Internationalization & Localization With PHP

Ben Ramsey

Longhorn PHP • November 3, 2023

WELCOME

A bit about me

- Senior Staff Engineer at Skillshare
- PHP 8.1 and 8.2 release manager
- Author of ramsey/uuid, et al.
- Creator of skillshare/formatphp (which we'll discuss)
- Working on PECL ecma_intl, a port of ECMA-402 to PHP

WHAT IS I18N & L10N?

“Internationalization is the process of designing a software application so that it can be adapted to various languages and regions without engineering changes.”

“Internationalization and localization,” Wikipedia.

“Localization is the process of adapting internationalized software for a specific region or language by translating text and adding locale-specific components.”

“Internationalization and localization,” Wikipedia.

I18N & L10N

Internationalization & localization

- Localization depends on the work of internationalization.
- Internationalization is the job of programmers.
- Localization is the job of copy writers, marketers, product owners, etc.).
- Software that has been internationalized can be localized.
- Internationalized means it’s ready to adapt to any place.
- Localized means it’s ready to use in a specific place.

i18n & l10n are numeronyms.

LOCALIZATION

It's not just translating words into another language

- Converting from one language to another
- Using the correct time zone
- Displaying prices in the user's currency and formatting foreign currencies
- Formatting dates and times and using the proper calendar system
- Formatting numbers
- Displaying the proper names for regions, currencies, languages, etc.

INTERNATIONALIZATION

Facilitates localization

- Ensure that strings/content can be extracted and translated
- Languages may be used across multiple locales, so ensure numbers, dates/times, currencies, time zones, country names, etc. render properly for those locales, **no matter what language is used**.
- Ensure translators have context to understand how to translate a string.
- Ensure strings have placeholders the translators can understand, and account for plural forms.

COMMON PITFALLS
Setting up

- Imagine a function with the following signature:

```php
function translate(string $message, mixed ...$values): string;
```

- Takes a message, looks it up in a translation table of some sort, replaces placeholders with the values, and returns the result.
- Assume the application sets a locale at some point, for this to work.

COMMON PITFALLS

Dates & times

```
echo translate(
    'Your reservation is confirmed for %s at %s.',
    date('F j, Y', $time),
    date('g:i A', $time),
);
```

COMMON PITFALLS

Numbers

```
echo translate(
    "You've cycled %s miles this year.",
    number_format(3328.2591, 2, '.', ','),
);
```

COMMON PITFALLS

Currency

```
echo translate('Your order total is $%s.', $total);
```

COMMON PITFALLS
Message formatting

```
echo translate('Read ')
     . $bookTitle
     . translate(' by ')
     . $authorName . '.';

echo translate(
     'There are %d item(s) in your order.',
     $itemCount,
);
```

COMMON PITFALLS

Display names

```
echo translate(
    'Congratulations on booking your trip to %s!',
    $countryName,
);
```

These are all mistakes.

COMMON PITFALLS

Why they are mistakes, or misguided attempts

- Assumes **translation** is the only requirement for internationalization.
- Assumes number, date, & currency formats are the same across locales.
- Assumes country names, month names, etc. are the same across locales.
- Assumes all languages follow the same subject-verb-object pattern.

TOOLS OFTEN USED

- gettext extension for PHP (via libintl, a.k.a. GNU gettext)
  - Uses PO (portable object) files for translations
- intl extension for PHP (via libicu)
- IMO, both are cumbersome and difficult to use
  - I’ve not encountered a translation vendor that works with PO files

FORMAT PHP &
FORMAT JS

INTRODUCING
FormatPHP & FormatJS

- **FormatJS** - formatjs.io - Set of JavaScript libraries for use on the client & server (Node.js)
- **FormatPHP** - docs.formatphp.dev - userland PHP library that ports the functionality of FormatJS to PHP
  - Created by *me* for Skillshare; we use both FormatJS and FormatPHP
- Both based on ICU (International Components for Unicode)
- Both follow conventions of and provide polyfills for ECMA-402

GOALS FOR FORMAT PHP

Why did we create it?

