



# Lesson(1) [Layout & Structure]

تحفيز: اكتب كودك كأنك بتعلمه لغيرك

## لترتيب العناصر جنب بعض (أفقيًا) - Row

✓ **Row** يعني إيه ؟

ببساطة، لو عايز تحط أكثر من ويدجت جنب بعض في سطر واحد (من الشمال لليمين) زي لما تحط زرار وجنبه أيقونة ونص ... **Row** هتستخدم

**Row: أبسط استخدام**

```
Row(  
  children: [  
    Icon(Icons.star),  
    Text('نجمة'),  
  ],  
)
```

## الناتج هيبقى:

أيقونة ★ وبعديها كلمة "نجمة" جنب بعض في نفس السطر

## طيب لو عايز أزود مسافة بينهم؟

بينهم `EdgeInsets` سهل جدًا، تحط

```
Row(  
  children: [  
    Icon(Icons.star),  
    SizedBox(width: 10), // المسافة دي  
    Text('نجمة'),  
  ],  
)
```

## طب إزاي أوزع العناصر جوا الصف؟

دي بتحدد توزيع العناصر أفقيًا ، `MainAxisAlignment` فيه خاصية اسمها

```
Row(  
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
  children: [  
    Icon(Icons.home),  
    Icon(Icons.favorite),  
    Icon(Icons.settings),  
  ],  
)
```

## النتيجة:

= كل أيقونة متوزعة بشكل متساوي على الصف

✓ يعني واحدة عالشمال وواحدة في النص وواحدة عالييمين

## عايز العناصر تطلع في النص بالظبط من فوق وتحت؟

هتستخدم `crossAxisAlignment` :

```
Row(  
  crossAxisAlignment: CrossAxisAlignment.center,  
  children: [  
    Icon(Icons.star, size: 40),  
    Text('نجمة'),  
  ],  
)
```

تحفيز: اتعب النهاردة، وعيش كمبرمج محترف بكرة

## 1. `mainAxisAlignment`

`Column` أو رأسياً في `Row` ده بيحدد طريقة توزيع العناصر أفقياً في

### إزاي بيشتغل؟

- بيحدد توزيع العناصر من اليمين لليساار : `Row` في
- بيحدد توزيع العناصر من فوق لتحت : `Column` في

👁️ `mainAxisAlignment` : القيم اللي ممكن تستخدمها في

- `MainAxisAlignment.start` :  
العناصر هتكون في البداية
- `MainAxisAlignment.end` :  
العناصر هتكون في النهاية
- `MainAxisAlignment.center` :  
العناصر هتكون في النص
- `MainAxisAlignment.spaceBetween` :  
المسافة بين العناصر متساوية، وأول وآخر عنصر في الأطراف
- `MainAxisAlignment.spaceAround` :  
المسافة بين العناصر متساوية، لكن الأطراف بيكون فيهم مسافة أكبر

- `MainAxisAlignment.spaceEvenly` :  
العناصر متوزعة بالتساوي في المساحة.

### كود يوضح `mainAxisAlignment` :

```
Row(  
  mainAxisAlignment: MainAxisAlignment.spaceBetween, // توزيع العناصر بالتساوي  
  children: [  
    Icon(Icons.home),  
    Icon(Icons.favorite),  
    Icon(Icons.settings),  
  ],  
)
```

### النتيجة:

الـ 3 أيقونات ستكون متوزعة بالتساوي في الصف.

## 2. `crossAxisAlignment`

. `Column` أو أفقي في `Row` ده بيحدد طريقة محاذاة العناصر بشكل رأسي في

### إزاي بيشتغل؟

- بيحدد المحاذاة من فوق لتحت : `Row` في
- بيحدد المحاذاة من الشمال لليمين : `Column` في

### القيم اللي ممكن تستخدمها في `crossAxisAlignment` :

- `CrossAxisAlignment.start` :  
( `Column` ) أو على اليسار في `Row` أعلى في) العناصر هتكون متراسة على بداية المحور
- `CrossAxisAlignment.end` :  
( `Column` ) أو على اليمين في `Row` أسفل في) العناصر هتكون متراسة على نهاية المحور

- **CrossAxisAlignment.center** :  
العناصر ستكون محاذاة في النص.
- **CrossAxisAlignment.stretch** :  
العناصر سيتمدد لتغطي المساحة المتاحة.

### **crossAxisAlignment** : كود يوضح


```
Row(  
  crossAxisAlignment: CrossAxisAlignment.center, // محاذاة العناصر في النص  
  children: [  
    Icon(Icons.home, size: 50),  
    Icon(Icons.favorite, size: 70),  
    Icon(Icons.settings, size: 60),  
  ],  
)
```

### **النتيجة:**

ال 3 أيقونات ستكون محاذاة رأسياً في النص.

