

Visualizing the Nonvisual

A Tool to Illustrate How NVDA Parses Data Visualizations

Flannery Currin

Department of Computer Science

Earlham College

Richmond, Indiana

fhcurren15@earlham.edu

ABSTRACT

Data visualizations allow many web users to quickly grasp key takeaway messages from analyses of large sets of data. However, over-reliance on these visualization techniques can actually obscure that information from visually impaired screen reader users. To convey information to visually impaired users effectively, many screen readers intentionally ignore purely aesthetic information like font size and color, but those details convey significant information in some data visualizations. Developers designing content that includes data visualizations do not typically focus on the accessibility of that content to screen reader users and may have trouble conceptualizing how a screen reader might go about parsing the information contained in that structure. To address this specific accessibility issue, I examined and recorded the way that NVDA – the most widely used open source screen reader – conveyed the information it managed to glean from a few simple, Javascript-built data visualizations. From there, I was able to design a simple visualization tool bundled as an NVDA app module and web application to help developers visualize how NVDA interprets and presents their products. To provide this basic illustration without requiring developers to install NVDA, I also created a standalone demo with a sample data visualization and compiled suggestions for how to handle certain data visualization accessibility problems. This work just scratches the surface of accessibility issues in technological fields, but aims to make creating accessible content easier for developers.

CCS CONCEPTS

• **Human-centered computing** → **Information visualization**; **Accessibility design and evaluation methods**; **Accessibility technologies**; **Accessibility systems and tools**; *Visualization systems and tools*; • **Social and professional topics** → *People with disabilities*;

KEYWORDS

NVDA, screen reader, data visualization, accessibility

1 INTRODUCTION

Data visualizations tell a narrative that help people understand the big picture told by a large amount of data, which is crucial in a world where larger and larger datasets can be created rapidly. However, not everyone can see data visualizations or easily extract meaning from them. Screen readers are designed to help visually impaired people navigate various applications, most of which are focused primarily on presenting information in a visual way. However, screen readers are imperfect and have trouble parsing out what information is actually pertinent. Even when a screen reader

can identify important information, that information can sometimes be obscured or overwhelmed by other information. It can be especially difficult for a screen reader to recognize when to parse aesthetic information like color and font size presented within a data visualization. The visualization could be presented as a photo without any alternative text to make information available to screen readers. The screen reader might not recognize the graphic as a whole and instead list each individual component as a separate piece, and in these cases might struggle to find a logical ordering of those pieces. By studying the interaction between screen readers and data visualizations and rethinking the way data is summarized and presented, developers can make large amounts of information more readily available to an entire population.

This study addresses the communication gap between screen reader users and web developers by creating an NVDA plugin and visualization tool that tracks how NVDA parses a data visualization and shows what the visualization would look like based solely on that information. This work systematically examined and recorded the way that NVDA – the most widely used open source screen reader – conveys the information it extracts from three simple data visualizations specially designed using D3, a JavaScript library [14]. The simplest of these visualizations was a representation of a Slopegraph, a type of visualization that connects pairs of points to show basic trends. I transcribed this parsing process using NVDA's capabilities to add plugins to an installation that allow for customized capabilities, and then built on that plugin to create a fuller, more visual representation of that process. Because the visualization creation process depends on tracking NVDA's parsing process, it must be bundled up and included in an installation of NVDA in order to work. To give developers a general sense of how a screen reader like NVDA approaches data visualizations, I also created a standalone demo of this process using the Slopegraph visualization. Gathering what I learned from the transcriptions of NVDA's parsing process of all three data visualizations, I also compiled suggestions for developers seeking to make their data visualization content more accessible to screen reader users.

