jsxlate

message extraction and translation for React

who am I?

Eric O'Connell

- I work at idealist.org
- I <3 compilers
- translation software is a theme

internationalization

(after typing that, i can see why they call it i18n)

what is it?

- **Internationalized** software supports the languages and cultural customs of people throughout the world. The Web reaches all parts of the world. Internationalized web apps provide a great user experience for people everywhere.

- **Localized** software adapts to a specific language and culture by translating text into the user's language and formatting data in accordance with the user's expectations. An app is typically localized for a small set of locales.

translation

- Hello → Hola
- Hello, {name} → Hola, {name}
- There are {count} fish. →
  Hay {count} pez., Hay {count} peces.
- !!!

translation, cont'd.

```
{gender_of_host, select,
  female {
    {num_guests, 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 {
    {num_guests, 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 {
    {num_guests, 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.}}}}
```

what about HTML?

- how do you translate
  - `<p className=“awesome”>You are {awesome} <img src=“/imgs/en/awesome.gif” alt=“You are awesome”/>!</p>`
- It might look like:
  - `<p className=“awesome”>¡Usted es {awesome} <img src=“/imgs/es/awesome.gif” alt=“¡Usted es increíble!”/>!</p>`

the problem

- if you have
  - `<p className=“awesome”>You are {awesome} <img src=“/imgs/en/awesome.gif” alt=“You are awesome”/>!</p>`
- other tools might ask you to translate:
  - `“You are “`
  - `“You are awesome”`
  - `“!”`

context!

- translators need context to do their work!
- jsxlate would ask them to translate:
  - `You are {awesome} <img src=“/imgs/en/awesome.gif” alt=“You are awesome”/>!`

demos!

how does it work

how to jsxlate

- mark the messages
- extract messages & send to translators
- create localized bundles of messages per locale
- wire up locale & messages to app
- transform the app source

mark up msgs

- wrap (non-DOM) strings in calls to i18n:
  - `i18n(You look nice today)`
- wrap React strings in the <I18N> component:
  - `<I18N>You look <em>nice</em> today</I18N>`
- when messages need to be pluralized:
  - `<Pluralize on={count}><Match when=one>…`

extract messages

- `$(npm bin)/extract-messages $(find src/ -name ‘*.js?’) > messages.json`
  - (sorry)
- Upload this JSON to your translation service
  - the format is currently compatible with transifex
  - we’d like to output .POT files as well because everything seems to support gettext

create bundles

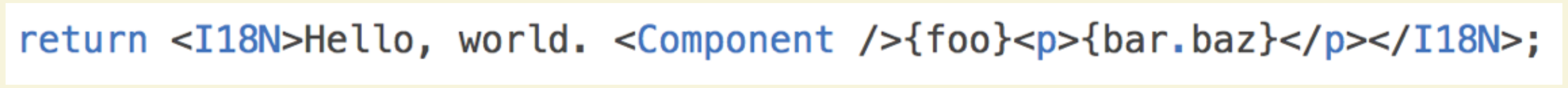
- now you have translated strings — yay!
- generate yo’ bundles:
  - `$(npm bin)/bundle-messages -t translations-es.json $(find src/ -name ‘*.js?’) > bundle-es.js`

hook up locale/messages

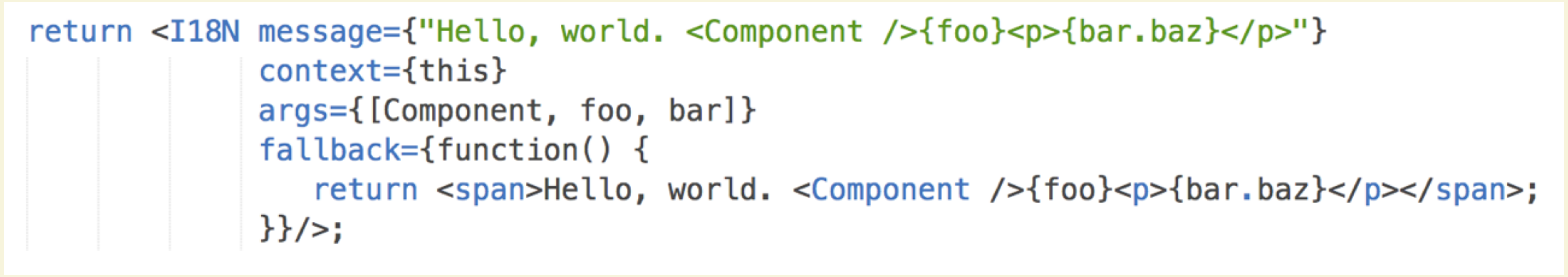
- `import {setMessages, setLocale}`
- `let bundle = require(‘./i18n/bundle-‘ + locale);`
- `setMessages(bundle);`
- `setLocale(locale);`

transform!

