# How Accessible is the Process of Web Interface Design?

Kirk Norman, Yevgeniy Arber and Ravi Kuber
UMBC
knorman1@umbc.edu

## ABSTRACT

This paper describes a data gathering study, examining the experiences and day-to-day challenges faced by blind web interface developers when designing sites and online applications. Findings have revealed that considerable amounts of time and cognitive effort can be spent checking code in text editing software and examining the content presented via the web browser. Participants highlighted the burden experienced from committing large sections of code to memory, and the restrictions associated with assistive technologies when performing collaborative tasks with sighted developers and clients. Our future work aims to focus on the development of a multimodal web editing and browsing solution, designed to support both blind and sighted parties during the design process.

## Categories and Subject Descriptors

H.5.2 User Interfaces – *User-centered design*

## General Terms

Human Factors.

## Keywords

Accessibility, blind, HTML, screen reader, web development.

## 1. INTRODUCTION

While the accessibility barriers encountered by individuals who are blind have been well documented by researchers, relatively limited attention has been given to the needs of interface developers who use assistive technologies to support their work. Research suggests that the greatest challenges facing blind developers include (1) the cognitive burden of committing the structure of program to memory to reduce parsing errors, (2) verifying program consistency using a screen reader, (3) difficulties accessing controls on editors, (4) lack of assistive tools integrating the environment for compiling and debugging programs, (5) once compiled, determining the position of content on the interface [1-5]. Furthermore, the restrictive nature of screen readers can impact the process of spatially-distributing content on a graphical user interface (e.g. a web page), resulting in designs which may not look as visually-appealing as those developed by sighted developers. As a result of the accessibility challenges faced, blind developers are thought to be at a competitive disadvantage in the workplace [5].

In this paper, we describe the findings from a data gathering study conducted to identify the ways in which blind web developers develop and edit web sites and online applications. We examined the difficulties faced when designing interfaces independently or when working as part of a team with sighted peers.

## 2. RELATED WORK

Non-visual interfaces have been developed to overcome the digital divide experienced accessing programming environments. Examples include the development of a scripting tool to create graphical forms in Visual Basic, without needing the 'point and click' approach used by sighted users [4]. Speech cues enable the user to manipulate the size the forms and objects. Tran et al. [5] developed a code editor and accessible interface to compile C# programs. Compiling errors are outputted using speech, which can then be remedied by the user. Sanchez and Agauyo [3] developed the Audio Programming Language, where commands are dynamically presented in a circular command list. The aim was to alleviate the need to commit commands to memory, enabling the programmer to focus their attention on the design process itself.

Assistive solutions have yet to focus upon supporting the process of web interface design. We aim to identify the experiences of web interface designers who are blind. The long-term goal is to design a solution to better support their needs.

## 3. DATA GATHERING STUDY

Six legally-blind participants who had experience with web development were recruited for the study. Interviews were conducted with each participant, either by telephone or by video conference. Questions covered a range of areas including use of assistive technologies, experience coding web pages/applications using HTML, scripting languages, and strategies to design for both blind and sighted audiences.

### 3.1 Participant Demographics

Participants recruited for the study were aged between 29 and 48 (6 male 0 female). Two participants described themselves as congenitally blind, while the other four lost their sight in later life. All six identified themselves as intermediate to expert users of screen reading technologies. Five used JAWS as their primary assistive screen reading tool, with one participant alternating between JAWS and NVDA.

## 4. RESULTS AND DISCUSSION

### 4.1 Developing and Checking Content

All six participants favored using simple text editing tools such as Notepad to code web pages using HTML. The code could be then checked within the text editor itself using JAWS, and amended as appropriate. Participants could then proof the content displayed on the page through the web browser. This check was performed to ensure that all objects were presented (e.g. images, hyperlinks etc.), information contained within tables or lists were correctly ordered, and that no items or objects seemed out of place. For example, if a string of characters listing part of a tag was presented via the browser (e.g. "</p"), the code would be amended to ensure that the tag was closed (e.g. "</p>"). Participants described the process of verifying code and content as time-consuming, as they would need to switch continuously between the code in the text editor to the browser page to ensure that all errors had been fixed. If further errors were identified, participants would often leave comments in the code to repair them at a later point in time. Two participants had previous experience with web development software (e.g. Dreamweaver and Frontpage). They suggested that after trying to learn these technologies, frustrations from not being able to access components in the software led them to using alternative solutions. Web editing technologies were thought to be designed with sighted users in mind, often encouraging users to drag-and-drop. However, for non-mouse users, performing these tasks using keystrokes could be difficult.

