OBJECT
COMPUTING

WEBINAR

Internationalization Using web-translate

Mark Volkmann, Partner and Principal Software Engineer
mark@objectcomputing.com

GOAL

- Translate text displayed in web applications to multiple languages
- Lookup translations as a build step, not at run-time
  - so translation costs are not incurred during web app usage
- Keep it simple!
- Use open source

SOLUTION

- Use web-translate at https://www.npmjs.com/package/web-translate
- Set of JavaScript functions and a command
- Not specific to any framework
  - will work with React, Vue, Angular, ...
- English-centric
  - translations are driven from initial text that is English

WEB-TRANSLATE GOALS ...

- Determine required translations by
  - parsing source files for calls to a certain function
  - parsing a JSON file containing English translations
- Use either of the most popular translation services
  - Google Cloud Translate API and Yandex Translate Service
- Easily generate additional translations when new languages need to be supported

... WEB-TRANSLATE GOALS

- Allow generated translations to be overridden
  - by manually creating language-specific JSON files that describe overrides
- Support build-time translation
  - to avoid incurring run-time translation costs during each user session
- Support run-time translations
  - for cases when dynamically generated text must be translated

API KEYS

- Google Cloud Translation API and Yandex Translate Service both require an API key
- For Google, see steps at https://www.npmjs.com/package/web-translate#no-free-lunch
- For Yandex, see https://tech.yandex.com/translate/

GOOGLE CLOUD TRANSLATION API

- Requires setup of a Google Cloud Platform (GCP) project
  - steps documented at https://www.npmjs.com/package/web-translate#no-free-lunch
- $20 (USD) per million characters translated

YANDEX TRANSLATE SERVICE

OBJECT COMPUTING

- Has free and commercial tiers
- Free tier
  - 1,000,000 characters per day, but not more than 10,000,000 per month
- Commercial tier
  - $15 (USD) per million characters translated up to 50 million
  - rates go down slowly above that

see details at https://translate.yandex.com/developers/offer/prices

WEB-TRANSLATE SETUP

1. **`npm install web-translate`**
2. Set environment variable **`TRANSLATE_ENGINE`** to **`google`** or **`yandex`** (defaults to **`yandex`** because that has a free tier)
3. Set environment variable **`API_KEY`** to API key for desired service
4. Add following npm script in **`package.json`** file for application

```
"gentran": "generate-translations",
```

SUPPORTED LANGUAGES ...

OBJECT COMPUTING

- Create file **`languages.json`** to list languages to be supported
- Specifies language codes like **`en`**
  - see list at https://www.wikipedia.org/wiki/List_of_ISO_639-1_codes/
- Example

```
{
  "Chinese": "zh",
  "English": "en",
  "French": "fr",
  "German": "de",
  "Russian": "ru",
  "Spanish": "es"
}
```

locale variations such as differences between **`en-US`** and **`en-GB`** are not currently supported

... SUPPORTED LANGUAGES

OBJECT COMPUTING

- **`languages.json`** must be accessible at web app domain
  - example: if web app is running on **`localhost:3000`**,
    an HTTP GET request to **`localhost:3000/languages.json`**
    must return content of this file

GETTING TRANSLATIONS ...

- Use **i18n** function to get translations

```
import {i18n} from 'web-translate';
```

  - example: when language is Spanish, calling **i18n('Hello')** might return **Hola**
  - string passed to **i18n** can be **English text or**
    a key used to lookup translation in a language-specific JSON file

... GETTING TRANSLATIONS

OBJECT COMPUTING

- Keys are useful for long phrases, sentences, and even paragraphs
- Example: `i18n('greet')` might return
  `"Welcome to my wonderful web application!"`
- Translations for keys must be defined in language-specific JSON files
  - example: file `en.json` could contain:

```
{
  "contact": "For more information, contact Mark Volkmann.",
  "greet": "Welcome to my wonderful web application!"
}
```

GENERATING TRANSLATIONS ...

- Manually looking up and recording translations can be very tedious
- Translations files for all languages to be supported can be generated by running **`npm run gentran`**
- Next two slides describe what this does

... GENERATING TRANSLATIONS ...

1. Get all languages to be supported from **`languages.json`**
2. Get all literal strings passed to **`i18n`** function
   in all source files under **`src`** directory
   that have extension of **`js`**, **`jsx`**, **`ts`**, **`tsx`**, or **`vue`**
3. Get all English translations from **`en.json`**

... GENERATING TRANSLATIONS

4. For each language to be supported except English ...
   - Set **`translations`** to a map of all translations found in an overrides file for current language (ex. **`fr-overrides.json`**); if none exist, **`translations`** begins empty
   - For each key in English translation file **`en.json`** ...
     - If there is not already a translation for this key, get translation from translation service and save in **`translations`**
   - For each literal string passed to **`i18n`** in source files ...
     - If there is not already a translation for this string, get translation from translation service and save in **`translations`**
   - Write **`translations`** to new translation file for current language

more on override files later

source of most translations at run-time

MORE TRANSLATION DETAILS

- Don't manually edit generate translation files
  - will be overwritten next time **`npm run gentran`** is run
- If **`i18n`** function is passed a variable instead of literal string
  - no translations will be provided, because
    tracing flow of code to find all possible values is a very hard problem
  - in these cases, manually enter desired translations in **`en.json`** file
  - translations for other supported languages are generated from this

ALLOWING USER TO SELECT LANGUAGE ...

