Start thinking declaratively

If you're coming to Flutter from an imperative framework (such as Android SDK or iOS UIKit), you need to start thinking about app development from a new perspective.

Many assumptions that you might have don't apply to Flutter. For example, in Flutter it's okay to rebuild parts of your UI from scra instead of modifying it. Flutter is fast enough to do that, even on every frame if needed.

Flutter is *declarative*. This means that Flutter builds its user interface to reflect the current state of your app:

When the state of your app changes (for example, the user flips a switch in the settings screen), you change the state, and that triggers a redraw of the user interface. There is no imperative changing of the UI itself (like `widget.setText`)—you change the sta and the UI rebuilds from scratch.

Read more about the declarative approach to UI programming in the get started guide.

The declarative style of UI programming has many benefits. Remarkably, there is only one code path for any state of the UI. You describe what the UI should look like for any given state, once—and that is it.

At first, this style of programming might not seem as intuitive as the imperative style. This is why this section is here. Read on.