The main contributions of this work are:

- (1) A simple visualization tool to help sighted developers visualize how NVDA interprets and presents their products
- (2) A standalone demo of this visualization tool with a sample data visualization
- (3) A compilation of suggestions for how developers might handle certain data visualization accessibility problems

The rest of the paper is organized as follows. Section 2 provides a summary of necessary context for this project. Section 3 summarizes prior work related to data visualization design and

accessibility. Section 4 outlines the basic framework and design process informing this work. Section 5 describes the visualization tool's implementation in detail. Section 6 describes the demo and the results from testing the visualization tool. Section 7 lists and explains the compiled suggestions and considerations for developers. Section 8 summarizes the contribution of this study and describes areas for future work.

2 PRELIMINARIES

2.1 General Visualization Design Guidelines

The field of data visualization design has an extensive history, pre-dating computer-based design. Friendly and Denis compiled an impressively thorough history of data visualization and statistics from 6200 B.C. through 2001 [6]. They document important milestones, providing dates, context, and links (many of which are now outdated but can be used to find working links). Friendly followed up on this work in 2005 with a paper highlighting the general patterns found in the timeline and explaining the criteria used to define a "milestone" [5]. These milestones span developments in theory, technology, data collection and analysis, design, and dissemination. In 2006, Wilkinson wrote about the concept of a "grammar of graphics," a set of rules delineating the most correct way to structure graphic data visualizations [20]. Wilkinson's work was instrumental in generalizing rules from specific types of charts to more general graphing in order to accommodate a much wider range of visualizations.

Other researchers have developed their own guidelines for creating specific types of data visualizations or visualizations in specific contexts. Kelleher and Wagener developed a set of ten guidelines for developing data visualizations specifically for scientific publications, but most of their principles hold true for general use cases of data visualizations [9]. This set of guidelines is based on a relatively large survey of the literature and provides a good summary of general practices in creating clear, communicative data visualizations. A few of these guidelines are more relevant to the translation of data visualizations into something a screen reader can parse than the others, but having the overall context helps identify ways in which accessibility could fit into existing guidelines.

Cawthon and Moere investigated the relationship between the aesthetic appeal of a data visualization and the visualization's effectiveness and efficiency [3]. They normalized the visualizations according to color, size, scale, positioning, typography, conveyance, and interaction. While they found that aesthetics and effectiveness are not always directly linked, in general, participants were more likely to give up on finding an answer they couldn't gather immediately when the visualization was less visually appealing. This has interesting implications for screen reader users who are not able to see these aesthetically appealing components and already tend to spend more time than the average web user trying to find information that is not readily available [2].

2.2 Legal Accessibility Requirements

The Access Board's 508 Standards can be found in part 1194 of the Electronic Code of Federal Regulations (e-CFR) [1]. The 508 Standards outline federal accessibility requirements for electronic information and define relevant terms and concepts. The main goals

of the 508 Standards are to allow all U.S. federal employees with disabilities to have access to all of the information they need to perform their jobs and to allow the general public to access all information on government-sponsored platforms. The 508 Standards specify that all information on government-sponsored sites must be accessible to those without vision or with limited vision, without perception of color, without hearing or with limited hearing, without speech, with limited manipulation, with limited reach and strength, and with limited language, cognitive, and learning abilities. This section focuses on U.S. government-sponsored websites for the sake of consistency with the set of 508 Standards as a metric for measuring compliance.

Jaeger and Matteson address the idea that by attempting to ensure that information is correct and legally accessible, government-sponsored websites often lag behind commercially sponsored sites [7]. Because commercially responsible sites are not legally required to make their content accessible, many do not bother if it would slow things down too much. And even with the legal incentive, Jaeger and Matteson find that many government-sponsored sites are still not fully compliant with the 508 Standards. Lazar and Olalere review various governmental agencies' best practices for ensuring continued 508 compliance with their websites, and highlight the U.S. Census Bureau as having one of the most accessible sites [11]. This is especially interesting because of the large quantities of raw data the Census Bureau gathers.

3 RELATED WORK

Data visualization design and accessibility design have both been studied extensively, but largely separately. This work seeks to reconcile some of the work done in each of those fields and provide context for those in one field of the work done in the other field. Unlike prior accessibility-focused work, this work seeks to bridge the gap between screen reader users and developers who create data visualizations which are not just static HTML and can be interpreted and sorted out differently by screen readers.

