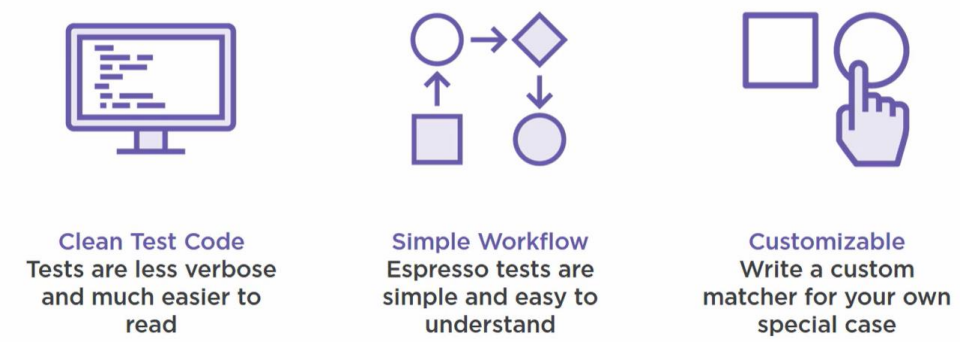
Espresso Fundamentals

Espresso is a user interface testing framework developed by Google for android.

In android, there are two types of tests we can write:
1. Unit test — It can be achieved by the JUnit framework.
2. Instrumentation test or android test — Espresso comes under this type there are other frameworks like Appium, Robolectric.

What does Espresso offer us?



Expresso Workflow

Overview of applying our workflow

```
onView( < matcher > )
    .perform( < action > );

onView( < matcher > )
    .check( < assertion < matcher > > );
```

Example

Let’s say we have an application that has button like this:

When we click our button, we want textView to look like this:

This is how we test it with Espresso

```
onView(withId(R.id.button))
    .perform(click());

onView(withId(R.id.results_textview))
    .check(matches(withText("The button was clicked!")));
```

Espresso Dependencies

First thing first, take a look at this

```
dependencies {
    testCompile 'junit:junit:4.12'
    androidTestCompile 'com.android.support.test:runner:0.5'
}
```

TestCompile-> compile for local tests only

AndroidTestCompile-> compile for Android (instrumented) tests only

We will be using both junits tests and apis provided by android support.

Now, we can use following **Espresso Dependencies depending on usage**

```
androidTestCompile ...
    'com.android.support.test.espresso:espresso-core:2.2.2'
    'com.android.support.test.espresso:espresso-contrib:2.2.2'
    'com.android.support.test.espresso:espresso-web:2.2.2'
    'com.android.support.test.espresso:espresso-idling-resources:2.2.2'
    'com.android.support.test.espresso:espresso-intents:2.2.2'
```

- If you want to use recycler view actions, drawer actions or accessibility checks, include espresso-contrib dependency
- Include espresso-web if you need WebView support in your tests
- For synchronizing background tasks, include espresso-idling-resources
- If you'll be validating or stubbing intents, include espresso-intents

Running Tests in Android Studio

Local Unit Test

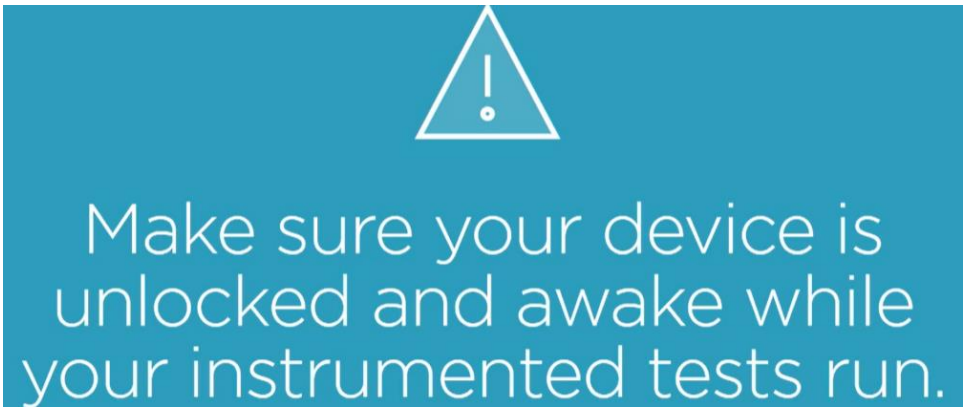
The local unit tests should go under this directory

Local unit tests are simple unit tests that run on JVM. They have no Android dependency, so that are run without Android Emulator/Device

Instrumented Unit Tests

The instrumented unit tests go under this directory

This is where our UI tests go. These unit tests rely on components of Android Framework. So, they do require an Android Emulator/Device to run. These can take a lot of time to run


Make sure your device is
unlocked and awake while
your instrumented tests run.