Internationalize your JavaScript Application: Prepare for "the next billion" internet users.

Kevin Hakanson

11-13 August 2014

Abstract

Are you prepared for "the next billion" internet users, most of whom don't use English as their primary language?

We will explore the globalization (internationalization and localization) of JavaScript based applications. It will look at the ECMAScript Internationalization API and popular open source projects like AngularJS, messageformat.js, jQuery Globalize and twitter-cldr-js.

Topics will include cultures/locales, character encoding, number formatting, date formatting, choice/plural formatting and translations.

inrule TECHNOLOGY®

API HEALTHCARE

WordPress.com

PayPal + Braintree Developer

SKYLINE TECHNOLOGIES

SafeNet

DevExpress

omni resources

corvisa cloud

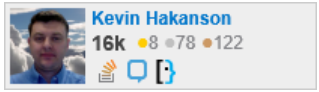
TechSmith®

Kevin Hakanson

THOMSON REUTERS

@hakanson

+KevinHakanson



Biography (en-us)

Language: English (United States) | Español (México) | Français (Canada)

Salutation: Mr. | Mrs. | Ms.

First Name: John

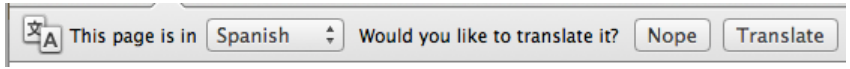
Last Name: Doe

Favorite Color: Blue | Green | Orange

Friends: 1000000

Summary:

Mr. John Doe
(*Saturday, July 26, 2014*)
His / her favorite color: Orange
He / she has 1,000,000 friend(s)



Biography (es-mx)

Idioma: English (United States) | Español (México) | Français (Canada)

Saludo: Sr. | Sra. | Srta.

Primer Nombre: John

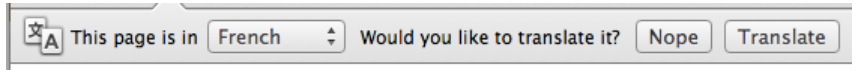
Apellido: Doe

Color Favorito: Azul | Verde | Naranja

Amigos: 1000000

Resumen:

Srta. John Doe
(*jueves, 7 de agosto de 2014*)
Su / su color favorito: Naranja
Él / ella tiene 1.000.000 amigo(s)



Biography (fr-ca)

Langue: English (United States) | Español (México) | Français (Canada)

Salutation: M. | Mme | Mlle

Prénom: John

Nom: Doe

Couleur préférée: Bleu | Vert | Orange

Amis: 1000000

Résumé:

M. John Doe
(*samedi 26 juillet 2014*)
Son / sa couleur préférée: Orange
Il / elle a 1 000 000 ami(s)

Internationalization

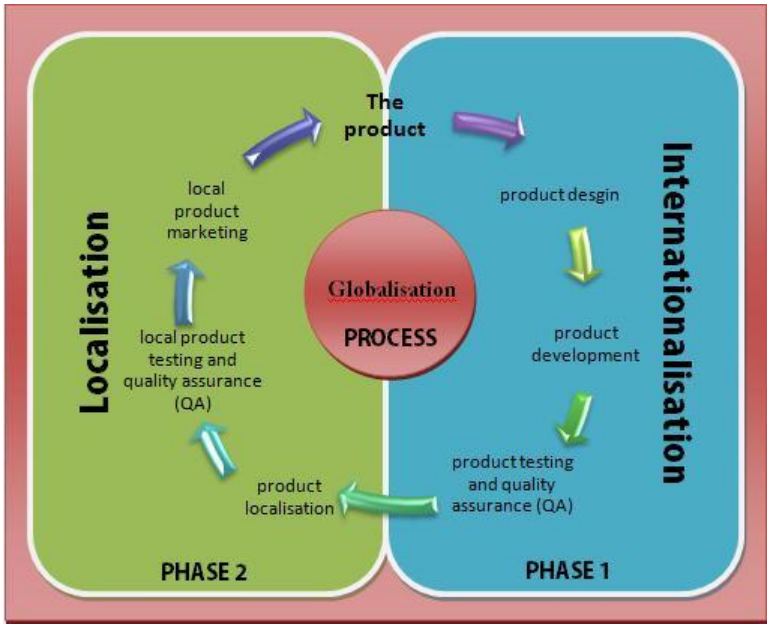
Internationalization of software means designing it such that it supports or can be easily adapted to support the needs of users speaking different languages and having different cultural expectations, and enables worldwide communication between them.

Localization

Localization then is the actual adaptation to a specific language and culture.

Globalization

Globalization of software is commonly understood to be the combination of internationalization and localization.



http://upload.wikimedia.org/wikipedia/commons/e/e3/Globalisationchart.jpg

I18N & L10N

The terms are frequently abbreviated to the numeronyms i18n (where 18 stands for the number of letters between the first i and last n in internationalization) and l10n respectively, due to the length of the words.

Other “correct” spellings are internationalisation and localisation.

http://en.wikipedia.org/wiki/Internationalization_and_localization

K3n H6n

~~Kevin Hakanson~~

ECMAScript Internationalization API

The ECMAScript Internationalization API provides key language-sensitive functionality as a complement to the ECMAScript Language Specification, 5.1 edition or successor.

Standard ECMA-402
1st Edition / December 2012
http://www.ecma-international.org/ecma-402/1.0/

ECMA-402

Its functionality has been selected from that of well-established internationalization APIs such as those of the:

- Internationalization Components for Unicode (ICU) library
- .NET framework
- Java platform

ICU - International Components for Unicode http://site.icu-project.org/

ICU is a mature, widely used set of C/C++ and Java libraries providing Unicode and Globalization support for software applications. ICU is widely portable and gives applications the same results on all platforms and between C/C++ and Java software.

ICU is released under a nonrestrictive open source license that is suitable for use with both commercial software and with other open source or free software.

Used by both Google Chrome and Mozilla Firefox.

ECMA-402

Three key pieces of language-sensitive functionality:

- String comparison (collation)
  `String.prototype.localeCompare`

- Number formatting
  `Number.prototype.toLocaleString`

- Date and time formatting
  `Date.prototype.toLocaleString`
  `Date.prototype.toLocaleDateString`
  `Date.prototype.toLocaleTimeString`

ECMA-402 Language Tags

Identifies locales using language tags as defined by IETF BCP 47 (RFCs 5646 and 4647 or their successors), which may include extensions such as those registered through RFC 6067. Their canonical form is specified in RFC 5646 section 4.5 or its successor.

