Lección: Comprendiendo el Uso de BoxFit en Flutter

Objetivos de la Lección

- Comprender el propósito y funcionamiento del widget **BoxFit** en Flutter.
- Identificar las diferentes opciones de **BoxFit** y cómo afectan la visualización de imágenes o elementos dentro de un contenedor.
- Implementar ejemplos prácticos de cada valor de **BoxFit** para ver cómo se adaptan las imágenes a un contenedor.

¿Qué es BoxFit en Flutter?

BoxFit es una enumeración en Flutter que define cómo debe ajustarse una imagen o cualquier elemento al tamaño de su contenedor. Al especificar un valor de **BoxFit** para un widget que contiene una imagen, podemos decidir si la imagen debe ajustarse completamente, rellenar, cubrir o mantener su proporción dentro del contenedor.

La propiedad `fit` se usa comúnmente con widgets de imagen como **Image** para controlar cómo una imagen se adapta a su caja contenedora. Esto es especialmente útil cuando el tamaño del contenedor no coincide con las dimensiones de la imagen.

Valores Principales de BoxFit

Flutter proporciona varios valores para **BoxFit**, cada uno de los cuales determina cómo se escala el contenido para ajustarse al contenedor:

1. **BoxFit.cover**: Escala la imagen para cubrir completamente el contenedor, recortando las partes que sobresalgan.
2. **BoxFit.contain**: Escala la imagen para que se ajuste completamente dentro del contenedor sin recortarla, manteniendo su proporción.

3. **BoxFit.fill**: Escala la imagen para llenar completamente el contenedor, ignorando la proporción original.
4. **BoxFit.fitWidth**: Escala la imagen para ajustarse al ancho del contenedor y recorta las partes verticales si es necesario.
5. **BoxFit.fitHeight**: Escala la imagen para ajustarse a la altura del contenedor y recorta las partes horizontales si es necesario.
6. **BoxFit.none**: No ajusta la imagen; la muestra en su tamaño original. Puede quedar recortada si es más grande que el contenedor.
7. **BoxFit.scaleDown**: Similar a `BoxFit.contain`, pero solo escala la imagen si es más grande que el contenedor. De lo contrario, la imagen mantiene su tamaño original.

Ejemplo Básico de BoxFit en Imágenes

A continuación, veremos ejemplos de cada valor de **BoxFit** usando una imagen simple. Los ejemplos muestran cómo la imagen se ajusta en un contenedor de tamaño fijo con cada valor de **BoxFit**.

1. BoxFit.cover

`BoxFit.cover` escala la imagen para cubrir todo el contenedor. La imagen puede ser recortada si su proporción no coincide con la del contenedor.

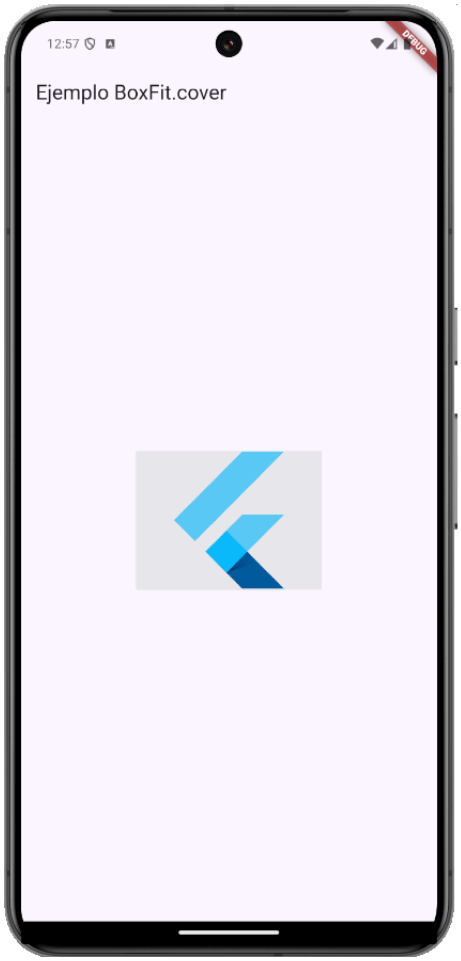
```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
```

```
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Ejemplo BoxFit.cover'),
        ),
        body: Center(
          child: Container(
            width: 200,
            height: 150,
            child: Image.network(
              'https://flutter.dev/images/flutter-logo.png',
              fit: BoxFit.cover,
            ),
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

- La imagen cubre todo el contenedor, ajustándose para llenar el espacio disponible.
    - Utiliza la siguiente dirección URL e imagen: https://static-00.iconduck.com/assets.00/flutter-icon-2048x2048-ufx4idi8.png

- Cualquier parte de la imagen que no encaje en la proporción del contenedor será recortada.



Captura de la pantalla del ejemplo presentado con el Widget BoxFit y su propiedad “cover” utilizando una imagen leída de Internet. Creado por Javier A. Dastas (2024)

2. BoxFit.contain

`BoxFit.contain` escala la imagen para que se ajuste completamente dentro del contenedor sin recortarla, manteniendo su proporción original.

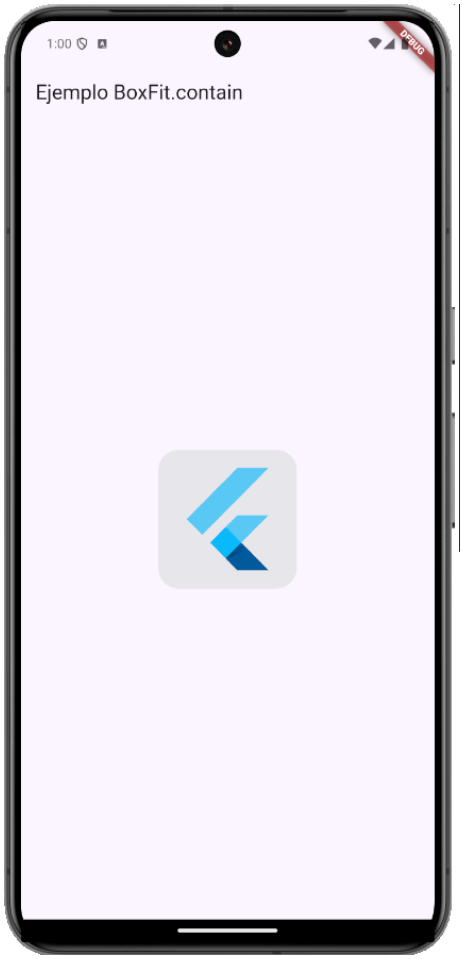
```
import 'package:flutter/material.dart';


