Spring 5-16-2023

Modern Practices for Responsive Web Design and Web Accessibility

Keyaun Washington
University of Nebraska-Lincoln

MODERN PRACTICES FOR RESPONSIVE WEB DESIGN AND WEB ACCESSIBILITY

An Undergraduate Honors Thesis

Submitted in Partial Fulfillment of

University Honors Program Requirements

University of Nebraska-Lincoln

by

Keyaun Washington, BA

Computer Science and Mathematics

College of Engineering and College of Arts and Sciences

May 16, 2023

Faculty Mentors:

Brady Garvin, PhD, School of Computing

Abstract

Responsive web design and web accessibility play crucial roles in ensuring an optimal user experience on the web. By designing websites with responsiveness and accessibility in mind, more opportunities are opened up for a wider audience to access and interact with our content. Through modern practices, responsive web design allows websites to reach several different devices ranging from compact smartwatches to expansive television screens. Designing for accessibility provides accommodations for individuals with impairments while also providing benefits for individuals without impairments. However, designing for responsiveness and accessibility can present challenges; a poor attempt at providing accessibility features can worsen a user's experience. It is vital that a website is designed properly to ensure usability for all users.

Key Words

Interaction Design, Human-Centered Computing, Responsive Web Design, Web Accessibility, Computer Science

Introduction

In the early days of the internet, desktop computers were the primary method of browsing the web. Even though users may have varying monitor resolutions, aspect ratios, and physical sizes, the variety was small enough that web pages designed for a specific width (and/or height) were viewable without much disruption. As a result, designing a web page to fit a monitor with this specific width was suitable for most people's needs.

However, technology has changed dramatically since then. We now have devices with both smaller and larger resolutions as well as devices of different aspect ratios and orientations. Since the web can be browsed on such a large variety of devices, designing a web page for any specific resolution is no longer feasible. Responsive web design tackles this issue by providing a one-size-fits-all solution to web design, allowing elements of a web page to reposition and scale based on the resolution of the device used to view the page. A responsive web page can be viewed by anyone on any device, regardless of the size or resolution of the device used.

However, the use of responsive web design can also lead to accessibility concerns if implemented improperly. When designing a responsive web page, it is important to consider all types of visitors including mobile device users, keyboard-only users, and users with disabilities. If accessibility is not taken into consideration when designing a responsive web page, the result might be unmanageable for those who need accessibility features.

The scope of this survey covers the modern practices and implementation methods for responsive web design and web accessibility. When designing for responsiveness, considerations are given to macro layouts, micro layouts, typography, images, icons, and

conversions from fixed layouts. Considerations for web accessibility encompass accessible rich internet applications, keyboard accessibility, images, video, audio, colors, and typography.

Background

In the beginning, desktop computers were the primary method of browsing the web. Even though users may have varying monitor resolutions, aspect ratios, and physical sizes, the variety was small enough that web pages designed for a specific width (and/or height) were viewable without much disruption. In the early 1990s, during the initial rise in popularity of the internet, “most monitors had screen dimensions of 640 pixels wide by 480 pixels tall” [11]. As a result, designing a web page to fit a monitor with this specific width was suitable for most people’s needs.

However, this resolution did not stick around for long. Soon, “most screens had dimensions of 800 by 600 pixels” and “web designs changed accordingly” [11]. Reasonably, this became the new “safe” resolution at which to design web pages. However, as we know today, this resolution did not stick around for long either. As technology advanced and more consumers gained access to larger resolutions, these new resolutions became the next “safe” resolution. This means that web page designers are constantly fighting to update and match the next common resolution. Once enough consumers gain access to a higher resolution, a web page design is now “behind” and designed for old monitors rather than the currently popular monitors.

When the common resolution of monitors increased over the years, web pages designed for fixed resolutions were still viewable in full. However, this increase in resolution resulted in a

block of blank, unused space on either side. For a small increase in resolution, the amount of wasted space is hardly an issue. For a large increase in resolution, the amount of blank space is a bigger problem; the content still fits on the page, but now valuable space is wasted when it could be filled with more content instead. Even still, the wasted space is still manageable to some since all of a web page's content would still fit on the screen.

