Rows, Columns & Icons: (Refer app: eyecon)

Now we’re talking, aren’t we?

Now is the time to add multiple widgets onto the screen at once. Because, up until now, our apps looked somewhat mundane to the eye. We’ll start with rows first:

A row means a horizontal arrangement of objects. Anything that we put into a row will be placed one beside the other. We need to add a widget called ‘Row’ to the body property. This widget has a property called ‘*children’* (Notice that the word children implies that it accepts more than one child—which is kind of the point of rows). This property accepts a *list of widgets.* As discussed in the DART primer, a list is nothing but an array in DART. It’s datatype is the objects of the Widget class.

Let’s add 3 widgets to the row:

1. A simple text widget
2. A flat button widget (we’ll learn about it in a moment, for now, just roll with it)
3. A container widget containing text

We’ll colour them differently in order to identify them. The code looks like:

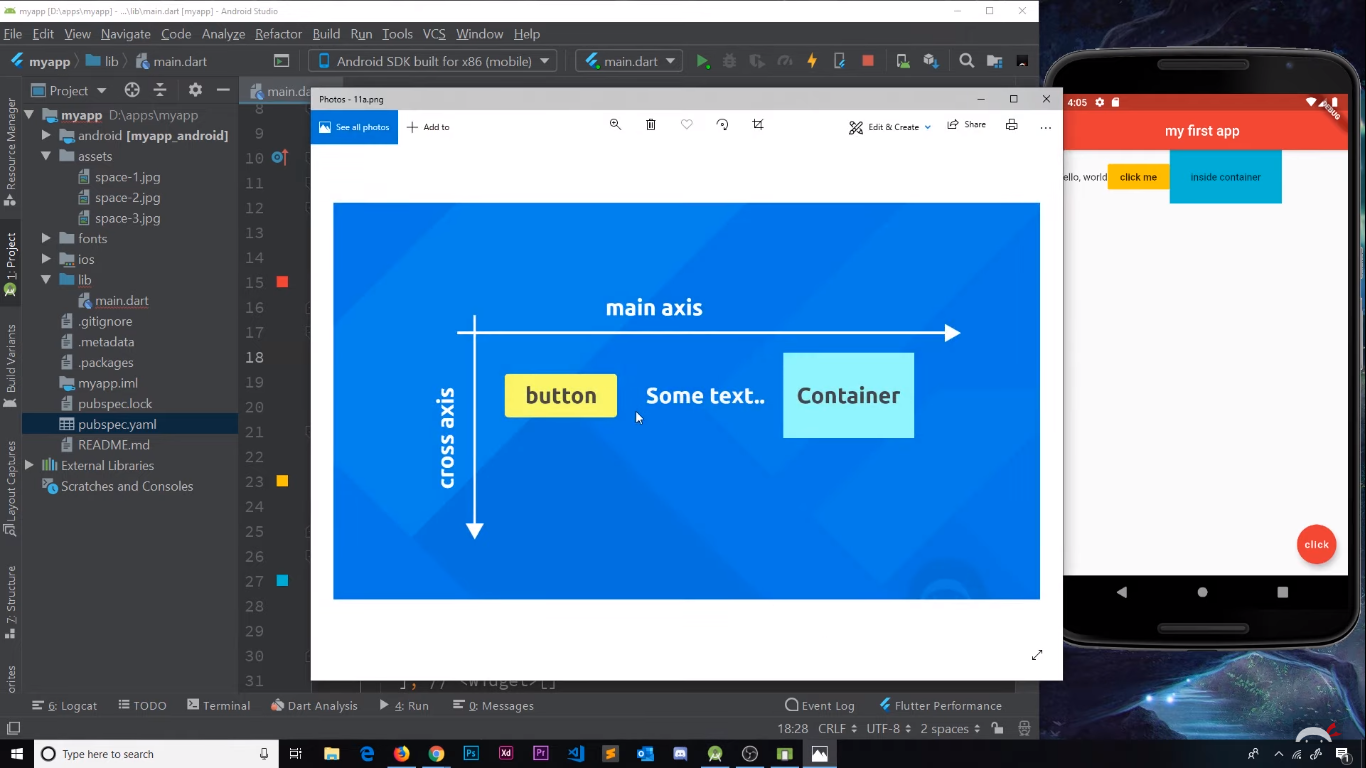
body: Row(  
 children: <Widget>[  
 Text(**'Your'**),  
 FlatButton(  
 onPressed: () {},  
 child: Text(**'Mom'**),  
 color: Colors.*orange*,  
 )  
 Container(  
 child: Text(**'Gay'**),  
 color: Colors.*cyan*,  
 padding: EdgeInsets.all(20.0),  
 )  
 ],  
),



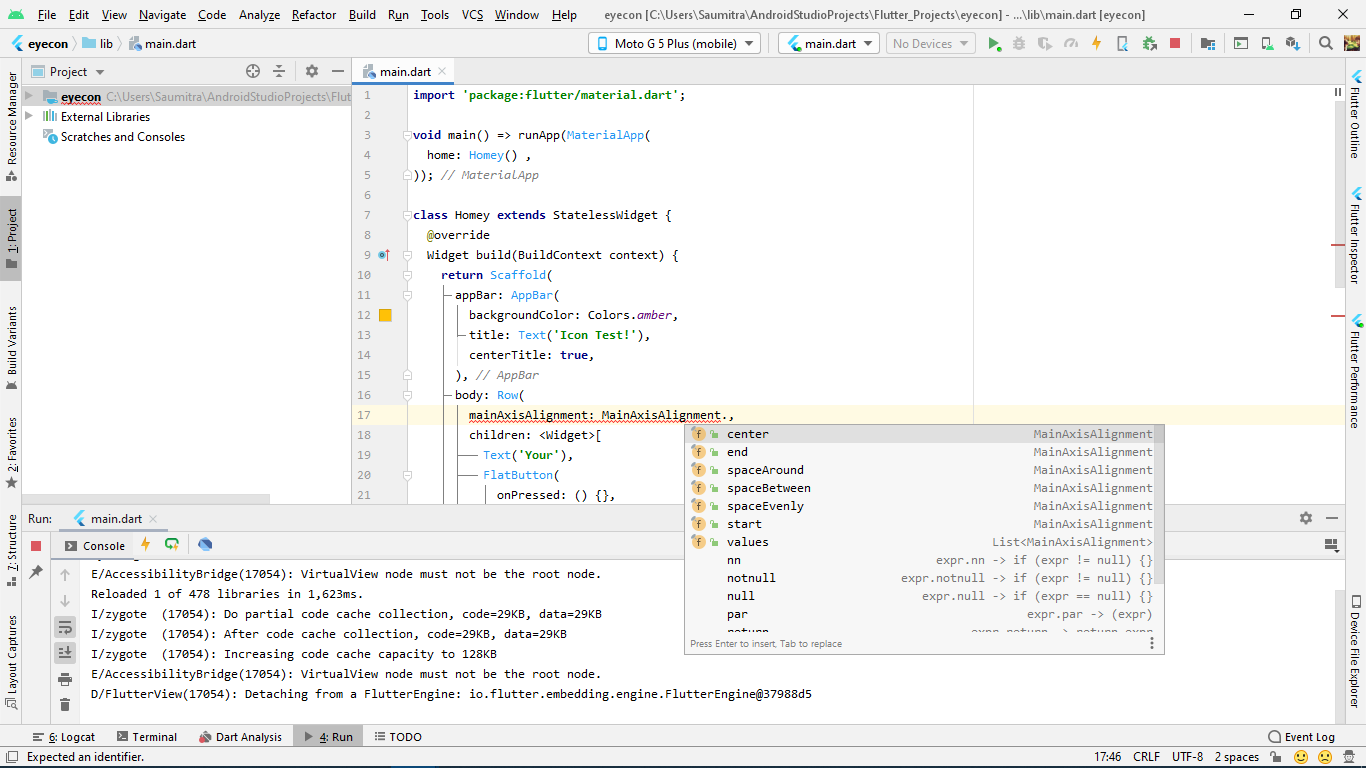
Et Voila! We now have multiple widgets on the screen!

