Dropdown Menu

Aşağıya doğru açılır bir menü yapmak ve bu menüden bir öğeyi seçmek için DropdownButton Widget'ı kullanılır

DropdownButton: (items ve onChanged parameterlerini kullanmak mecburi)

Bir menü butonu olduğu için ve menüdeki herhangi bir seçeneğin seçilmesi ve dolayısıyla ekran yenilenmesi durumu olduğu için Durumlu bir widget içerisinden kullanılmalıdır.

items : Listede gösterilecek elemanlar String bir liste şeklinde verilir. Bunun için DropdownMenuItem sınıfı kullanılabilir yada String bir liste değişkeni de kullanılabilir.

DropdownMenuItem(child: Text('Elma'), value: 'Elma',),
child : Menüde listelenecek öğe, string olmalıdır.
Value: Seçilen öğenin program içerisinde kullanılabilmesini sağlamak için öğenin değerini tutan parametre.

onChanged: (onChanged){}, Menüden seçim yapıldığında seçilen öğeyi göstermek ve değerini kullanmak için kullanılan parametre, parantez içindeki onChanged parametresine DropdownMenuItem'ın value parametresindeki değeri gelir.

value: Menüde seçilen öğenin değerin tutulduğu parametre

hint: Menüde ilk başta gösterilecek bir açıklama ifadesi vermek istenirse

icon: Menüde ikon görüntüler

iconSize: İkon boyutunu belirler

underline: Menüde öğenin alt kısmına bir çizgi oluşturmak için kullanılır

style: Yazı stilini belirler.

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

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Giris(),
    );
  }
```

```
}

class Giris extends StatefulWidget {
  @override
  _GirisState createState() => _GirisState();
}

class _GirisState extends State<Giris> {
  var secilen='Elma';
  //var secilen='';
  //String? secilen=null;

// List<String> meyveler=['Elma','Kiraz','Armut'];
// List<String> meyveler=['Elma','Kiraz','Armut','Çilek'];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: DropdownButton(
          items: [
            DropdownMenuItem(child: Text('Elma'), value: 'Elma',),
            DropdownMenuItem(child: Text('Kiraz'), value: 'Kiraz',),
            DropdownMenuItem(child: Text('Armut'), value: 'Armut',),
          ],

//  items: meyveler.map((String e) => DropdownMenuItem(child: Text(e),
value: e,)).toList(),

          //value: 'Elma',
          value: secilen,


          onChanged: (gelendeger){             // (String? gelendeger)
            print(gelendeger);
            secilen = gelendeger.toString();  // secilen=gelendeger;
            print(secilen);

        /*    setState(() {
              print(gelendeger);
              secilen = gelendeger;
            });
        */
          },

          hint: Text('Meyve Seç'),
          icon: Icon(Icons.arrow_downward),
          iconSize: 60,
          underline: Container(height: 3, color: Colors.purple,),
          style: TextStyle(color: Colors.purple, fontSize: 20,
              backgroundColor: Colors.yellow),
        ),
      ),
    );
  }
}
```

DropdownMenude items parametresini daha fonksiyonel kullanmak için, yani tek tek yazmamak için,

Menüdeki listeyi bir List olarak sınıfın hemen altında tanımlarız.

```
List<String> meyveler=['Elma','Kiraz','Armut'];
```

Daha sonra items parametresinde listelerin map metodundan faydalanırız,

```
items: meyveler.map((String e) => DropdownMenuItem(child: Text(e), value:
e,)).toList(),
```

Böylece listeleri istediğimiz kadar ekleyip çıkararak daha kolay bir liste oluşturmuş oluruz.

PopupMenu:

Default olarak üst üste üç nokta ikon gösterimi ile oluşturulan menüdür. İstenirse ikon değiştirilebilinir.

```
PopupMenuButton(itemBuilder: itemBuilder)
```

itemBuilder: List<PopupMenuEntry<dynamic>> Function(BuildContext)

itemBuilder parametresine Context bilgisi alan bir fonksiyon tanımlamak gerekir. Bu fonksiyon bize PopupMenuEntry<dynamic> nesnelerinden oluşan bir liste dönmelidir. PopupMenuEntry<dynamic> nesneleri menüde gösterilecek olan öğeleri belirler.

onSelected: Bu parametre menüden seçilen öğenin alınmasını sağlar. void Function(dynamic)? Tipinde bir fonksiyon kullanılması gerekir. Fonksiyonda kullanılan parametreye menüden seçilen öğe gelmiş olur.

child: Parametresine Text ile bir öğe ismi yazdırılabilir. O zaman üst üste ikon yerine bu öğe gelmiş olur. Bu durumda DropdownMenu'den bir farkı kalmaz.

icon: Bu parametre ile ikon değişikliği yapılır.

iconSize: İkonun büyüklüğünü belirler.

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

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
```

```
        primarySwatch: Colors.blue,
      ),
      home: Giris(),
    );
  }
}

class Giris extends StatefulWidget {
  @override
  _GirisState createState() => _GirisState();
}

class _GirisState extends State<Giris> {
  var secilen='aa';
  // List<String> meyveler=['Elma','Kiraz','Armut'];
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: PopupMenuButton(
          itemBuilder: (BuildContext context){
            return <PopupMenuEntry>[
              PopupMenuItem(child: Text('Elma'), value: 'Elma',),
              PopupMenuItem(child: Text('Kiraz'), value: 'Kiraz',),
              PopupMenuItem(child: Text('Armut'), value: 'Armut',),
            ];
            //   return meyveler.map((e) => PopupMenuItem(child: Text(e),
value: e,)).toList();
          },
          onSelected: (meyve){
            setState(() {
              print(meyve);
              secilen=meyve.toString();
            });
          },
          //icon: Icon(Icons.add),
          //child: Text(secilen),
        ),
      ),
    );
  }
}
```