void main() {
  runApp(MyApp());
```

```
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Ejemplo BoxFit.contain'),
        ),
        body: Center(
          child: Container(
            width: 200,
            height: 150,
            child: Image.network(
              'https://flutter.dev/images/flutter-logo.png',
              fit: BoxFit.contain,
            ),
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

- La imagen se escala para caber dentro del contenedor sin recortar ninguna parte.

- Se mantiene la proporción de la imagen, y el espacio sobrante se muestra como áreas en blanco.



Captura de la pantalla del ejemplo presentado con el Widget BoxFit y su propiedad “contain” utilizando una imagen leída de Internet. Creado por Javier A. Dastas (2024)

3. BoxFit.fill

`BoxFit.fill` ajusta la imagen para llenar completamente el contenedor, ignorando la proporción original. Esto puede distorsionar la imagen.

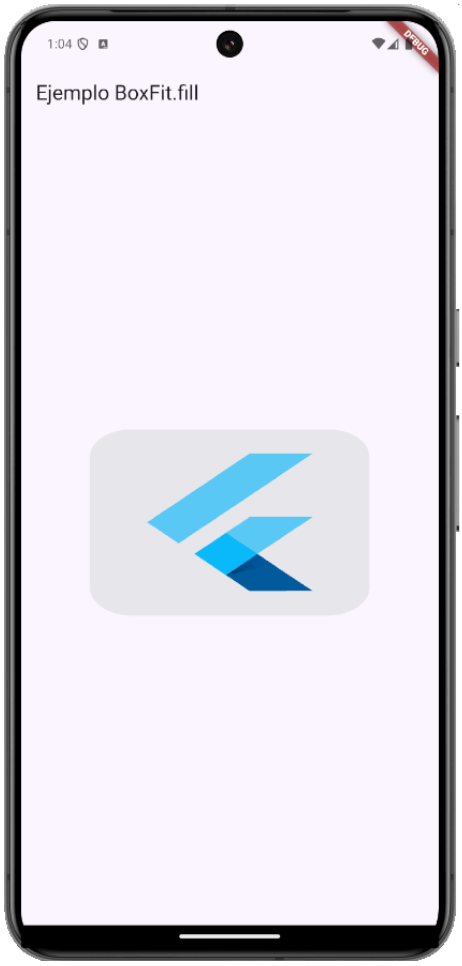
```
import 'package:flutter/material.dart';


void main() {
  runApp(MyApp());
```

```
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Ejemplo BoxFit.fill'),
        ),
        body: Center(
          child: Container(
            width: 200,
            height: 150,
            child: Image.network(
              'https://flutter.dev/images/flutter-logo.png',
              fit: BoxFit.fill,
            ),
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

- La imagen se estira o se comprime para llenar el contenedor por completo.

- Esto puede causar una **distorsión** en la imagen, ya que la proporción original no se mantiene.



Captura de la pantalla del ejemplo presentado con el Widget BoxFit y su propiedad “fill” utilizando una imagen leída de Internet. Creado por Javier A. Dastas (2024)

4. BoxFit.fitWidth

`BoxFit.fitWidth` escala la imagen para que se ajuste al ancho del contenedor, recortando las partes verticales si es necesario.

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


void main() {
  runApp(MyApp());
```

```
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Ejemplo BoxFit.fitWidth'),
        ),
        body: Center(
          child: Container(
            width: 200,
            height: 150,
            child: Image.network(
              'https://flutter.dev/images/flutter-logo.png',
              fit: BoxFit.fitWidth,
            ),
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

- La imagen se escala para que su ancho coincida con el del contenedor.

- Si la altura no encaja, se recorta verticalmente para ajustarse al contenedor.

5. BoxFit.fitHeight

`BoxFit.fitHeight` escala la imagen para que se ajuste a la altura del contenedor, recortando las partes horizontales si es necesario.

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

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Ejemplo BoxFit.fitHeight'),
        ),
        body: Center(
          child: Container(
            width: 200,
            height: 150,
            child: Image.network(
              'https://flutter.dev/images/flutter-logo.png',
              fit: BoxFit.fitHeight,
            ),
```

```
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

