Intl.DisplayNames

There are times when our applications need to reference infoormation such as languages, regions, scripts, and currencies. Some of these may be easier and some may be hardm but either way we shouldn't hardcode them into the apps themselves, this makes them brittle and harder to localize.

Rather than download a fresh copy of the locale data every time there is a change, we can leverage the versions available in JavaScript runtimes. The `Intl.DisplayNames` API gives JavaScript developers direct access to those translations, allowing applications to more easily display localized names.

The following example creates four constants reflecting the languages, regions, scripts and currency as used in Chile (`es-cl` as the first parameter to `DisplayNames`).

```
const LanguageNames = new Intl.DisplayNames(['es-cl'], { type: 'language'
const RegionNames = new Intl.DisplayNames(['es-cl'], { type: 'region' });
const ScriptNames = new Intl.DisplayNames(['es-cl'], { type: 'script' });
const CurrencyNames = new Intl.DisplayNames(['es-cl'], {type: 'currency'}
```

We then use the constants to query the CLDR data based on our locale.

The values that we use for each type of query are different:

- If the type is "region", the *code* should be either an ISO-3166 two letters region code, or a three digits UN M49 Geographic Regions
- If the type is "script", the *code* should be an ISO-15924 four letters script code
- If the type is "language", the *code* should be a *languageCode* ["-" *scriptCode*] ["-" *regionCode*] *("-" *variant*) subsequence of the unicode_language_id grammar in UTS 35's Unicode Language and Locale Identifiers grammar. *languageCode* is either a two letters ISO 639-1 language code or a three letters ISO 639-2 language code.
- If the type is "currency", the *code* should be a 3-letter ISO 4217 currency code

The first group of examples query the names of the langaugues.

```
LanguageNames.of('fr');
"francés"

LanguageNames.of('es-es');
"español de España"

LanguageNames.of('JPN');
"japonés"

LanguageNames.of('zho');
"chino"
```

Same with name of regions and countries. There are some regions that don't appear in the listings for region names and where they appear they don't work with the code.

```
RegionNames.of('UK');
"Reino Unido"

RegionNames.of('DE');
"Alemania"

RegionNames.of('EU');
"Unión Europea"
```

The currency values are, to me the most confusing as they use only the first two letters of the country and then the name of the currency itself, which is not an intuitive name if you don't look at the table.

```
CurrencyNames.of('JPY');
"yen"

CurrencyNames.of('clp');
"Peso Chileno"

CurrencyNames.of('GBP')
```

```
"libra esterlina"

CurrencyNames.of('USD');
"dólar estadounidense"

CurrencyNames.of('CNY');
"yuan"
```

The names for scripts and languages are a little more complicated. The names of the scripts/languages are not as intuitive as the names of languages or regions and not all regions have names available to us.

Entering an unknown script name will return either the string you entered or a syntax error when you use them.

```
ScriptNames.of('Latn');
"latino"

ScriptNames.of('Cyrl');
"cirílico"

ScriptNames.of('Arab');
"árabe"

ScriptNames.of('Kana');
"katakana"
```

This gives you a lot of flexibility when building multilingual applications. I need to fully learn how to use it but even as I stumble along, the possibilities for internation e-commerce sites are awesome.

Links

- V8 release v8.1
- Intl.DisplayNames Proposal