These widgets however, look kind of bunched up or stacked together. In order to change the alignment of these widgets, we’ll use properties of the row widget called ‘MainAxisAlignment’ and ‘CrossAxisAlignment’.

In the case of rows, the horizontal axis is the main axis, and the axis that crosses them is the cross axis.

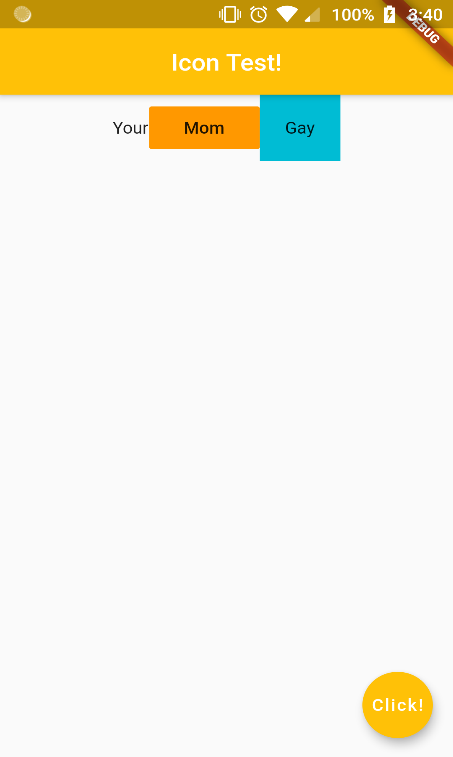


Let’s start with mainAxisAlignment. It has a value of MainAxisAlignment.<various options>.



1. center : Positions the elements at the centre of the row.

mainAxisAlignment: MainAxisAlignment.**center**



1. end : Bunches the elements to the right hand side of the screen (The “end” of the screen).

mainAxisAlignment: MainAxisAlignment.**end**



1. spaceAround : Similar to spaceBetween, the difference is that the space between the widgets is double that between the widget and the end of the screen. Essentially each widget has like a margin around it.

mainAxisAlignment: MainAxisAlignment.**spaceAround**



1. spaceBetween : Places elements with spaces in between, but no spaces at the ends. Start and end elements are stuck to the LHS and RHS of the screen.

mainAxisAlignment: MainAxisAlignment.**spaceBetween**,



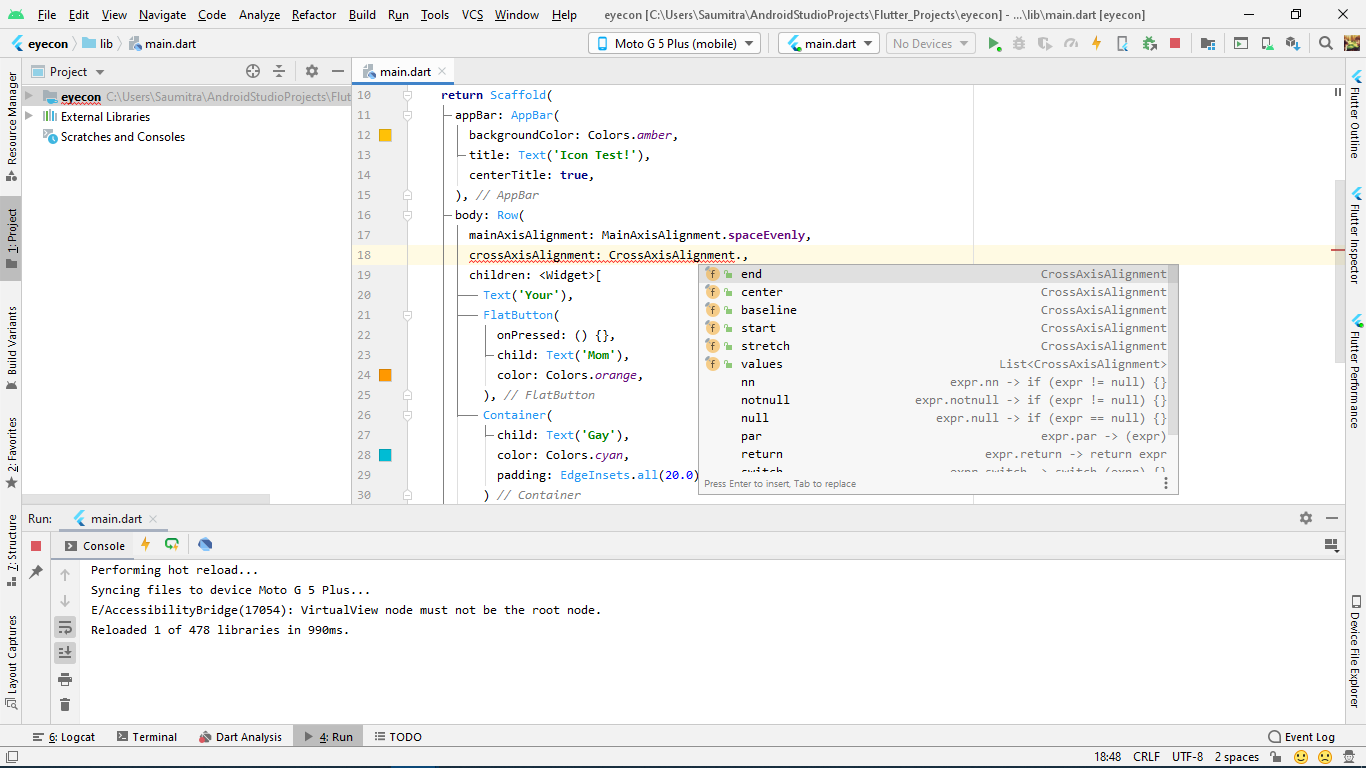
1. spaceEvenly : Spaces elements evenly across the entire row. Elements are also separated from the ends of the screen.

mainAxisAlignment: MainAxisAlignment.**spaceEvenly**

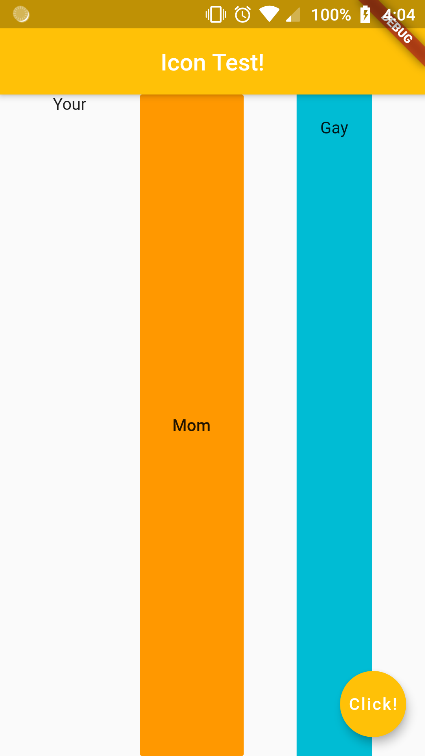


1. start: This the default setting. Elements are positioned from the start of the screen (LHS)

In a similar manner, we have the crossAxisAlignment property, with multiple options. This property adjusts how the elements look with respect to the vertical axis of the row.



1. Stretch: Stretches elements across the entire cross axis of the row. Because there is only one row, the cross axis is the entire screen height.



