flexible

- `Expanded`:
  - `child`: الويجت اللي هيأخذ المساحة الفاضية.
- `Flexible`:
  - `flex`: بتحدد نسبة المساحة اللي هياخدها الويجت بالنسبة للعناصر الأخرى. القيمة `1` الافتراضية هي.

🔲 SizedBox

- `height`: بتحدد الارتفاع.
- `width`: بتحدد العرض.

📦 Spacer

- `flex`: `1` بتحدد نسبة المساحة اللي هيأخذها الفراغ بين العناصر. القيمة الافتراضية هي.

🎨 Align و Center

- `Align`:

- `alignment` : بتحدد مكان العنصر داخل الحاوية (مثلًا: `Alignment.center` , `Alignment.topLeft`).
- `child` : الويجت اللي عايز تحطه جوه الـ `Align` .

- `Center` :
  - `child` : الويجت اللي هيتوسّط في الحاوية.

🔄 `Stack` , `Positioned` , و `Wrap`

- `Stack` :
  - `children` : مجموعة العناصر اللي هتتراكم فوق بعضها.
- `Positioned` :
  - `top` , `left` , `right` , `bottom` : بتحدد مكان العنصر داخل الـ `Stack` .
  - `child` : الويجت اللي عايز تضيفه.
- `Wrap` :
  - `children` : مجموعة العناصر اللي هتتوزع في صفوف وأعمدة.
  - `spacing` : المسافة بين العناصر.
  - `runSpacing` : المسافة بين الصفوف.

🎯 نصائح سريعة:

- `Row` و `Column` هما الأساس لترتيب العناصر أفقيًا وعموديًا.
- `Expanded` و `Flexible` بيساعدوك تتحكم في توزيع المساحة.
- `SizedBox` و `Spacer` مناسبين لو حبيت تضيف فواصل بين العناصر.
- `Align` و `Center` مهمين عشان تتحكم في مكان العناصر داخل الحاوية.

تحفيز: كل سطر كود بيعلمك ازاي تفكر بشكل مختلف

🔘 `ElevatedButton` , `TextButton` , `OutlinedButton` , `IconButton` , `FloatingActionButton` , `MaterialButton` , `GestureDetector`

✅ `ElevatedButton`

- `onPressed` : الفانكشن اللي بتتنفذ لما المستخدم يضغط على الزر.

- `child` : الويجت اللي هيتحط جوه الزر (عادةً بيكون نص).
- `style` : بتحدد شكل الزر (مثلًا: اللون، الحدود، الظلال).

📝 TextButton

- `onPressed` : الفانكشن اللي بتتنفذ عند الضغط.
- `child` : الويجت اللي هيتحط جوه الزر.
- `style` : لتخصيص شكل الزر (مثلًا: اللون، نوع الخط، إلخ).

🔲 OutlinedButton

- `onPressed` : الفانكشن اللي بتتنفذ عند الضغط.
- `child` : الويجت اللي هيتحط جوه الزر.
- `style` : بتحدد شكل الزر (مثلًا: اللون، الحدود المرسومة حول الزر).

⚡ IconButton

- `onPressed` : الفانكشن اللي بتتنفذ عند الضغط.
- `icon` : الأيقونة اللي هتظهر في الزر.
- `tooltip` : النص اللي يظهر عندما يمر المستخدم على الأيقونة.

✨ FloatingActionButton

- `onPressed` : الفانكشن اللي بتتنفذ عند الضغط.
- `child` : الويجت اللي هيتحط جوه الزر (عادةً أيقونة).
- `tooltip` : النص اللي يظهر عند مرور المستخدم فوق الزر.
- `backgroundColor` : بتحدد لون خلفية الزر.

🔘 MaterialButton

- `onPressed` : الفانكشن اللي بتتنفذ عند الضغط.
- `child` : الويجت اللي هيتحط جوه الزر.

- `color` : اللون الأساسي للزر.
- `elevation` : قيمة الظل تحت الزر.

🎯 GestureDetector

- `onTap` : الفانكشن اللي بتتنفذ عند الضغط على العنصر.
- `onLongPress` : الفانكشن اللي بتتنفذ عند الضغط الطويل.
- `onPanUpdate` : الفانكشن اللي بتتنفذ أثناء السحب.
- `child` : الويجت اللي بيتم عليه الكشف عن الحركة.

🎯 **نصائح سريعة:**

- `ElevatedButton` و `TextButton` و `OutlinedButton` هما الأزرار الأساسية في Flutter.
- `FloatingActionButton` مهم عشان تضع أزرار عائمة في الواجهة.
- `GestureDetector` مفيد جدًا لما تحتاج تتفاعل مع الحركات زي الضغط والسحب.

