The Final Project: Gender Neutralizer

Aesthetic Programming
Digital Design - 3. Semester
Group 3
Mads Dixen, Jesper Vendelbo, Jakob Terkelsen, Thomas Matthiesen & Mark Nielsen
Characters: 14.536
Link to video: https://vimeo.com/268583940
Link to github: https://goo.gl/5zn7J8

Table of content

Introduction

We mainly want to focus on the topic of code and language in the digital culture. We have chosen to approach this topic by creating a program that can manipulate with what is visually and literarily accessible to a user of a web-browser, who might use this software for social media, articles, forums, search engines, etc., and by engaging this digital world the users are exposed to different usage of language of which they do not have any control. We want to challenge the lack of choice you have of the content in the contemporary constant feeds and data streams that we experience on the web, by creating something that allows the individual to choose, and take control of what is shown to them and therefore make web content from external sources less powerful and influential. As a reference point, it can be thought of as the browser extension AdBlock (AdBlock, n.d.) that gets rid of all ads that are visually forced upon you while browsing the internet, or the Facebook Demetricator (Grosser, n.d.) that removes or changes all the metrics that are shown on Facebook, therefore granting you more control as an individual by partially giving you power over how the web is going to be viewed by you as a user. With our program, the "Gender Neutralizer" we want to address the problem, that if you are continually experiencing a factor in the same relations, it will act as a catalyzer for the creation of stereotypes. If we are constantly exposed to specific genders being related to specific statements, there is a good chance that we will automatically start to perceive them as being true.

Our program is an extension for the web browser. The extension is integrated in the browser and it is running alongside while the user is browsing different websites. Google Chrome says the following about the definition of a browser extension: *"Extensions are small software programs that customize the browsing experience. They enable users to tailor Chrome functionality and behavior to individual needs or preferences. They are built on web technologies such as HTML, JavaScript, and CSS."* (Chrome, n.d.) Following this definition our program can examine content on all URL's and locate gender notations when loading a webpage with gender words appearing on it, then our program will go through them and replace them with a neutralized alternative - tailoring the behavior of the browser to the user's preferences.

The Technicalities of the Program

Check **appendix 1** for a flow chart of the program, it works as a simplified overview of the programs functionalities and workflow. Before our extension can change anything on the website it is used on, it first has to define what words to look for and replace, and what to replace them with. We have included all of this in a single object called "genders". This object contains two arrays genders.checkWords and genders.replacementWords. The latter is, as the name suggests, a list of all the gender-neutral terms we have chosen to use as replacement words. In the first

array, each element is an array in itself containing all the words which the program replaces with the pertaining replacement word. For example genders.checkWords[1] is an array containing “boy”, “girl”, “daughter” and “son” and all of these will be replaced by genders.replacementWords[1] which is “child”.

Walk()

The rest of our code is built around a single function we’ve made: walk(node), which is more specifically used as walk(document.body). What this function does is to “walk” through all the HTML nodes on the page. It does this by checking the nodeType of the node it walks through with the switch()statement, which is very similar to using if() statements, except it can have several conditions and different outcomes depending on which of the conditions is met. If the nodeType is a document fragment it will run the walk() function for each of the child nodes of that node, that’s how it runs through all HTML nodes rather than just the one it starts with, and if it then ends up on a node which has the text node as its nodeType, it will run another function we have made for that text node; the neutralizeGender() function. As the name suggests neutralizeGender() is the part of our code that actually replaces gender-based words. What it does first is define the nodeValue of the node (which since the node is a text node, is a string of text) as a variable named v.

Loops

For the next part of the code we have used nested loops. First, we have a loop that runs once for every replacement word. It does this by using i < genders.replacementWords.length as a condition for running. Inside this we have a loop that runs once for every word that needs to be replaced with this particular replacement word, this is of course done using j < genders.checkWords[i].length (we use j as our loop variable instead of i to differentiate, because this loop is inside a loop using i). So, the result of this nested loop is that the code within the second loop is run once for every single word our program wants to replace. That piece of code starts out by defining genders.checkWords[i][j] and genders.replacementWords[i] as two variables. The way we replace the words then, is to use the v.replace(), but if we just used this function with the variables as is, it would also replace substrings within words. For example, since our program replaces “he” by “they”, “helium” would become “theylium”. There is however a way to avoid this. What we do is, to use new RegExp(), which is a function that can create a regular expression from a string. “*Regular expressions are a way to describe patterns in string data. They form a small, separate language that is part of JavaScript and many other languages and systems*” (Haverbeke, 2018, p. 143). In our case we create the regular expression from genders.checkWords[i][j], and to that we add the “special characters” (these are characters that can change properties of a regular expression) “\b” and “g”. “\b” is the one that fixes the problem we have talked about earlier, it adds the “matches word