1. Center: The default setting, at the centre of the height of the tallest widget.
2. Start: Positions elements at the start of the cross axis



1. End: Positions elements at the bottom of the tallest widget

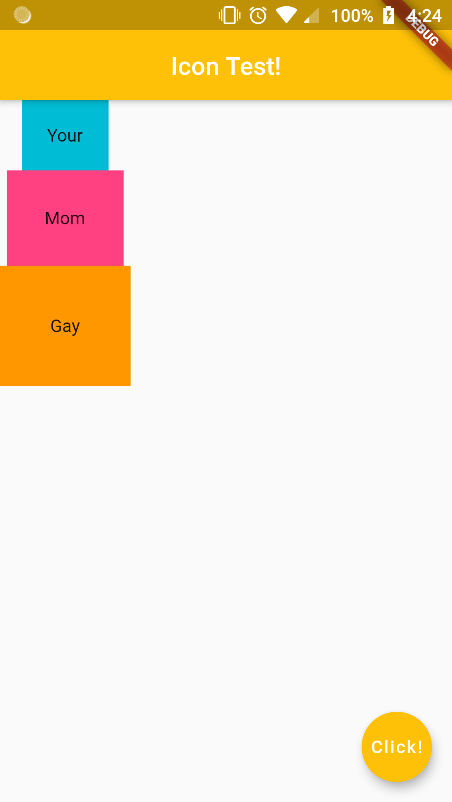


The entire code looks like:

body: Row(  
 mainAxisAlignment: MainAxisAlignment.**spaceEvenly**,  
 crossAxisAlignment: CrossAxisAlignment.**end**,  
 children: <Widget>[  
 Text(**'Your'**),  
 FlatButton(  
 onPressed: () {},  
 child: Text(**'Mom'**),  
 color: Colors.*orange*,  
 ),  
 Container(  
 child: Text(**'Gay'**),  
 color: Colors.*cyan*,  
 padding: EdgeInsets.all(20.0),  
 )  
 ],  
),

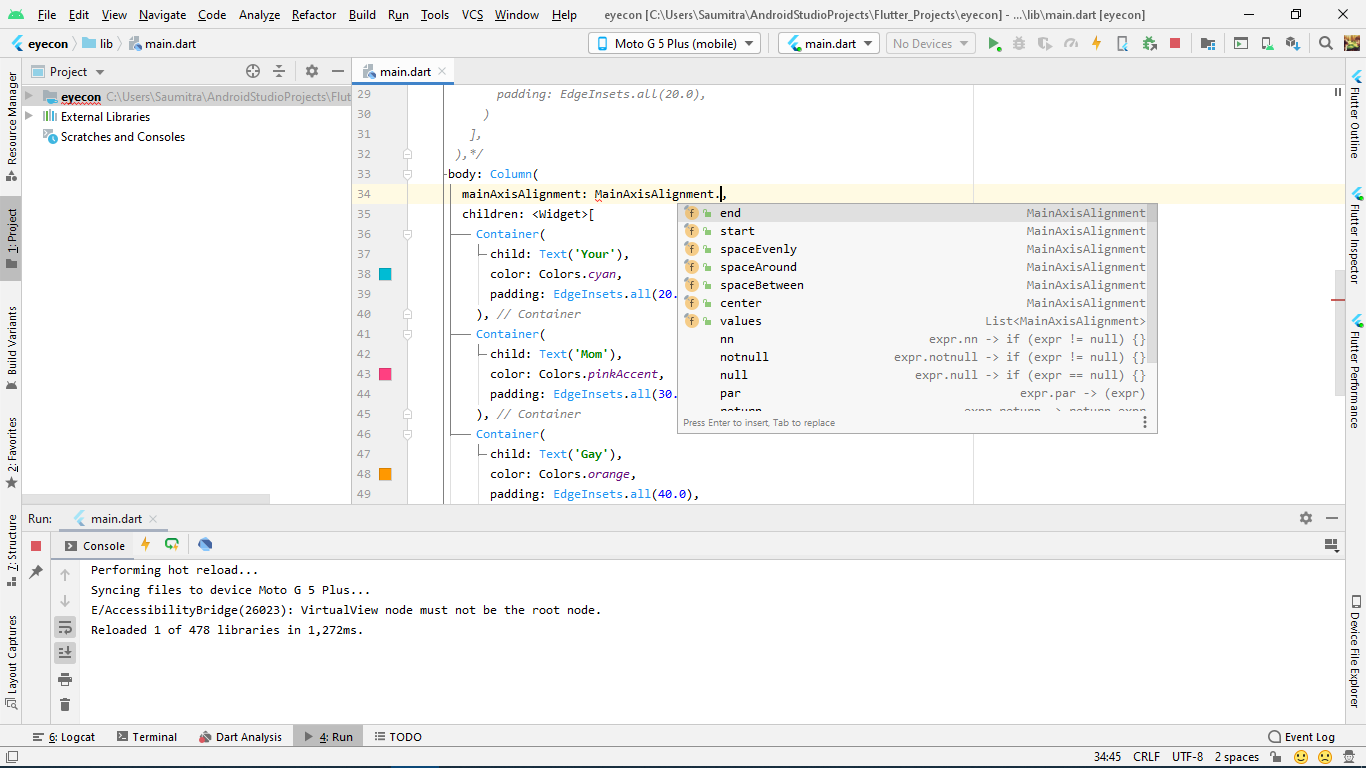
Moving on to columns, we have the exact similar format of children as that of rows. The code now looks like:

body: Column(  
 children: <Widget>[  
 Container(  
 child: Text(**'Your'**),  
 color: Colors.*cyan*,  
 padding: EdgeInsets.all(20.0),  
 ),  
 Container(  
 child: Text(**'Mom'**),  
 color: Colors.*pinkAccent*,  
 padding: EdgeInsets.all(30.0),  
 ),  
 Container(  
 child: Text(**'Gay'**),  
 color: Colors.*orange*,  
 padding: EdgeInsets.all(40.0),  
 )  
 ],  
),

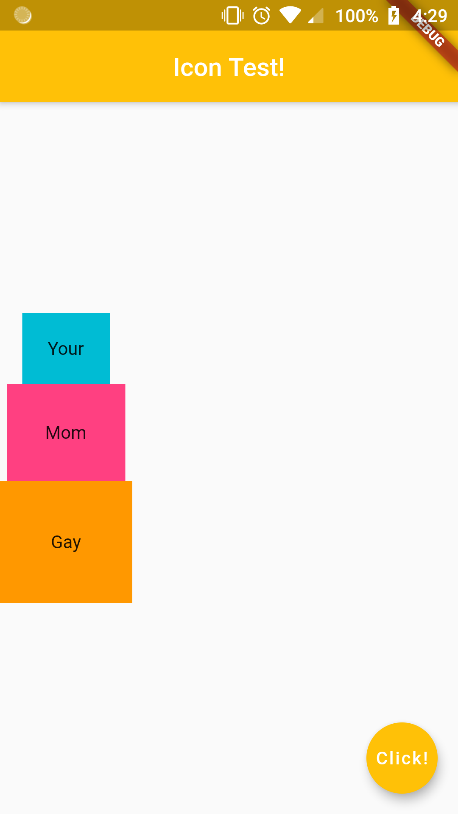


For the alignments, we need to understand that the ‘MainAxis’ for the column widget is VERTICAL, and the ‘CrossAxis’ is HORIZONTAL.