(Note: RFC 5646 uses ISO 639 codes for language)


LANGUAGES
ENGLISH, SARCASM, SPANGLISH

Example Language Tags

en-US

English (United States)

es-MX

Español (México)

fr-CA

français (Canada)

```
> var locales = ["en-US", "es-MX", "fr-CA"]
> Intl.DateTimeFormat.supportedLocalesOf(locales)
["en-US", "es-MX", "fr-CA"]
```

ECMA-402 Browser Support

Supported:

Chrome, Firefox, IE11+, Opera

Not Supported:

Safari

No status reporting on either:

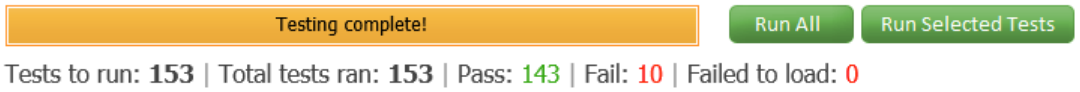
http://caniuse.com/
http://status.modern.ie/ (Issue #100)

ECMAScript **Internationalization** test402

http://test262.ecmascript.org/testcases_intl402.html#

Firefox 31.0

Internet Explorer 11.0.9600.17207



ECMAScript **Internationalization** test402

Google Chrome 36.0.1985.125

Opera 23.0.1522.60

Safari 7.0.5 (9537.77.4)

ECMA-402 Blog Posts

Norbert Lindenberg (Dec 2012)
http://norbertlindenberg.com/2012/12/ecmascript-internationalization-api/

David Storey (Aug 2013)
http://generatedcontent.org/post/59403168016/esintlapi

Dr. Alex Rauschmayer (Sep 2013)
http://www.2ality.com/2013/09/ecmascript-i18n-api.html

EMCA-402 Polyfill/Backport?

Compatibility implementation of the ECMAScript Internationalization API (ECMA-402) for JavaScript
https://github.com/andyearnshaw/Intl.js

> “`Intl.js` attempts to fill the void of availability for this API...so that developers can take advantage of the native API in environments that support it, or `Intl.js` for legacy or unsupporting environments.”

> “`Intl.js` is designed to be compatible with ECMAScript 3.1 environments in order to follow the specification as closely as possible.”

jQuery Globalize

Originally a jQuery plugin (jquery-global), now a standalone JavaScript library called Globalize.
http://wiki.jqueryui.com/w/page/39118647/Globalize

Enables complex culture-aware number and date parsing and formatting, including the raw culture information for hundreds of different languages and countries, as well as an extensible system for localization.

Globalize Cultures

Each culture is given a unique code that is a combination of an ISO 639 two-letter lowercase culture code for the language, and a two-letter uppercase code for the country or region.

For example, "en-US" is the culture code for English in the United States.

"Neutral" cultures based on accepted set of rules by anyone speaking that language.

For example, "es" is the neutral culture for Spanish.

Globalize API

```
Globalize.addCultureInfo( cultureName,
    extendCultureName, info )
Globalize.cultures
Globalize.culture( selector )
Globalize.findClosestCulture( selector )
Globalize.format( value, format, culture )
Globalize.localize( key, culture )
Globalize.parseInt( value, radix, culture )
Globalize.parseFloat( value, radix, culture )
Globalize.parseDate( value, formats, culture )
```

Globalize Culture Examples

```
> Globalize.culture("en-US")
Object {name: "en-US", englishName: "English (United States)",
nativeName: "English", isRTL: false, language: "en"…}

> Globalize.culture("es-MX")
Object {name: "es-MX", englishName: "Spanish (Mexico)",
nativeName: "Español (México)", isRTL: false, language: "es"…}

> Globalize.culture("fr-CA")
Object {name: "fr-CA", englishName: "French (Canada)",
nativeName: "français (Canada)", isRTL: false, language: "fr"…}
```

HTML is great for declaring static documents, but it falters when we try to use it for declaring dynamic views in web-applications. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop.

http://www.angularjs.org/

AngularJS I18N/L10N Support

Supports i18n/l10n for datetime, number and currency `filters`.

Localizable pluralization support provided by the `ngPluralize directive`.

All localizable components depend on locale-specific rule sets managed by the `$locale service`.
http://docs.angularjs.org/guide/i18n

Extending AngularJS

i18n for your Angular apps, made easy
http://pascalprecht.github.io/angular-translate/#/guide

Internazionalization (i18n) with AngularJS
http://blog.brunoscopelliti.com/internazionalization-i18n-with-angularjs

Localizing Your AngularJS App
http://codingsmackdown.tv/blog/2012/12/14/localizing-your-angularjs-app/

Angular i18n library wrapping Jed (gettext for js)
https://github.com/ErikAndreas/lingua

...

messageformat.js

ICU MessageFormat for Javascript - i18n Plural and Gender Capable Messages

- Implements SelectFormat and PluralFormat
- Plan to pull in locale-aware NumberFormat parsing as a "plugin"

https://github.com/SlexAxton/messageformat.js

Example Formatted Message

```
{GENDER, select,
    male {He has}
  female {She has}
   other {They have}
} {NUM_FRIENDS, plural,
              =0 {no friends}
             one {1 friend}
           other {# friends}
}
```

Example Usage

```
> var mf = new MessageFormat('en')
> var messageTemplate = "{...}" // see previous slide
> var message = mf.compile(messageTemplate)

> message({ "GENDER" : "male" , "NUM_FRIENDS" : 0 })
"He has no friends."

> message({ "GENDER" : "female" , "NUM_FRIENDS" : 1 })
"She has 1 friend."

> message({ "GENDER" : "other" , "NUM_FRIENDS" : 2 })
"They have 2 friends."
```

messageformat.js

Alex Sexton: Client Side Internationalization

https://www.youtube.com/watch?v=uXS_-JRsB8M

twitter-cldr-js

JavaScript implementation of the ICU (International Components for Unicode) that uses the Common Locale Data Repository (CLDR) to format dates, plurals, and more.

CLDR - Unicode Common Locale Data Repository http://cldr.unicode.org/

The Unicode CLDR provides key building blocks for software to support the world's languages, with the largest and most extensive standard repository of locale data available.

This data is used by a wide spectrum of companies for their software internationalization and localization, adapting software to the conventions of different languages for such common software tasks.

CLDR Includes:

- **Locale-specific patterns for formatting and parsing** dates, times, timezones, numbers and currency values

CLDR Includes:

- **Locale-specific patterns for formatting and parsing**
- **Translations of names**
  languages, scripts, countries and regions, currencies, eras, months, weekdays, day periods, timezones, cities, and time units

CLDR Includes:

- **Locale-specific patterns for formatting and parsing**
- **Translations of names**
- **Language & script information**
  characters used; plural cases; gender of lists; capitalization; rules for sorting & searching; writing direction; transliteration rules; rules for spelling out numbers; rules for segmenting text into graphemes, words, and sentences

CLDR Includes:

- **Locale-specific patterns for formatting and parsing**
- **Translations of names**
- **Language & script information**
- **Country information**
  language usage, currency information, calendar preference and week conventions, postal and telephone codes

CLDR Includes:

- **Locale-specific patterns for formatting and parsing**
- **Translations of names**
- **Language & script information**
- **Country information**
- **Other**
  ISO & BCP 47 code support (cross mappings, etc.), keyboard layouts

twitter-cldr-js

JavaScript implementation of the ICU (International Components for Unicode) that uses the Common Locale Data Repository (CLDR) to format dates, plurals, and more.