3.1 Usability Concerns for Screen Reader Users

Most accessibility research focuses more on how usable a site is rather than how legally compliant it is. Lazar et al. were some of the first researchers to formally assess the specific frustrations of screen reader users [10]. Participants were asked to record frustrations during multi-hour computer use sessions in non-web-based electronic time diaries. They found that a majority of the frustrations reported by screen reader users would be easy to fix on the development side of things. Surprisingly, they also found that while screen reader users reported relatively high levels of frustration when they ran into issues, they actually wasted less time on those issues than student and worker populations previously studied. The researchers stated that this might be because screen reader users were more likely to engage with the issue and find a way around it, rather than defaulting to rebooting. However, Bigham et al. found that screen reader users tend to spend more time looking for information that is not available on a site when asked to find it because they believe that it is there and just difficult to access – a problem that sighted web users do not face nearly as often [2].

Voykinska et al. conducted surveys and interviews of blind or nearly blind individuals who frequently use social networking services (SNSs) to gain an understanding of how blind users process visual information on these platforms [19]. They identified certain accessibility issues: clutter, confusion resulting from information that is inconsistently arranged either horizontally or vertically, buttons with text descriptions that do not match their visual appearance, and infinitely scrolling pages. They also documented strategies that users developed to overcome these challenges: using different versions of a site to accomplish different tasks, using client applications to de-clutter SNSs interfaces, utilizing email notifications, asking sighted people for help, and avoiding specific features of SNSs. Many participants interacted with visual content in much the same way as sighted users, but relied on context clues and help from sighted people more than on technology to accomplish those tasks.

Shinohara and Wobbrock identified that assistive technology users have concerns about social usability in addition to functional usability [17]. They, along with Bennett and Pratt, developed a framework for teaching an ability-based version of the Design Thinking paradigm, which posits that empathizing with users is the first step of the design process and therefore should include considerations of social usability as well as functional usability [16]. They emphasized an approach that draws primarily from user testing and incorporating the ideas and feedback of disabled experts. While that approach definitely produces the most detailed and accurate usability feedback, developers typically do not conduct user testing for data visualizations unless it is an entirely new form of data visualization. Tools that can help developers empathize with potential users with varying abilities can therefore prove useful in these use cases.

Keeping these specific usability concerns in mind helped guide this work by providing specific questions to ask when looking at how NVDA parses data visualizations and how to convey the answers to those questions to developers who do not necessarily know to ask those questions.

3.2 Accessibility-Focused Design Tools

Inaccessible or difficult to access content can typically only be made truly accessible by the designers and developers of that content, as was the case when Mori et al. went about making Google Docs more accessible to screen reader users [13]. Tagaki et al. created a tool for developers, aDesigner – shown in Figure 1, to visualize a site’s usability beyond simple legal compliance using color and gradations in order to help web designers assess their products’ real-world accessibility [18]. In essence, the visualization simulates the environment in which visually impaired users are processing the site. The design highlights the amount of time required to access pieces of information on a page, the actual effects of adding various “skip-links”, the effectiveness of headings, and the order in which content is revealed to the user. This paper has been fundamental in understanding how to effectively simulate usability so sighted developers can have a similar experience to screen reader users. aDesigner is still available, but it can only parse the HTML files provided and does not work with JavaScript files. Michalska et al. referenced aDesigner and recommended it 10 years after its

development in their list of concrete steps designers can take to help ensure that their products are both legally accessible and realistically usable [12]. aDesigner in many ways set the standard for development tools intended to make creating accessible web content easier for developers new to accessibility concepts. This work seeks to build on aDesigner and create a tool that conveys similar information to users, but in the setting of a browser, not a desktop application. aDesigner is limited to parsing HTML files and cannot find issues with Javascript-generated content since it only parses the HTML file loaded in the tool. Creating a tool that gathers the same information a screen reader would in a real browser helps identify some of those issues aDesigner would normally miss.