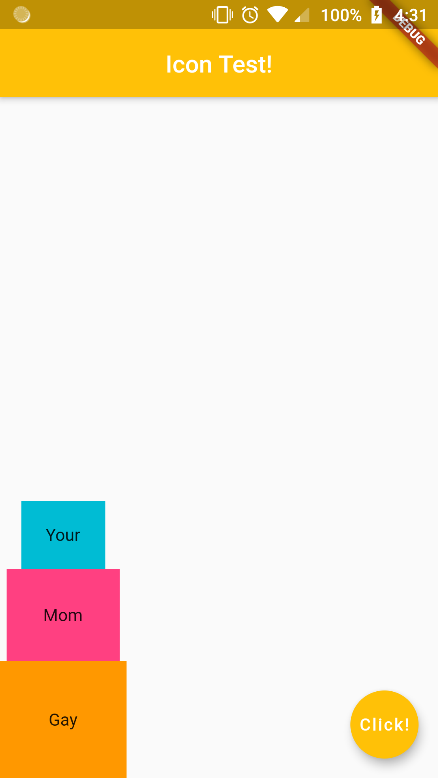
Let’s start with mainAxisAlignment. It has a value of MainAxisAlignment.<various options>.



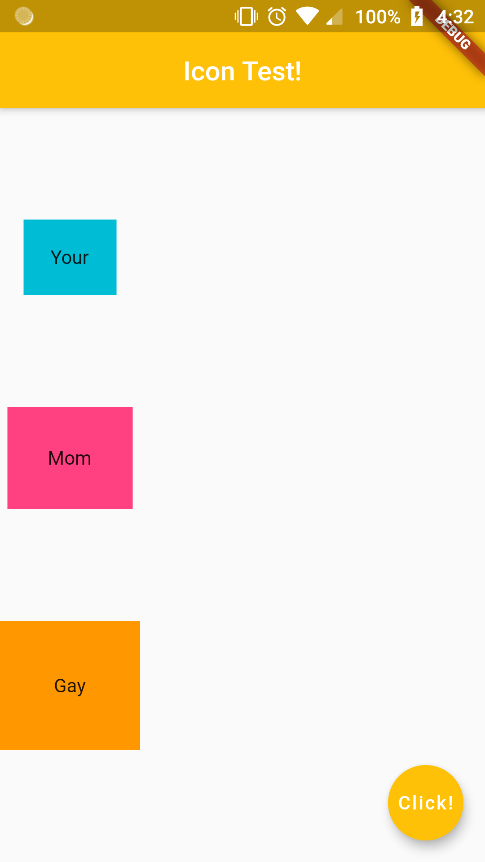
1. Center: Positions the elements at the centre of the column.



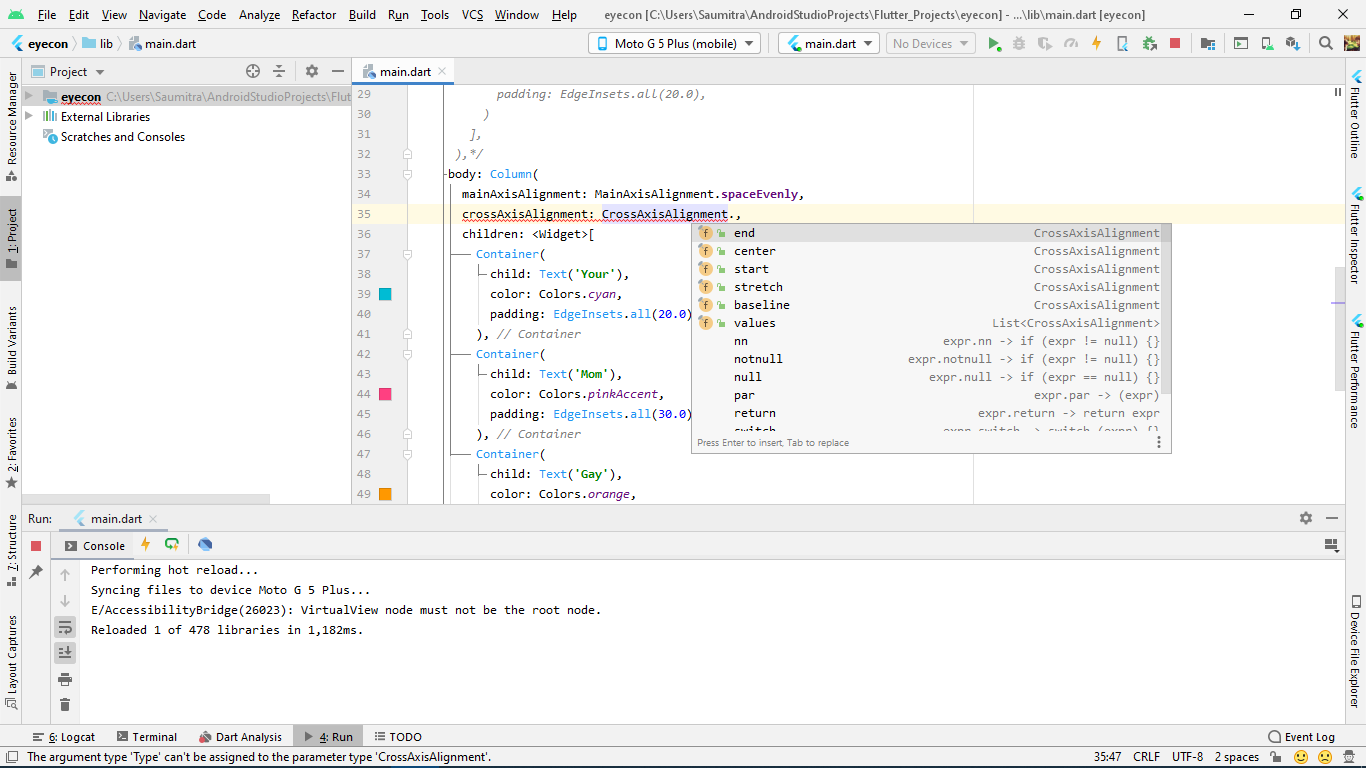
1. End: Bunches the elements to the bottom of the main axis (The “end” of the screen).



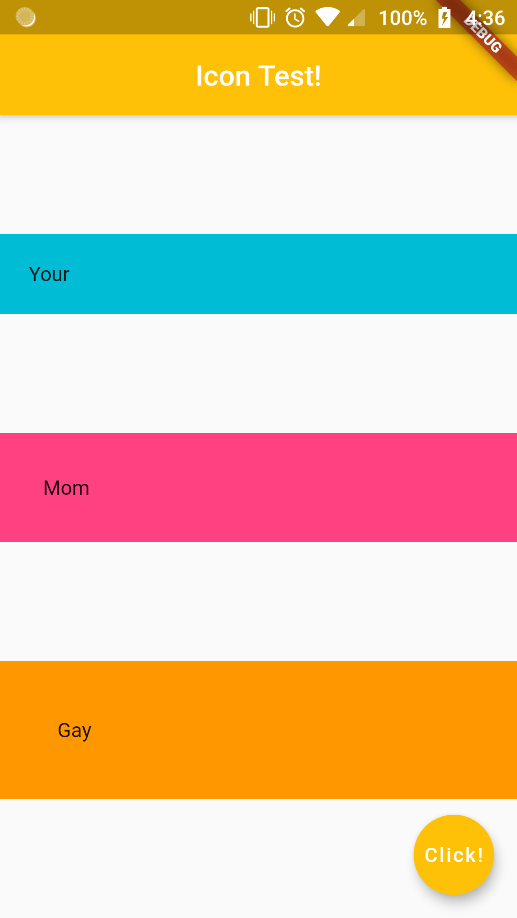
1. spaceEvenly : Spaces elements evenly across the entire column. Elements are also separated from the ends of the screen.