- Already decided on FormatJS, a well-supported set of internationalization tools used by many in the JavaScript community.
- Wanted a set of PHP tools:
  - As easy to use as FormatJS.
  - Shared the same/similar APIs with FormatJS (reduce cognitive load).
  - Allowed us to use the same translation workflow and formats.

Nothing like it existed,
so we had to create it.

FEATURES
Of FormatPHP & FormatJS

- Ability to declare i18n-friendly messages (FormatJS, FormatPHP)
- Linter that enforces such messages (FormatJS, FormatPHP†)
- CLI for extraction & compilation (FormatJS, FormatPHP)
- Polyfills for ECMA-402 functionality (FormatJS, FormatPHP‡)
- Bundler plugin for compiling TypeScript/JavaScript (FormatJS)

† Not yet. I'd love to see a PR for a PHP_CodeSniffer "sniff" that provides this functionality. 😉
‡ Sort of. ECMA-402 is a specification for JavaScript, and FormatPHP provides some of this functionality.

GETTING STARTED

FormatPHP & FormatJS

```
yarn add @formatjs/intl @formatjs/cli

composer require skillshare/formatphp
```

FORMAT PHP
intl.php (part 1)

```php
use FormatPHP\{Config, FormatPHP, Intl, Message, MessageCollection};

$messagesInSpanish = new MessageCollection([
  new Message('myMessage', 'Hoy es {ts, date, ::yyyyMMdd}'),
]);

$intl = new FormatPHP(
  config: new Config(
    locale: new Intl\Locale('es-ES'),
    defaultLocale: new Intl\Locale('en-US'),
  ),
  messages: $messagesInSpanish,
);
```

FORMAT PHP
intl.php (part 2)

```php
echo $intl->formatMessage([
  'id' => 'myMessage',
  'defaultMessage' => 'Today is {ts, date, ::yyyyMMdd}',
], [
  'ts' => new DateTimeImmutable(),
]) . "\n";

echo $intl->formatNumber(19, new Intl\NumberFormatOptions([
  'style' => 'currency',
  'currency' => 'EUR',
])) . "\n";
```

FORMAT PHP

intl.php - output

```
❯ php intl.php
Hoy es 03/11/2023
19,00 €
```

FORMAT JS

intl.mjs (part 1)

```
import {createIntl} from '@formatjs/intl'

const messagesInSpanish = {
  myMessage: 'Hoy es {ts, date, ::yyyyMMdd}',
}

const intl = createIntl({
  locale: 'es-ES',
  defaultLocale: 'en-US',
  messages: messagesInSpanish,
})
```

FORMAT JS

intl.mjs (part 2)

```
console.log(intl.formatMessage({
  id: 'myMessage',
  defaultMessage: 'Today is {ts, date, ::yyyyMMdd}',
}, {
  ts: Date.now(),
}))

console.log(intl.formatNumber(19, {
  style: 'currency',
  currency: 'EUR',
}))
```

FORMAT JS

intl.mjs - output

```
❯ node intl.mjs
Hoy es 03/11/2023
19,00 €
```

FORMATTING STRINGS

DATES & TIMES

FormatPHP

```php
$date = new DateTimeImmutable('now');

echo $intl->formatDate($date); // e.g., "03/11/2023"
echo $intl->formatTime($date); // e.g., "22:31"
```

DATES & TIMES

FormatPHP

```php
echo $intl->formatDate($date, new Intl\DateTimeFormatOptions([
  'day' => 'numeric',
  'month' => 'short',
  'weekday' => 'short',
  'year' => 'numeric',
])); // e.g., "vie, 3 nov 2023"

echo $intl->formatTime($date, new Intl\DateTimeFormatOptions([
  'timeStyle' => 'full',
  'timeZone' => 'America/Chicago',
])); // e.g., "17:31:23 (hora de verano central)"
```

DATES & TIMES

FormatJS

```
const date = Date.now()

console.log(intl.formatDate(date)) // e.g., "03/11/2023"
console.log(intl.formatTime(date)) // e.g., "22:31"
```

DATES & TIMES

FormatJS

```
console.log(intl.formatDate(date, {
  day: 'numeric',
  month: 'short',
  weekday: 'short',
  year: 'numeric',
})) // e.g., "vie, 3 nov 2023"

console.log(intl.formatTime(date, {
  timeStyle: 'full',
  timeZone: 'America/Chicago',
})) // e.g., "17:31:23 (hora de verano central)"
```