تحفيز: النجاح مش في الكود نفسه، النجاح في استمراريتك

📝 Text , TextStyle , RichText , TextSpan

✅ Text

- `data` : النص اللي هيتعرض في الواجهة.
- `style` : لتخصيص الخط (مثلًا: اللون، الحجم، النوع).
- `textAlign` : لتحديد محاذاة النص (يسار، يمين، وسط).
- `overflow` : يحدد إذا كان النص هيتقطع ولا هيظهر كله.
- `maxLines` : عدد الأسطر اللي النص يقدر يظهر فيها.

🎨 TextStyle

- `color` : لتحديد لون النص.
- `fontSize` : لتحديد حجم الخط.
- `fontWeight` : لتحديد سماكة الخط (مثلًا: عادي أو ثقيل).

- `fontStyle` : لتحديد إذا كان النص مائل.
- `letterSpacing` : المسافة بين الحروف.

📜 RichText

- بيتم استخدامه لعرض نصوص مركبة أو مزخرفة.
- يتم دمج أكتر من `TextSpan` في عنصر واحد.
- `text` : الويجت الأساسي اللي بيحتوي على الـ `TextSpan` .
- `textAlign` : لتحديد محاذاة النص.

📝 TextSpan

- `text` : النص نفسه.
- `style` : ستايل النص (مثلًا: اللون، الحجم، نوع الخط).
- `children` : مجموعة من النصوص أو العناصر الفرعية زي `TextSpan` .

🎯 نصائح سريعة:

- لو عندك نص واحد عادي، استخدم `Text` .
- لو عايز تخصيص أكثر وتنسيق النصوص بطريقة معقدة، استخدم `RichText` و `TextSpan` .
- `TextStyle` هي الأداة اللي من خلالها تقدر تغير خصائص النص زي اللون والحجم.

تحفيز: خلي كل يوم فرصة لتعلم شيء جديد

⌨️ TextField , TextFormField , Form

✅ TextField

- ويدجت إدخال نصوص أساسي، بيسمح للمستخدم بكتابة نص.
- `controller` : لتحديد قيمة النص في الـ `TextField` والتحكم فيه.
- `keyboardType` : لتحديد نوع لوحة المفاتيح (مثلًا: لو حابب يكتب أرقام أو نص).
- `obscureText` : إذا كنت عايز تخفي النص (مثال: كلمة المرور).
- `decoration` : لتخصيص الشكل زي إضافة `labelText` , `hintText` , `border` , وغيره.

📦 TextFormField

- مشابه للـ `TextField` لكن مع إضافة **التحقق من الصحة** (validation).
- `validator`: دالة لتحديد الشروط اللي لازم النص يتبعها (مثال: لازم يكون البريد الإلكتروني صحيح).
- `onSaved`: لتخزين البيانات بعد التحقق.

📝 Form

- حاوية بتجمع مجموعة من الـ `TextFormField` أو أي حقل إدخال تاني.
- `key`: `GlobalKey<FormState>` لتمكين التحقق من صحة البيانات.
- `child`: يحتوي على الحقول اللي عايز تستخدمها داخل الـ `Form`.

🎯 نصائح سريعة:

- لو محتاج حقل إدخال نص بدون تحقق، استخدم `TextField`.
- لو هتحتاج تحقق من النصوص المدخلة، استخدم `TextFormField` مع `Form`.
- لازم تستخدم `Form` عشان تنظم التحقق من الصحة وتخزن البيانات بشكل مرتب.

تحفيز: ما فيش حاجة صعبة، كل حاجة تتعلمها خطوة بخطوة

🖼️ Image.asset , Image.network , Icon , CircleAvatar

✅ Image.asset

- بيعرض صورة موجودة في مجلد المشروع (محلية).
- `assetName`: اسم الصورة داخل المجلد (مثال: `assets/images/logo.png`).
- `fit`: كيفية ملء الصورة داخل الـ `Image`. ممكن تستخدم `BoxFit.cover` عشان الصورة تملأ المساحة بشكل مناسب.
- `width` و `height`: تحدد أبعاد الصورة.

📦 Image.network

- لعرض صورة من الإنترنت باستخدام رابط URL.

- `url`: رابط الصورة على الإنترنت.
- `fit`: نفس الخاصية لملء الصورة.
- `loadingBuilder`: لو حابب تعرض صورة مؤقتة أثناء تحميل الصورة الأصلية.

🎨 Icon

- لعرض أيقونة من مجموعة الأيقونات الجاهزة في Flutter.
- `icon`: الأيقونة اللي عايز تستخدمها (مثلًا: `Icons.home`).
- `size`: حجم الأيقونة.
- `color`: لون الأيقونة.