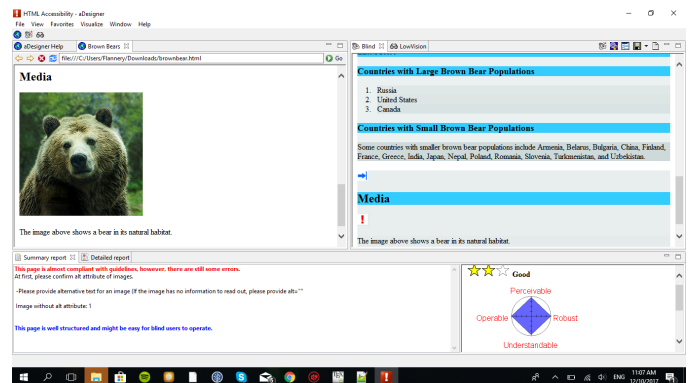


Figure 1: An example of aDesigner [18] catching the presence of an image without alternative text. HTML obtained from Codecademy [4]

4 DESIGN

4.1 Existing Structure

The visualization tool is intended to bridge the gap in communication between screen reader users and developers. The types of data visualizations this project focused on were constructed using D3[8], a popular JavaScript library. To create such visualizations, developers write a JavaScript file that constructs a visualization using data that is typically stored in a JSON file. To create the display in a browser, the JavaScript file is then referenced by an HTML file. NVDA tries to pull out information from that display, so it does not have direct access to the underlying structure of the original data or even the JavaScript file. Instead, it pulls out all the information it can gather from the display on the page and attempts to flatten it into a tree called a virtual buffer. NVDA’s interpretation of the content on the page is typically all the screen reader user has access to.

4.2 Added Framework

The basic architecture of this project is illustrated in Figure 2. The existing links are marked by solid lines while the links added by this project are marked by dotted lines. Most developers know what goes into creating JavaScript files and can debug based on the display they see, but they do not necessarily know what a screen

reader is able to access, unable to access, or has trouble accessing. This visualization tool is able to track NVDA's parsing process and generate a visual representation of the process that more closely matches how the developer is used to processing information than a speech or braille display created by NVDA. I wrote a plugin for NVDA to pull out the relevant information as NVDA is able to access it and write it to a text file (called the transcript). It does so by accessing the virtual buffer NVDA generates when Firefox becomes the active application and recording the elements of that virtual buffer one at a time. That generated text file can be fed into a tool I created that generates a new visualization to show the developer what their visualization would "look" like to a screen reader user. This generation takes the input provided by the text file and splits it to demonstrate the parsing process. It displays the words one at a time to simulate the way screen reader users receive that information and eliminate the details like color and spacing that could convey information to a sighted individual but are not picked up by NVDA. This should help the developer identify specific accessibility issues with their visualizations and address those problems using the list of accessibility suggestions included in this work or others.