TwitterCldr supports the following:

- Date and time formatting
- Relative date and time formatting (eg. 1 month ago)
- Number formatting (decimal, currency, and percentage)
- Long/short decimals
- Plural rules
- Bidirectional reordering

https://github.com/twitter/twitter-cldr-js

Cultures / Locales

http://www.jsonline.com/entertainment/dining/15-beer-and-wine-facts-that-may-surprise-you-b99117052z1-227119841.html

EMCA-402

Implementation dependent

David Storey's Appendix A includes locale support tables for each browser that supports the Internationalization API

http://generatedcontent.org/post/59403168016/esintlapi

Globalize (~353)

```
// globalize.js
Globalize.cultures[ "default" ] = {...};

days: {
    // full day names
    names: [ "Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday" ],

// globalize.cultures.es-MX.js
Globalize.addCultureInfo( "es-MX", "default", {...});

days: {
    names: ["domingo", "lunes", "martes", "miércoles",
"jueves", "viernes", "sábado"],
```

AngularJS "en-US" $locale

Bundled into `angular.js`

```
angularModule('ngLocale', []).provider('$locale', $LocaleProvider);
function $LocaleProvider() {
    this.$get = function () {
        return { id: 'en-us', /* … */ };
    };
}
```

In separate `angular-locale_en-US.js`

```
angular.module("ngLocale", [], ["$provide", function($provide) {
    $provide.value("$locale", {
        "id": "en-us", /* … */
    });
```

AngularJS (~446)

angular-locale_en-us.js

```
"DAY": [
    "Sunday",
    "Monday",
    "Tuesday",
    "Wednesday",
    "Thursday",
    "Friday",
    "Saturday"
],
```

angular-locale_es-mx.js

```
"DAY": [
    "domingo",
    "lunes",
    "martes",
    "mi\u00e9rcoles",
    "jueves",
    "viernes",
    "s\u00e1bado"
],
```

AngularJS `locale.js` File Encoding

Bug Fix from 1.0.7 monochromatic-rainbow (2013-05-22)

> **i18n**: escape all chars above \u007f in locale files (695c54c1, #2417)
> https://github.com/angular/angular.js/blob/master/CHANGELOG.md

Example: src/ngLocale/angular-locale_es-mx.js

```
-       "3": "miércoles",
+       "3": "mi\u00e9rcoles",
```

Globalize Culture Examples (redux)

```
> Globalize.culture("en-US")
Object {name: "en-US", englishName: "English (United States)",
nativeName: "English", isRTL: false, language: "en"…}

> Globalize.culture("es-MX")
Object {name: "es-MX", englishName: "Spanish (Mexico)",
nativeName: "Español (México)", isRTL: false, language: "es"…}

> Globalize.culture("fr-CA")
Object {name: "fr-CA", englishName: "French (Canada)",
nativeName: "français (Canada)", isRTL: false, language: "fr"…}
```

Globalize File Encoding (UTF-8)

WebStorm integrated web server without `<meta charset='utf-8'/>` in demo.html

```
>       Globalize.culture("es-MX")
    Object {name: "es-MX", englishName: "Spanish (Mexico)",
    nativeName: "EspaÃ±ol (MÃ©xico)", isRTL: false, language:
    "es"…}
>       Globalize.culture("fr-CA")
    Object {name: "fr-CA", englishName: "French (Canada)",
    nativeName: "franÃ§ais (Canada)", isRTL: false, language:
    "fr"…}
```

X-Powered-By: Express

```
var express = require('express');
var app = express();
app.configure(function(){
  app.use(function(req, res, next) {
    if (/.*\.js/.test(req.path)) {
      res.charset = "utf-8";
      // Content-Type: application/javascript; charset=utf-8
    }
    next();
  });
  app.use(express.static('../jsi18n'));
});
app.listen(1337);  // http://localhost:1337/demo.html#/
```

More Options

Convert source code to ASCII (7 bit)

Before:

```
var σ = 'Köln';
```

```
> uglifyjs -b beautify=false,ascii-only=true file.js
```

After:

```
var \u03c3 = "K\xf6ln";
```

Load source with charset attribute on script tag

```
var π = Math.PI, ε = 1e-6;
```

```
<script charset="utf-8" src="d3.js"></script>
```

http://www.2ality.com/2013/09/javascript-unicode.html

é (small e, acute accent)

HTML Entity

```
&eacute;
&#xE9;
&#233;
```

JavaScript

```
"\u00E9"
"\xE9"
```

UTF-8

```
Ã©
C3 A9
```

URL

```
%e9
```

```
%c3%a9
```

ñ (small n, tilde)

HTML Entity

```
&ntilde;
&#xF1;
&#241;
```

URL

```
%f1
```

JavaScript

```
"\u00F1"
"\xF1"
```

UTF-8

```
Ã±
C3 B1
```

```
%c3%b1
```

ç (small c, cedilla)

HTML Entity

```
&ccedil;
&#xE7;
&#231;
```

URL

```
%e7
```

JavaScript

```
"\u00E7"
"\xE7"
```

UTF-8

```
Ã§
C3 A7

%c3%a7
```

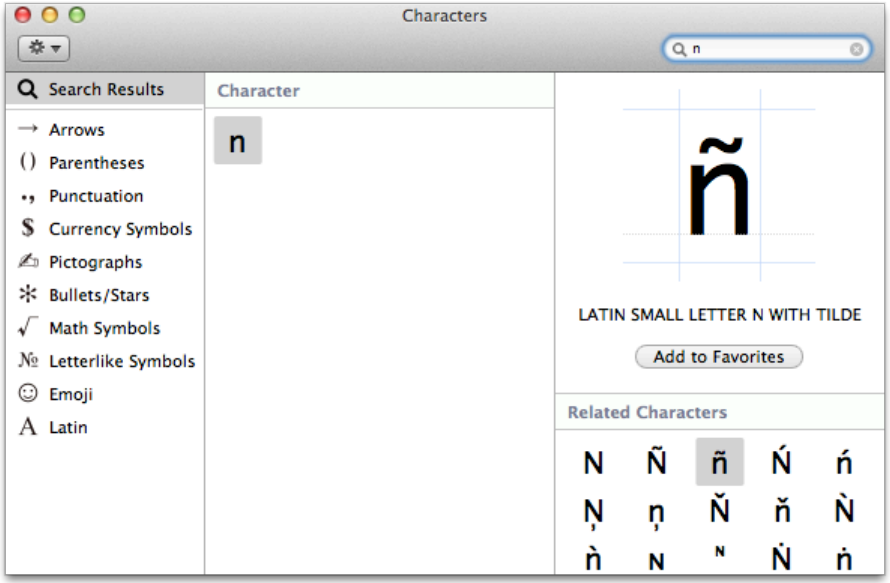
```
> div = document.createElement("div")
<div> </div>

> div.innerHTML = "&ntilde;&eacute;&ccedil;"
"&ntilde;&eacute;&ccedil;"
> div.innerText
"ñéç"
> div.innerHTML
"ñéç"
```

```
>     div.innerHTML = "&ntilde;&eacute;&ccedil;
&#xF1;&#xE9;&#xE7; &#241;&#233;&#231;"
   "&ntilde;&eacute;&ccedil; &#xF1;&#xE9;&#xE7;
&#241;&#233;&#231;"
>     div.innerHTML
   "ñéç ñéç ñéç"

>     div.innerText = "\u00F1\u00E9\u00E7 \xF1\xE9\xE7"
   "ñéç ñéç"
>     encodeURIComponent("ñéç")
   "%C3%B1%C3%A9%C3%A7"
>     escape("ñéç")
   "%F1%E9%E7"
```