### **الملخص:**

- **mainAxisAlignment** :
  - **Column** أو **Row** رأسياً في **Row** يحدد توزيع العناصر أفقياً في **Column**.
    - Row أفقي
    - Column رأسي
- **crossAxisAlignment** :
  - **Column** أو **أفقياً** في **Row** يحدد محاذاة العناصر بشكل عمودي في **Column**.
    - Row رأسي
    - Column أفقي

 **تحفيز: فشل اليوم = فهم أقوى بكرة**

## Column

هي الويجت اللي بتستخدم لو عايز ترتب العناصر عمودياً (من فوق لتحت) ال `Column`

### ✓ إزاي بيشتغل؟

- هيتم ترتيبهم واحد فوق الثاني بشكل عمودي ، `Column` لو حطيت أكثر من عنصر جوا ال

### 📦 كود يوضح ال `Column` :

```
Column(  
  children: [  
    Text('الصفحة 1'),  
    Text('الصفحة 2'),  
    Text('الصفحة 3'),  
  ],  
)
```

### 🖼️ النتيجة:

ال 3 نصوص هتظهر بشكل عمودي

## مع `Column` `crossAxisAlignment`

ده هيحدد المحاذاة الأفقية للعناصر ، `Column` في `crossAxisAlignment` لو استخدمت

### ✓ القيم اللي تستخدمها:

- `CrossAxisAlignment.start` : هتكون العناصر على الشمال
- `CrossAxisAlignment.end` : هتكون العناصر على اليمين
- `CrossAxisAlignment.center` : هتكون العناصر في النص أفقياً
- `CrossAxisAlignment.stretch` : العناصر هتمدد علشان تاخذ العرض الكامل

### 📦 كود يوضح المحاذاة الأفقية باستخدام `crossAxisAlignment` :

```
Column(
  crossAxisAlignment: CrossAxisAlignment.center, // محاذاة العناصر أفقيًا في النص
  children: [
    Text('الصفحة 1'),
    Text('الصفحة 2'),
    Text('الصفحة 3'),
  ],
)
```

### النتيجة:

الـ 3 نصوص ستكون محاذاة في النص.

## Column مع mainAxisAlignment

ده هحدد المحاذاة الرأسية (من فوق لتحت) ، Column في mainAxisAlignment لو استخدمت للعناصر.

### ✓ القيم اللي تستخدمها:

- **MainAxisAlignment.start** : هتكون العناصر من فوق.
- **MainAxisAlignment.end** : هتكون العناصر من تحت.
- **MainAxisAlignment.center** : العناصر هتكون في النص رأسياً.
- **MainAxisAlignment.spaceBetween** : المسافة بين العناصر هتكون متساوية.
- **MainAxisAlignment.spaceAround** : العناصر هتكون متوزعة مع مساحة أكبر بين الأطراف.
- **MainAxisAlignment.spaceEvenly** : العناصر هتكون متوزعة بشكل متساوي على كامل المساحة.

### كود يوضح Column في mainAxisAlignment :

```
Column(
  mainAxisAlignment: MainAxisAlignment.spaceBetween, // توزيع العناصر بشكل متساوي عمودياً
)
```

```
children: [
  Text('الصفحة 1'),
  Text('الصفحة 2'),
  Text('الصفحة 3'),
],
)
```

### النتيجة:

المسافة بين النصوص ستكون متساوية، وأول وآخر نص ستكون على الأطراف.

متخيلش error يوقفك

## Expanded و Flexible

. **Column** أو **Row** دول من أكثر الويجتس اللي بتساعد في توسيع العناصر داخل الـ

- بتخلي العنصر يأخذ المساحة المتاحة بالكامل : **Expanded**.
- **flex** (نسبة) بتسمح للعنصر إنه يأخذ جزء من المساحة المتاحة بناءً على الـ **Flexible** : (المساحة).

### **Expanded** :

- بتخلي العنصر يأخذ المساحة المتاحة بعد توزيع المساحة بين العناصر الأخرى.

### **Expanded** : كود يوضح

```
Column(
  children: [
    Expanded(child: Container(color: Colors.red)), // هياخذ مساحة كبيرة
    Container(color: Colors.green, height: 50), // هياخذ المساحة المحددة
  ],
)
```

### النتيجة:

الأخضر هياخذ المساحة المحددة له **Container** الأحمر هياخذ المساحة المتاحة، والـ **Container** الـ



## ✓ Flexible :

- `flex` . لكن هنا بتحدد نسبة المساحة عن طريق الـ `Expanded` بيعمل نفس فكرة

## 📦 Flexible : كود يوضح

```
Row(  
  children: [  
    Flexible(flex: 2, child: Container(color: Colors.red)), // هياخذ 2/3 من المساحة  
    Flexible(flex: 1, child: Container(color: Colors.green)), // هياخذ 1/3 من المساحة  
  ],  
)
```

