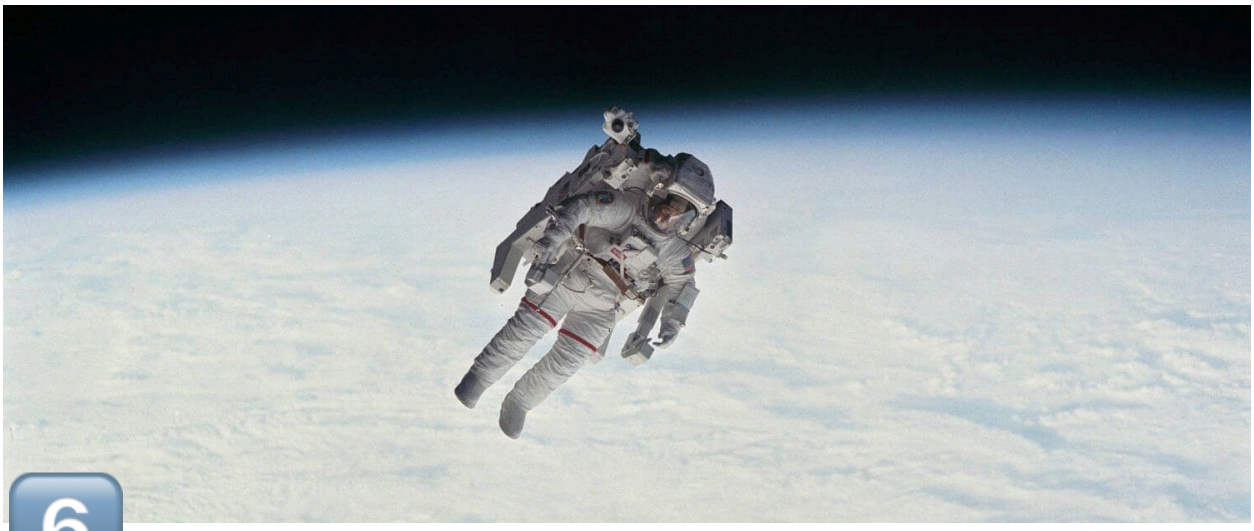

6️⃣

Lesson(6) [Lists & Scroll]

تحفيز: التعلم هو أداة إبداعك

📜 ListView – لعرض قائمة قابلة للتمرير 📱

بيتيح لك عرض **قوائم قابلة للتمرير** سواء كانت عمودية أو أفقية في Flutter **ListView**!

نقدر نستخدمه في التطبيقات لعرض **النصوص، الصور، أو أي محتوى آخر** بشكل مرتب وسهل التمرير.

الأنواع الأساسية لـ ListView 👇:

1. **ListView – قائمة ثابتة (Static List).**
2. **ListView.builder – قائمة ديناميكية (Lazy Loading).**
3. **ListView.separated – قائمة مع فواصل بين العناصر.**
4. **ListView.custom – قائمة مخصصة باستخدام Sliver.**

1️⃣ ListView() – قائمة ثابتة (Static List)

⚙️ ده لو عندك **عدد ثابت من العناصر**، زي ما بتعرض بيانات ثابتة، مثلاً في إعدادات التطبيق.

- **Attributes** :

- **children**: مجموعة العناصر اللي هتظهر في القائمة.
- **scrollDirection**: هل التمرير عمودي (افتراضي) أم أفقي؟
- **padding**: المسافة بين الحواف والعناصر.

- **الكود**:

```
ListView(
  children: [
    ListTile(title: Text('عنصر 1')),
    ListTile(title: Text('عنصر 2')),
    ListTile(title: Text('عنصر 3')),
  ],
  scrollDirection: Axis.vertical,  // التمرير عمودي
  padding: EdgeInsets.all(8),  // المسافة حول العناصر
)
```

🔑 **الـ Output:**

- هتظهر **قائمة ثابتة** من 3 عناصر (عنصر 1، عنصر 2، عنصر 3)، مع **تمرير عمودي**، وعناصر مفرغة من جميع الجوانب.