Mac Character Viewer

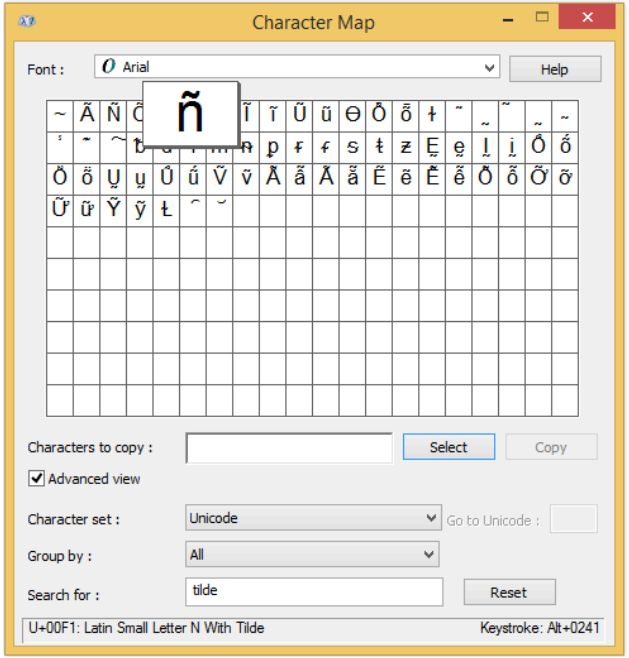
System Preferences - Keyboard

☑ Show Keyboard & Character Viewers in menu bar



Windows Character Map

Start - Run - charmap.exe



JavaScript has a Unicode problem

```
> "mañana" == "mañana"
  false
> "ma\xF1ana" == "man\u0303ana"
  false
```

The first string contains U+00F1 LATIN SMALL LETTER N WITH TILDE, while the second string uses two separate code points (U+006E LATIN SMALL LETTER N and U+0303 COMBINING TILDE) to create the same glyph.

http://mathiasbynens.be/notes/javascript-unicode

TextEncoder

Script API to allow encoding/decoding of strings from binary data. Common scenario: decoding a binary data file fetched via XHR into an ArrayBuffer that contains strings encoded as UTF-8.

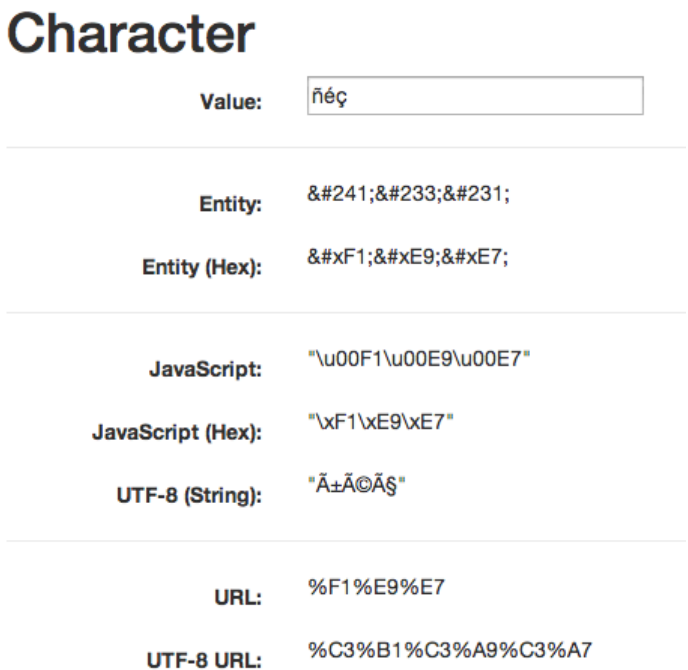
http://encoding.spec.whatwg.org/#textencoder

Firefox 19

https://developer.mozilla.org/en-US/docs/Web/API/TextEncoder

Intent to Ship in Chrome 38

Issue 243354: Implement Text Encoding API


Character
Value:
ñéç
Entity:
ñéç
Entity (Hex):
ñéç
JavaScript:
"\u00F1\u00E9\u00E7"
JavaScript (Hex):
"\xF1\xE9\xE7"
UTF-8 (String):
"Ã±Ã©Ã§"
URL:
%F1%E9%E7
UTF-8 URL:
%C3%B1%C3%A9%C3%A7

DEMO

Dynamically Changing Culture / Locale

ECMA-402

No settable default locale list:

- global communication channel (security risk)
- application may require multiple components and contexts; higher-level concern of application/library vs ECMAScript

Every API takes a locale list as parameter

Globalize

Include culture files

```
<script src="globalize/globalize.js"></script>
<script src="globalize/cultures/globalize.culture.en-US.js"></script>
<script src="globalize/cultures/globalize.culture.es-MX.js"></script>
<script src="globalize/cultures/globalize.culture.fr-CA.js"></script>
<script>
    // see https://github.com/jquery/globalize/blob/master/doc/cldr.md
    $.get( "cldr/main/en/ca-gregorian.json", Globalize.load );
    $.get( "cldr/main/es/ca-gregorian.json", Globalize.load );
    $.get( "cldr/main/fr/ca-gregorian.json", Globalize.load );
</script>
```

Call function to set default

```
Globalize.culture("en-US")
```

AngularJS $locale Binding

AngularJS binds value of `$locale` and doesn't allow updates to locale or including multiple locale `<script>` files.