NUMBERS
FormatPHP

```php
$number = -12_345.678;

echo $intl->formatNumber($number); // e.g., "-12.345,678"

echo $intl->formatNumber(1562.25, new Intl\NumberFormatOptions([
  'style' => 'unit',
  'unit' => 'kilometer',
])); // e.g., "1562,25 km"
```

NUMBERS

FormatJS

```
const number = -12_345.678;

console.log(intl.formatNumber(number)); // e.g., "-12.345,678"

console.log(intl.formatNumber(1562.25, {
  style: 'unit',
  unit: 'kilometer',
})) // e.g., "1562,25 km"
```

CURRENCY

FormatPHP

```php
echo $intl->formatCurrency(123.0, 'USD');
// e.g., "123,00 US$"

echo $intl->formatCurrency(
  value: 123.0,
  currencyCode: 'USD',
  options: new Intl\NumberFormatOptions([
    'currencyDisplay' => 'narrowSymbol',
    'trailingZeroDisplay' => 'stripIfInteger',
  ]),
); // e.g., "123 $"
```

CURRENCY

FormatJS

```
console.log(intl.formatNumber(123.0, {
  style: 'currency',
  currency: 'USD',
})); // e.g., "123,00 US$"

console.log(intl.formatNumber(123.0, {
  style: 'currency',
  currency: 'USD',
  currencyDisplay: 'narrowSymbol',
  trailingZeroDisplay: 'stripIfInteger',
})) // e.g., "123,00 $"
```

DISPLAY NAMES

FormatPHP

```php
echo $intl->formatDisplayName(
  'US',
  new Intl\DisplayNamesOptions(['type' => 'region']),
);
```

DISPLAY NAMES

FormatJS

```
console.log(intl.formatDisplayName('US', {
  type: 'region',
}))
```

MESSAGES
Notes

- FormatPHP & FormatJS support ICU message syntax.
- Starts out simple and can support very complex statements.
- ICU is currently working to update and define MessageFormat 2.
  - ECMA-402 is tracking the work of MessageFormat 2 closely.
  - Unclear how it will affect FormatJS and FormatPHP.

MESSAGES
FormatPHP

```php
$user = (object) ['name' => 'Bilbo'];

echo $intl->formatMessage([
  'id' => 'greeting',
  'description' => 'Greets a user after they log in.',
  'defaultMessage' => 'Hello, {personName}!',
], [
  'personName' => $user->name,
]);
```

```
echo $intl->formatMessage([
  'id' => 'sale',
  'description' => 'Explains how far people walked for the sale prices.',
  'defaultMessage' => <<<'EOD'
    On {actionDate, date, ::dMMMM} at {actionDate, time, ::jmm},
    they walked {distance, number, ::unit/kilometer unit-width-full-name .#}
    to pay only {amount, number, ::currency/EUR unit-width-short
    precision-currency-standard/w} in the
    {percentage, number, ::percent precision-integer} off sale
    on furniture.
    EOD,
], [
  'actionDate' => new DateTimeImmutable('now'),
  'distance' => 5.358,
  'amount' => 150.00123,
  'percentage' => 0.25,
]);
```

```
echo $intl->formatMessage([
  'id' => 'sale',
  'description' => 'Explains how far people walked for the sale prices.',
  'defaultMessage' => <<<'EOD'
    On {actionDate, date, ::dMMMM} at {actionDate, time, ::jmm},
    they walked {distance, number, ::unit/kilometer unit-width-full-name .#}
    to pay only {amount, number, ::currency/EUR unit-width-short
    precision-currency-standard/w} in the
    {percentage, number, ::percent precision-integer} off sale
    on furniture.
    EOD,
], [
  'actionDate' => new DateTimeImmutable('now'),
  'distance' => 5.358,
  'amount' => 150.00123,
  'percentage' => 0.25,
]);
```

```
echo $intl->formatMessage([
  'id' => 'sale',
  'description' => 'Explains how far people walked for the sale prices.',
  'defaultMessage' => <<<'EOD'
    On {actionDate, date, ::dMMMM} at {actionDate, time, ::jmm},
    they walked {distance, number, ::unit/kilometer unit-width-full-name .#}
    to pay only {amount, number, ::currency/EUR unit-width-short
    precision-currency-standard/w} in the
    {percentage, number, ::percent precision-integer} off sale
    on furniture.
    EOD,
], [
  'actionDate' => new DateTimeImmutable('now'),
  'distance' => 5.358,
  'amount' => 150.00123,
  'percentage' => 0.25,
]);
```

```
echo $intl->formatMessage([
  'id' => 'sale',
  'description' => 'Explains how far people walked for the sale prices.',
  'defaultMessage' => <<<'EOD'
    On {actionDate, date, ::dMMMM} at {actionDate, time, ::jmm},
    they walked {distance, number, ::unit/kilometer unit-width-full-name .#}
    to pay only {amount, number, ::currency/EUR unit-width-short
    precision-currency-standard/w} in the
    {percentage, number, ::percent precision-integer} off sale
    on furniture.
    EOD,
], [
  'actionDate' => new DateTimeImmutable('now'),
  'distance' => 5.358,
  'amount' => 150.00123,
  'percentage' => 0.25,
]);
```

```
echo $intl->formatMessage([
  'id' => 'sale',
  'description' => 'Explains how far people walked for the sale prices.',
  'defaultMessage' => <<<'EOD'
    On {actionDate, date, ::dMMMM} at {actionDate, time, ::jmm},
    they walked {distance, number, ::unit/kilometer unit-width-full-name .#}
    to pay only {amount, number, ::currency/EUR unit-width-short
    precision-currency-standard/w} in the
    {percentage, number, ::percent precision-integer} off sale
    on furniture.
    EOD,
], [
  'actionDate' => new DateTimeImmutable('now'),
  'distance' => 5.358,
  'amount' => 150.00123,
  'percentage' => 0.25,
]);
```