2️⃣ ListView.builder() – قائمة ديناميكية (Lazy Loading)

مفيد جداً لو عندك **عدد غير محدد من العناصر**، لأن القوائم دي بيتم الـ **ListView.builder** تحميلها عند التمرير، مش كلها من الأول. 👌

- **Attributes**:
  - **itemCount**: عدد العناصر.
  - **itemBuilder**: دالة لبناء العناصر حسب الـ `index`.
  - **scrollDirection**: التمرير عمودي أو أفقي.
  - **shrinkWrap**: لو `true`، هتأخذ المساحة المطلوبة فقط.
  - **physics**: سلوك التمرير.
- **الكود**:

```
ListView.builder(
  itemCount: 10,  // عدد العناصر
  itemBuilder: (context, index) {
    return ListTile(title: Text('عنصر رقم $index'));
  },
  scrollDirection: Axis.vertical,  // التمرير عمودي
  shrinkWrap: true,  // المساحة المطلوبة فقط
  padding: EdgeInsets.all(8),  // المسافة حول العناصر
  physics: BouncingScrollPhysics(),  // سلوك التمرير
)
```

🔑 **الـ Output:**

- هتظهر **10 عناصر** من نوع `ListTile` في (عنصر رقم 0، عنصر رقم 1، وهكذا) بترتيب أرقام **قائمة تمرير عمودي**.
- **يتم تحميل العناصر فقط عند الحاجة** عند التمرير، (Lazy Loading). 😎

3️⃣ ListView.separated() – قائمة مع فواصل بين العناصر

ListView.separated! 🎯 تستخدم `ListView`، لو حابب تضيف **فواصل** بين العناصر داخل الـ

- **Attributes**:
  - **itemCount**: عدد العناصر.
  - **itemBuilder**: دالة لبناء العناصر.
  - **separatorBuilder**: دالة لبناء الفاصل بين العناصر.
  - **scrollDirection**: التمرير عمودي أو أفقي.
  - **shrinkWrap**: لو `true` هتأخذ المساحة المطلوبة فقط.
- **الكود**:

```
ListView.separated(
  itemCount: 10,  // عدد العناصر
  itemBuilder: (context, index) {
    return ListTile(title: Text('عنصر رقم $index'));
  },
```

```
  separatorBuilder: (context, index) {
    return Divider(color: Colors.grey);  // الفاصل بين العناصر
  },
  scrollDirection: Axis.vertical,  // التمرير عمودي
  shrinkWrap: true,  // المساحة المطلوبة فقط
  padding: EdgeInsets.all(8),  // المسافة حول العناصر
  physics: BouncingScrollPhysics(),  // سلوك التمرير
)
```

🔑 **الـ Output:**

- هتظهر `ListTile` عبارة عن خط فاصل، **10 عناصر** من نوع مع **فاصل بين كل عنصر وآخر** (`Divider`).
- التمرير هيفضل عمودي و**العناصر هتأخذ المساحة المطلوبة فقط**. ✨

4️⃣ ListView.custom() – قائمة مخصصة (Custom ListView)

ListView.custom بتديك تحكم كامل في بناء العناصر باستخدام **Sliver**. 🛠️

- **Attributes**:
  - **childrenDelegate**: بتستخدم `SliverChildBuilderDelegate` لبناء العناصر.
  - **scrollDirection**: التمرير عمودي أو أفقي.
  - **shrinkWrap**: تقليص المساحة المطلوبة.
  - **padding**: المسافة حول العناصر.
- **الكود**:

```
ListView.custom(
  childrenDelegate: SliverChildBuilderDelegate(
    (context, index) {
      return ListTile(title: Text('عنصر رقم $index'));
    },
    childCount: 10,  // عدد العناصر
  ),
  scrollDirection: Axis.vertical,  // التمرير عمودي
  shrinkWrap: true,  // فقط المساحة المطلوبة
```

```
  padding: EdgeInsets.all(8),  // المسافة حول العناصر
  physics: BouncingScrollPhysics(),  // سلوك التمرير
)
```

🔑 **الـ Output:**

- مع **تحكم كامل** في الطريقة اللي بتعرض بها العناصر، هتظهر **10 عناصر** من نوع `ListTile` باستخدام `Sliver`!
- التمرير هيكون عمودي والعناصر هتأخذ المساحة المطلوبة فقط. 💪

🎯 ملخص سريع:

- **ListView**: لعرض **قائمة ثابتة** من العناصر.
- **ListView.builder**: لعرض **قائمة ديناميكية** (تحميل العناصر عند التمرير).
- **ListView.separated**: لإضافة **فواصل** بين العناصر.
- **ListView.custom**: والتحكم في طريقة عرض العناصر **Sliver** لعرض قائمة باستخدام.

تحفيز: كل عقبة هي فرصة لتصبح أقوى

📊 GridView – لعرض محتوى في شبكة (Grid) 📱

GridView في Flutter بيتيح لك عرض **محتوى في شبكة**. سواء كانت الشبكة تتكون من أعمدة وصفوف ثابتة أو متغيرة، يمكنك استخدامه لعرض **صور، نصوص، أو أي محتوى آخر** بطريقة منظمة وسهلة.