Angular Dynamic Locale

A library to change the locale at runtime. It changes the underlying `$locale` object, so changing the locale will change the formatting of all elements that use `$locale`

```
tmhDynamicLocale.set('en-us');
```

http://lgalfaso.github.io/angular-dynamic-locale/

twitter-cldr-js

Include single locale

```
<script src="twitter_cldr/en.js"></script>
```

Dynamically load locale

```
$.getScript("twitter_cldr/en.js", function () {
    // do updates
});
```

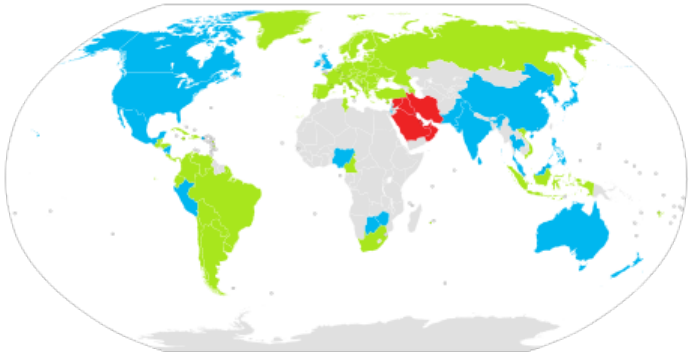
Number / Currency Formatting

“ONE MILLION DOLLARS”

Decimal Mark

A symbol used to separate the integer part from the fractional part of a number written in decimal form.
Different cultures use different symbols for the decimal mark. The choice of symbol for the decimal mark also affects the choice of symbol for the thousands separator used in digit grouping.

Full Stop "." – Blue
Comma "," – Green
Eastern Arabic numerals – Red
Data unavailable – Grey



http://en.wikipedia.org/wiki/Decimal_mark

Globalize

Four main types of number formatting:

- **n** for number
- **d** for decimal digits
- **p** for percentage
- **c** for currency

Each format token may also be followed by a number determining how many decimal places to display (with **d** determines the minimum number of digits to display, zero padded)

```
> Globalize.format(12345.6789, "n")
"12,345.68"

> Globalize.format(12345.6789, "n4")
"12,345.6789"

> Globalize.format(12345.6789, "n9")
"12,345.678900000"
```

```
> Globalize.format(12345.6789, "d")
"12345"

> Globalize.format(12345.6789, "d4")
"12345"

> Globalize.format(12345.6789, "d9")
"000012345"
```

```
> Globalize.format(12345.6789, "n")
"12,345.68"

> Globalize.format(12345.6789, "n", "es-MX")
"12,345.68"

> Globalize.format(12345.6789, "n", "fr-CA")
"12 345,68"
```

EMCA-402

Three format styles for NumberFormat objects and `Number.prototype.toLocaleString`:

- decimal
- currency
- percent

Number of digits used to represent a number can be constrained

Grouping separators can be disabled

ECMA-402 Components of Number and Currency Formats

Property	Values
style	"decimal", "currency", "percent"
currency	string
currencyDisplay	"code", "symbol", "name"
minimumIntegerDigits	non-negative integer
minimumFractionDigits	non-negative integer
maximumFractionDigits	non-negative integer
minimumSignificantDigits	positive integer
maximumSignificantDigits	positive integer
useGrouping	boolean

```
> num = 12345.6789
12345.6789

> num.toLocaleString("en-us")
"12,345.679"

> num.toLocaleString("es-MX")
"12,345.679"

> num.toLocaleString("fr-CA")
"12 345,679"
```

```
> num = 12345.6789
12345.6789

> num.toLocaleString("en-us", { useGrouping: true,
  minimumSignificantDigits: 4})
"12,345.6789"

> num.toLocaleString("en-US", { useGrouping: false,
  maximumFractionDigits: 0})
"12346"

> num.toLocaleString("en-US", { useGrouping: false,
  maximumFractionDigits: 0, minimumIntegerDigits: 9})
"000012346"
```

Chromium Issue 304722

Intl NumberFormat minimumFractionDigits options property not honored

Reported by kevin.ha...@gmail.com, Oct 6

Steps to reproduce the problem:

```
(12345.6789).toLocaleString("en-us", { useGrouping: false, minimumFractionDigits: 4 })
"12345.679"
```

What is the expected behavior?

```
"12345.6789"
```

https://code.google.com/p/chromium/issues/detail?id=304722

```
(12345.67).toLocaleString("en-US", { style: "currency",
useGrouping: true, currency: "USD", currencyDisplay: "name" })
```

- Chrome

  12,345.67 US Dollars
- Internet Explorer 11

  USD 12,345.67
- Firefox 31.0

  Error: internal error while computing Intl data

  https://bugzilla.mozilla.org/show_bug.cgi?id=866372
- Intl.js

  USD12,345.67

  https://github.com/andyearnshaw/Intl.js/issues/57

twitter-cldr-js

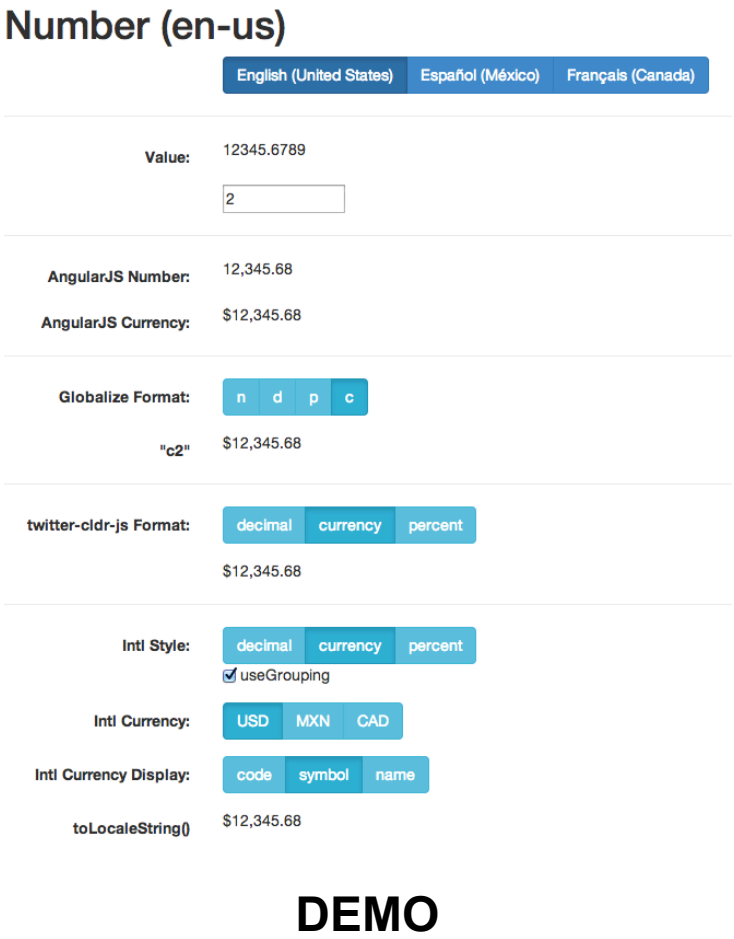
Number formatting supports decimals, currencies, and percentages.

In addition to formatting regular decimals, supports short and long decimals.