- Import **`web-translate`** functions

```
import {
  getLanguageCode,
  getSupportedLanguages,
  i18n,
  setLanguage
} from 'web-translate';
```

- Get current language code

```
const languageCode = getLanguageCode();
```

defaults to browser setting in **`navigator.language`**

... ALLOWING USER TO SELECT LANGUAGE ...

- Get array of supported languages

```
const languages = await getSupportedLanguages();
```

- Render **`select`** element using supported languages as options with current language selected

... ALLOWING USER TO SELECT LANGUAGE

- When user selects a language ...

```
await setLanguage(languageCode);
```

- Cause current page to re-render
  - web framework specific

DYNAMIC TRANSLATION ...

OBJECT COMPUTING

- Some apps dynamically generate text that requires translation
- Accomplished with **`translate`** function
  - takes "from" language code, "to" language code, and text to be translated
- Example

could use value returned by **`getLanguageCode()`** here

```
const text = 'I like strawberry pie!';
const translatedText = await translate('en', 'fr', text);
```

... DYNAMIC TRANSLATION

- Requires **`TRANSLATE_ENGINE`** and **`API_KEY`** environment variables to be set in environment where web app is running
- Run-time translation incurs per user session charges from selected translation service, so **avoid** when possible

EXAMPLE SCENARIO ...

OBJECT COMPUTING

- We'll begin with these files

languages.json

```
{
  "English": "en",
  "French": "fr",
  "Spanish": "es"
}
```

en.json

```
{
  "some-key": "My English key"
}
```

es.json

```
{
  "Hello": "Hola"
}
```

... EXAMPLE SCENARIO ...

OBJECT COMPUTING

- Render translations for **`Hello`** and **`some-key`** in UI code with **`i18n('Hello')`** and **`i18n('some-key')`**
- Before running **`npm run gentran`**,
  if English is selected language,
  results are **`Hello`** and **`My English Key`**

... EXAMPLE SCENARIO ...

OBJECT COMPUTING

- Changing language to French
  - gives **`Hello`** and **`some-key`**, because file **`fr.json`** does not exist yet
- Changing language to Spanish
  - gives **`Hola`** and **`some-key`**
  - **`Hola`** because it's the translation for **`Hello`** in **`es.json`**
  - **`some-key`** because **`es.json`** does not yet provide a translation for that key

... EXAMPLE SCENARIO ...

OBJECT COMPUTING

- Run **`npm run gentran`**
  - generates **`es.json`** and **`fr.json`** which now contain translations for **`Hello`** and **`some-key`**
  - when language is French, we get **`Bonjour`** and **`Ma clé anglaise`** which are French translations for **`Hello`** and **`My English Key`**
  - when language is Spanish, we see still **`Hola`**, but we now see Spanish translation for **`My English Key`** which is **`Mi clave en inglés`**

... EXAMPLE SCENARIO ...

- Add lines to **`languages.json`**

```
"German": "de",
"Russian": "ru",
```

- Run **`npm run gentran`** again
- Translations for those languages are now available

... EXAMPLE SCENARIO

- Add following call in web UI code

```
i18n('Where is your pencil?')
```

- Run **`npm run gentran`** again
- Translations for that text are now available for all languages being supported

OVERRIDING TRANSLATIONS ...

- Sometimes translations provided by translation service are not ideal for the application
- To override translations,
  create language-specific **`-override.json`** files
  and run **`npm run gentran`** again

... OVERRIDING TRANSLATIONS

OBJECT COMPUTING

- Example
  - to use **Oh La** instead of **Hola** for Spanish translation of **Hello**, create file **es-override.json** with following content:

```
{
  "Hello": "Oh La"
}
```

  - run **npm run gentran** again
  - set language to Spanish and new translation will be rendered

PLACEHOLDERS

- Translation text can contain placeholders for inserting dynamic text
- Text outside placeholders will be translated
- Example: **`en.json`** could contain:

```
  "greet": "Hello ${name}, today is ${dayOfWeek}.",
```

- To use this, pass second argument to **`i18n`** that is an object where keys are placeholder names and values are values to be inserted

```
{i18n('greet', {name: 'Mark', dayOfWeek: 'Tuesday'});
```

HTML TAGS

- Translation text can contain HTML tags that are not translated
- Text outside tags will be translated
- Example: **`en.json`** could contain:

```
"greet": "<i>Please</i> be <b>careful<b>!",
```

WRAP UP

- web-translate provides the simplest approach to language translations in web applications I have seen!
- Please post suggestions for improvements and any issues at https://github.com/mvolkmann/web-translate/issues

LEARN MORE ABOUT OCI EVENTS AND TRAINING

Events:

- objectcomputing.com/events

Training:

- objectcomputing.com/training
- grailstraining.com
- micronauttraining.com

Or email info@ocitraining.com to schedule a custom training program for your team online, on site, or in our state-of-the-art, Midwest training lab.

OBJECT
COMPUTING

CONNECT WITH US

1+ (314) 579-0066

@objectcomputing

objectcomputing.com