{gender*render}
Specification

Template system and implementation specification for rendering gender-neutral email templates with pronoun information

phseiff
from phseiff.com

November 13, 2020

Contents

1 Abstract **1**

2 Requirements **2**

3 Design Decisions **3**

4 Standard **4**
4.1 Template Language 4

1 Abstract

Our society, as well as the way we perceive gender, are steadily evolving. This evolution does not hold in light of technological questions, and it is our -the "IT people"'s duty- to address and do our best to solve the social issues that arise from our technology. One such technology are email- and other text templates, which are becoming increasingly popular to automate customer interactions of any kind, be it in newsletters, notifications or program menus. Many such templates are gender-specific, in that they address the reader in a gendered fashion

("Dear Mrs. Dursley, ..."). Such templates are relatively easily implemented by providing two versions of the email, one for every binary gender. However, some texts are far more complicated, because they address multiple people (each with their own unknown at the time of writing), or people in the third person (throwing their pronouns into the mix). In addition, an increasingly height amount of people use non-binary pronouns, or gender-neutral pronouns, many of whom might now yet be discovered at the time of writing, which makes these people marginalized when it comes to being correctly addressed even in automated emails.

This creates the requirement for creating template systems for the english language, and, in extention, any language (since all languages work differently), that support writing complex texts in a gender-neutral fashion and later "render" them to correctly gendered texts.

{gender*render} is an attempt at creating one such template language, including a Specification, to serve as a proof of concept as well as a starting point for people who want to implement similar things. The vision behind this proof of concept is not only to show *how* addressing people with unconventional preferred pronouns can be automized, but also to show *that* it can be easily automized, to debunk the myth that properly addressing nonbinary people in an automated fashion is simply technically impossible.

2 Requirements

There are multiple requirements for such a template language, whom I will list here, including short explanation of why they are required wherever I deem it necessary:

- The language must be easy to use even for less tech affine people. This means that the atoms of the language, such as tags et cetera, must be as short as possible, and should not clash with commonly used words or signs, so the amount of escape characters the user needs to use is minimal.
- The language must support different scenarios:
  - One person being addressed versus multiple people being addressed
  - Only people mentioned in first person, only people mentioned in third person, or a mixture of both
  - Everyone using pronouns versus some people preferring not to use any pronouns
- The fact that multiple scenarios are supported may not make using the template language for only a subset of them more complicated that it needs to be.

- Rendering templates may only require the information needed for rendering the template. For example, rendering a template that never addresses anyone in the first person should not require providing information as to whether the person goes by "Mr", "Mrs" or any other form of address. This is especially relevant since users do not want and should not need to require more information that necessary for rendering the templates, especially considering the intimate nature of preferred pronouns.
- The syntax should be describable using a context-free grammar in conjunctive normal form, which allows easy syntax checking and syntax highlighting.
- The data containing a persons preferred pronouns should be given in a widely-used, standardized format, such as JSON.

3 Design Decisions

The following decisions where made based on the the technical requirements ruled out in the corresponding section:

- The language uses a syntax similar to pythons build-in string formatting syntax, using curly brackets to annotate gender-specific parts of a sentence. Backslashes are used as escape characters for the rare occurrences where curly brackets are actually needed.
- In addition to terms like "possessive pronoun", using the gender-neutral form ("their") in tags is supported, potentially making texts more fluid to write and easier to read in their un-rendered form.
- If tags contain IDs to annotate which person is referred to, a mapping of IDs to pronoun preferences is accepted for rendering. If no such IDs are added to the document because only one person of unknown gender is addressed in the document, the pronoun preferences are directly accepted by the renderer, without having to be part of a person-to-pronoun-mapping. This supports referring to multiple persons in one text without making the writing of texts that refer to only one person any more troublesome.
- The pronoun information is given to the renderer as a piece of JSON data (or a similar object if the language used by the implementation supports such objects, e.g. dicts in Python). Information that is not required by the template may be left out in the template.
- Templates can be parsed before being rendered and then used for multiple renderings. This should debunk the idea that gender-sensitive template systems are to inefficient to use them.