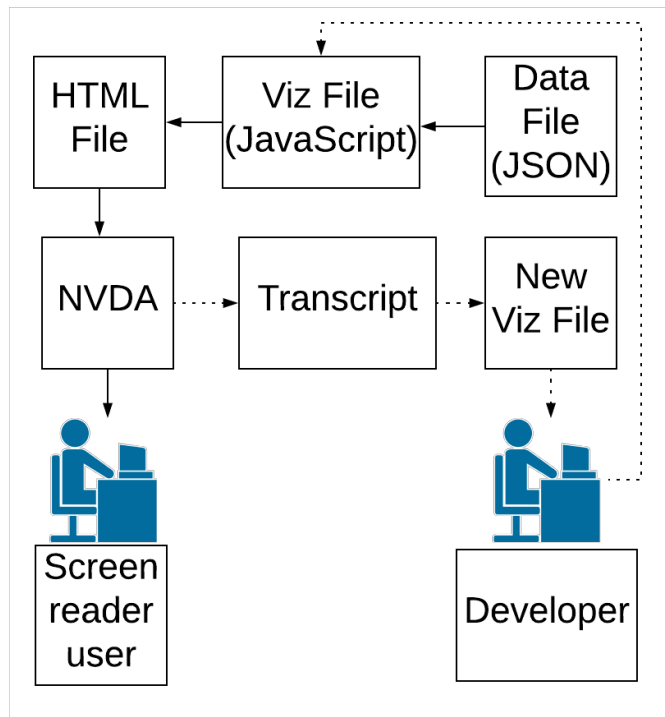


Figure 2: The framework for this visualization tool. Pre-existing links are represented by solid lines. Links added by this work are represented by dotted lines.

5 IMPLEMENTATION

5.1 Tools Used

All data visualizations created for this study were made using the D3 JavaScript library and had a data-JavaScript-HTML structure like the one described in the top row of Figure 2. NVDA was built using Python, and uses an internal installation of Python so that its users do not have to worry about making sure that dependencies are installed. The folder structure of NVDA includes folders where users can create and import python app modules or global plugins that affect the way that a local installation of NVDA behaves either in a specific application or across applications. For initial guidance, I looked to an existing global plugin written for NVDA, Takuya Nishimoto's Focus Highlight add on [15], but my final plugin was actually built as an app-specific module for Firefox. The module can be renamed to run in the user's browser of choice, but since NVDA runs best in Firefox, that browser seemed like the logical choice for a default. To create the visualization tool itself, I used JavaScript and the D3 library.

5.2 Visualization Tool

When a user installs NVDA and adds this plugin to the appModule folder, all they need to do in order to generate the raw data for the visualization tool is start running NVDA, navigate to the content they want to test in Firefox, and then quit NVDA. This produces a text file with a record of what NVDA was able to gather from the content of the web page. The developer can then navigate to the visualization tool which is initially a black screen with a button in the upper left-hand corner the user can use to select an input file. Once a file has been selected, the first word that NVDA was able to pull out of the web page is displayed on the screen. The user can move to the next word by pressing any key. This conveys what information NVDA is able to find and the order in which it presents that information to its users.

At this point in time, the visualization tool works better in Firefox than in other web browsers, consistent with NVDA's general performance. This particular issue is less pressing than some of the other accessibility issues because individual developers do not typically have the ability to make significant changes to the browsers they use to display their content, and NVDA users are typically aware that performance in Firefox is generally better than it is in other browsers. However, this does indicate that developers still need to be aware of the fact that environment factors such as web browsers can impact their accessibility testing experience, at least to some degree through no fault of their own or their product. Using Firefox as the default program for the app module encourages developers to acknowledge those performance differences and understand how much to attribute to that problem instead of their content.

6 RESULTS

To test this tool, I used two simple data visualizations that I created based on examples provided in the D3 gallery. These visualizations included a slopegraph, a bubblegraph, and a Sankey diagram. These tests confirmed that the transcription and visualization process works for more than one type of original visualization. Of these visualizations, the slopegraph proved to be the simplest in terms

of illustrating how the tool works and what types of accessibility concerns it can point out, so it was selected to create a demo.

The demo illustrates a couple of simple usability issues with the original slopegraph visualization. Purely visual information like the physical distance between points and the slopes of the lines are not reinforced through non-visual means. The information about what the chart is actually attempting to convey appears only after all of the points have already been read off, and the links between points are also not made clear by the order in which the screen reader grabs them. These issues are easy to pick up from viewing the visualization, but might go completely unnoticed if the developer relied solely on the appearance of the original visualization.

The demo allows users to see how the visualization tool presents information without first needing to install NVDA. The demo is primarily intended to demonstrate how the tool works, but can also illustrate some of the issues NVDA has when trying to process that particular visualization and similar ones. The original slopegraph can be seen in Figure 3. The first screen of the data visualization in the demo can be seen in Figure 4, and the full demo can be found at <https://github.com/fcurrin/viz>.