Big issues arise when a resolution *decreases*. With the introduction of tablets and smartphones, the internet is regularly viewed on much smaller resolutions than normal for a desktop device. Since web pages were designed for much higher resolutions than these smaller devices could support, they were hard to view or navigate on small resolutions. Without adjusting a web page's content dynamically, the scale of the web page was disproportionate to a smartphone's resolution; either the web page was scaled to fit the content within the display, resulting in tiny and unreadable text, or scaled to have readable text, resulting in content overflowing beyond the screen and requiring the user to scroll horizontally and vertically to see everything.

Terminology

HTML (HyperText Markup Language) and **CSS (Cascading Style Sheets)** are the two most important technologies used in web design. HTML is used to define the content and structure of a web page. It uses tags to show different types of elements such as paragraphs, images, lists, and tables. HTML allows web developers to structure content in a logical and meaningful way.

CSS is used to define the layout and appearance of a web page. It allows web designers to add colors, fonts, backgrounds, spacing, and other visual effects to HTML elements, creating a more visually appealing and engaging user experience. CSS is designed to work seamlessly with HTML, customizing the appearance of the content defined in HTML.

A **fixed layout** is a web design approach where the content of a website is set to a specific width. In a fixed layout, the content is the same width no matter the resolution, and the content's width is not adjusted when the viewport dimensions are changed. Fixed layouts were common in the past when the diversity of display resolutions was small, but they have since become outdated with the rise of much smaller and much larger devices.

Media queries are a set of rules that web developers can use to apply different styles based on the characteristics of the device that is being used. A media query comprises two parts: the media type and the media feature. The **media type** is used to specify which categories of devices the styles will apply to (screen, print, or all), and the **media feature** describes the characteristics that the device must have for the styles to apply. These characteristics include viewport width and height, orientation (portrait or landscape), hovering capabilities, preferred color scheme (light or dark mode), and more. For example, a media query can be used to apply a set of styles only if the viewport is at least `1000px` wide. Media queries can also be combined. For example, a media query can be used to apply a set of styles only if the viewport is in landscape orientation, the user prefers dark mode, and the height is less than `500px`.

An **adaptive layout** is a web design approach that uses media queries to dynamically fit content to different devices and screen sizes. Typically, an adaptive layout will have several

“breakpoints”, or measurements at which the layout needs to be adjusted to fit a new range of resolutions. As such, a web designer will create separate styles for each range of resolutions that they deem necessary. While an adaptive layout is a valid option when it comes to web design (especially when a web designer wants radically different designs for different resolutions), its main drawbacks are poor maintainability and scalability. A web designer has to create several different sets of styles for each range defined in the media queries. When a change or addition is made to a website’s style and/or content that requires rescaling or repositioning elements, the change would need to be reflected in all of the relevant media queries. For websites with many breakpoints for many different resolutions, this issue is only exacerbated.

A **responsive layout** or **liquid layout** is a web design approach that uses flexible elements and grids to dynamically fit content to different devices and screen sizes. Unlike adaptive layouts, responsive layouts provide a one-size-fits-all solution, allowing any user to view a page on any screen size while also allowing the web developer to create only one set of styles rather than a separate set of styles for each individual resolution. While a responsive layout may not be completely void of media queries, this type of layout provides the same benefits as the adaptive layout without the maintainability drawbacks. As such, responsive web design is an important practice in the modern web development world.

Methodology

To understand the importance, impact, and implementation of responsive web design and web accessibility, I browsed several online conferences and journals, courses, and

documentation pages. Through web searches (with Google Search), I found implementation details and other technical information used in modern responsive web design and web accessibility. I used varying search queries to find broader or more specific information, such as “responsive web design” to learn about this topic as a whole and “HTML tabindex” to learn more about the implementation and use cases for this specific accessibility feature. The resources I found with this method include online courses (such as web.dev’s “Learn Responsive Design” [19]) and documentation pages (such as MDN’s “Web Technology for Developers” [20]). The World Wide Web Consortium (W3C) also provides many industry standards for web design including accessibility in the Web Accessibility Initiative (WAI) [21]. These resources provided me with a deeper understanding of the underlying mechanics and applications of modern responsive web design and web accessibility.