On November 3 at 5:35 PM, they walked 5.4 kilometers to pay only EUR 150.00 in the 25% off sale on furniture.

```
$host = (object) ['name' => 'Bilbo', 'gender' => 'male'];
$party = (object) ['guests' => [(object) ['name' => 'Frodo']]];

echo $intl->formatMessage([
  'id' => 'party',
  'description' => 'Tells who is inviting whom to their party.',
  'defaultMessage' => <<<'EOD'
    {hostGender, select,
      female {{numGuests, plural, offset:1
        =0 {{host} does not give a party.}
        =1 {{host} invites {guest} to her party.}
        =2 {{host} invites {guest} and one other person to her party.}
        other {{host} invites {guest} and # other people to her party.}
      }}
      male {{numGuests, plural, offset:1
        =0 {{host} does not give a party.}
        =1 {{host} invites {guest} to his party.}
        =2 {{host} invites {guest} and one other person to his party.}
        other {{host} invites {guest} and # other people to his party.}
      }}
      other {{numGuests, plural, offset:1
        =0 {{host} does not give a party.}
        =1 {{host} invites {guest} to their party.}
        =2 {{host} invites {guest} and one other person to their party.}
        other {{host} invites {guest} and # other people to their party.}
      }}
    }
    EOD,
], [
  'hostGender' => $host->gender,
  'host' => $host->name,
  'numGuests' => count($party->guests),
  'guest' => $party->guests[0]->name,
]);
```

```
'defaultMessage' => <<<'EOD'
  {hostGender, select,
    female {{numGuests, plural, offset:1
      =0 {{host} does not give a party.}
      =1 {{host} invites {guest} to her party.}
      =2 {{host} invites {guest} and one other person to her party.}
      other {{host} invites {guest} and # other people to her party.}
    }}
    male {{numGuests, plural, offset:1
      =0 {{host} does not give a party.}
      =1 {{host} invites {guest} to his party.}
      =2 {{host} invites {guest} and one other person to his party.}
      other {{host} invites {guest} and # other people to his party.}
    }}
    other {{numGuests, plural, offset:1
      =0 {{host} does not give a party.}
      =1 {{host} invites {guest} to their party.}
      =2 {{host} invites {guest} and one other person to their party.}
      other {{host} invites {guest} and # other people to their party.}
    }}
  }
  EOD,
```

```
'defaultMessage' => <<<'EOD'
  {hostGender, select,
    female {{numGuests, plural, offset:1
      =0 {{host} does not give a party.}
      =1 {{host} invites {guest} to her party.}
      =2 {{host} invites {guest} and one other person to her party.}
      other {{host} invites {guest} and # other people to her party.}
    }}
    male {{numGuests, plural, offset:1
      =0 {{host} does not give a party.}
      =1 {{host} invites {guest} to his party.}
      =2 {{host} invites {guest} and one other person to his party.}
      other {{host} invites {guest} and # other people to his party.}
    }}
    other {{numGuests, plural, offset:1
      =0 {{host} does not give a party.}
      =1 {{host} invites {guest} to their party.}
      =2 {{host} invites {guest} and one other person to their party.}
      other {{host} invites {guest} and # other people to their party.}
    }}
  }
  EOD,
```

```
'defaultMessage' => <<<'EOD'
  {hostGender, select,
    female {{numGuests, plural, offset:1
      =0 {{host} does not give a party.}
      =1 {{host} invites {guest} to her party.}
      =2 {{host} invites {guest} and one other person to her party.}
      other {{host} invites {guest} and # other people to her party.}
    }}
    male {{numGuests, plural, offset:1
      =0 {{host} does not give a party.}
      =1 {{host} invites {guest} to his party.}
      =2 {{host} invites {guest} and one other person to his party.}
      other {{host} invites {guest} and # other people to his party.}
    }}
    other {{numGuests, plural, offset:1
      =0 {{host} does not give a party.}
      =1 {{host} invites {guest} to their party.}
      =2 {{host} invites {guest} and one other person to their party.}
      other {{host} invites {guest} and # other people to their party.}
    }}
  }
  EOD,
```