boundary” property to regular expression, which means that when used with v.replace(), it will only look for entire words that match our used string and not parts of words. “g” simply makes sure that v.replace() replaces globally, which means that it will find every instance of the word in the text node rather than just the first one it encounters. Now, with the word we are checking for, converted to a regular expression, we use v.replace() and define v as the result. So the line of code becomes v=v.replace(checkExpression, replaceWord).

Formatting and adapting

This however is not always enough to change all instances of the word, since v.replace() is case sensitive. We have written all words with the first letter as uppercase in the object gender, so that is the ones it replaces first. After that we go through the same process of creating a regular expression and using v.replace(), but before that we use checkWord=checkWord.toUpperCase() and replaceWord=replaceWord.toUpperCase(), this changes the check word to all uppercase, which makes the program look for the word in this form (such as: HIM) and replaces it with an all uppercase replacement word. The same thing is done with only lowercase word, using the .toLowerCase() function, except this time we also add the “i” special character which makes our regular expression case insensitive, so just in case someone had written one of the words our program look for with mixed lettering (for example: wOMaN) our program will still find it and replace it with our replacement word with lower case lettering. Finally after having done this for every single word in our checkWords array (the program having run through the entirety of the nested loops), our program will set textNode.nodeValue()=v, thus applying the changes we have made to the text node itself. Then when the neutralizeGender() function has been run for every text node on the site our extension is done and the user will see the results in their browser.

The conceptual view of the program

Our program is not functional in such a way that it improves how fast the webpage loads or removes ads. It is a work that shows the possibility to manipulate the text that is shown to you. When using our program on a webpage like e.g. New York Times, it becomes clear that algorithms, like our own program, can easily manipulate web content, without you knowing about it at first glance. Other than this, our program shows how text would look like, if the internet and texts in general were gender neutralized. The program puts writing a text in perspective when it is possible to use an algorithm to edit the text afterwards, engaging in a continuous manipulation of what was thought of as a final form. This can create some suspicion towards what you are reading, if the text could have been manipulated and changed from its original form written by the author.

Altering language through code

Our program touches upon a couple of themes from the AP course: Language Plus Code and Algorithms. The theme we want to reflect on the most in relation to our program is Language Plus Code. The text Vocable Code by Geoff Coxx and Alex McLean (2013) mainly addresses how computer code is able to speak for itself. Like that actual source code in a text editor can have a message, other than just what the message of what the program does, despite the fact that it is not a language that is used to be read out loud. In relation to this, they also comment on how software programs have a voice for themselves: "*Through the connection to speech, programs can be called into action as utterances that have wider political resonance; they express themselves as if possessing a voice*" (Cox & Mclean, 2013, p. 19). Our program also has a specific voice for itself. It has its own set of rules to swap words on every URL. It becomes visible for the users using our browser extension, that our program filters the original text, and then delivers a gender neutralized text. In this way our program has a very strong voice, that edits original text, and make the text into the program's own.

Why Gender Neutralize?

So why have we chosen to neutralize the internet from gender specific words, when there are so many options like war, hate-crimes, terrorism, disasters, etc., that could have been the focus for our alteration of web content through code? There has been a struggle to reach a world with gender equality for decades and the debate is still one of huge interest to citizens, businesses, activists and politicians around the world. In the western society this comes to show through female employees having a lower pay rate than their fellow male employees (Hinsliff, 2018), discussions about equal maternity rights between parents (Tucker, 2017) and the existence of classic gender stereotypes that sets the frame for how to raise a child, how to behave when being of a specific gender and what to expect of a certain gender (Olson, 2015). This has led to political and cultural actions towards achieving gender equality, which is ridiculed by some and praised by others - for instance the usage of the word 'hen' in Sweden as a gender-neutral pronoun (AFP in Stockholm, 2015). The aim for our program is not to take direct part in this discussion, but rather to visualize how code can alter language and therefore the content of the internet - creating groundworks for reflection upon the solidity of the content, and of course, inevitably the meaning of stereotypes and the meaning of gender. We chose the neutralization of genders as the content for alteration in our program due to the subject's relevancy in our contemporary western world and because of the fact that it is relevant to every human being.