PopupMenude items parametresini daha fonksiyonel kullanmak için, yani tek tek yazmamak için,

Menüdeki listeyi bir List olarak sınıfın hemen altında tanımlarız.

```
List meyveler=['Elma','Kiraz','Armut'];
```

Daha sonra return ile geriye listelerin map metodundan faydalanarak popupitem nesnelerini göndeririz,

```
return meyveler.map((e) => PopupMenuItem(child: Text(e), value: e,)).toList();
```

Böylece listeleri istediğimiz kadar ekleyip çıkararak daha kolay bir liste oluşturmuş oluruz.

ListView() Widget:

Ekranda sabit bir liste görüntülenmesini sağlar. Liste adedi ekranın alt sınırını geçerse yukarı aşağı kaydırılarak liste öğleri görünebilir.

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

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

class MyApp extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Liste(),
    );
  }
}

class Liste extends StatelessWidget {
  const Liste({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: ListView(
                  children: [
                    Text('Elma'),
                    Text('Kiraz'),    //
                    Text('Armut'),
                  ],
                ),
    );
  }
}
```

Yukarıdaki meyve listesini ekranı aşacak şekilde artırarak tekrar deneyiniz.

Expanded() Widget:

Expanded, row ve column içindeki widgetlara ana eksende müsait olan tüm boş alanı doldurmalarını söyler.

Bu örnekte Expanded, Sütun widgetında ListViewin görünmesi için bir bağdaştırıcı olarak kullanılmalıdır. Bunu kullanmadan ekranda görüntüleyemeyiz.

Yukarıdaki örneğin Liste sınıfını aşağıdaki gibi yazabiliriz.

```
class Liste extends StatelessWidget {
  const Liste({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: Column(
        children: [
          Expanded(
            child: ListView(
                        children: [
                          Text('Elma'),
                          Text('Kiraz'),
                          Text('Armut'),
                        ],
                      ),
          ),
        ],
      ),
    );
  }
}
```

Dinamik bir liste yapmak istersek;

Liste elemanlarını bir liste halinde tanımlayıp, ListView.builder kullanarak listeleme yapabiliriz.

```
ListView.builder(itemBuilder: itemBuilder)
```

itemBuilder: Widget Function(BuildContext, int) int parametresine bir isim değişken ismi verilir. Bu değişken değeri 0 dan başlar ve döngü sonuna kadar 1 artarak gider.

itemCount: Listede görünecek öğe sayısını belirler. Bunu vermek zorunludur yoksa itemBuilder parametresine verilen fonksiyon sonsuz döngüye girer ve program hata verir.

```
class Liste extends StatelessWidget {
  List meyve=["Elma","Kiraz","Armut"];
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: Column(
        children: [
          Expanded(
            child: ListView.builder(
              itemCount: meyve.length,
              itemBuilder: (BuildContext context, int index){
                return Text(meyve[index],);
              },
            ),
          )
        ],
      ),
    );
  }
}
```

GridView() Widget:

Ekranı kareli defter misali bölümlere ayırarak öğeleri bu bölümler içerisinde göstermek için kullanılır

```
GridView.count(
    crossAxisCount: 2,     // Bir satırda kaç tane nesne olacak
    childAspectRatio: 2/1,   // Nesnelerin yatay dikey oranı
    padding: EdgeInsets.all(10),  // Köşelerdeki boşluk değeri
    crossAxisSpacing: 15,    // Sütunlar arası boşluk
    mainAxisSpacing: 15,    // İki satır arasındaki boşluk
    scrollDirection: Axis.horizontal,   // Dikey olan gridview yatay olarak
konumlanır, nesneler yatay olarak yerleşir ve kaydırma yönüde yatay olarak
değişmiş olur.
    reverse: true,   // Nesnelerin yerleşimi ters sırada olur.
    children: [
            Text("Elma"),
            Text("Kiraz"),
            Text("Armut"),
    ],
)
```

```
class Liste extends StatelessWidget {
  //List meyve=["Elma","Kiraz","Armut"];
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: GridView.count(
          crossAxisCount: 2,
          childAspectRatio: 1,
          //childAspectRatio: 2/1,
          //padding: EdgeInsets.all(10),
          //crossAxisSpacing: 15,
          //mainAxisSpacing: 15,
          //scrollDirection: Axis.horizontal,
          //reverse: true,
          children: [
            Text("Elma"),
            Text("Kiraz"),
            Text("Armut"),
          ],
      ),
    );
  }
}
```

GridView.extent(): GridView.count() yerine kullanılabilir. Extent'te bir satırda kaç eleman nesne olacağı bilgisi yerine bir nesnenin ekranda ne kadar yer kaplayacağı bilgisi verilir. Nesneler ona göre yerleşir.