Bilbo invites Frodo to his party.

MESSAGES
FormatJS

```
const user = {name: 'Bilbo'}

console.log(intl.formatMessage({
  id: 'greeting',
  description: 'Greets a user after they log in.',
  defaultMessage: 'Hello, {personName}!',
}, {
  personName: user.name,
}))
```

```
console.log(intl.formatMessage({
  id: 'sale',
  description: 'Explains how far people walked for the sale prices.',
  defaultMessage: `
    On {actionDate, date, ::dMMMM} at {actionDate, time, ::jmm},
    they walked {distance, number, ::unit/kilometer unit-width-full-name .#}
    to pay only {amount, number, ::currency/EUR unit-width-short
    precision-currency-standard/w} in the
    {percentage, number, ::percent precision-integer} off sale
    on furniture.`,
}, {
  actionDate: Date.now(),
  distance: 5.358,
  amount: 150.00123,
  percentage: 0.25,
}))
```

```
const host = {name: 'Bilbo', gender: 'male'}
const party = {guests: [{name: 'Frodo'}]}

console.log(intl.formatMessage({
  id: 'party',
  description: 'Tells who is inviting whom to their party.',
  defaultMessage: `
    {hostGender, select,
      female {{numGuests, plural, offset:1
        =0 {{host} does not give a party.}
        =1 {{host} invites {guest} to her party.}
        =2 {{host} invites {guest} and one other person to her party.}
        other {{host} invites {guest} and # other people to her party.}
      }}
      male {{numGuests, plural, offset:1
        =0 {{host} does not give a party.}
        =1 {{host} invites {guest} to his party.}
        =2 {{host} invites {guest} and one other person to his party.}
        other {{host} invites {guest} and # other people to his party.}
      }}
      other {{numGuests, plural, offset:1
        =0 {{host} does not give a party.}
        =1 {{host} invites {guest} to their party.}
        =2 {{host} invites {guest} and one other person to their party.}
        other {{host} invites {guest} and # other people to their party.}
      }}
    }`,
}, {
  hostGender: host.gender,
  host: host.name,
  numGuests: party.guests.length,
  guest: party.guests[0].name,
}))
```

TRANSLATION

EXTRACTING STRINGS

FormatPHP

- Extract all the message strings from PHP source files:

```
❯ ./vendor/bin/formatphp extract \
    --out-file=locales/en.json \
    '**/*.php' \
    '**/*.phtml'

[notice] Message descriptors extracted and written to locales/en.json
```