Participants were aware that web editors offer graphical cues to support the interface design process. For example, web editors clearly delineate different types of code from each other using color-coded variables (e.g. comments in green etc). However, these cues are not always accessible to screen reader users, meaning that individuals who are blind may not be benefiting from the support provided to sighted users.

To reduce the number of coding errors, participants stated that they often committed sections of HTML or scripting code to memory. Alternatively, they would create template files which could be reused within web pages. In contrast to web editing solutions, text editors (e.g Notepad) do not offer features such as autocomplete or automatic closure of tags once opened, meaning that the user would need to concentrate on the task to reduce the likelihood of errors.

### 4.2 Determining the spatial layout of objects

Participants were found to design web pages for both sighted and blind audiences, and were aware that information would need to be spatially-distributed across the page to be visually appealing to the users. To gain an overview of layout, one participant mentioned using the Screen Layout mode in JAWS, enabling him to explore the objects present. Two developers kept a CSS file on hand, created by sighted designers that would provide the formatting required for a sighted audience. One participant who developed content management systems favored the use of Drupal templates for layout. Separation from style and substance of web pages appeared to be a common theme arising from the interviews.

While the participants wanted to perform tasks independently, some asked sighted colleagues to double-check the visual appearance of the site, just to ensure that content was appropriately displayed. However, for sites tailored to blind audiences, it appeared that the majority preferred to develop content without assistance from sighted peers, utilizing commands such as the Simple Layout function in JAWS, enabling the user to identify table-based content present on the page. Participants highlighted workarounds that they had employed to work with color on a web site. One participant had memorized pantone colors from when he could still depend on his residual sight. These colors could then be coded.

### 4.3 Scripting Languages

Participants highlighted difficulties with screen readers interpreting Javascript output, making it challenging in certain instances to verify whether the scripting code which they had written was appropriate. As a result, scripting languages were often used with caution.

### 4.4 Collaborative design

For nearly all participants surveyed, sites had been created collaboratively, either with other sighted developers or clients. Participants suggested that it was difficult to modify the layout of objects using the screen reader, due to the lack of synchronization between the visual and non-visual presentation of web page content. This was particularly an issue when prototyping on-the-fly. A need was identified for additional support in this process.

## 5. DESIGN IMPLICATIONS

Analysis of the data revealed that web interface developers who are blind may benefit from the following features:

- An editing tool that can check code, in addition to providing information about the layout of objects. The aim would be to reduce the time and effort spent verifying output.
- Synchronization between visual and non-visual presentation of web pages, enabling blind and sighted users to collaborate more effectively in the web design process.
- Providing more support to assist the process of coding. For example, non-visual methods of conveying different variables, and autocomplete of tags.
- Alleviating the memory burden, by reducing the need to remember commands and pieces of programming code.

Future work will focus on the development of a multimodal web editing and browsing solution to assist the development of code generation and to provide the structural information needed to support decisions associated with layout of content. We aim to identify whether the assistive solution can provide an effective alternative to current methods of developing sites, and to specifically examine whether it can augment the process of prototyping on-the-fly.

## 6. REFERENCES

[1] Hayhoe, S., 2011. Non-Visual Programming, Perceptual Culture and Mulsemedia. In: Multiple Sensorial Media Advances and Applications. IGI Global, 80-98.

[2] Konecki, M. et al. 2011. Making Programming Accessible to the Blinds. In *Proc. MIPRO*, 820-824.

[3] Sanchez, J. and Agauyo, F. 2006. APL: Audio Programming Language for Blind Learners. In *Proc. ICCHP*, 1334-1341.

[4] Siegfried, R.M. 2006. Visual Programming and the Blind: the Challenge and the Opportunity. In *ACM SIGCSE Bulletin*, 38, 275-278.

[5] Tran, D. et al. 2007. Text-to-Speech Technology-Based Programming Tool. In *Proc. SSIP*, 173-176.