Material and Cupertino Libraries Localizations

The `.arb` files in this directory contain localized values (primarily strings) used by the Material and Cupertino libraries. The `generated_material_localizations.dart` and `generated_cupertino_localizations.dart` files combine all of the localizations into a single Map that is linked with the rest of flutter_localizations package.

If you're looking for information about internationalizing Flutter apps in general, see the Internationalizing Flutter Apps tutorial.

Translations for one locale: .arb files

The Material and Cupertino libraries use Application Resource Bundle files, which have a `.arb` extension, to store localized translations of messages, format strings, and other values. This format is also used by the Dart intl package and it is supported by the Google Translators Toolkit.

The Material and Cupertino libraries only depend on a small subset of the ARB format. Each .arb file contains a single JSON table that maps from resource IDs to localized values.

Filenames contain the locale that the values have been translated for. For example `material_de.arb` contains German translations, and `material_ar.arb` contains Arabic translations. Files that contain regional translations have names that include the locale's regional suffix. For example `material_en_GB.arb` contains additional English translations that are specific to Great Britain.

There is one language-specific .arb file for each supported locale. If an additional file with a regional suffix is present, the regional localizations are automatically merged with the language-specific ones.

The JSON table's keys, called resource IDs, are valid Dart variable names. They correspond to methods from the `MaterialLocalizations` or `CupertinoLocalizations` classes. For example:

```
Widget build(BuildContext context) {
  return FlatButton(
    child: Text(
      MaterialLocalizations.of(context).cancelButtonLabel,
    ),
  );
}
```

This widget build method creates a button whose label is the local translation of "CANCEL" which is defined for the `cancelButtonLabel` resource ID.

Each of the language-specific .arb files contains an entry for `cancelButtonLabel`. They're all represented by the `Map` in the generated `localizations.dart` file. The Map is used by the MaterialLocalizations class.

material_en.arb and cupertino_en.arb Define all of the resource IDs

All of the `material_*.arb` files whose names do not include a regional suffix contain translations for the same set of resource IDs as `material_en.arb`.

Similarly all of the `cupertino_*.arb` files whose names do not include a regional suffix contain translations for the same set of resource IDs as `cupertino_en.arb`.

For each resource ID defined for English, there is an additional resource with an '@' prefix. These '@' resources are not used by the generated Dart code at run time, they just exist to inform translators about how the value will be used, and to inform the code generator about what code to write.

```
"cancelButtonLabel": "CANCEL",
"@cancelButtonLabel": {
  "description": "The label for cancel buttons and menu items.",
  "type": "text"
},
```

Values with Parameters, Plurals

A few of material translations contain `$variable` tokens. The Material and Cupertino libraries replace these tokens with values at run-time. For example:

```
"aboutListTileTitle": "About $applicationName",
```

The value for this resource ID is retrieved with a parameterized method instead of a simple getter:

```
MaterialLocalizations.of(context).aboutListTileTitle(yourAppTitle)
```

The names of the `$variable` tokens must match the names of the `MaterialLocalizations` method parameters.

Plurals are handled similarly, with a lookup method that includes a quantity parameter. For example `selectedRowCountTitle` returns a string like "1 item selected" or "no items selected".

```
MaterialLocalizations.of(context).selectedRowCountTitle(yourRowCount)
```

Plural translations can be provided for several quantities: 0, 1, 2, "few", "many", "other". The variations are identified by a resource ID suffix which must be one of "Zero", "One", "Two", "Few", "Many", "Other". The "Other" variation is used when none of the other quantities apply. All plural resources must include a resource with the "Other" suffix. For example the English translations ('material_en.arb') for `selectedRowCountTitle` are:

```
"selectedRowCountTitleZero": "No items selected",
"selectedRowCountTitleOne": "1 item selected",
"selectedRowCountTitleOther": "$selectedRowCount items selected",
```

When defining new resources that handle pluralizations, the "One" and the "Other" forms must, at minimum, always be defined in the source English ARB files.

scriptCategory and timeOfDayFormat for Material library

In `material_en.arb`, the values of these resource IDs are not translations, they're keywords that help define an app's text theme and time picker layout respectively.

The value of `timeOfDayFormat` defines how a time picker displayed by showTimePicker() formats and lays out its time controls. The value of `timeOfDayFormat` must be a string that matches one of the formats defined by https://api.flutter.dev/flutter/material/TimeOfDayFormat-class.html. It is converted to an enum value because the `material_en.arb` file has this value labeled as `"x-flutter-type": "icuShortTimePattern"`.

The value of `scriptCategory` is based on the Language categories reference section in the Material spec. The Material theme uses the `scriptCategory` value to lookup a localized version of the default `TextTheme`, see Typography.geometryThemeFor.

'generated_*_localizations.dart': all of the localizations as a Map

If you look at the comment at the top of the `generated_material_localizations.dart` and `generated_cupertino_localizations.dart` files, you'll see that it was manually generated using a `dev/tools/localizations` app called `gen_localizations`.

You can see what that script would generate by running this command:

```
dart dev/tools/localizations/bin/gen_localizations.dart packages/flutter_localizations/lib/s
```

The gen_localizations app just combines the contents of all of the .arb files into a single `Map` per library that has entries for each .arb file's locale. The `MaterialLocalizations` and `CupertinoLocalizations` class implementations use these Maps to implement the methods that lookup localized resource values.

The gen_localizations app must be run by hand after .arb files have been updated. The app's first parameter is the path to this directory, the second is the file name prefix (the file name less the locale suffix) for the .arb files in this directory.

To in-place update the generated localizations file using the default values, you can just run:

```
dart dev/tools/localizations/bin/gen_localizations.dart --overwrite
```

Special handling for the Kannada (kn) translations

Originally, the cupertino_kn.arb and material_kn.arb files contained unicode characters that can cause current versions of Emacs on Linux to crash. There is

more information here: https://github.com/flutter/flutter/issues/36704.

Rather than risking developers' editor sessions, the strings in these arb files (and the code generated for them) have been encoded using the appropriate escapes for JSON and Dart. The JSON format arb files were rewritten with dev/tools/localization/bin/encode_kn_arb_files.dart. The localizations code generator uses generateEncodedString() from dev/tools/localization/localizations_utils.dart.

Support for Pashto (ps) translations

When Flutter first set up i18n for the Material library, Pashto (ps) translations were included for the first set of Material widgets. However, Pashto was never set up to be continuously maintained in Flutter by Google, so material_ps.arb was never updated beyond the initial commit.

To prevent breaking applications that rely on these original Pashto translations, they will be kept. However, all new strings will have the English translation until support for Pashto is provided. See https://github.com/flutter/flutter/issues/60598.

Translations Status, Reporting Errors

The translations (the `.arb` files) in this directory are based on the English translations in `material_en.arb` and `cupertino_en.arb`. Google contributes translations for all the languages supported by this package. (Googlers, for more details see .)

If you have feedback about the translations please file an issue on the Flutter github repo.

See Also

The Internationalizing Flutter Apps tutorial describes how to use the internationalization APIs in an ordinary Flutter app.

Application Resource Bundle covers the `.arb` file format used to store localized translations of messages, format strings, and other values.

The Dart intl package supports internationalization.