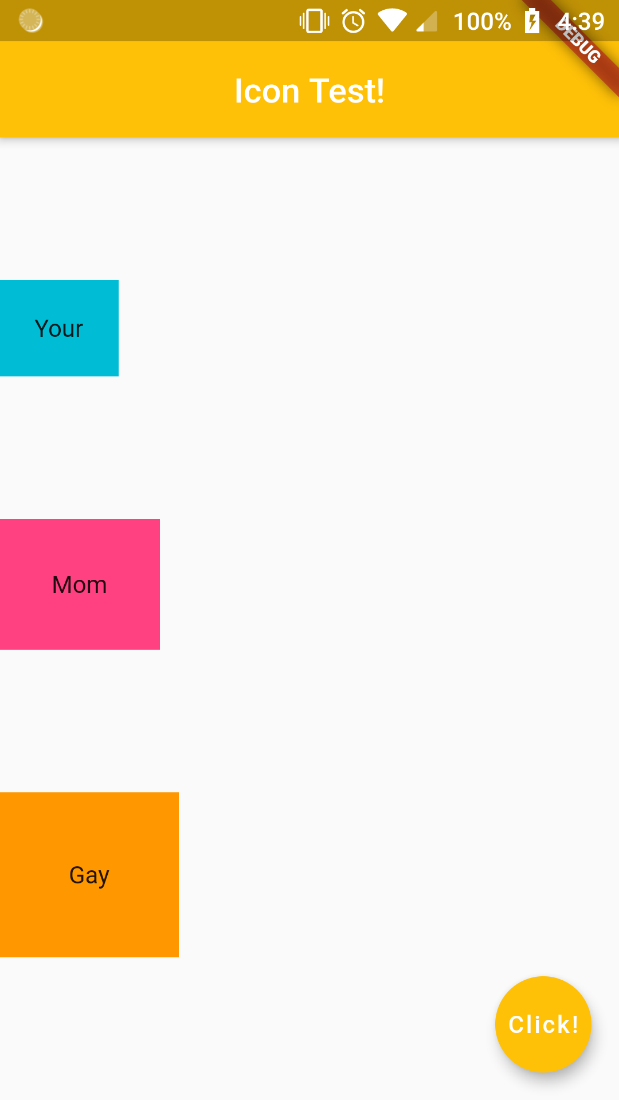
Now, let’s try the cross axis alignment…



1. Stretch: Stretches elements across the entire cross axis of the row. Because there is only one row, the cross axis is the entire screen width.



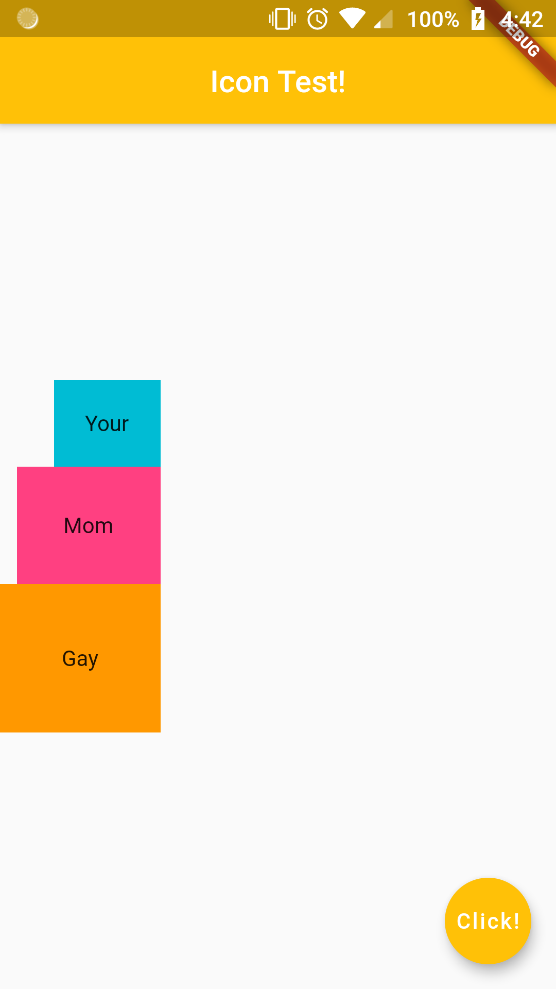
1. Center: The default setting, at the centre of the width of the widest widget.
2. Start: Positions elements at the start of the cross axis



1. End: Positions elements at the bottom of the widest widget.

\*Here we have also changed the mainAxisAlignment

mainAxisAlignment: MainAxisAlignment.**center**,  
crossAxisAlignment: CrossAxisAlignment.**end**



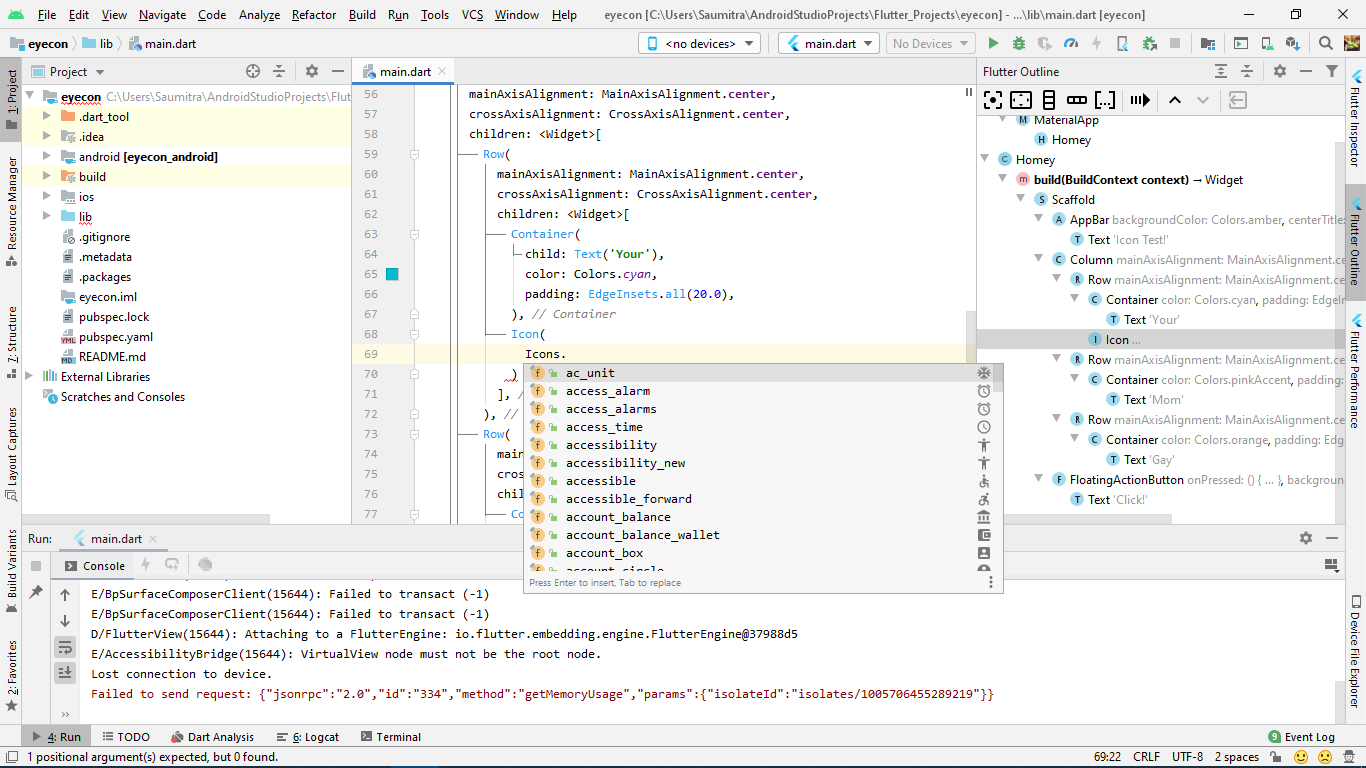
Icons and Buttons:

Let’s play around with icons and buttons, in combination with rows and columns, so that we get a good practice of both.