```
GridView.extent(

      maxcrossAxisExtent: 100,   // Nesnenin ekrandaki en/boy boyutu belirlenir.

)
```

Dinamik GridView kullanmak için;

```
GridView.builder(
        gridDelegate: gridDelegate, kaç satır ve öğe ekran oranı gibi
özellerin verildiği kısım. Burada SliverGridDelegateWithFixedCrossAxisCount
sınıfını kullanırız
        itemBuilder: itemBuilder), GridView de gösterilecek nesnelerin
oluşmasını sağlayan contex bilgisi alan bir fonksiyon
```

```
class Liste extends StatelessWidget {
  List meyve=["Elma","Kiraz","Armut"];
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: GridView.builder(
          gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
              crossAxisCount: 2,
              childAspectRatio: 2/1
          ),
          itemCount: meyve.length,
          itemBuilder: (BuildContext context, int index){
            return Text(meyve[index]);
          }),
    );
  }
}
```

Alttaki Widgetların hepsi en sonda verilen uygulama ile örneklendirildi.

ListTile() Widget


```
ListTile(
  leading: Icon(Icons.add),  // Icon yerine CircleAvatar(child:
Icon(Icons.add),),
  title: Text('Başlık'),
  subtitle: Text('Alt Başlık'),
  trailing: Icon(Icons.ac_unit),
  tileColor: Colors.red,
  onTap: (){     // Tıklandığında çalışan fonksiyon
    print('Tıklandı');
  },
),
```

Card() Widget: Kartvizit oluşturmak için kullanılır 


```
Card(
  color: Colors.blue[100],   //renk verir
  shadowColor: Colors.purple,  //gölge rengi verir
  elevation: 12,   //gölgenin genişliğini verir
  shape: RoundedRectangleBorder( // köşeleri yuvarlar
    borderRadius: BorderRadius.circular(20),
    ),
```

```
  child: Text('Elma',
    style: TextStyle(fontSize: 30),
    ),
),
```

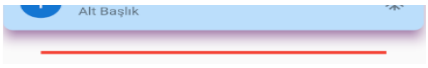
Child: parametresine Text yerine listelemelerde kullanılan **ListTile** sınıfı kullanabiliriz. Daha güzel bir görünüm elde edebiliriz.

GestureDetector() Widget:

Card Widget veya Container gibi tıklandığında herhangi bir çalışacak parametresi olmayan nesneler için, tıklandığında çalışacak özellik eklemeyi sağlayan bir widget.

```
GestureDetector(
  onTap: (){print("tıkla");},
  child: Card(
    color: Colors.blue,
    child: Text("Meyve"),
  ),
),
```

Divider() Widget: Yatay Bir Çizgi




```
Divider(
  color: Colors.red, //rengi
  thickness: 4,   //kalınlığı
  height: 10,   //üst nesneden uzaklık
  indent: 40,   //sol kenardan uzaklık
  endIndent: 40,  //sağ kenardan uzaklık
)
```

Örnek uygulama:

```
class Liste extends StatelessWidget {
  List meyve=["Elma","Kiraz","Armut"];
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: Column(
        children: [
          ListTile(
            leading: Icon(Icons.add),  // Icon yerine CircleAvatar(child:
Icon(Icons.add),),
            title: Text('Başlık'),
            subtitle: Text('Alt Başlık'),
            trailing: Icon(Icons.ac_unit),
            tileColor: Colors.red,
            onTap: (){     // Tıklandığında çalışan fonksiyon
              print('Tıklandı');
            },
          ),
          Card(
```

```
            color: Colors.blue[100],   //renk verir
            shadowColor: Colors.purple,  //gölge rengi verir
            elevation: 12,   //gölgenin genişliğini verir
            shape: RoundedRectangleBorder( // köşeleri yuvarlar
              borderRadius: BorderRadius.circular(20),
              ),
            child: Text('Elma',
              style: TextStyle(fontSize: 30),
              ),
          ),
          GestureDetector(
            onTap: (){print("tıkla");},
            child: Card(
              color: Colors.blue,
              child: Text("Meyve"),
            ),
          ),
          Divider(
            color: Colors.red, //rengi
            thickness: 4,   //kalınlığı
            height: 10,   //üst nesneden uzaklık
            indent: 40,   //sol kenardan uzaklık
            endIndent: 40,  //sağ kenardan uzaklık
          ),
        ],
      ),
    );
  }
}
```