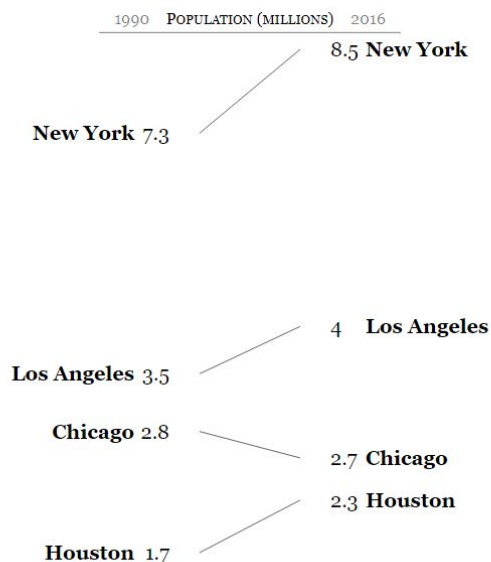


Figure 3: A simple slopegraph showing the change in population in a few major U.S. cities between 1990 and 2016.

7 SUGGESTIONS FOR ACCESSIBLE VISUALIZATION DESIGN

Through the work to develop this accessibility visualization tool, I have also created a list of suggestions for developers interested in making their data visualizations more accessible.

- (1) If possible, start thinking about accessibility in the early stages of the design process. Think about what attributes and organizational structure will be added or obscured in each layer of a visualization. Does information need to be

Slopegraph

Figure 4: The first slide in the visualization generated by the slopegraph in Figure 3. The rest of the slides can be found in the Appendix.

portrayed using color or can it also be portrayed in some other fashion?

- (2) Use a consistent ordering scheme. If information should be parsed row by row in one section, it is unreasonable to expect a screen reader to pick up on a sudden switch to parsing information column by column in another section. The order in which elements in a data visualization are read can greatly impact how easily users can understand how different elements are related to one another.
- (3) Make groupings of elements and connections between them explicit. If possible, label elements and connecting lines so that relevant details like size are accessible even when those elements are not visible and likely to be discarded as purely aesthetic detail.
- (4) Consider bundling your data visualization as an image and writing an alt text caption. This option is probably the most time consuming in the long run because it needs to be done from scratch for each visualization, but might be the most effective because the developer knows better than anyone else what message they want people who access their content to take away from it.

8 CONCLUSION

This work aimed to identify common accessibility problems that make data visualizations difficult for screen readers to parse, and to make testing content for accessibility and creating accessible content easier for individual developers by compiling a short list of specific suggestions that can be used in combination with existing data visualization and accessibility guidelines, developing a tool to help developers visualize how a screen reader parses their work, and creating a standalone demo of that tool.

My immediate next steps will be to conduct user testing to measure their effectiveness. This testing will begin in January of 2019. Future steps to extend this particular tool could include looking at how this visualization tool works with other types of data visualizations and visualizations in different contexts and environments.

While this work is intended to help developers learn how to create more accessible content and ease some of the burden screen reader users experience when trying to navigate inaccessible or difficult to access information contained in data visualizations, it

does not fully address the underlying pervasive accessibility issues of rich web content. While individual web developers can make intentional choices that make their content more accessible, tools are being developed to make developing content easier for people without a programming background [8]. Those tools do not necessarily prioritize accessibility, so trying to incorporate strategies to help developers create more accessible content into tools that make developing content easier for a broader range of people could have a more overarching impact on the accessibility of web content as a whole.