These design decisions contain only those that are relevant to the requirements listet in the previous section; in-depth explanation and definition of the way the template system works are given in the next section.

4 Standard

This section contains the actual standard. It is divided into three subsections; one for defining the template language and how gender-neutral texts are described with it, one for defining the data structure used to describe the pronoun preferences of all people mentioned in a template, and one for guidelines and specification on implementing a renderer for the template language.

4.1 Template Language

Any text that follows the syntax of the following definition is considered a valid {gender*render}-template. Any text that does not follow the following is not considered a valid {gender*render}-template. Files whose content is a valid {gender*render}-template are referred to as files containing {gender*render}-templates in the following section, and *not* as {gender*render}-templates on their own. It is recommended to save such files with the file type `.grt` (short for "gender render template").

The purpose of {gender*render}-templates is to write texts in a gender-neutral way (at least in regards to some of the individuals they refer to), and to be valid input for the {gender*render}-renderer, who is described in a later section.

{gender*render}-templates may contain an arbitrary number (including zero) of {gender*render}-tags. A {gender*render}-tag is defined a sequence of characters that starts with an unescaped left curly bracket ("{", `U+007B`) and ends with an unescaped right curly bracket ("}", `U+007D`) without containing any unescaped curly brackets (`U+007B` as well as `U+007D`) in between. The purpose of {gender*render}-tags is to describe gender-specific sentence components in a gender-neutral fashion, these usually being mentions of a person in the third person singular.

A character is considered escaped if it is proceeded by an unescaped backslash ("\", `U+005C`) or by a backslash which is not proceeded by other backslash. A backslash which is not escaped is called an escape-character. A template which contains backslashes which are neither escaped nor escape characters is not considered a valid {gender*render}-template, as is any template which contains unescaped curly brackets who are not part of any valid {gender*render}-tag.

Every character of a {gender*render}-tag except the first and last characters (the brackets) is considered part of its content. Said content is divided into sections through unescaped asterisks ("*", `U+002A`). A section of a {gender*render}-tag does not contain any unescaped asterisks, and it must

contain at least one non-whitespace[1] character. Colons (":", `U+003A`) are considered special characters in sections, and may thus appear at most once per section, and neither as the first nor as the last non-whitespace character of the section. If a section contains a colon, the characters of the section beforehand the colon (minus all leading or trailing whitespace) are called the sections *type descriptors*, and the characters following the colon (after having all their whitespace collapsed into one `U+0020`-space each, except for trailing and leading whitespace, which is removed completely) are called the sections *value*. If a section does not contain a colon, its *value* is defined as all of its characters (having all their whitespace collapsed into one `U+0020`-space each, except for trailing and leading whitespace, which is removed completely).

There are multiple different types of section, assigned to sections by their type descriptor. A section whose type is "foo" is called a "foo-section". Every type of section has a unique priority, as a real number between 0 and 1000, assigned by this specification. The right-most section with no type descriptor and no assigned section type is assigned the section type with the highest priority of all section types that no section of the tag is assigned by this rule or its section descriptor yet. Every {gender*render}-tag must have at least one section, and may only have one section of every type; this takes into account the assigned section type of sections without a type descriptor. In addition, a tag may not contain more sections than there are section types defined by the spec.

The most basic type of section is the `context`-type, which describes the syntactic context of the {gender*render}-tag. Every {gender*render}-tag must have one context-section. The following table lists the possible values a `context`-section's value may have, as well as their meanings, though the syntactic validity of the template does not depend on whether the values and types of the the {gender*render}-tags are listed in this spec:

syntactic context indicated by the value	possible values, comma separated	short explanation, where necessary
Subject	they, subject, subj	
Object	them, object, obj	
Dependant possessive Determiner	their, dposs, dpossessive	
Independent possessive Determiner	theirs, iposs, ipossessive	
Reflexive	themself, reflexive, reflex	

[1] "Whitespace" as defined by the HTML Living Standard.