In addition, I looked into several online conferences and journals for research findings and experiments. Some of the conferences I looked into during my search included the ACM Conference on Human Factors in Computing Systems (CHI) and the ACM Symposium on User Interface Software and Technology (UIST). I used search terms such as “responsive web design”, “UI/UX”, and “web accessibility” to find a large sample of papers, and then I looked more into specific papers after reading the abstracts to know which papers were the most relevant to modern responsive web design and web accessibility. These papers provided me with insight into the latest findings and advancements in these areas. These findings, along with the courses and documentation pages, gave me a more comprehensive understanding of the topic as a whole.

Creating an Inclusive and Adaptive Web Experience

When designing a website, both responsiveness and accessibility should be considered. Responsive web design can cause some accessibility concerns, so it is important to have both in mind while designing a web page. Designing for accessibility provides accommodations to those with disabilities while also providing benefits to everyone. It is essential to design for responsiveness while simultaneously addressing any potential compromises in accessibility.

Designing for Responsiveness

When creating a responsive layout from scratch, it is important to design a page-wide layout as well as the layouts inside of elements or components. Each individual element, like a block of text or an image, also needs specific styling to achieve responsiveness.

Macro Layouts

The term **macro layout** refers to the overall arrangement of elements and components of a webpage. This type of layout typically involves organizing components across the entire screen, rearranging and shuffling content where there is room. With responsive design, these components scale and reposition based on the size of the screen. To achieve this functionality, the element containing all of the components can be given a different display type in CSS. The "flexbox" and "grid" layouts are suitable for macro layouts as they both allow objects to be positioned and scaled dynamically.

The "`flex`" display option is used for flexbox layouts. In a flexbox layout, elements are scaled and arranged to fill up the space in their containers; the elements have "flexibility".

Flexbox layouts are “designed for one-dimensional content” and “excel at taking a bunch of items which have different sizes and returning the best layout for those items” [12]. This type of layout is particularly useful for distributing a series of items of varying sizes across several rows or columns. A common design pattern in modern websites involves a main section (such as an article) accompanied by a sidebar. By applying a display of “`flex`” to an element, its child elements are automatically positioned in a row by default but do not grow to fill the container [12]. To allow these elements to scale and fill the container, they can be given a scale factor through the “`flex-grow`” CSS property (or the “`flex`” CSS property, which is a shorthand to set `flex-grow` among other flex properties). An element with a `flex-grow` of `2` will grow twice as much compared to an element with a `flex-grow` of `1`, and an element with a `flex-grow` of `0` will not grow at all. To achieve the article-sidebar pattern, this `flex-grow` property can be used to grow the article and fill the space not occupied by the sidebar. However, at small viewport widths, the article and sidebar can appear to be too squished and difficult to read. In this case, the “`flex-wrap`” property can be applied to tell elements to wrap around to the next row or column when there’s not enough space for them to fit nicely. This article-sidebar pattern is a typical use case for the `flex-wrap` property as elements can be rearranged in a responsive manner without the need for media queries [12].

The “`grid`” display option is suited for exactly what one would expect: grid layouts. In CSS, an element can be given a display value of “`grid`”, and the child elements will be positioned according to a grid defined by the parent element. Using the “`grid-template-rows`” and “`grid-template-columns`” properties, one can specify how many rows or columns are in the grid and how large those rows or columns are. For

example, to achieve a layout with two columns with the first column twice as wide as the second, the container element can be styled with the grid-template-columns property of value "`2fr 1fr`" (fr is a fractional unit; `2fr` takes up 2 units of the total space). Grid items can also be repositioned and can span multiple rows or columns using the "`grid-row`" and "`grid-column`" CSS properties. Unlike flex layouts, grid items do not have the ability to wrap elements around to another row or column when space is minimal. However, with the clever use of CSS functions and values such as "`repeat()`" and "`auto-fill`", one can achieve a grid with an automatically determined maximal number of rows or columns. Alternatively, one can use media queries to force a certain number of rows or columns based on some condition (such as viewport width), but managing multiple breakpoints (`500px`, `800px`, `1000px`, etc.) can be "quite tedious and difficult to maintain" [13].