- La imagen se escala para que su altura coincida con la del contenedor.
- Si el ancho no encaja, la imagen se recorta horizontalmente para ajustarse al contenedor.

6. BoxFit.none

`BoxFit.none` no ajusta la imagen. Esta se muestra en su tamaño original y puede quedar recortada si es más grande que el contenedor.

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

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
```

```
          title: Text('Ejemplo BoxFit.none'),
        ),
        body: Center(
          child: Container(
            width: 200,
            height: 150,
            child: Image.network(
              'https://flutter.dev/images/flutter-logo.png',
              fit: BoxFit.none,
            ),
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

- La imagen conserva su tamaño original.
- Si la imagen es más grande que el contenedor, se recorta en los bordes.

7. BoxFit.scaleDown

`BoxFit.scaleDown` es similar a `BoxFit.contain`, pero solo reduce el tamaño de la imagen si es más grande que el contenedor. Si es más pequeña, no la escala.

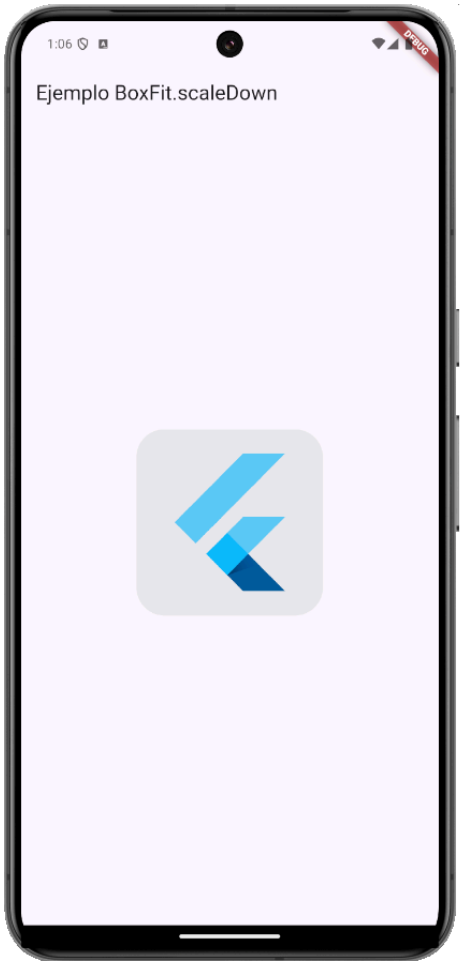
```
import 'package:flutter/material.dart';


void main() {
```

```
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Ejemplo BoxFit.scaleDown'),
        ),
        body: Center(
          child: Container(
            width: 200,
            height: 150,
            child: Image.network(
              'https://flutter.dev/images/flutter-logo.png',
              fit: BoxFit.scaleDown,
            ),
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

- La imagen se escala solo si es más grande que el contenedor.

- Mantiene su tamaño original si ya es más pequeña que el contenedor.



Captura de la pantalla del ejemplo presentado con el Widget BoxFit y su propiedad “scaleDown” utilizando una imagen leída de Internet.
Creado por Javier A. Dastas (2024)

Resumen de la Lección

En esta lección, hemos aprendido sobre el uso del **BoxFit** en Flutter y cómo se puede usar para ajustar imágenes dentro de un contenedor. Exploramos los diferentes valores de **BoxFit** (`cover`, `contain`, `fill`, `fitWidth`, `fitHeight`, `none` y `scaleDown`) y vimos ejemplos prácticos para comprender cómo cada uno afecta la visualización de una imagen.

El uso de **BoxFit** es esencial para crear interfaces visualmente agradables y optimizadas, ya que permite controlar cómo se ajustan las imágenes a diferentes tamaños de pantalla y resoluciones.

Actividad de la Lección

Esta actividad te ayudará a aplicar y experimentar con los diferentes valores de **BoxFit** y a comprender cómo afectan la visualización de las imágenes en Flutter.

Instrucciones:

1. Crea una aplicación en Flutter con una imagen cargada desde Internet. Implementa la imagen en un contenedor y prueba cada valor de **BoxFit** para observar cómo se ajusta en el contenedor.
2. Explica cuál de los valores de **BoxFit** crees que es el más adecuado para una imagen de fondo que deba cubrir toda la pantalla sin distorsionarse y por qué.
3. Entrega un documento en formato PDF con copia de tu código y copia de imágenes o capturas de pantalla demostrando que tu código funciona.