**A visual guide to Input Decorations for Flutter TextField**

*A thousand pictures is worth a word*



Consider this a visual supplement the InputDecoration [documentation](https://api.flutter.dev/flutter/material/InputDecoration-class.html). Sometimes it’s easier to just see and understand rather than reading a long description.

**No decoration**

TextField(),



**Icon**

TextField(  
 decoration: InputDecoration(  
 icon: Icon(Icons.star),  
 ),  
),



**Prefix icon**

TextField(  
 decoration: InputDecoration(  
 prefixIcon: Icon(Icons.star),  
 ),  
),



**Suffix icon**

TextField(  
 decoration: InputDecoration(  
 suffixIcon: Icon(Icons.star),  
 ),  
),



**Prefix**

This can be any widget.

TextField(  
 decoration: InputDecoration(  
 prefix: Container(width: 10, height: 10, color: Colors.red,)  
 ),  
),



**Prefix text**

TextField(  
 decoration: InputDecoration(  
 prefixText: 'Hello'  
 ),  
),



**Hint text**

TextField(  
 decoration: InputDecoration(  
 hintText: 'Hello',  
 ),  
),



**Suffix text**

TextField(  
 decoration: InputDecoration(  
 suffixText: 'Hello',  
 ),  
),



**Label text**

TextField(  
 decoration: InputDecoration(  
 labelText: 'Hello',  
 ),  
),

**Unfocused**

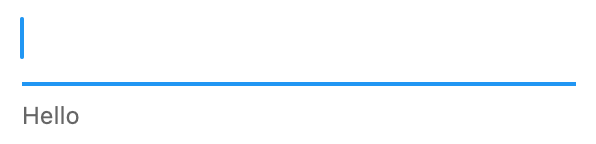


**Focused**



**Helper text**

TextField(  
 decoration: InputDecoration(  
 helperText: 'Hello',  
 ),  
),



**Error text**

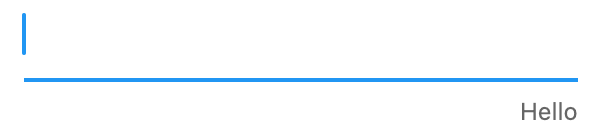
TextField(  
 decoration: InputDecoration(  
 errorText: 'Hello',  
 ),  
),



**Counter text**

TextField(  
 decoration: InputDecoration(  
 counterText: 'Hello',  
 ),  
),

You could replace that text with a character count and rebuild when the input changes.

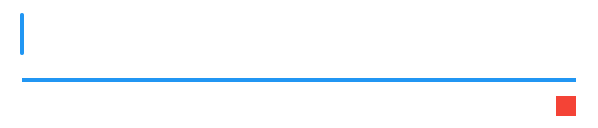


**Counter**

This could be any widget.

TextField(  
 decoration: InputDecoration(  
 counter: Container(width: 10, height: 10, color: Colors.red,)  
 ),  
),

You could make the widget change based on the number of characters that have been entered.



**Style**

This example shows hintStyle, but it works the same to set a TextStyle for labelStyle, counterStyle, errorStyle, prefixStyle, suffixStyle, and helperStyle.

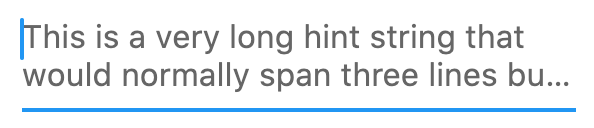
TextField(  
 decoration: InputDecoration(  
 hintText: 'Hello',  
 hintStyle: TextStyle(color: Colors.green),  
 ),  
),



**Max lines**

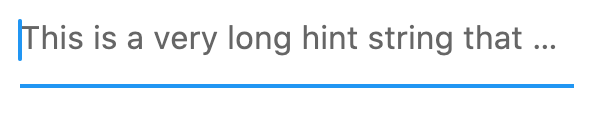
The example here shows hintMaxLines, but helperMaxLines and errorMaxLines work similarly.

