Widgets - Cheat Sheet

A Quick Overview

What's a "Widget"?

A building block with which you compose your app's user interfaces.
Flutter has no drag-and-drop editor or complex markup language, instead you work with widgets in your Dart code to compose the UI you want to present.

You work with both built-in widgets (e.g. `RaisedButton()` to show a button) as well as custom widgets which in the are composed of built-in widgets (to build your app's pages or more complex widget combinations which you then can also re-use).

Example:

```
class MyCustomButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Column(children: [
      Text('Tap to do something amazing!'),
      RaisedButton(child: Text('Tap!'), onPressed: () {}),
    ]);
  }
}
```

(Re-)use this widget wherever you want a button with a text above it.

Building a Widget Tree

Since you nest widgets into each other (by passing widgets to widgets through constructor arguments), you typically build your UI via a so-called "widget tree").

In the example above, the `Column()` widget holds two child/ nested widgets for example: `Text()` and `RaisedButton()`. And the `RaisedButton()` holds yet another child widget: Another `Text()` widget.

The build() Method

Each (custom) widget you build must have a `build()` method - `StatelessWidget` / `StatefulWidget` which you have to extend forces you to include such a method.

Flutter calls the `build()` method when drawing your widgets onto the app's screen. The `build()` method returns a widget => The content Flutter should draw onto the screen.

Stateless vs Stateful

Flutter has two types of widgets you can build.

Widgets which extend `StatelessWidget` are meant to present data. They return a widget (tree) which simply outputs some data. You can't change internal data (i.e. class properties) in your widget's class and re-render the UI upon such a change.

Widgets which extend `StatefulWidget` also return a widget (tree) but there, you can also change some internal data (class properties) in a way such that the user interface is re-built - i.e. your changes are reflected in your app's user interface.

Official Docs & Widget Overview

Flutter has a lot of built-in widgets which you can freely use, configure and combine. You also often will have multiple widgets that do almost the same thing / produce almost the same output, depending on how you configure it. This flexibility is normal in Flutter and whilst it can be irritating at first, it allows you to build any user interface you want, since you'll not face any restrictions.

The official widget catalog is a great place to get an overview (with categories) of the important widgets Flutter provides: https://flutter.dev/docs/development/ui/widgets