For the framework

Dart tests are written using the `flutter_test` package's API, named with the suffix `_test.dart`, and placed inside the `test/` subdirectory of the package under test.

We support several kinds of tests:

- Unit tests, e.g. using `flutter_test`. See below.
- Unit tests that use golden-file testing, comparing pixels. See [[Writing a golden-file test for package:flutter]].
- End-to-end tests, e.g. using `flutter_driver` and our device lab.

Our bots run on our test and build infrastructure.

Running unit tests

Flutter tests use the `flutter_test` package (source, API documentation), which provides flutter-specific extensions on top of the Dart `test` package.

To automatically find all files named `*_test.dart` inside a package's `test/` subdirectory, and run them inside the headless flutter shell as a test, use the `flutter test` command, e.g:

- `cd examples/hello_world`
- `flutter test`

Individual tests can also be run directly, e.g.: `flutter test lib/my_app_test.dart`

You can view these tests on a device by running them directly using `flutter run`. For tests inside the `packages/flutter` directory, you will need to copy them to (or symlink to them from) the `test/` directory of an actual app (e.g. the flutter gallery), since the `flutter` package itself is not set up to execute as an application (which is necessary to use `flutter run` with a test).

Unit tests run with `flutter test` run inside a headless flutter shell on your workstation, you won't see any UI. You can use `print` to generate console output or you can interact with the Dart VM via the Dart Observatory at http://localhost:8181/.

To debug tests in Observatory, use the `--start-paused` option to start the test in a paused state and wait for connection from a debugger. This option lets you set breakpoints before the test runs.

To run analysis and all the tests for the entire Flutter repository, the same way that Cirrus runs them, run `dart dev/bots/test.dart` and `dart dev/bots/analyze.dart`.

If you've built your own flutter engine (see [[Setting up the Engine development environment]]), you can pass `--local-engine` to change what flutter shell `flutter test` uses. For example, if you built an engine in the `out/host_debug_unopt` directory, you can pass `--local-engine=host_debug_unopt` to run the tests in that engine.

To learn how to see how well tested the codebase is, see [[Test coverage for package:flutter]].

Running device lab tests locally

Flutter runs a number of end-to-end tests in a device lab. The Flutter repo contains code for bootstrapping and executing these tests, in addition to the tests themselves.

The code that runs the device lab end-to-end tests can be found here:

```
dev/devicelab
```

The tests that run in the device lab can be found here:

```
dev/integration_tests
```

When a device lab test fails, it is important to be able to run the test locally to verify the problem and intended solution. To execute a device lab test locally, do the following:

1. Navigate in your terminal to the `dev/devicelab` directory.
2. Ensure that a physical device, simulator, or emulator is connected.
3. Ensure that the current locale is en_US by executing the command: `export LANG=en_US.UTF-8`.
4. Execute the command: `../../bin/cache/dart-sdk/bin/dart bin/run.dart -t [name_of_test]` where `[name_of_test]` is replaced by the name of the test you want to run as defined within `dev/devicelab/manifest.yaml`.

Device lab tests with a local engine

Sometimes a device lab test fails due to engine changes that you've made. In these cases, you'd like to run the impacted device lab tests locally with your local version of the engine. To do this, pass the appropriate flags to `run.dart`:

```
../../bin/cache/dart-sdk/bin/dart bin/run.dart \
  --local-engine-src-path=[path_to_src] \
  --local-engine=[engine_build_for_your_device] \
  -t [name_of_test]
```

If your local Flutter engine is in the same directory as your `flutter/` directory then you can omit the `--local-engine-src-path` parameter because it will be resolved automatically:

```
../../bin/cache/dart-sdk/bin/dart bin/run.dart \
  --local-engine=[engine_build_for_your_device] \
  -t [name_of_test]
```

The following is an example of what running the local engine command might look like:

```
../../bin/cache/dart-sdk/bin/dart bin/run.dart \
  --local-engine-src-path=/Users/myname/flutter/engine/src \
  --local-engine=android_debug_unopt_x86 \
  -t external_ui_integration_test
```

The above command would use the local Flutter engine located at `/Users/myname/flutter/engine` to execute the `external_ui_integration_test` test on an Android emulator, which is why the `android_debug_unopt_x86` version of the engine is used.

For the engine

See the [[Testing the engine]] wiki.