Our Program as Software Art

Art can be defined as: "*The expression or application of human creative skill and imagination […] producing works to be appreciated primarily for their beauty or emotional power."* (Oxford Dictionaries, n.d.). If we perceive our software and its functionality as a piece of software art and therefore as a human expression, which evokes emotional responses by its perceivers, the emotional responses will vary from person to person, depending on their previous experiences and their knowledge of the discussion about gender, equality and the connotations that are connected to gender words. Our goal when it comes to our program's artistic expression is to create a visualized groundwork for thinking and reflecting upon the idea of gender and stereotypes. The purpose of the program's functionality is to gender neutralize the internet – though, we do not expect most people to find this helpful in their daily use of websites – but the result might create a visualized example of how a gender neutralized internet could look like. Hopefully this would evoke a reaction and a raising awareness to all of the connotations and stereotypes that might be loaded into words defining gender.

Software is often seen as an invisible medium, with which it is possible to create, display and perform art in the form of music, images and audiovisual content (Cramer, 2002). If the software in itself is perceived as a piece of art and the programmer as the artist, the program running through rules and algorithms might suddenly change character – being just as much a part of the art piece as any sensually perceivable output it creates. *"If any algorithm can be executed mentally, as it was common before computers were invented, then of course software can exist and run without hardware."* (Cramer, 2002). What is expressed here by Florian Cramer is a relevant entry to perceive our gender neutralizer program as an artwork. The output of our program could be created by any person with the ability to, or the will to learn how to change the HTML source code in a browser manually – or could even be created as a mental vision by any person. But instead of relying on people actually putting effort into having this experience with a gender neutralization themselves, the software allows us to visualize the concept for the spectators quickly and effortlessly – therefore leaving cognitive space for the audience to reflect upon the meaning of genders and their connotations. What is actually executed by our program is the movements of the *'artist's hands'* taking place in the browser window every time a page is refreshed by the spectator – the running program and its functions becoming a piece of art in itself.

References

AdBlock. (n.d.). *AdBlock*. Retrieved from AdBlock: https://getadblock.com/

AFP in Stockholm. (2015, March 24). *Sweden adds gender-neutral pronoun to dictionary*. Retrieved from The Guardian: https://www.theguardian.com/world/2015/mar/24/sweden-adds-gender-neutral-pronoun-to-dictionary

Chrome. (n.d.). *What are extensions?* Retrieved from Chrome: https://developer.chrome.com/extensions

Cox, G., & Mclean, A. (2013). Vocable Code. In G. Cox, & A. Mclean, *Speaking Code* (pp. 17-38). MIT Press.

Cramer, F. (2002). *Concepts, Notations, Software, Art.* Netzlitteratur.

Grosser, B. (n.d.). *Facebook Demetricator*. Retrieved from Benjamin Grosser: https://bengrosser.com/projects/facebook-demetricator/

Haverbeke, M. (2018). *Eloquent Javascript.* MIT.

Hinsliff, G. (2018, February 2). *Gender equality at work is a matter of respect, not just money*. Retrieved from The Guardian: https://www.theguardian.com/commentisfree/2018/feb/02/gender-equality-work-respect-money

Olson, S. (2015, March 22). *What It Means To 'Be A Man': How Male Gender Stereotypes Try To Fit Growing Boys Into A Mold, And Fail*. Retrieved from Medical Daily: https://www.medicaldaily.com/what-it-means-be-man-how-male-gender-stereotypes-try-fit-growing-boys-mold-and-fail-326450

Oxford Dictionaries. (n.d.). *Art*. Retrieved from Oxford Dictionaries: https://en.oxforddictionaries.com/definition/art

Tucker, L. (2017, december 6). *On Pregnancy and Gender Equality*. Retrieved from Huffpost: https://www.huffingtonpost.com/lindsay-tucker/on-pregnancy-and-gender-equality_b_7259072.html

Appendix 1 - Flowchart