## 🖼️ النتيجة:

الأحمر هياخذ ضعف المساحة مقارنة بالأخضر الـ `Container`

## 🎯 الملخص:

- ترتيب العناصر عموديًا : `Column`
- تحديد المحاذاة الأفقية في `Column` : `crossAxisAlignment`
- تحديد المحاذاة الرأسية في `Column` : `mainAxisAlignment`
- بيعطي مساحة كاملة للعنصر : `Expanded`
- بتحدد المساحة بناءً على نسبة مئوية : `Flexible`

## تحفيز: ابني العالم اللي تحلم بيه بسطر كود

## SizedBox

هي وبيجت بتستخدم لتحديد حجم ثابت للمسافة أو للعنصر نفسه. يعني لو حبيت `SizedBox` الـ بتحدد عرض أو ارتفاع معين بين العناصر أو حتى تحدد حجم عنصر ثابت، بتستخدمه

## ✓ إزاي بيشتغل؟

- بتقدر تستخدمه ، `Column` أو `Row` لو حبيت تحدد عرض أو ارتفاع ثابت بين عناصر الـ

## كود يوضح ال `SizedBox` :

```
Column(  
  children: [  
    Text('الصفحة 1'),  
    SizedBox(height: 20), // المسافة بين النصوص  
    Text('الصفحة 2'),  
  ],  
)
```

### النتيجة:

فيه مسافة 20 بين النص الأول والثاني

## استخدامه في تحديد الحجم:

```
SizedBox(  
  width: 100, // العرض  
  height: 50, // الارتفاع  
  child: Container(  
    color: Colors.blue,  
  ),  
)
```

### النتيجة:

هياخذ الحجم المحدد له  `Container`  ال

## `Spacer`

هو وسيلة لتوزيع المسافات بشكل ذكي بين العناصر. يحاول يملأ الفراغ المتاح  `Spacer`  ال ويفصل العناصر بشكل متساوي.

## إزاي بيشتغل؟

- بيشتغل المساحة الفارغة بين العناصر بشكل تلقائي.

## **Spacer :** كود يوضح ال

```
Row(  
  children: [  
    Text('الصفحة 1'),  
    Spacer(), // هيفصل بين العناصر بشكل متساوي  
    Text('الصفحة 2'),  
  ],  
)
```

### النتيجة:

هيفصل بين النصين بشكل متساوي، بحيث يوزع المساحة المتاحة بين العناصر **Spacer** ال

### **لو حبيت تستخدمه مع أكثر من عنصر**

```
Row(  
  children: [  
    Text('الصفحة 1'),  
    Spacer(),  
    Text('الصفحة 2'),  
    Spacer(),  
    Text('الصفحة 3'),  
  ],  
)
```

### النتيجة:

هتلاقى النصوص متوزعة بين بعضها والمسافات الفارغة متوزعة بالتساوي

### الملخص:

- **SizedBox** : بتحدد حجم ثابت للعنصر أو المسافة.
- **Spacer** : بيقسم الفراغ المتاح بين العناصر بشكل متساوي.

تحفيز: عيشها مبرمج، وفكر بمنطق 

## Align

يستخدم عشان يحدد مكان العنصر في المساحة المتاحة ليه. يعني لو حابب تحط `Align` ال العنصر في مكان معين زي (يمين، شمال، فوق، تحت) هتستخدمه.

### ✓ إزاي بيشتغل؟

- اللي بتخليك تختار الاتجاه اللي تحب ، `alignment` بيحدد لك مكان العنصر بناءً على خاصية `تحط فيه العنصر`.

### 📦 `Align` : كود يوضح ال

```
Align(  
  alignment: Alignment.topRight, // حط العنصر في أعلى اليمين  
  child: Container(  
    width: 100,  
    height: 100,  
    color: Colors.blue,  
  ),  
)
```

### 🖼️ النتيجة:

هيتحط في أعلى اليمين `Container` ال.

### ✓ `Alignment` : مع خيارات ال

- `Alignment.topLeft` (أعلى يسار)
- `Alignment.topCenter` (أعلى المنتصف)
- `Alignment.center` (في المنتصف)
- `Alignment.bottomRight` (أسفل اليمين)
- `Alignment.bottomLeft` (أسفل اليسار)

## Center

يخلي العنصر يتوسط الشاشة أو المساحة المتاحة له. يعني لو حابب تحط العنصر **Center** ال  
في المنتصف بدون تعقيد، هتستخدمه مباشرة.

## ✓ إزاي بيشتغل؟

- بيحط العنصر في المنتصف تمامًا.

## 📦 **Center** : كود يوضح الـ

```
Center(  
  child: Container(  
    width: 100,  
    height: 100,  
    color: Colors.blue,  
  ),  
)
```

## 🖼️ النتيجة:

الـ **Container** هيتوسط الشاشة.