🔵 CircleAvatar

- لعرض صورة أو حرف في شكل دائرة.
- `backgroundImage`: لتحديد صورة الخلفية.
- `child`: لو حابب تحط عنصر نص داخل الدائرة بدل الصورة.
- `radius`: لتحديد حجم الدائرة.

🎯 **نصائح سريعة:**

- لو الصورة من جهازك، استخدم `Image.asset`.
- لو الصورة من الإنترنت، استخدم `Image.network`.
- `Icon` تستخدمه لما تحتاج تعرض أيقونات معروفة مثل أزرار أو رموز.
- `CircleAvatar` ممتاز لعرض صور شخصية أو أول حرف من اسم.

تحفيز: لو قدرت تحل مشكلة في الكود، تقدر تحل أي مشكلة في الحياة

📜 ListView , ListView.builder , GridView

✅ ListView

- الويجت الأساسي لعرض قائمة من العناصر بشكل عمودي.
- بيتم تحريك العناصر بسهولة باستخدام **Scroll**.

- `children` : مجموعة العناصر اللي هتظهر في القائمة.
- `scrollDirection` : ممكن تخليه `Axis.horizontal` لو حابب عرض العناصر بشكل أفقي.

📦 ListView.builder

- بيعرض العناصر في قائمة لكن بشكل ديناميكي، بحيث يخلق العناصر فقط لما يتم تمرير الـ List.
- `itemCount` : عدد العناصر اللي هتظهر في القائمة.
- `itemBuilder` : الفانكشن اللي بتحدد بيها شكل العنصر.

🎨 GridView

- بدل من قائمة (Grid) بيعرض العناصر في شبكة.
- `crossAxisCount` : عدد الأعمدة في الشبكة.
- `children` : العناصر اللي هتظهر.
- `scrollDirection` : نفس الخاصية لتمرير العناصر بشكل أفقي أو عمودي.

🎯 **نصائح سريعة**:

- لو عندك قائمة ثابتة من العناصر استخدم `ListView`.
- لو القائمة ديناميكية أو العناصر بتتغير، استخدم `ListView.builder`.
- لو عايز تعرض العناصر في شبكة، استخدم `GridView`.

تحفيز: من الكود البسيط تبدأ الفكرة الكبيرة

💳 الدرس 7: بطاقات وواجهات جذابة

✅ Card

- الويجت اللي بيستخدم لعمل بطاقات بتحتوي على محتوى.
- بتكون مفيدة لعرض المعلومات بطريقة منظمة وجميلة.
- `child` : العنصر اللي هيظهر جوا الكارد، زي نص أو صورة.
- `elevation` : التحكم في الظل تحت الكارد.

- `shape` : تعديل الشكل، زي زوايا دائرية.

✅ ListTile

- ويدجت مصممة لعرض العناصر في شكل قائمة.
- بتنظم العناصر بشكل عمودي زي العناوين والأيقونات.
- `title` : النص الرئيسي.
- `subtitle` : النص الفرعي.
- `leading` : الأيقونة أو الصورة اللي هتظهر قبل النص.
- `trailing` : الأيقونة أو الزرار اللي هتظهر بعد النص.

✅ Divider

- ويدجت بتستخدم لفصل العناصر في القوائم أو بين الأقسام.
- بتضيف خط فاصل بين العناصر.
- `height` : ارتفاع الخط.
- `color` : تحديد لون الخط الفاصل.

🎯 **نصائح سريعة:**

- `Card` بتساعدك تعطي مظهر جذاب للعناصر.
- `ListTile` مثالية لعرض العناصر بشكل قائمة منظمة.
- `Divider` بيقسم لك المحتوى بشكل واضح وسهل.

تحفيز: بمرور الوقت، هتشوف تطورك بوضوح لما تتقدم خطوة بخطوة

🎨 الدرس 8: الألوان والستايلات العامة

✅ BoxDecoration

- ويدجت تستخدم لتزيين الـ `Container` (أو أي ويدجت تانية).
- بتقدر تضيف خلفية ملونة، حدود، ظل، وأشكال دائرية.
- `color` : لتحديد اللون الخلفي.

- `border` : لتحديد الحدود حول العنصر.
- `borderRadius` : عشان تضيف زوايا دائرية.
- `boxShadow` : لإضافة ظل للعنصر.
- `gradient` : عشان تضيف تأثير تدرج لوني.

✅ `Theme` و `ThemeData`

- بتستخدم لتحديد الستايل العام للتطبيق.
- بتقدر تحدد فيه الألوان، الخطوط، حجم النصوص وكل حاجة تخص الستايل.
- `primaryColor` : اللون الأساسي للتطبيق.
- `accentColor` : اللون اللي بيظهر في عناصر تانية زي الأزرار.
- `textTheme` : لتحديد استايل النصوص في التطبيق.

