# Pseudolocalization

*This document is published to the web in the public* [*Angular Design Docs*](https://drive.google.com/#folders/0BxgtL8yFJbacUnUxc3l5aTZrbVk)*/*[*i18n*](https://drive.google.com/folderview?id=0BxyFnbmFLIVxfkpVQzMtcjlTb00zM0xaS0dISXZ4RllnSmp4NGJ6WmNsS21SOFM5ckltZjQ) *folder*

*Chirayu Krishnappa <chirayu@google.com>  
Igor Minar <iminar@google.com>  
Status: Draft  
  
Last updated: Jan 26, 2015*

## What is Pseudolocalization?

Pseudolocalization is the process of localizing/translating from the primary language to a "fake" language. For instance, one could translate English into fake English by replaceing all the English characters with accented versions, or reversing the words, etc.

## Why use Pseudolocalization?

It's primarily useful for testing localization. As a developer, you want to see how your application looks in different locales. Requiring a full translation to test is both expensive, time consuming and does not cover sufficient cases (unless you're testing every single locale.) With pseudolocalization, you can run a tool to exercise various fake translations that can reveal potential problems in translated locales. It doesn't cover all the cases either, but it can go a long way.

## What's tested by Pseudolocalization?

* Spacing. Translated messages can be longer, sometimes significantly longer, than their originals. The UI should have sufficient room or rules to accomodate longer messages.
* Catch untranslated messages. The pseudolocalized messages stand out making it easy to pick out messages that aren't being pick up for translation.
* Catch messages that are incorrectly translated. For instance, words like "POP" in "POP/POP3 mail" should not be translated. They should be replaced with an opaque placeholder in the extracted message to protect against translation.
* UI behavior with BiDi text

## Forms of Pseudolocalization

**NOTE**: You can combine multiple of these forms and in different orders to generate a few standard pseudolocales for testing your application.

* Adding accents. Replace every English word with an accented version. You could pick the same kind of accent all the time or choose different kinds. e.g "The quick brown fox jumped over the lazy dog" becomes "T̈ḧë q̈üïc̈k̈ b̈r̈öẅn̈ f̈öẍ j̈üm̈p̈ëd̈ öv̈ër̈ ẗḧë l̈äz̈ÿ d̈ög̈".
* Increasing the word length. You could repeat words to double them. You can also combine it with adding accents or by capitalizing the added word. e.g "Hello, World" could become "Hello HELLO, World WORLD" or "Ḧël̈l̈ö HELLO, Ẅör̈l̈d̈ WORLD". Another method would be to replace specific known problematic words that translate into long phrases in some locales with appropriately sized ones.
* Add brackets around word. e.g. "Hello" → "[Hello]".
* Add colorful emoji's to make them stick out even more. e.g. "Hello" to "Ḧël̈l̈ö 🎍🎎🎒🎒🎓" when combined with repeating the word. This can more easily highlight areas where the text is not highlighted. (If your browser can’t display emoji, this may not work for you. screenshot: )
* Replace original text with pseudo text in a directionally flipped language. e.g. Use an RTL language when the original text is in a LTR language.

**NOTE**: While pseudolocalization strictly works at the translation layer (producing translations for original messages) for testing, you can combine it with different styles when viewing in a testing mode (you can generate special test builds) that apply different style such as a border or background color to highlight translated messages, etc.

See also: [code.google.com/pseudolocalization-tool](https://code.google.com/p/pseudolocalization-tool/)

## Pseudo locale names

You would test pseudo locales by using the locale switcher in your application to change the locale in your application's UI to pick a pseudo locale. (You could support these fake locales in just your development environment.) Or you could also have the development server automatically override your locale selection based on some debug query param to a pseudo locale. Regardless, you are now faced with having to make up some names for your pseudo locales. Here are some options:

### Use the reserved namespace

For example, you could make up locale names like en-x-pseudoaccented, en-x-pseudodoubled, etc.

The single-letter subtag, x, introduces a sequence of private use subtags ([RFC 5646: Tags for Identifying Languages](https://tools.ietf.org/html/rfc5646))

This has the downside that some tools don't support private use subtags. If you need to use such tools, this is not an option.

### Make up non-conflicting locale names

You could use locale names like en-pseudoaccented, en-pseudodoubled, etc. that are not recognized but are unlikely to ever conflict with real locales.