TextField(  
 decoration: InputDecoration(  
 hintMaxLines: 2,  
 hintText: 'This is a very long hint string that would normally span three lines but it cannot.',  
 ),  
),



The default is one line:

TextField(  
 decoration: InputDecoration(  
 hintText: 'This is a very long hint string that would normally span three lines but it cannot.',  
 ),  
),



**Hint text direction**

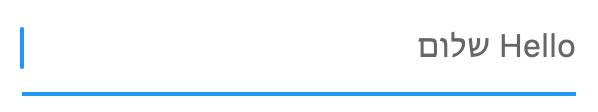
**Left-to-right**

TextField(  
 decoration: InputDecoration(  
 hintText: 'Hello שלום',  
 hintTextDirection: TextDirection.ltr,  
 ),  
),



**Right-to-left**

TextField(  
 decoration: InputDecoration(  
 hintText: 'Hello שלום',  
 hintTextDirection: TextDirection.rtl,  
 ),  
),



**Floating label behavior**

**Auto**

TextField(  
 decoration: InputDecoration(  
 labelText: 'Hello',  
 floatingLabelBehavior: FloatingLabelBehavior.auto,  
 ),  
),



**Always**

TextField(  
 decoration: InputDecoration(  
 labelText: 'Hello',  
 floatingLabelBehavior: FloatingLabelBehavior.always,  
 ),  
),



**Never**

TextField(  
 decoration: InputDecoration(  
 labelText: 'Hello',  
 floatingLabelBehavior: FloatingLabelBehavior.never,  
 ),  
),



**Border**

**None**

TextField(  
 decoration: InputDecoration(  
 border: InputBorder.none,  
 ),  
),



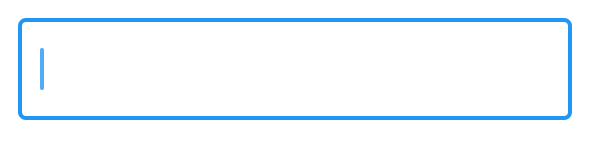
**Underline**

TextField(  
 decoration: InputDecoration(  
 border: UnderlineInputBorder(),  
 ),  
),



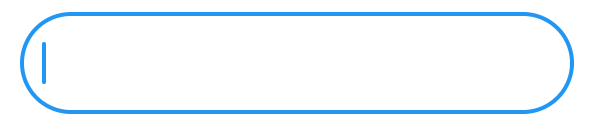
**Outline**

TextField(  
 decoration: InputDecoration(  
 border: OutlineInputBorder(),  
 ),  
),



**Outline with border radius**

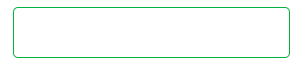
TextField(  
 decoration: InputDecoration(  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(30),  
 ),  
 ),  
),



**Specific border behavior**

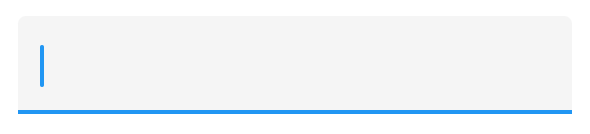
The border is the general case, but you can also modify that for specific situations by setting focusedBorder, enabledBorder, disabledBorder, errorBorder, and focusedErrorBorder.

TextField(  
 decoration: InputDecoration(  
 enabledBorder: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(5),  
 borderSide: BorderSide(  
 color: Colors.green,  
 width: 1.0,  
 ),  
 ),  
 focusedBorder: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(30),  
 borderSide: BorderSide(  
 color: Colors.purple,  
 width: 2.0,  
 ),  
 ),  
 ),  
),



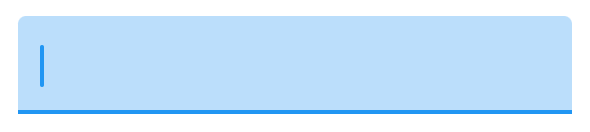
**Filled**

TextField(  
 decoration: InputDecoration(  
 filled: true,  
 ),  
),



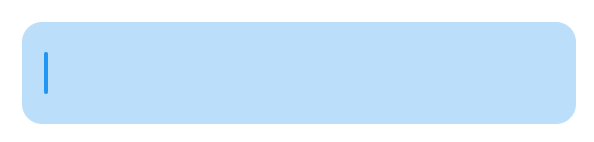
**Color**

TextField(  
 decoration: InputDecoration(  
 filled: true,  
 fillColor: Colors.blue.shade100,  
 ),  
),



**Color and border**

TextField(  
 decoration: InputDecoration(  
 filled: true,  
 fillColor: Colors.blue.shade100,  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(10),  
 borderSide: BorderSide.none,  
 )  
 ),  
),