✅ `Color` و `Gradient` و `Border` و `BoxShadow`

- `Color` : لتحديد اللون.
  - `Colors.blue` : لون أزرق.
  - `Colors.red` : لون أحمر.
- `Gradient` : لتحديد تدرج لوني.
  - `LinearGradient` : تدرج لوني من نقطة لأخرى.
  - `RadialGradient` : تدرج دائري من المنتصف.
- `Border` : لتحديد الحدود حول الـ Widget.
  - `Border.all()` : لتحديد الحدود بجميع الاتجاهات.
- `BoxShadow` : لتحديد الظلال حول العنصر.
  - `BoxShadow(blurRadius: 10)` : عشان تضيف ظل ضبابي حول العنصر.

🎯 نصائح سريعة:

- استخدم `BoxDecoration` لتزيين العناصر بشكل مرن.
- `Theme` بيساعدك تحافظ على ستايل موحد في التطبيق.

- `Gradient` والـ `BoxShadow` ممكن تدي مظهر جذاب جدا.

تحفيز: كل ما تكتب كود أكتر، بتتحسن أكتر

🔄 الدرس 9: التفاعل وحالة التطبيق

✅ StatefulWidget

- الـ `StatefulWidget` هو ويدجت بيغير حالته أثناء تشغيل التطبيق.
- لو عايز تغير شيء في الواجهة (مثلاً: زر بيظهر نص مختلف بعد ما يتم الضغط عليه)، هتستخدمه.
- بيسمح بالتفاعل مع المستخدم مثل الأزرار أو التغييرات في البيانات.
- مهم لو كنت هتحتاج تحديث البيانات بشكل ديناميكي.

أهم حاجة فيه:

- بيحتاج إلى `createState()` عشان يعرف يعمل تحديث للواجهة.

✅ StatelessWidget

- الـ `StatelessWidget` هو ويدجت مش هيتغير حالته بعد ما يتعرض.
- بيستخدم في الحالات اللي مفيش فيها تفاعل أو تغيير للبيانات.
- زي الصور أو النصوص الثابتة.

✅ setState()

- `setState()` دي دالة مهمة في الـ `StatefulWidget`.
- عشان تحدّث الواجهة لما تغيّر قيمة معينة (مثلًا: تغيير نص أو حالة عنصر) هتستخدم `setState()`.
- بيقوم بإعادة بناء نفسه عشان يظهر التغيير الجديد widget بعد ما تستخدمها، الـ.

🎯 نصائح سريعة:

- استخدم `StatefulWidget` لما يكون عندك حاجة هتتغير أثناء تشغيل التطبيق.
- لو الواجهة ثابتة، استخدم `StatelessWidget`.

- `setState()` هي السر اللي بيخلي التطبيق يتفاعل مع المستخدم.

تحفيز: الحلول مش دايمًا سهلة، لكن الكود دايمًا هو الحل

🌍 الدرس 10: التنقل بين الصفحات

✅ `Navigator.push` و `pop`

- `Navigator.push`: بتستخدمها علشان تنتقل من صفحة لصفحة جديدة.
  - بحيث المستخدم يقدر يرجع لها، (stack) "بتدفع الشاشة الجديدة في "المكدس
- `Navigator.pop`: بتستخدمها علشان ترجّع الصفحة القديمة بعد ما تخرج من صفحة جديدة.
  - بتحذف الشاشة الحالية من "المكدس" وترجع للمستخدم الصفحة السابقة.

✅ `Drawer`

- الـ `Drawer` هو قائمة جانبية بتظهر من الجنب (غالبًا على اليسار).
- بيتم استخدامها لوضع روابط أو اختيارات مثل الإعدادات، أو المساعدة، أو التنقل بين الصفحات.
- بتظهر لما المستخدم يضغط على الأيقونة الخاصة بالقائمة في الـ `AppBar`.

✅ `BottomNavigationBar`

- `BottomNavigationBar` هو شريط تنقل بيظهر في أسفل الشاشة.
- بيسمح للمستخدم بالتنقل بين الصفحات باستخدام أيقونات أو نصوص.
- بتظهر غالبًا في التطبيقات اللي بتحتاج تبويب بين أكتر من صفحة (زي: الرئيسية، الإعدادات، الملف الشخصي).

🎯 **نصائح سريعة:**

- `Navigator.push` علشان تنتقل لصفحة جديدة.
- `Navigator.pop` علشان ترجع للصفحة السابقة.
- `Drawer` بيكون للقوائم الجانبية.
- `BottomNavigationBar` ممتاز للتنقل بين الصفحات في أسفل التطبيق.

تحفيز: طول ما أنت بتتعلم، أنت ماشي في الطريق الصح