- the places you called i18n() or <I18N> need to be transformed
- webpack: jsxlate-loader
- other build systems:
  - `$(npm bin)/transform < foo.jsx > out/foo.jsx`
  - `require(‘jsxlate’).transformMessages(src);`



```
return <I18N>Hello, world. <Component />{foo}<p>{bar.baz}</p></I18N>;
```



```
return <I18N message={"Hello, world. <Component />{foo}<p>{bar.baz}</p>"}
             context={this}
             args={[Component, foo, bar]}
             fallback={function() {
                return <span>Hello, world. <Component />{foo}<p>{bar.baz}</p></span>;
             }}/>;
```

other i18n solutions

we evaluated:

- FormatJS / React Intl (Yahoo)
- react-translate-component
- jsx-i18n
- react-i18n

React Intl

- there is a lot of nice work in React Intl
- recommended for number/date/time formatting
- ICU MessageFormat

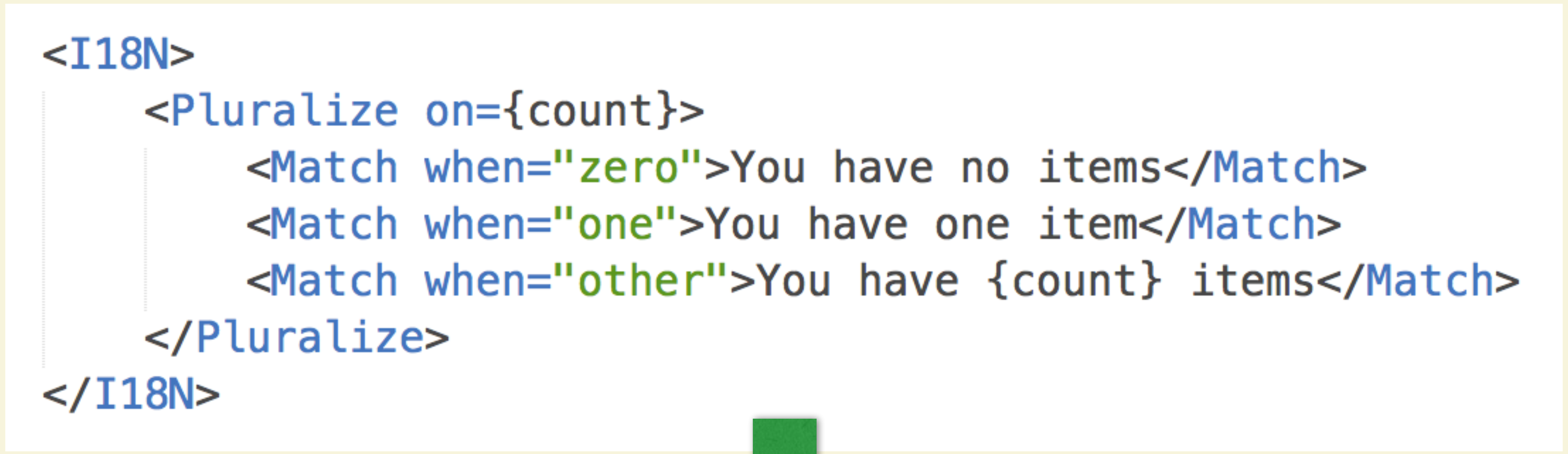
others

- mostly lacking in documentation
- none support nested tags
- jsx-i18n does support extraction
- meh?

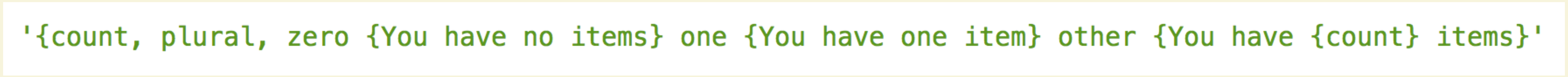
TODO.md

pluralization

- top priority is to add support for pluralization:



```
<I18N>
    <Pluralize on={count}>
        <Match when="zero">You have no items</Match>
        <Match when="one">You have one item</Match>
        <Match when="other">You have {count} items</Match>
    </Pluralize>
</I18N>
```



```
'{count, plural, zero {You have no items} one {You have one item} other {You have {count} items}'
```

optimization

- several things can be made faster
- jsxlate-loader is slooow, rewrite as babel plugin
- more profiling to do

docs & examples

- show deeper integration with React Intl/FormatJS
- examples using grunt/gulp/broccoli
- browserify -t jsxlatify
- react native..?

& more!

- support more attributes & configuration
  - `placeholder, alt, etc.`
- locale fallback support
- alternative message syntax

Thanks

impossible without:

- Dave McCabe, for writing the original react-extract
- FormatJS & React Intl, for doing a lot of the hard work
- Idealist, for sponsoring this work

@compassing

github.com/drd