Icons:

Flutter gives us the ability to use any icon from the material design library. This can be done with the help of the Icon() widget. The first parameter that we can specify is “what” icon to use, which is some icon from the ‘Icons’ library.

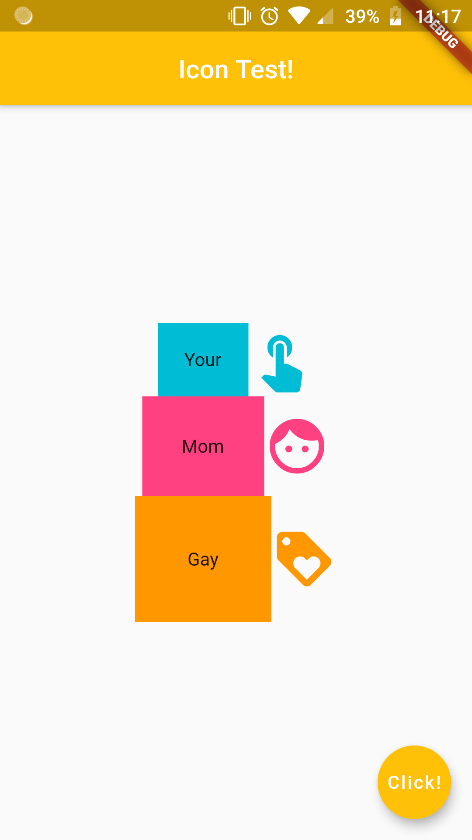
Just pressing a dot after Icons makes the android studio autocomplete tool fetch little graphic images of the icon we want to use.



Each icon has a colour and size property which we can adjust as per our specifications. Fiddling with the rows and columns, we inserted 3 rows in one column. Each row has one container and one icon. The code looks like:

body: Column(  
 mainAxisAlignment: MainAxisAlignment.**center**,  
 crossAxisAlignment: CrossAxisAlignment.**center**,  
 children: <Widget>[  
 Row(  
 mainAxisAlignment: MainAxisAlignment.**center**,  
 crossAxisAlignment: CrossAxisAlignment.**center**,  
 children: <Widget>[  
 Container(  
 child: Text(**'Your'**),  
 color: Colors.*cyan*,  
 padding: EdgeInsets.all(20.0),  
 ),  
 Icon(  
 Icons.*touch\_app*,  
 color: Colors.*cyan*,  
 size: 50.0,  
 )  
 ],  
 ),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.**center**,  
 crossAxisAlignment: CrossAxisAlignment.**center**,  
 children: <Widget>[  
 Container(  
 child: Text(**'Mom'**),  
 color: Colors.*pinkAccent*,  
 padding: EdgeInsets.all(30.0),  
 ),  
 Icon(  
 Icons.*face*,  
 color: Colors.*pinkAccent*,  
 size: 50.0,  
 )  
 ],  
 ),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.**center**,  
 crossAxisAlignment: CrossAxisAlignment.**center**,  
 children: <Widget>[  
 Container(  
 child: Text(**'Gay'**),  
 color: Colors.*orange*,  
 padding: EdgeInsets.all(40.0),  
 ),  
 Icon(  
 Icons.*loyalty*,  
 color: Colors.*orange*,  
 size: 50.0,  
 )  
 ],  
 )  
  
 ],  
),

The output looks like:



Buttons:

The concept of buttons is quite simple in flutter, and is similar to icons. There are a lot of different kind of buttons like the flatButton, raisedButton, iconButton, and their names are quite self-explanatory. The one thing that needs to be understood about buttons is that they COMPULSORILY NEED to have the onPressed() method. It should be no surprise that a button has ONE JOB and that is to do something when someone presses it. We can use the ‘print’ method inside the onPressed() method to print out statements onto the console whenever the user clicks on the button.

Raised Button: The interesting thing about this button is that it gives a slight 3D appeal with a shadow below it. This is what gives it the ‘raised’ feel. It accepts various parameters like:

Colour, child (to insert text), size, etc.

Flat Button: It doesn’t have the shadow below it. It is simply an ordinary rectangle on the screen. It accepts various parameters like:

Colour, child (to insert text), size, etc.

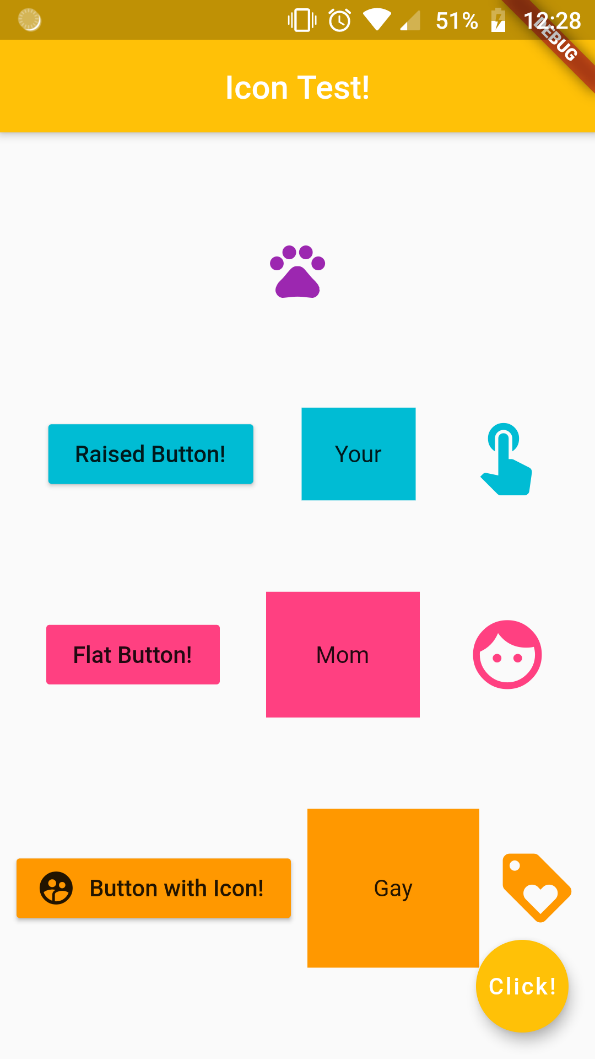
Icon Button: It is exactly what is says. It is an icon which can be pressed, and therefore acts like a button. It accepts various parameters like:

Icon,Colour, child (to insert text), size, etc.

RaisedButton.Icon(or FlatButton.Icon) : This is a mechanism to add icons INSIDE buttons. It allows us to specify an icon, a label (a text next to the icon), the colour of the button, the colour of the icon, and much more. It therefore is a combination of the button and the icon.

