The Intl.PluralRules API

By Mathias Bynens
V8 JavaScript whisperer

Iñtërnâtiônàlizætiøn is hard. Handling plurals is one of many problems that might seem simple, until you realize every language has its own pluralization rules.

For English pluralization, there are only two possible outcomes. Let's use the word "cat" as an example:

- 1 cat, i.e. the `'one'` form, known as the singular in English
- 2 cats, but also 42 cats, 0.5 cats, etc., i.e. the `'other'` form (the only other), known as the plural in English.

The brand new `Intl.PluralRules` API tells you which form applies in a language of your choice based on a given number.

```
const pr = new Intl.PluralRules('en-US');
pr.select(0);   // 'other' (e.g. '0 cats')
pr.select(0.5); // 'other' (e.g. '0.5 cats')
pr.select(1);   // 'one'   (e.g. '1 cat')
pr.select(1.5); // 'other' (e.g. '0.5 cats')
pr.select(2);   // 'other' (e.g. '0.5 cats')
```

Unlike other internationalization APIs, `Intl.PluralRules` is a low-level API that does not perform any formatting itself. Instead, you can build your own formatter on top of it:

```
const suffixes = new Map([
    // Note: in real-world scenarios, you wouldn't hardcode the plurals
    // like this; they'd be part of your translation files.
    ['one',   'cat'],
    ['other', 'cats'],
]);
const pr = new Intl.PluralRules('en-US');
const formatCats = (n) => {
    const rule = pr.select(n);
    const suffix = suffixes.get(rule);
    return `${n} ${suffix}`;
};

formatCats(1);   // '1 cat'
```

```
formatCats(0);   // '0 cats'
formatCats(0.5); // '0.5 cats'
formatCats(1.5); // '1.5 cats'
formatCats(2);   // '2 cats'
```

For the relatively simple English pluralization rules, this might seem like overkill; however, not all languages follow the same rules. Some languages have only a single pluralization form, and some languages have multiple forms. Welsh, for example, has six different pluralization forms!

```
const suffixes = new Map([
    ['zero',  'cathod'],
    ['one',   'gath'],
    // Note: the `two` form happens to be the same as the ``'one'``
    // form for this word specifically, but that is not true for
    // all words in Welsh.
    ['two',   'gath'],
    ['few',   'cath'],
    ['many',  'chath'],
    ['other', 'cath'],
]);
const pr = new Intl.PluralRules('cy');
const formatWelshCats = (n) => {
    const rule = pr.select(n);
    const suffix = suffixes.get(rule);
    return `${n} ${suffix}`;
};

formatWelshCats(0);   // '0 cathod'
formatWelshCats(1);   // '1 gath'
formatWelshCats(1.5); // '1.5 cath'
formatWelshCats(2);   // '2 gath'
formatWelshCats(3);   // '3 cath'
formatWelshCats(6);   // '6 chath'
formatWelshCats(42);  // '42 cath'
```

To implement correct pluralization while supporting multiple languages, a database of languages and their pluralization rules is needed. The Unicode CLDR includes this data, but to use it in JavaScript, it has to be embedded and shipped alongside your other JavaScript code, increasing load times, parse times, and memory usage. The **`Intl.PluralRules`** API shifts that burden to the JavaScript engine, enabling more performant internationalized pluralizations.

Note: While CLDR data includes the form mappings per language, it doesn't come with a list of singular/plural forms for individual words. You still have to translate and provide those yourself, just like

before.

Ordinal numbers

The **`Intl.PluralRules`** API supports various selection rules through the **`type`** property on the optional **`options`** argument. Its implicit default value (as used in the above examples) is **`'cardinal'`**. To figure out the ordinal indicator for a given number instead (e.g. **1** → **1st**, **2** → **2nd**, etc.), use **`{ type: 'ordinal' }`**:

```
const pr = new Intl.PluralRules('en-US', {
    type: 'ordinal'
});
const suffixes = new Map([
    ['one',    'st'],
    ['two',    'nd'],
    ['few',    'rd'],
    ['other',  'th'],
]);
const formatOrdinals = (n) => {
    const rule = pr.select(n);
    const suffix = suffixes.get(rule);
    return `${n}${suffix}`;
};

formatOrdinals(0);   // '0th'
formatOrdinals(1);   // '1st'
formatOrdinals(2);   // '2nd'
formatOrdinals(3);   // '3rd'
formatOrdinals(4);   // '4th'
formatOrdinals(11);  // '11th'
formatOrdinals(21);  // '21st'
formatOrdinals(42);  // '42nd'
formatOrdinals(103); // '103rd'
```

`Intl.PluralRules` is a low-level API, especially when compared to other internationalization features. As such, even if you're not using it directly, you might be using a library or framework that depends on it.

`Intl.PluralRules` is available by default in V8 v6.3.172, Chrome 63, and Firefox 58. As this API becomes more widely available, you'll find libraries such as Globalize dropping their dependency on hardcoded CLDR databases in favor of the native functionality, thereby improving load-time performance, parse-time performance, run-time performance, and memory usage.

Last updated July 2, 2018.