- Short decimals abbreviate the notation for the appropriate power of ten, for example "1M" for 1,000,000 or "2K" for 2,000.
- Long decimals include the full notation, for example "1 million" or "2 thousand".

```
> num = 12345.6789
12345.6789

> fmt = new TwitterCldr.DecimalFormatter()
DecimalFormatter {all_tokens: Object, tokens: Array[0],
symbols: Object, default_symbols: Object, constructor: function…}


> fmt.format(num)
"12,345.6789"

> fmt.format(num, {precision: 2})
"12,345.68"
```

```
> num = 12345.6789
12345.6789

> fmt = new TwitterCldr.CurrencyFormatter();
CurrencyFormatter {default_currency_symbol: "$",
default_precision: 2, all_tokens: Object, tokens: Array[0], symbols:
Object…}

> fmt.format(num, {currency: "USD"})
"$12,345.6789"

> fmt.format(num, {currency: "USD", precision: 2})
"$12,345.68"
```


Number (en-us)
English (United States)
Español (México)
Français (Canada)
Value:
12345.6789
2
AngularJS Number:
12,345.68
AngularJS Currency:
$12,345.68
Globalize Format:
n
d
p
c
"c2"
$12,345.68
twitter-cldr-js Format:
decimal
currency
percent
$12,345.68
Intl Style:
decimal
currency
percent
useGrouping
Intl Currency:
USD
MXN
CAD
Intl Currency Display:
code
symbol
name
toLocaleString()
$12,345.68

DEMO

Date Formatting

International Dates, not Dating

ECMA-402 Components of Date and Time Formats

Property	Values
weekday	"narrow", "short", "long"
era	"narrow", "short", "long"
year	"2-digit", "numeric"
month	"2-digit", "numeric", "narrow", "short", "long"
day	"2-digit", "numeric"
hour	"2-digit", "numeric"
minute	"2-digit", "numeric"
second	"2-digit", "numeric"
timeZoneName	"short", "long"

Globalize Standard Date Formats

Format	Meaning	"en-US"	new Date (2013,0,1,13,1,1)
f	Long Date, Short Time	dddd, MMMM dd, yyyy h:mm tt	Tuesday, January 01, 2013 1:01 PM
F	Long Date, Long Time	dddd, MMMM dd, yyyy h:mm:ss tt	Tuesday, January 01, 2013 1:01:01 PM
t	Short Time	h:mm tt	1:01 PM
T	Long Time	h:mm:ss tt	1:01:01 PM
d	Short Date	M/d/yyyy	1/1/2013
D	Long Date	dddd, MMMM dd, yyyy	Tuesday, January 01, 2013
Y	Month/Year	yyyy MMMM	2013 January
M	Month/Day	MMMM dd	January 01

AngularJS Predefined Date Formats

Format	**"en-US"**	**new Date (2013,0,1,13,1,1)**
medium	MMM d, y h:mm:ss a	Jan 1, 2013 1:01:01 PM
short	M/d/yy h:mm a	1/1/13 1:01 PM
fullDate	EEEE, MMMM d,y	Tuesday, January 1, 2013
longDate	MMM d, y	January 1, 2013
mediumDate	MMM d, y	Jan 1, 2013
shortDate	M/d/yy	1/1/13
mediumTime	h:mm:ss a	1:01:01 PM
shortTime	h:mm a	1:01 PM

Date/Time Format Patterns

jQuery Globalize

```
// short date pattern
d: "M/d/yyyy",
// long date pattern
D: "dddd, MMMM dd, yyyy",
// short time pattern
t: "h:mm tt",
// long time pattern
T: "h:mm:ss tt",
// long date, short time pattern
f: "dddd, MMMM dd, yyyy h:mm tt",
// long date, long time pattern
F: "dddd, MMMM dd, yyyy h:mm:ss tt",
// month/day pattern
M: "MMMM dd",
// month/year pattern
Y: "yyyy MMMM"
```

AngularJS

```
"fullDate": "EEEE, MMMM d, y",
"longDate": "MMMM d, y",
"medium": "MMM d, y h:mm:ss a",
"mediumDate": "MMM d, y",
"mediumTime": "h:mm:ss a",
"short": "M/d/yy h:mm a",
"shortDate": "M/d/yy",
"shortTime": "h:mm a"
```

twitter-cldr-js

The default CLDR data set only includes 4 date formats, `full`, `long`, `medium`, and `short`. See below for a list of additional formats.

```
> TwitterCldr.DateTimeFormatter.additional_formats()
["EHm", "EHms", "Ed", "Ehm", "Ehms", "Gy", "GyMMM",
"GyMMMEd", "GyMMMd", "H", "Hm", "Hms", "M", "MEd", "MMM",
"MMMEd", "MMMd", "Md", "d", "h", "hm", "hms", "ms", "y", "yM",
"yMEd", "yMMM", "yMMMEd", "yMMMd", "yMd", "yQQQ",
"yQQQQ"]
```

```
> then = new Date (2013,0,1,13,1,1)
Tue Jan 01 2013 13:01:01 GMT-0600 (CST)

> Globalize.format( then , 'f' )
"Tuesday, January 01, 2013 1:01 PM"

> then.toLocaleString("en-US", { weekday : "long", month : "long", day : "2-digit", year : "numeric", hour : "numeric", minute : "2-digit",  hour12 : true  } )
"Tuesday, January 01, 2013 1:01 PM"

> fmt = new TwitterCldr.DateTimeFormatter()
DateTimeFormatter {tokens: Object, weekday_keys: Array[7], methods: Object, format: function, get_tokens: function…}
> fmt.format(then, {"type": "full"})
"Tuesday, January 1, 2013 at 1:01:01 PM UTC-06:00"
```

```
> then = new Date (2013,0,1,13,1,1)
Tue Jan 01 2013 13:01:01 GMT-0600 (CST)

> Globalize.format( then , 't' )
"1:01 PM"

> then.toLocaleTimeString("en-US", { hour : "numeric",
minute : "2-digit",  hour12 : true } )
"1:01 PM"

> fmt = new TwitterCldr.DateTimeFormatter()
DateTimeFormatter {tokens: Object, weekday_keys: Array[7],
methods: Object, format: function, get_tokens: function…}
> fmt.format(then, {"format": "time", "type": "short"})
"1:01 PM"
```