Form of Address	Mr, Mrs, Mr_s, address	
Surname	Smith, name, surname, family-name	(It should be mentioned that Smith is the most common US surname[2])
Personal name	Avery, personalname, first-name	(It should be mentioned that Avery is the most popular unisex name in the US today[3])
Any Noun	*any nominative*, with whitespaces replaced by hyphens ("-", U+002D)	If the value of the `section` does not match any of the above, its content is understood as being a noun which either server as a substitution or as a description of a person. For example, the sentence "{name} is an {actor}" or "the {actor} asked for applause" would be good candidates for using said type of value since "actor" has two different gendered forms ("actor" and "actress") in english.

The priority the `context`-section type is `1000`. If the `context`-section's value contains multiple strings, each separated from each other by whitespace, such as "{`foo:bar * context:Mr_s Smith`}", the {gender*render}-tag is interpreted as if it was "{`foo:bar * context:Mr_s`}{`foo:bar * context:Smith`}".

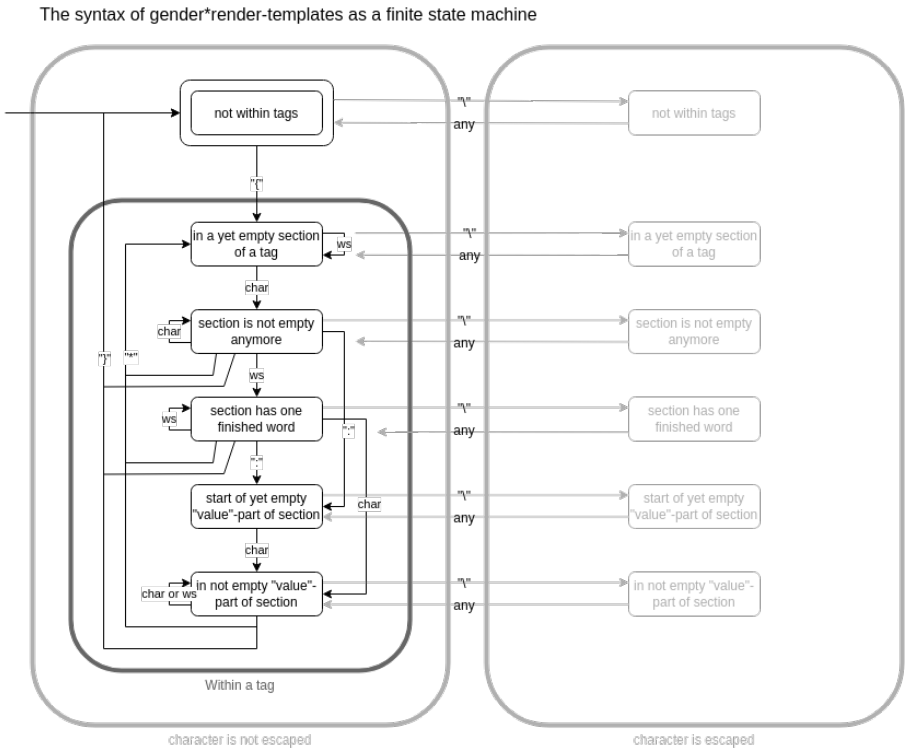
The other section type supported by this version of this Specification is the `id`-type. The value of an `id`-section may take any value as long as it does not contain any whitespace. It describes which individual the {gender*render}-tag refers to. Two {gender*render}-tags with the same id-value refer to the same individual. The id-value can be omitted by the user if there is only one individual mentioned in the whole template, and in some other cases; this is explored further in the renderer section. Whether there is an `id`-section is not part of the template specification, since it is not clear until the pronoun information is given.

Since there are only two section types defined by this specification, and one of them is mandatory, there is no practical need to use any section descriptors. They are still defined as a language feature in this template to provide a way to

[2] according to voanews.com
[3] according to nameberry.com

port the template language to other natural languages that might require additional information without having to introduce new syntax elements for every language.

To end this section of the spec, here is a graphic of the {gender*render}-template syntax described as a finite state machine (not taking into account the fact that not every section type is valid, and the rules about assigned sections and every type of section only existing once):



Explanation:
ws: Whitespace
char: Any character except whitespace, ":", "*", "{", "}", "\"
any: Any character

4.2 Pronoun Description Data