# Lección 11: Introducción al Uso del Widget `Container` en Flutter

## Objetivos de la Lección

- Comprender las características y uso básico del widget `Container` en Flutter.
- Configurar el tamaño, color, margen y bordes de un `Container`.
- Implementar un diseño sencillo usando `Container` y otros widgets básicos como `Text`, `SizedBox`, `Column` y `Row`.

## Introducción de la Lección

El widget `Container` es uno de los widgets más versátiles en Flutter. Se usa frecuentemente como un contenedor flexible para organizar otros widgets y configurar propiedades como el tamaño, el color de fondo, el margen, los bordes y mucho más. En esta lección, explicaremos cómo personalizar y organizar elementos en una aplicación Flutter usando `Container` junto con algunos otros widgets básicos.

## Desarrollo del Tema Central

### 1. Concepto de Container

Un `Container` en Flutter es un widget que permite envolver otros widgets para añadir estilos y configuraciones visuales. Con él, se pueden ajustar parámetros como el color de fondo, el tamaño, los márgenes, y las sombras.

La estructura básica de un `Container` es la siguiente:

```
Container(
  width: 100, // Ancho del container
  height: 100, // Alto del container
  color: Colors.blue, // Color de fondo
  child: Text('Texto en el Container'), // Widget hijo
);
```

## 2. Personalización del Container

**Tamaño**

El tamaño del `Container` se puede ajustar usando los parámetros `width` (ancho) y `height` (alto). Si no se especifican, el `Container` se ajustará según el tamaño de su contenido.

**Ejemplo**: Crear un `Container` con tamaño específico

```
Container(
  width: 150,
  height: 100,
  color: Colors.lightBlue,
  child: Center(child: Text('Tamaño fijo')),
);
```

**Color de Fondo**

El color de fondo se establece con la propiedad `color`. Esto le da al `Container` un color de fondo uniforme.

**Ejemplo**: Añadir un color de fondo a un `Container`

```
Container(
  width: 200,
  height: 100,
  color: Colors.greenAccent,
  child: Center(child: Text('Color de fondo')),
);
```

**Margen y Padding**

- **Margin**: Espacio entre el `Container` y otros widgets alrededor.
- **Padding**: Espacio dentro del `Container` alrededor de su contenido.

**Ejemplo**: Añadir margen y padding a un `Container`

```
Container(
  margin: EdgeInsets.all(20), // Margen externo de 20 puntos
  padding: EdgeInsets.all(10), // Espacio interno de 10 puntos
alrededor del texto
  color: Colors.orange,
  child: Text('Margen y Padding'),
);
```

**Bordes y Bordes Redondeados**

Con la propiedad `decoration` podemos añadir bordes personalizados y bordes redondeados usando `BoxDecoration`.

**Ejemplo**: Crear un `Container` con bordes redondeados

```
Container(
  width: 200,
  height: 100,
  decoration: BoxDecoration(
    color: Colors.purple,
    borderRadius: BorderRadius.circular(15),
  ),
  child: Center(child: Text('Bordes redondeados', style:
TextStyle(color: Colors.white))),
);
```

## 3. Ejemplos Prácticos

**Ejemplo 1: Crear una Tarjeta Simple**

Este ejemplo organiza varios `Container` dentro de una columna para crear una tarjeta básica.

```
Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Container(
      width: 200,
      height: 100,
      color: Colors.blueGrey,
      child: Center(child: Text('Título de la Tarjeta', style:
TextStyle(color: Colors.white))),
```

```
    ),
    SizedBox(height: 10), // Espacio entre los containers
    Container(
      width: 200,
      padding: EdgeInsets.all(10),
      color: Colors.white,
      child: Text('Esta es una tarjeta de ejemplo hecha con
Container.'),
    ),
  ],
);
```

**Ejemplo 2: Usando `Container` en un Layout de Columna y Fila**

En este ejemplo, utilizamos `Row` y `Column` para organizar varios `Container` de diferentes colores.

```
Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Container(
          width: 80,
          height: 80,
          color: Colors.red,
```

```
          child: Center(child: Text('1', style: TextStyle(color:
Colors.white))),
        ),
        SizedBox(width: 10),
        Container(
          width: 80,
          height: 80,
          color: Colors.green,
          child: Center(child: Text('2', style: TextStyle(color:
Colors.white))),
        ),
      ],
    ),
    SizedBox(height: 10),
    Container(
      width: 180,
      height: 80,
      color: Colors.blue,
      child: Center(child: Text('3', style: TextStyle(color:
Colors.white))),
    ),
  ],
);
```

## Ejemplo de Sombra en un Container

El uso de sombra puede añadir profundidad y hacer que el `Container` destaque visualmente.

```
Container(
  width: 200,
  height: 100,
  decoration: BoxDecoration(
    color: Colors.teal,
    boxShadow: [
      BoxShadow(
        color: Colors.black.withOpacity(0.3),
        spreadRadius: 5,
        blurRadius: 7,
        offset: Offset(0, 3),
      ),
    ],
  ),
  child: Center(child: Text('Container con Sombra', style:
TextStyle(color: Colors.white))),
);
```

## Relación con Otros Conceptos o Lecciones

El widget `Container` es un componente clave en Flutter y se usará en aplicaciones de interfaz más complejas. Es útil para crear diseño básico y, al combinarlo con otros widgets como `Column`, `Row`, y `Text`, permite construir interfaces de usuario flexibles y personalizables.

## Resumen de la Lección

En esta lección, exploramos el widget `Container` y aprendimos cómo configurar su tamaño, color, márgenes, bordes, y cómo organizar varios `Container` dentro de una aplicación. También vimos ejemplos de cómo crear tarjetas sencillas y cómo aplicar sombras para mejorar el diseño visual.

## Actividad de la Lección

1. Crea una tarjeta en Flutter utilizando un `Container` que incluya:
    - Un título centrado en la parte superior.
    - Un cuerpo de texto con descripción.
    - Un color de fondo para cada sección (título y cuerpo).
2. Usa `margin` y `padding` para separar cada sección y agrega bordes redondeados y sombra a la tarjeta.