Intl AngularJS Filter

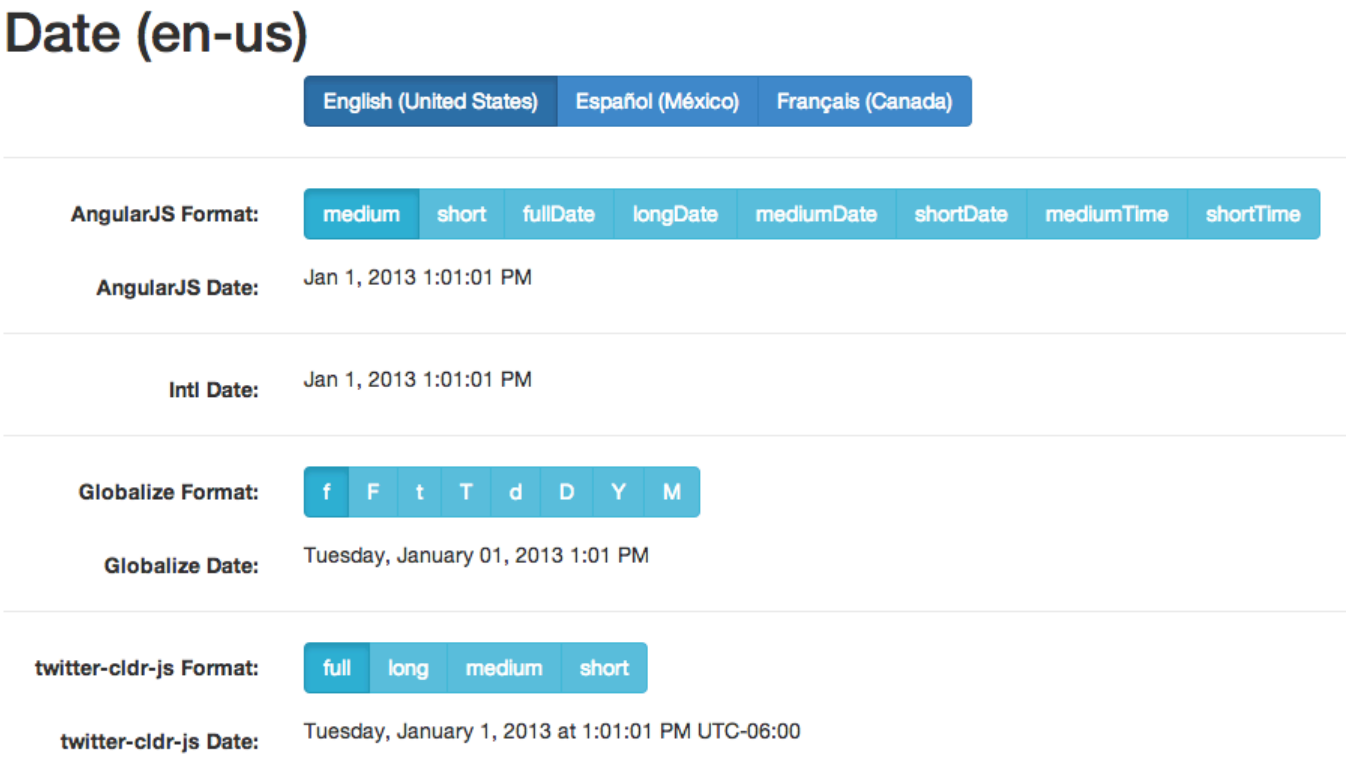
Proof of concept AngularJS filter illustrating ECMA-402 API. Takes two parameters:

- arg0 - culture (optional)
- arg1 - format (predefined string or Intl struct)

Sample usage:

```
{{ then | dateIntl:"en-US":"longDate" }}
{{ then | dateIntl:"es-MX":"longDate" }}
{{ then | dateIntl:"longDate" }}
{{ then | dateIntl:{ hour : "numeric", minute : "2-digit",  hour12 : true } }}
```

```
var IntlFilters = angular.module('Intl.filters', []);
IntlFilters.filter('dateIntl', ['$locale', function ($locale) {
    return function (date, arg0, arg1) {
        var culture = (arg1 ? arg0 : $locale.id);
        var format = (arg1 ? arg1 : arg0);
        var s = date.toISOString();
        if (typeof format == "string") {
            switch (format) {
                case "medium":
                    s = date.toLocaleString(culture, { month: "short", day:
"numeric", year: "numeric", hour: "numeric", minute: "2-digit", second: "2-digit",
hour12: true });
                    break;
                    // case "longdate", "shorttime", ...
            }
        } else {
            s = date.toLocaleString(culture, format);
        }
        return s;
    };
}]);
```

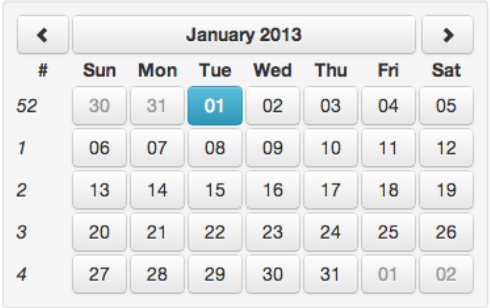

Date (en-us)
English (United States)
Español (México)
Français (Canada)
AngularJS Format:
medium
short
fullDate
longDate
mediumDate
shortDate
mediumTime
shortTime
AngularJS Date:
Jan 1, 2013 1:01:01 PM
Intl Date:
Jan 1, 2013 1:01:01 PM
Globalize Format:
f
F
t
T
d
D
Y
M
Globalize Date:
Tuesday, January 01, 2013 1:01 PM
twitter-cldr-js Format:
full
long
medium
short
twitter-cldr-js Date:
Tuesday, January 1, 2013 at 1:01:01 PM UTC-06:00

DEMO

AngularUI Bootstrap

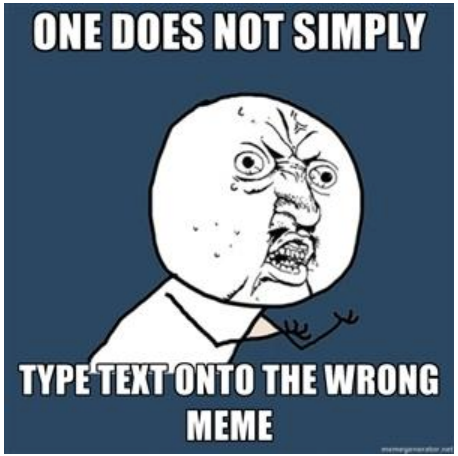
Datepicker (ui.bootstrap.datepicker)

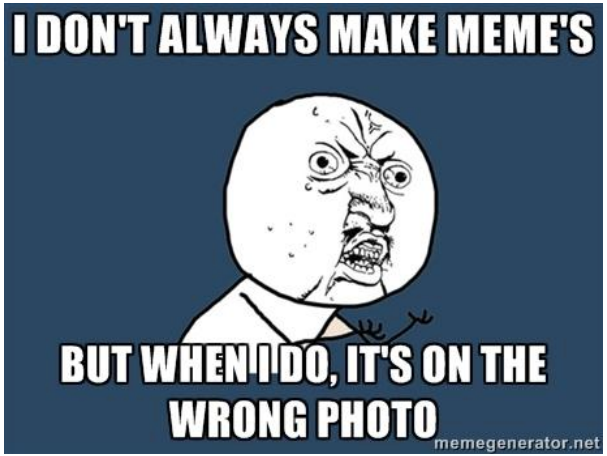
- A clean, flexible, and fully customizable date picker.
- Everything is formatted using the date filter and thus is also localized.