الأنواع الأساسية لـ GridView 👇:

1. **GridView – شبكة ثابتة (Static Grid).**
2. **GridView.builder – شبكة ديناميكية (Lazy Loading).**
3. **GridView.count – شبكة مع عدد أعمدة ثابت (Fixed Column Grid).**
4. **GridView.extent – شبكة مع أعمدة متغيرة حسب الحجم (Flexible Column Grid).**

1️⃣ GridView() – شبكة ثابتة (Static Grid)

ده لو عندك **عدد ثابت من العناصر** وتريد عرضهم في شبكة ثابتة. 🤩

- **Attributes** :
  - **children**: مجموعة العناصر التي ستظهر في الشبكة.
  - **gridDelegate**: تحديد ترتيب الأعمدة والصفوف في الشبكة.
  - **padding**: المسافة بين الحواف والعناصر.
- **الكود**:

```
GridView(
  children: [
    GridTile(child: Text('1 عنصر')),
    GridTile(child: Text('2 عنصر')),
    GridTile(child: Text('3 عنصر')),
  ],
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: 2,  // عدد الأعمدة
  ),
  padding: EdgeInsets.all(8),  // المسافة حول العناصر
)
```

🔑 **الـ Output:**

- هتظهر **شبكة ثابتة** من 3 عناصر في **شبكة من عمودين**.

2️⃣ GridView.builder() – شبكة ديناميكية (Lazy Loading)

لو عندك **عدد غير محدد من العناصر** وتريد تحميلهم عند التمرير، نستخدم **GridView.builder**. 👌

- **Attributes**:
  - **itemCount**: عدد العناصر.
  - **itemBuilder**: دالة لبناء العناصر حسب الـ `index`.
  - **gridDelegate**: ترتيب الأعمدة والصفوف.
  - **padding**: المسافة بين الحواف والعناصر.

- **الكود**:

```
GridView.builder(
  itemCount: 10,  // عدد العناصر
  itemBuilder: (context, index) {
    return GridTile(child: Text('عنصر رقم $index'));
  },
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: 2,  // عدد الأعمدة
  ),
  padding: EdgeInsets.all(8),  // المسافة حول العناصر
)
```

🔑 **الـ Output:**

- هتظهر (Lazy Loading) **10 عناصر** في **شبكة من عمودين**، و**يتم تحميل العناصر عند التمرير**.

3️⃣ GridView.count() – شبكة مع عدد أعمدة ثابت (Fixed Column Grid)

لو عايز **عدد ثابت من الأعمدة** في الشبكة، استخدم **GridView.count**. 🤩

- **Attributes**:
  - **crossAxisCount**: عدد الأعمدة.
  - **children**: مجموعة العناصر التي ستظهر في الشبكة.
  - **padding**: المسافة بين الحواف والعناصر.
- **الكود**:

```
GridView.count(
  crossAxisCount: 3,  // عدد الأعمدة
  children: List.generate(10, (index) {
    return GridTile(child: Text('عنصر رقم $index'));
  }),
  padding: EdgeInsets.all(8),  // المسافة حول العناصر
```

```
)
```

🔑 **الـ Output:**

- هتظهر **10 عناصر** في **شبكة تحتوي على 3 أعمدة ثابتة**. ✨

4️⃣ GridView.extent() – شبكة مع أعمدة متغيرة حسب الحجم (Flexible Column Grid)

لو عايز عدد الأعمدة يكون **مرن** بناءً على المساحة المتاحة، نستخدم **GridView.extent**. 🎯

- **Attributes**:
  - **maxCrossAxisExtent**: أقصى عرض للعمود.
  - **children**: مجموعة العناصر.
  - **padding**: المسافة بين الحواف والعناصر.
- **الكود**:

```
GridView.extent(
  maxCrossAxisExtent: 150,  // أقصى عرض للعمود
  children: List.generate(10, (index) {
    return GridTile(child: Text('عنصر رقم $index'));
  }),
  padding: EdgeInsets.all(8),  // المسافة حول العناصر
)
```

🔑 **الـ Output:**

- هتظهر **10 عناصر** في **شبكة بأعمدة مرنة** حسب الحجم المحدد (150px). 💪

🎯 ملخص سريع:

- **GridView**: لعرض **شبكة ثابتة** من العناصر.
- **GridView.builder**: لعرض **شبكة ديناميكية** (تحميل العناصر عند التمرير).
- **GridView.count**: لعرض شبكة مع **عدد أعمدة ثابت**.

- **GridView.extent**: لعرض شبكة مع **أعمدة مرنة** حسب الحجم.

تحفيز: كل خطوة في الكود تقربك من النجاح