## 🎯 الملخص:

- **Align**: بيحط العنصر في مكان معين حسب الإتجاه اللي تختاره.
- **Center**: بيخلي العنصر في المنتصف.

## 🔥 تحفيز: البرمجة سلاحك للمستقبل

## Stack

بتستخدم عشان تحط العناصر فوق بعضها. يعني لو حابب تظهر أكثر من عنصر في **Stack** ال  
نفس المكان، وواحد يكون فوق الثاني، هتستخدمها.

## ✓ إزاي بيشتغل؟

- **Stack** بيحدد مكان كل عنصر بداخل الـ **Positioned** بتجمع عناصر كتير في نفس المكان، والـ

## 📦 Stack : كود يوضح ال

```
Stack(  
  children: [  
    Container(  
      width: 200,  
      height: 200,  
      color: Colors.blue,  
    ),  
    Positioned(  
      left: 50, // حط العنصر في المكان ده من اليسار  
      top: 50, // حط العنصر في المكان ده من فوق  
      child: Container(  
        width: 100,  
        height: 100,  
        color: Colors.red,  
      ),  
    ),  
  ],  
)
```

## 🖼️ النتيجة:

الأحمر فوقه في مكان معين **Container** الأزرق في الخلفية، والـ **Container** هتلاقى ال

## ✅ **Positioned** : استخدام ال

- زي المسافات من الأعلى، من اليسار، من ( **Stack** بتحدد مكان العناصر بداخل الـ **Positioned** الـ (اليمين، من الأسفل).

## **Positioned**

عشان تحدد موقع العناصر بداخلها بشكل دقيق **Stack** بتستخدم داخل الـ **Positioned** الـ

## ✅ إزاي بيشتغل؟

- **Stack** بتحدد مكان العنصر بالنسبة للأبعاد الخاصة بالـ

## كود يوضح ال Positioned :

```
Stack(  
  children: [  
    Container(  
      width: 200,  
      height: 200,  
      color: Colors.blue,  
    ),  
    Positioned(  
      left: 20,  
      top: 20,  
      child: Container(  
        width: 100,  
        height: 100,  
        color: Colors.red,  
      ),  
    ),  
    Positioned(  
      right: 20,  
      bottom: 20,  
      child: Container(  
        width: 50,  
        height: 50,  
        color: Colors.green,  
      ),  
    ),  
  ],  
)
```

## النتيجة:

- هتلاقى العنصر الأحمر في أعلى اليسار.
- والعنصر الأخضر في أسفل اليمين.

## 🎯 الملخص:

- بتخلي العناصر تظهر فوق بعضها : **Stack**.
- **Positioned** : بتحدد مكان العنصر داخل الـ **Stack** باستخدام **left** , **top** , **right** , و **bottom** .

## تحفيز: الكود هو الفكرة... وانت المبدع

### Wrap

بتستخدم عشان تنظم العناصر بشكل تلقائي في صفوف أو أعمدة، زي ما بتشوف الـ **Wrap** الـ في قائمة صور أو أي واجهة فيها مجموعة من العناصر اللي محتاجة تتحط بشكل مرن.

### ✅ **Wrap** ؟ إزاي بيشتغل الـ

- بيرتب العناصر في صفوف أو أعمدة (حسب الحجم المتاح) الـ **Wrap** الـ
- بيحط العنصر في صف ثاني أو عمود ثاني تلقائياً **Wrap** أول ما مكان العنصر يخلص، الـ

### 📦 كود يوضح الـ **Wrap** :

```
Wrap(  
  spacing: 10, // المسافة بين العناصر أفقياً  
  runSpacing: 10, // المسافة بين الصفوف عمودياً  
  children: [  
    Container(width: 100, height: 100, color: Colors.blue),  
    Container(width: 100, height: 100, color: Colors.red),  
    Container(width: 100, height: 100, color: Colors.green),  
    Container(width: 100, height: 100, color: Colors.yellow),  
  ],  
)
```

### 🖼️ النتيجة:

- العناصر هتترتب بشكل تلقائي في صفوف جديدة لو المكان مش كافي.
- بيحدد المسافة بين العناصر أفقياً **spacing** الـ
- بيحدد المسافة بين الصفوف عمودياً **runSpacing** الـ



## ✓ يمكن استخدامه في حاجات زي **Wrap** الـ:

- لو عندك مجموعة صور أو أكواد لعرضها في صفوف وأعمدة.
- لو عايز تتأكد إن العناصر هتظهر بشكل جيد في أي حجم شاشة.

## 🎯 الملخص:

- بتستخدم لترتيب العناصر بشكل تلقائي في صفوف أو أعمدة: **Wrap**.

تحفيز: كل سطر كود هو خطوة نحو النجاح