generate-locales-tool

Angular relies on CLDR for internationalization. The locale data is split up into the following filesets:

1. A default locale used within `@angular/core`. This is the `locale_en.ts` file.
2. Information about currencies with their symbols and fraction digits (based on the default locale). The base currency information resides in `@angular/common/src/i18n`.
3. Locale files containing date, plural, currency information and more.

This is a high-level overview of the tool and how it organizes/optimizes the locale data within Angular. In the following section the individual filesets are described in detail.

Default locale file in `@angular/core`

The default locale file resides within `@angular/core` and provides locale information for the `en` locale. The `en` locale is commonly used for local development. This default locale file does not differ from other locale files generated within `@angular/common/locales`, so the details are discussed below.

Base currency information within `@angular/common`

The `@angular/core` package is not providing directives or syntax for dealing with currencies. This is part of the `@angular/common` package and its i18n features (such as the `currency` pipe). Due to this separation of concerns, the default locale in `core` does not contain any information about currencies, and the currency information is stored as a separate file within `@angular/common/src/i18n`.

The currency information is stored within an object that maps each currency code (like `USD`) to an array describing the symbol, narrow symbol and number of digits to display for fractions. e.g.

```
const baseCurrencies = {
  'CAD': ['CA$', '$', 2],
  'NZD': ['NZ$', '$'],
  'USD': ['$'],
}
```

In the snippet above, you might wonder why values for `NZD` or `USD` are missing. This is intentional as for `USD` there is no narrow symbol (given `$` already being the symbol). The tool does not set an explicit value for byte savings. Same applies for the fraction digits.

Locale files

Aside from the default locale (`en`) which is generated separately, this tool generates locale files for all other locales the CLDR package provides data for. Locale files currently provide the following locale data:

- Date time settings (describes how dates are formatted, e.g. the locale-based label for Monday)
- Number settings (describing how numbers are formatted, e.g. what the plus symbol is)
- Currency settings (describing how to format currencies for current locale)
- Currency symbols based on the given locale. e.g. in `en-AU` `USD` is the currency symbol.
- Directionality of the locale (i.e. whether text is displayed in RTL or LTR)

All the data is stored in an array where the Angular framework can read specific data using the corresponding index (see `LocaleDataIndex` for details on how data is read).

Performance optimizations

Given we already have currency symbols for the default locale within `@angular/common/src/i18n`, we do not need to capture currencies that aren't different within a given locale. We just omit the data for such currencies and let the Angular framework fall back to the currency symbols from the default locale (this avoids unnecessary duplication).

Additionally, if locale data is equal to locale data at a previous index, then the generation tool will not repeat the data but instead set `undefined`. The Angular framework will look for the previous value in that case. This reduces the payload size for locales even further. For example:

```
[
  // ...
  labelsForDayPeriodsNarrow,
  labelsForDayPeriodsAbbreviated
]
```

If `labelsForDayPeriodsAbbreviated` for example is equal to `labelsForDayPeriodsNarrow`, then the tool will not set a value for the abbreviated labels. Instead, it will set `undefined` as that minifies to: `[labelsForDayPeriodsNarrow, undefined]`.

Where are the CLDR data files stored?

The CLDR data is fetched through a Bazel repository. See the project-level `WORKSPACE` file for more information. The generation tool resolves all locale data through the `@cldr_data` Bazel repository.

Locale files, or the default currencies file are not source files that would end up in the Git repository. This is different to the old Gulp setup where such files were added as part of the repository. The benefit of no longer adding such files to the Git repository is that updating CLDR is no longer resulting in large diffs. Another significant benefit is that files cannot be modified manually. Previously, developers could have manually made changes within locale files and no tooling would complain. This could result in locale files which are not in sync with CLDR / or files which are broken.