The code which demonstrates all of this is as follows:

**import 'package:flutter/material.dart'**;  
**import 'package:fluttertoast/fluttertoast.dart'**;  
  
**void** main() => runApp(MaterialApp(  
 home: Homey() ,  
));  
  
**class** Homey **extends** StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 **return** Scaffold(  
 appBar: AppBar(  
 backgroundColor: Colors.*amber*,  
 title: Text(**'Icon Test!'**),  
 centerTitle: **true**,  
 ),  
 */\* body: Row(  
 mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
 crossAxisAlignment: CrossAxisAlignment.end,  
 children: <Widget>[  
 Text('Your'),  
 FlatButton(  
 onPressed: () {},  
 child: Text('Mom'),  
 color: Colors.orange,  
 ),  
 Container(  
 child: Text('Gay'),  
 color: Colors.cyan,  
 padding: EdgeInsets.all(20.0),  
 )  
 ],  
 ),\*/  
 /\*body: Column(  
 mainAxisAlignment: MainAxisAlignment.center,  
 crossAxisAlignment: CrossAxisAlignment.end,  
 children: <Widget>[  
 Container(  
 child: Text('Your'),  
 color: Colors.cyan,  
 padding: EdgeInsets.all(20.0),  
 ),  
 Container(  
 child: Text('Mom'),  
 color: Colors.pinkAccent,  
 padding: EdgeInsets.all(30.0),  
 ),  
 Container(  
 child: Text('Gay'),  
 color: Colors.orange,  
 padding: EdgeInsets.all(40.0),  
 )  
 ],  
 ),  
 \*/* body: Center(  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.**spaceEvenly**,  
 crossAxisAlignment: CrossAxisAlignment.**center**,  
 children: <Widget>[  
 Row(  
 mainAxisAlignment: MainAxisAlignment.**center**,  
 crossAxisAlignment: CrossAxisAlignment.**center**,  
 children: <Widget>[  
 IconButton(  
 onPressed: (){  
 print(**'You pressed the icon button'**);  
 Fluttertoast.*showToast*(  
 msg: **"Fuck off 😾"**,  
 toastLength: Toast.**LENGTH\_SHORT**,  
 gravity: ToastGravity.**BOTTOM**,  
 timeInSecForIosWeb: 1,  
 backgroundColor: Colors.*grey*,  
 textColor: Colors.*white*,  
 fontSize: 16.0  
 );  
 },  
 icon: Icon(  
 Icons.*pets* ),  
 color: Colors.*purple*,  
 iconSize: 40.0,  
 )  
 ],  
 ),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.**spaceEvenly**,  
 crossAxisAlignment: CrossAxisAlignment.**center**,  
 children: <Widget>[  
 RaisedButton(  
 onPressed: () {  
 print(**'You clicked ze Raised button'**);  
 Fluttertoast.*showToast*(  
 msg: **"Please 😾"**,  
 toastLength: Toast.**LENGTH\_SHORT**,  
 gravity: ToastGravity.**BOTTOM**,  
 timeInSecForIosWeb: 1,  
 backgroundColor: Colors.*grey*,  
 textColor: Colors.*white*,  
 fontSize: 16.0  
 );  
 },  
 color: Colors.*cyan*,  
 child: Text(**'Raised Button!'**),  
 ),  
 Container(  
 child: Text(**'Your'**),  
 color: Colors.*cyan*,  
 padding: EdgeInsets.all(20.0),  
 ),  
 Icon(  
 Icons.*touch\_app*,  
 color: Colors.*cyan*,  
 size: 50.0,  
 )  
 ],  
 ),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.**spaceEvenly**,  
 crossAxisAlignment: CrossAxisAlignment.**center**,  
 children: <Widget>[  
 FlatButton(  
 onPressed: () {  
 print(**'You clicked ze Flat button'**);  
 Fluttertoast.*showToast*(  
 msg: **"Fuck 😾"**,  
 toastLength: Toast.**LENGTH\_SHORT**,  
 gravity: ToastGravity.**BOTTOM**,  
 timeInSecForIosWeb: 1,  
 backgroundColor: Colors.*grey*,  
 textColor: Colors.*white*,  
 fontSize: 16.0  
 );  
 },  
 color: Colors.*pinkAccent*,  
 child: Text(**'Flat Button!'**),  
 ),  
 Container(  
 child: Text(**'Mom'**),  
 color: Colors.*pinkAccent*,  
 padding: EdgeInsets.all(30.0),  
 ),  
 Icon(  
 Icons.*face*,  
 color: Colors.*pinkAccent*,  
 size: 50.0,  
 )  
 ],  
 ),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.**spaceEvenly**,  
 crossAxisAlignment: CrossAxisAlignment.**center**,  
 children: <Widget>[  
 RaisedButton.icon(  
 onPressed: (){  
 print(**'You pressed ze weird button!'**);  
 Fluttertoast.*showToast*(  
 msg: **"Off 😾"**,  
 toastLength: Toast.**LENGTH\_SHORT**,  
 gravity: ToastGravity.**BOTTOM**,  
 timeInSecForIosWeb: 1,  
 backgroundColor: Colors.*grey*,  
 textColor: Colors.*white*,  
 fontSize: 16.0  
 );  
 },  
 icon: Icon(  
 Icons.*supervised\_user\_circle* ),  
 label: Text(**'Button with Icon!'**),  
 color: Colors.*orange*,  
 ),  
 Container(  
 child: Text(**'Gay'**),  
 color: Colors.*orange*,  
 padding: EdgeInsets.all(40.0),  
 ),  
 Icon(  
 Icons.*loyalty*,  
 color: Colors.*orange*,  
 size: 50.0,  
 )  
 ],  
 )  
  
 ],  
 ),  
 ),  
 floatingActionButton: FloatingActionButton(  
 onPressed: () {  
 Fluttertoast.*showToast*(  
 msg: **"Yeah, you heard the cat! 😾"**,  
 toastLength: Toast.**LENGTH\_SHORT**,  
 gravity: ToastGravity.**BOTTOM**,  
 timeInSecForIosWeb: 1,  
 backgroundColor: Colors.*grey*,  
 textColor: Colors.*white*,  
 fontSize: 16.0  
 );  
  
 },  
 backgroundColor: Colors.*amber*,  
 child: Text(**'Click!'**),  
 ),  
 );  
 }  
}



Expanded Widgets:

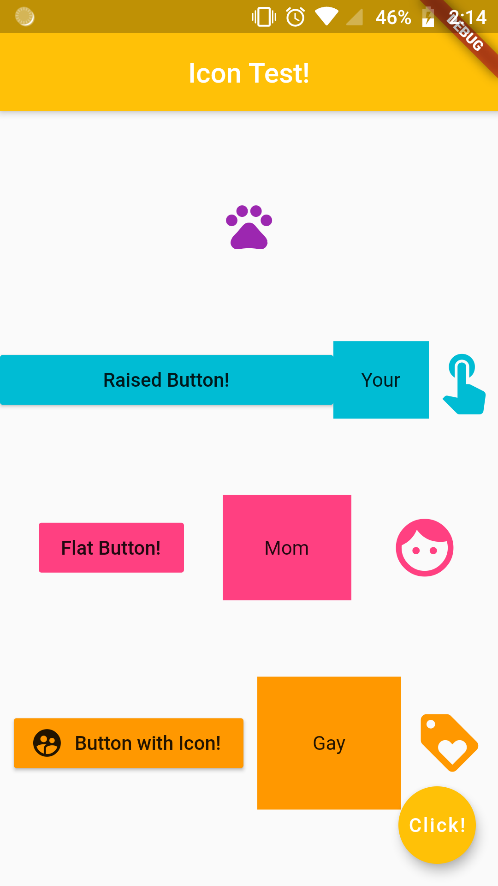
We’ve seen containers, rows and columns as types of layout widgets. Let’s explore another layout widget called the ‘Expanded’ widget.

In any row or column, the widgets that we add take up only that amount of space which is equal to their size. But what if we wanted them to take up the entire row? By doing so, we ensure that the widgets are packed together like sardines on the row or column.

We do that by enclosing these widgets inside another widget called ‘Expanded’. This is a wrapper widget that expands the child widget in such a way that the entire row is occupied by all the widgets in it. If there are other widgets in the row, then they are pushed to the end, and the current widget is expanded to fill in the row/column.

As an example, let’s wrap the raised button in our app inside the expanded widget.