Micro Layouts

The term "micro layout" refers to the layout within a component of a webpage rather than the layout between components as with macro layouts. These types of layouts are self-contained and are designed to adapt to the component's container. Responsive micro layouts are important because they allow components to be reused across various situations; a component would not need a specific width or height to be displayed properly. As with macro layouts, the flexbox and grid layouts can also be used to design micro layouts.

One additional CSS feature that is especially useful to micro layouts is container queries. Similar to media queries, container queries are a set of rules that can be used to apply different styles. However, unlike media queries, container queries apply specifically to an element and its

parent/container rather than the viewport as a whole. Through the use of container queries, components can be styled in an independent way [14]; each component can be designed modularly without needing to have the context of the entire page.

Typography

By default, text elements will wrap around to the next line when they become too long. In this way, “text on the web is responsive by default—it flows to fit the user's viewport” [15]. However, just because text can wrap around when necessary does not mean that wrapping is appropriate. The biggest concern with responsive typography is the font size; text needs to be small enough to fit on small devices but large enough to be readable on large devices. So, the default behavior of typography is not enough to suit the needs of all devices and users.

To adjust the font size of text elements in a responsive manner, one can use the “`font-size`” CSS property along with a responsive unit such as “`vw`” (viewport width). However, using only a responsive unit as the font size has two major flaws: the font size can be difficult to read at the extremes and cannot be manually adjusted by the user. For example, consider a font size of `1.25vw` (`1.25vw` means 1.25% of the viewport width). At a viewport width of `1920px`, this font appears at `24px` (which is an appropriate font size). However, at a viewport width of `400px`, this font appears at `5px` (which is too small to be readable). The font also cannot be zoomed or scaled manually by the user; the font is always 1.25% of the viewport width no matter how far in or out the page is zoomed. A better approach is to combine responsive units with fixed units using the `calc()` CSS function, which does the unit calculation and applies it. So, for example, one can use both `vw` units to scale text responsively

and em units to give the text a base font size. The use of fixed units such as px also allows the user to adjust font size with their browser. To give the font size a lower and upper bound, one can use the clamp() CSS function. Similar to calc(), one provides the font size along with the lower and upper bound. Using this method, the text scales responsively, is legible at the extremes, and allows the user to adjust the font size if they choose.

Images and Icons

Without applying any additional styles, images and icons will appear at their intrinsic width and height. For high-resolution images, this may be an issue because the image content can overflow its container or the viewport. A quick and easy solution to this issue is to apply the "max-width" or "max-inline-css" properties with a value of "100%" [16]. This allows any image to keep its size unless it is too large, in which case it shrinks to fit its container. To ensure that the image maintains its original aspect ratio, one can add the "block-size" CSS property with a value of "auto" to keep the aspect ratio constant [16]. Additionally, the "object-fit" CSS property can be used to resize an image to fill the container while preserving any aspect ratio. In particular, assigning the object-fit a value of "contain" will add additional spacing around the image to ensure that the whole image is visible while preserving the aspect ratio, and assigning the object-fit a value of "cover" will crop the image while preserving the aspect ratio.

Converting from a Fixed Layout

Although fixed layouts were much more abundant in the first few years of the internet, they are certainly not extinct. For one, it is much easier to create a prototype web page with a fixed layout than it is a responsive layout; creating a design that looks nice in one specific resolution is simpler than creating a one-size-fits-all design for all resolutions. As such, some web developers may prefer to create a fixed layout first and then convert it to a responsive layout. Some websites may still use fixed layouts simply because they were created long ago and have not been updated. Regardless of the reason, fixed layouts may pose a problem in terms of user experience, so some web developers may wish to upgrade a pre-existing fixed layout to a responsive layout.

To convert a fixed layout into a responsive layout, the first step is to convert absolute units (such as `px` or `pt`) into relative units (such as `%` and `vw`) for the width of containers/layouts. For example, if a web page was designed at `1000px` and a sidebar was given a width of `200px`, then converting the sidebar width to `20vw` will keep the sidebar at its original width when the screen is `1000px` in width, and the sidebar will scale accordingly for smaller or higher resolutions. Similarly, the sidebar width can be set to `20%` to span `200px` if its parent element has a width of `1000px`.