EXTRACTING STRINGS

FormatJS

- Extract all the message strings from JS source files:

```
❯ yarn formatjs extract '**/*.mjs' --out-file locales/en2.json
yarn run v1.22.19
$ /path/to/project/node_modules/.bin/formatjs extract '**/*.mjs' --out-file locales/en2.json
✨  Done in 0.59s.
```

```
{
  "greeting": {
   "defaultMessage": "Hello, {personName}!",
   "description": "Greets a user after they log in."
  },
  "myMessage": {
    "defaultMessage": "Today is {ts, date, ::yyyyMMdd}"
  },
  "party": {
    "defaultMessage": "{hostGender, select, female {{numGuests, plural, o
    "description": "Tells who is inviting whom to their party."
  },
  "sale": {
    "defaultMessage": "On {actionDate, date, ::dMMMM} at {actionDate, tim
    "description": "Explains how far people walked for the sale prices."
  }
}
```

EXTRACTING STRINGS

Notes

- Extract from PHP and JS source to two different locales files.
- These are your “default” locale strings.
- Consider merging these files, but be careful when IDs & messages overlap.
- Merging isn’t necessary, though.
- Send these files to your translators.

TMS
Translation Management Systems

- TMSs are what translators use to translate your messages.
- FormatPHP and FormatJS support extracting messages into JSON format supported by a wide-range of TMSs.
- You can automate the process of sending receiving locale translations.
  - e.g., CrowdIn supports automatic workflows that open GitHub PRs on your feature branches when your default locale files have changes.

TMS
Translation Management Systems

```
❯ ./vendor/bin/formatphp extract \
    --format=crowdin \
    --out-file=locales/en.json \
    '**/*.php' \
    '**/*.phtml'

❯ yarn formatjs extract '**/*.mjs' \
    --format crowdin \
    --out-file locales/en2.json
```

TMS
Translation Management Systems

TMS	`--format`
Smartling	`smartling`
Lingohub	`simple`
Phrase	`simple`
Crowdin	`crowdin`
SimpleLocalize	`simple`
POEditor	`simple`
Localize	`simple`
locize	`simple`

LOADING STRINGS

FormatPHP

```php
use FormatPHP\{Config, FormatPHP, Intl, MessageLoader};

$config = new Config(new Intl\Locale('es-419'));

$messageLoader = new MessageLoader(
  // The path to your locale JSON files (i.e., en.json, es.json, etc.).
  '/path/to/app/locales',
  $config,
);

$intl = new FormatPHP($config, $messageLoader->loadMessages());
```

TESTING
With pseudo-locales

Locale	Message
en-XA	ṁẏ ńâṁè íś {name}
en-XB	[!! ṁẏ ńâṁṁṁè íííś !!]{name}
xx-AC	MY NAME IS {name}
xx-HA	[javascript]my name is {name}
xx-LS	my name is {name}SSSSSSSSSSSSSSSSSSSSSSSS

LOOKING AHEAD

LOOKING AHEAD

- ICU MessageFormat 2
- ECMA-402 - Internationalization API Specification
- PHP-FIG: PSR-21 - Common Interfaces and Functionality for Interoperability of Message Translation and Formatting
- PECL ecma_intl extension - Ports ECMA-402 to PHP

THANK YOU!

Keep in touch

- ben.ramsey.dev
- phpc.social/@ramsey
- github.com/ramsey
- speakerdeck.com/ramsey
- www.linkedin.com/in/benramsey
- ben@ramsey.dev

joind.in/talk/3e1e4

ATTRIBUTION

- Fonts
  - Archivo Black by Omnibus-Type, SIL Open Font License, Version 1.1
  - DM Mono by Colophon Foundry, SIL Open Font License, Version 1.1
  - Playfair Display by Claus Eggers Sørensen, SIL Open Font License, Version 1.1
  - Saira by Omnibus-Type, SIL Open Font License, Version 1.1
- OpenMoji
  - “winking face” by Emily Jäger, CC BY-SA 4.0
  - “weary cat” by Emily Jäger, CC BY-SA 4.0