**Hover color**

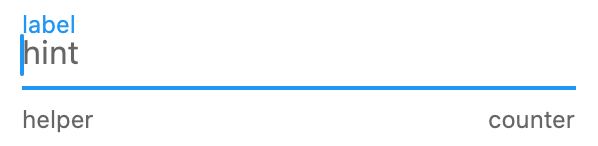
TextField(  
 decoration: InputDecoration(  
 filled: true,  
 hoverColor: Colors.blue.shade100,  
 border: OutlineInputBorder(),  
 ),  
),



**Enabled**

**true**

TextField(  
 decoration: InputDecoration(  
 enabled: true,  
 hintText: 'hint',  
 helperText: 'helper',  
 labelText: 'label',  
 counterText: 'counter'  
 ),  
),



**false**

TextField(  
 decoration: InputDecoration(  
 enabled: false,  
 hintText: 'hint',  
 helperText: 'helper',  
 labelText: 'label',  
 counterText: 'counter'  
 ),  
),



**Condensed**

The yellow container is there to show you the size of the TextField.

**false**

Container(  
 color: Colors.yellow,  
 child: TextField(  
 decoration: InputDecoration(  
 isCollapsed: false,  
 hintText: 'hello',  
 ),  
 ),  
),



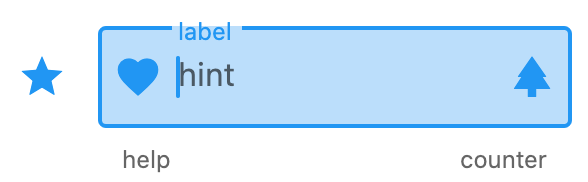
**true**

color: Colors.yellow,  
 child: TextField(  
 decoration: InputDecoration(  
 isCollapsed: true,  
 hintText: 'hello',  
 ),  
 ),  
),



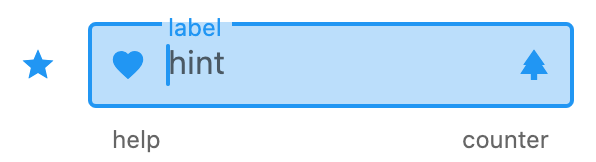
**Combined effects**

TextField(  
 decoration: InputDecoration(  
 filled: true,  
 fillColor: Colors.blue.shade100,  
 border: OutlineInputBorder(),  
 labelText: 'label',  
 hintText: 'hint',  
 helperText: 'help',  
 counterText: 'counter',  
 icon: Icon(Icons.star),  
 prefixIcon: Icon(Icons.favorite),  
 suffixIcon: Icon(Icons.park),  
 ),  
),



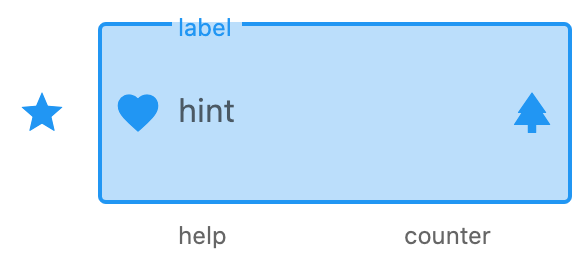
**Dense**

TextField(  
 decoration: InputDecoration(  
 **isDense: true,**  
 filled: true,  
 fillColor: Colors.blue.shade100,  
 border: OutlineInputBorder(),  
 labelText: 'label',  
 hintText: 'hint',  
 helperText: 'help',  
 counterText: 'counter',  
 icon: Icon(Icons.star),  
 prefixIcon: Icon(Icons.favorite),  
 suffixIcon: Icon(Icons.park),  
 ),  
),



**Content padding**

TextField(  
 decoration: InputDecoration(  
 **contentPadding: EdgeInsets.all(40),**  
 filled: true,  
 fillColor: Colors.blue.shade100,  
 border: OutlineInputBorder(),  
 labelText: 'label',  
 hintText: 'hint',  
 helperText: 'help',  
 counterText: 'counter',  
 icon: Icon(Icons.star),  
 prefixIcon: Icon(Icons.favorite),  
 suffixIcon: Icon(Icons.park),  
 ),  
),



**Mystery**

I couldn’t figure these two out:

* alignLabelWithHint
* focusColor

Feel free to leave a comment.