Not all styles and properties with absolute units should be converted to relative units, however. The height of an element such as a navigation bar may not need to change based on the screen's width. In general, the height of elements may use absolute units since modern websites favor the vertical scrollbar over the horizontal scrollbar. Similarly, font sizes should also

not be converted to relative units (or should at least mix relative and absolute units as mentioned previously).

Designing for Accessibility

Web accessibility refers to how “websites, tools, and technologies are designed and developed so that people with disabilities can use them” [9]. Although those with disabilities are the target audience when it comes to accessibility features, designing for accessibility means making web content available and usable by as many people as possible, regardless of their abilities or disabilities. Consequently, designing for accessibility also benefits those without disabilities (for example, those using tiny smartwatches) and those with “temporary disabilities” (for example, those with broken arms).

Accessible Rich Internet Applications (ARIA)

The introduction of HTML5 brought several key features, one of which being semantic tags. Semantic tags are used to define the content and structure of a web page in a more meaningful and descriptive way. Structurally, semantic tags are the same as other HTML tags; to a visitor of a website, elements would look exactly the same. However, semantic tags provide more context to the content they contain, which makes it much easier for tools such as search engines, screen readers, and web crawlers to understand the structure of a page.

ARIA (Accessible Rich Internet Applications) goes beyond the capabilities of semantic tags and provides additional information about the structure and functionality of a web page which may not be apparent from the HTML alone. Even with the addition of semantic tags,

some elements or components may not have a good semantic tag to describe their function. This leaves screen readers and other assistive technologies clueless as to what meaning these elements might have. ARIA works to resolve this issue by allowing web developers to give elements specific attributes (roles, properties, and states).

Keyboard Accessibility

Most modern browsers have some level of built-in keyboard accessibility. For example, pressing the Tab button will generally navigate between elements such as buttons, links, input boxes, and forms. Pressing the Enter key will select or trigger certain elements. The arrow keys can be used to move between elements, particularly in a form. However, web developers may find themselves creating custom elements that resemble existing HTML elements to provide their own functionality or styling that may not be possible with the default elements. As a result, these elements lose their native keyboard accessibility.

To provide (or prohibit) keyboard navigation to typically non-focusable (or non-selectable) elements, the "`tabindex`" attribute can be applied. Giving an element a `tabindex` attribute of value 0 allows for that element to be focused in sequential keyboard navigation with the order determined by its position in the HTML. A positive `tabindex` is used to define the order in which the elements should be navigated; an element with a `tabindex` of 1 will be selected before an element with a `tabindex` of 2, but all elements with a positive `tabindex` are selected before those with a `tabindex` of 0. A negative `tabindex` is used to specify that an element is not reachable using keyboard navigation. When elements of a webpage may change order and position (which is typical of a responsive webpage), a positive

`tabindex` should be avoided as it can be difficult for users with assistive technology to navigate a webpage if elements move out of the expected order [17].

Images, Video, and Audio

Visual and auditory elements such as images, videos, and audio sources are commonly used to provide information or add interest. However, if done improperly, these elements can pose a significant challenge to individuals with visual and auditory disabilities. For example, a blind person (or anyone unable to see their screen) will not be able to see any images or videos embedded into a web page, making it difficult or impossible for them to understand the content or the meaning behind the content. A deaf person (or somebody without headphones or speakers, for example) will not be able to hear audio sources from videos, music, or sound effects, making it difficult or impossible to gain any information or feedback in this manner.

Images are not required to have any attributes attached to them, although it is expected that an image has a "`src`" attribute (to specify what image to show) and likely a `width` and `height` attribute. To make images more accessible to users, the "`alt`" attribute can be applied. This attribute defines the alternate text shown to users when an image fails to load and cannot be displayed; in the image's place, the text is displayed instead. Assistive technologies use this `alt` attribute to supply additional information to the user. A screen reader can read this alternative text to the user to inform them of what the image is and what meaning it has. An empty `alt` attribute has a special meaning and is used to tell assistive technology that the image is for purely presentational purposes only (such as a background image or pattern). In

this way, an empty `alt` attribute has a different, specific meaning compared to an omitted alt attribute. An empty `alt` attribute informs assistive technology that it can be skipped over.