REFERENCES

- [1] 2017. Part 1194 - Information and Communication Technology Standards and Guidelines. <https://www.access-board.gov/guidelines-and-standards>
- [2] Jeffrey P Bigham, Irene Lin, and Saiph Savage. 2017. The Effects of Not Knowing What You Don't Know on Web Accessibility for Blind Web Users. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, 101–109.
- [3] Nick Cawthon and Andrew Vande Moere. 2007. The effect of aesthetic on the usability of data visualization. In *Information Visualization, 2007. IV'07. 11th International Conference*. IEEE, 637–648.
- [4] Codecademy. 2011. Learn HTML: Elements and Structure. <https://www.codecademy.com/learn/learn-html>.
- [5] M. Friendly. 2005. Milestones in the History of Data Visualization: A Case Study in Statistical Historiography. In *Classification: The Ubiquitous Challenge*, C. Weihs and W. Gaul (Eds.). Springer, New York, 34–52.
- [6] Michael Friendly and Daniel J Denis. 2001. Milestones in the history of thematic cartography, statistical graphics, and data visualization. URL <http://www.datavis.ca/milestones> 32 (2001).
- [7] Paul Jaeger and Miriam Matteson. 2009. e-Government and Technology Acceptance: The Case of the Implementation of Section 508 Guidelines for Websites. *Electronic Journal of E-Government* 7, 1 (2009).
- [8] Abhijit Jain. 2014. Data visualization with the D3. JS JavaScript library. *Journal of Computing Sciences in Colleges* 30, 2 (2014), 139–141.
- [9] Christa Kelleher and Thorsten Wagener. 2011. Ten guidelines for effective data visualization in scientific publications. *Environmental Modelling & Software* 26, 6 (2011), 822–827.
- [10] Jonathan Lazar, Aaron Allen, Jason Kleinman, and Chris Malarkey. 2007. What frustrates screen reader users on the web: A study of 100 blind users. *International Journal of human-computer interaction* 22, 3 (2007), 247–269.
- [11] Jonathan Lazar and Abiodun Olalere. 2011. Investigation of best practices for maintaining Section 508 compliance in US federal Web sites. *Universal Access in Human-Computer Interaction. Design for All and eInclusion* (2011), 498–506.
- [12] Anna M Michalska, Cindy X You, Ariana M Nicolini, Vincent J Ippolito, and Wolfgang Fink. 2014. Accessible web page design for the visually impaired: a case study. *International Journal of Human-Computer Interaction* 30, 12 (2014), 995–1002.
- [13] Giulio Mori, Maria Claudia Buzzi, Marina Buzzi, Barbara Leporini, and Victor MR Penichet. 2010. Making "Google Docs" user interface more accessible for blind people. In *International Conference on Advances in New Technologies, Interactive Interfaces, and Communicability*. Springer, 20–29.
- [14] Mozilla Developer Network. 2011. JavaScript. url: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (visited on 12/10/2017) (2011).
- [15] Takuya Nishimoto. 2018. Focus Highlight NVDA Addon v. 5. <https://addons.nvda-project.org/addons/focusHighlight.en.html>.
- [16] Kristen Shinohara, Cynthia L. Bennett, Wanda Pratt, and Jacob O. Wobbrock. 2018. Tenets for Social Accessibility: Towards Humanizing Disabled People in Design. *ACM Trans. Access. Comput.* 11, 1, Article 6 (March 2018), 31 pages. <https://doi.org/10.1145/3178855>
- [17] Kristen Shinohara and Jacob O. Wobbrock. 2016. Self-Conscious or Self-Confident? A Diary Study Conceptualizing the Social Accessibility of Assistive Technology. *ACM Trans. Access. Comput.* 8, 2, Article 5 (Jan. 2016), 31 pages. <https://doi.org/10.1145/2827857>
- [18] Hironobu Takagi, Chieko Asakawa, Kentarou Fukuda, and Junji Maeda. 2004. Accessibility designer: visualizing usability for the blind. In *ACM SIGACCESS Accessibility and Computing*. ACM, 177–184.
- [19] Violeta Voykinska, Shiri Azenkot, Shaomei Wu, and Gilly Leshed. 2016. How Blind People Interact with Visual Content on Social Networking Services. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. ACM, 1584–1595.
- [20] Leland Wilkinson. 2006. *The grammar of graphics*. Springer Science & Business Media.