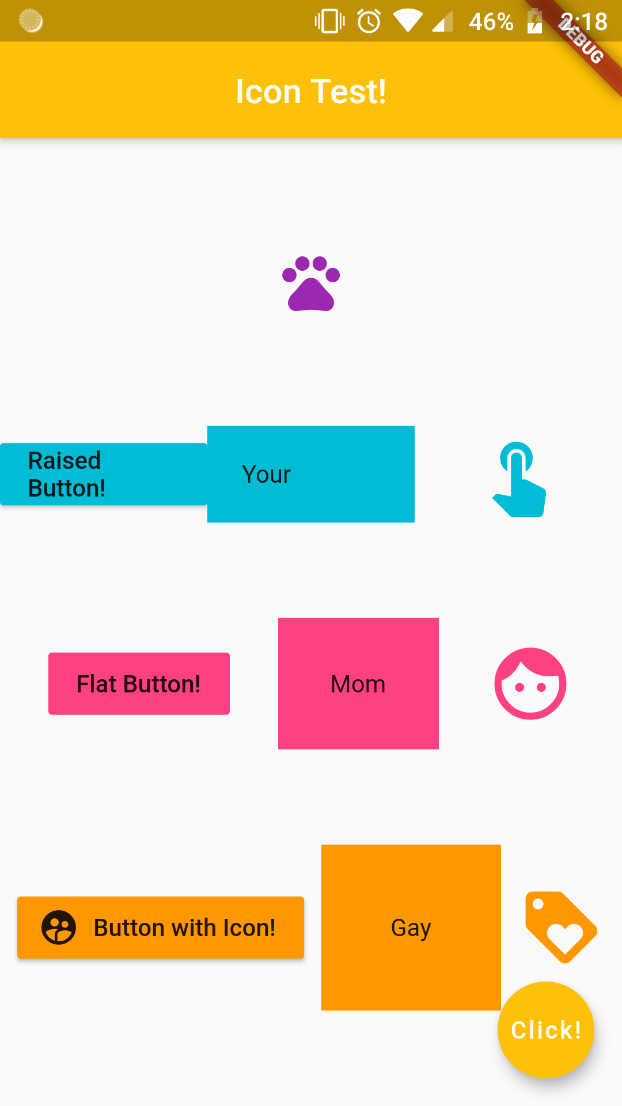
Row(  
 mainAxisAlignment: MainAxisAlignment.**spaceEvenly**,  
 crossAxisAlignment: CrossAxisAlignment.**center**,  
 children: <Widget>[  
 Expanded(  
 child: RaisedButton(  
 onPressed: () {  
 print(**'You clicked ze Raised button'**);  
 Fluttertoast.*showToast*(  
 msg: **"Please 😾"**,  
 toastLength: Toast.**LENGTH\_SHORT**,  
 gravity: ToastGravity.**BOTTOM**,  
 timeInSecForIosWeb: 1,  
 backgroundColor: Colors.*grey*,  
 textColor: Colors.*white*,  
 fontSize: 16.0  
 );  
 },  
 color: Colors.*cyan*,  
 child: Text(**'Raised Button!'**),  
 ),  
 ),  
 Container(  
 child: Text(**'Your'**),  
 color: Colors.*cyan*,  
 padding: EdgeInsets.all(20.0),  
 ),  
 Icon(  
 Icons.*touch\_app*,  
 color: Colors.*cyan*,  
 size: 50.0,  
 )  
 ],  
),



Let’s wrap the container as well as the icon inside the expanded widget also!

Expanded(  
 child: RaisedButton(  
 onPressed: () {  
 print(**'You clicked ze Raised button'**);  
 Fluttertoast.*showToast*(  
 msg: **"Please 😾"**,  
 toastLength: Toast.**LENGTH\_SHORT**,  
 gravity: ToastGravity.**BOTTOM**,  
 timeInSecForIosWeb: 1,  
 backgroundColor: Colors.*grey*,  
 textColor: Colors.*white*,  
 fontSize: 16.0  
 );  
 },  
 color: Colors.*cyan*,  
 child: Text(**'Raised Button!'**),  
 ),  
 ),  
   
 Expanded(  
 child: Container(  
 child: Text(**'Your'**),  
 color: Colors.*cyan*,  
 padding: EdgeInsets.all(20.0),  
 ),  
 ),  
   
 Expanded(  
 child: Icon(  
 Icons.*touch\_app*,  
 color: Colors.*cyan*,  
 size: 50.0,  
 ),  
 )  
 ],  
),

The output looks like:



There is also another interesting property of the expanded widget called as ‘Flex’. The flex value is used to divide the width of the row between the widgets that occupy it. For example, if we include all the widgets in a row inside an Expanded widget, we can assign flex values to all these Expanded widgets. The flex value is a representative fraction of the width. For example, if we have flex values as:

3,2,1 for widgets 1,2 & 3.

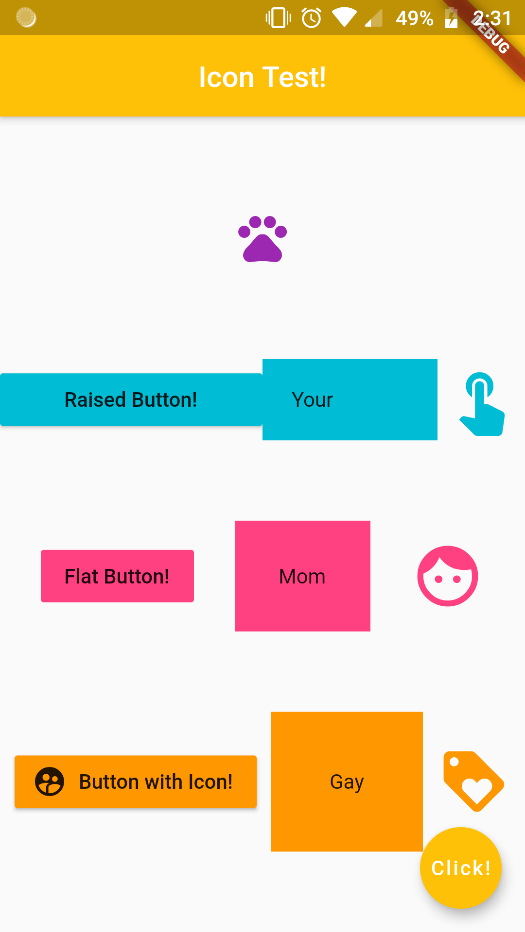
Then we can say that widget 1 occupies 3/3+2+1 th of the width of the row. Or ½ of the row.

Similarly, widget 2 occupies 2/3+2+1 th of the width of the row. Or 1/3rd of the row.

And finally widget 3 occupies 1/6th of the width of the row.

Placing these values in our code:

Row(  
 mainAxisAlignment: MainAxisAlignment.**spaceEvenly**,  
 crossAxisAlignment: CrossAxisAlignment.**center**,  
 children: <Widget>[  
 Expanded(  
 flex:3,  
 child: RaisedButton(  
 onPressed: () {  
 print(**'You clicked ze Raised button'**);  
 Fluttertoast.*showToast*(  
 msg: **"Please 😾"**,  
 toastLength: Toast.**LENGTH\_SHORT**,  
 gravity: ToastGravity.**BOTTOM**,  
 timeInSecForIosWeb: 1,  
 backgroundColor: Colors.*grey*,  
 textColor: Colors.*white*,  
 fontSize: 16.0  
 );  
 },  
 color: Colors.*cyan*,  
 child: Text(**'Raised Button!'**),  
 ),  
 ),  
   
 Expanded(  
 flex: 2,  
 child: Container(  
 child: Text(**'Your'**),  
 color: Colors.*cyan*,  
 padding: EdgeInsets.all(20.0),  
 ),  
 ),  
   
 Expanded(  
 flex: 1,  
 child: Icon(  
 Icons.*touch\_app*,  
 color: Colors.*cyan*,  
 size: 50.0,  
 ),  
 )  
 ],  
),



A good use case of expanded widgets is when we want to contain images within the boundaries of a widget. It is a helpful tool to resize large images where we want to position them inside closed boundaries.