Timepicker (ui.bootstrap.timepicker)

Choice / Plural Formatting





In the field of memetics, a **metameme** (or **meta-meme**) is defined as a meme about a meme.
http://en.wikipedia.org/wiki/Metameme

messageformat.js (redux)

```
{GENDER, select,
    male {He has}
  female {She has}
   other {They have}
} {NUM_FRIENDS, plural,
              =0 {no friends}
             one {1 friend}
           other {# friends}
}
```

AngularJS: ngSwitch

Conditionally swap DOM structure on your template based on a scope expression

- Place an expression `ng-switch="..."` attribute

Choses one of the nested elements and makes it visible based on which element matches the value obtained from the evaluated expression

- `ng-switch-when="..."` attribute is used to inform ngSwitch which element to display

http://docs.angularjs.org/api/ng.directive:ngSwitch

AngularJS: ngPluralize

There are two plural categories in Angular's default en-US locale: "one" and "other".

- An explicit number rule can only match one number.

Configure `ngPluralize` by providing two attributes: `count` and `when`.

- `count` can be either a string or an Angular expression
- `when` specifies the mappings between plural categories and the actual string to be displayed

http://docs.angularjs.org/api/ng.directive:ngPluralize

AngularJS Formatted Message

```
<div>
    <span ng-switch="salutationkey">
        <span ng-switch-when="MR">He</span>
        <span ng-switch-when="MRS">She</span>
        <span ng-switch-when="MS">She</span>
    </span>
    has
    <span ng-pluralize count="friends"
          when="{'0': 'no friends.',
                 'one': '1 friend.',
                 'other': '{{friends|number}} friends.'}">
    </span>
</div>
```

twitter-cldr-js (en)

```
TwitterCldr.PluralRules = (function () {
    function PluralRules() { }
    PluralRules.rules = {
        "keys": ["one", "other"], "rule": function (n) {
            return (function () { if (n == 1) { return "one" } else { return "other" } })();
        }
    };
    PluralRules.all = function () {
        return this.rules.keys;
    };
    PluralRules.rule_for = function (number) {
        var error;
        try {
            return this.rules.rule(number);
        } catch (_error) {
            error = _error;
            return "other";
        }
    };
    return PluralRules;
})();
```

Translations

http://untitledmagazine.net/category/international-cats/

AngularJS module that makes your life much easier when it comes to i18n and l10n including lazy loading and pluralization.

Provides components like filters and directives, asynchronous loading of i18n data, full pluralization support through MessageFormat and much more!

http://pascalprecht.github.io/angular-translate/#/guide

angular translate Usage

Set Preferred

```
$translateProvider.preferredLanguage('en-US');
```

Update

```
$translate.use('en-US');
```

Use with HTML Template Binding

```
    {{ 'FAV_COLOR' | translate }}:
```

Use in JavaScript

```
    $filter('translate')('FAV_COLOR');
```

```
$translateProvider.translations('en-US', {
    FAV_COLOR: "Favorite Color",
    BLUE: "Blue",
});


$translateProvider.translations('es-MX', {
    FAV_COLOR: "Color Favorito",
    BLUE: "Azul",
});


$translateProvider.translations('fr-CA', {
    FAV_COLOR: "Couleur préférée",
    BLUE: "Bleu",
});
```

Globalize Usage

Set Culture

```
Globalize.culture( "en-US" );
```

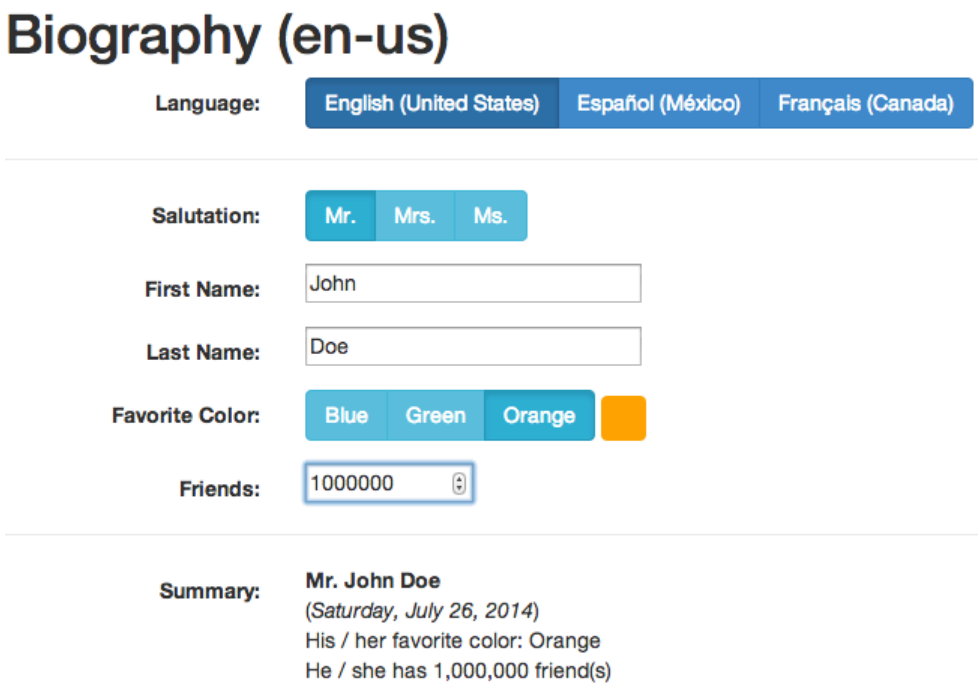
Localize

```
Globalize.localize( "FAV_COLOR" )
```

Localize for Culture

```
Globalize.localize( "FAV_COLOR", "en-US" )
Globalize.localize( "FAV_COLOR", "fr-CA" )
```

```
Globalize.addCultureInfo("en-US", {
    messages: {
        "FAV_COLOR": "Favorite Color",
        "BLUE": "Blue"
    }
});

Globalize.addCultureInfo("es-MX", {
    messages: {
        "FAV_COLOR": "Color Favorito",
        "BLUE": "Azul"
    }
});

Globalize.addCultureInfo("fr-CA", {
    messages: {
        "FAV_COLOR": "Couleur préférée",
        "BLUE": "Bleu"
    }
});
```


Biography (en-us)
Language:
English (United States)
Español (México)
Français (Canada)
Salutation:
Mr.
Mrs.
Ms.
First Name:
John
Last Name:
Doe
Favorite Color:
Blue
Green
Orange
Friends:
1000000
Summary:
Mr. John Doe
(Saturday, July 26, 2014)
His / her favorite color: Orange
He / she has 1,000,000 friend(s)

DEMO

¿preguntas?

(questions?)