Unlike images, video and audio elements do not have an `alt` attribute to inform assistive technologies of their meaning. Typically, embedded videos and audio sources are provided by third-party services such as YouTube and Spotify as a means of removing the burden from the web developer, and a good provider may already provide accessibility features for a user. If a web developer creates a video on their own, track tags can be used to supply subtitles and closed captions to the user which are displayed over the video. However, this method of supporting accessibility can require a lot of effort from the web developer, especially for longer videos. W3C recommends including “visible links to transcripts of audio” and “visible links to audio-described versions of videos” to provide alternatives for media such as video and audio. For content that plays automatically, it recommends providing “visible controls to allow users to stop any animations or auto-playing sound” [10].

Colors

Color is often used to theme and stylize websites and other interfaces. Additionally, it can be used to highlight or bring attention to certain elements of a page. However, users who suffer from colorblindness or other vision impairments may face usability issues if a webpage relies too heavily on color for conveying important information.

When color is used in conjunction with text, readability and color contrast are critical factors to consider. The contrast between the text color and the background color of a webpage should be sufficient enough to ensure legibility, especially for those with visual impairments.

Most people may find discomfort while attempting to read bright yellow text on a solid white background, but some people may have extra difficulty with other color combinations, such as red and green for individuals with deuteranopia (red-green colorblindness) or white and gray for individuals with visual impairments or reduced contrast sensitivity.

When color is used to distinguish certain elements, it should be accompanied by an additional identifier that is perceptible to those with color vision deficiencies. Without this extra cue, some users will not be able to differentiate elements based on only the color, meaning these users may not be able to pick up important information. For example, a website for quiz-taking may mark correct answers in green and incorrect answers in red. For a colorblind individual, the colors may be difficult or impossible to tell apart, so an extra identifier such as a checkmark or an 'X' symbol should be used to help convey this information.

Typography

Font style often contributes to the overall feel and theme of a website. Although some font styles are more elaborate and artistic than others, the chosen font style (along with font size and font weight/thickness) plays a major role in the accessibility of a web page. For text that is meant to be read thoroughly (such as an article or a paragraph), font style is primarily important for legibility and reading comprehension. Complex and intricate fonts (like cursive fonts, for example) may be difficult to read to the average reader and may be exceptionally difficult for those with reading disabilities such as dyslexia or ADHD. Letters and numbers should also be easily discernible from one another. For example, the lowercase 'L', the uppercase 'i', and the number '1' may all appear as a straight line among various fonts, making it nearly

impossible to tell them apart. Kerning (the spacing between characters) can also make it difficult to tell where words or characters start and end. For example, the letter pairs “r n” and “c l” may look like the letters “m” and “d” if there is not enough spacing between the letters.

One of the best ways to mitigate readability issues is to use a common font style such as Arial or Times New Roman. Even though common font styles may not be inherently more accessible than other fonts, some people (including those with reading disabilities) may have an easier time reading them because they have had a lot of experience with these fonts [18].

Conclusions

Both responsive web design and web accessibility play crucial roles in ensuring an optimal user experience on the web. By designing websites with responsiveness and accessibility in mind, more opportunities are opened up for a wider audience to access and interact with our content. Even though designing for responsiveness and accessibility can present challenges, their importance cannot be overstated.

In a world where technology is becoming increasingly intertwined with our daily lives, it is our responsibility as designers and developers to ensure that our websites and applications can be accessed by as many people as possible. Responsive web design allows pages to adapt to various devices and screen sizes, ensuring compatibility and usability across the board. Individuals can undergo consistent and intuitive experiences regardless of what device they are using. Designing for accessibility benefits all users, accommodating individuals with impairments but also providing a better experience for individuals without impairments. It

benefits everyone, regardless of their abilities. By incorporating responsive and accessible practices, we create a more inclusive digital environment.

Bibliography

[1] Asad Jameel, Khurram Shahzad, Afia Zafar, Usman Ahmed, Syed Jawad Hussain, and Ahthasham Sajid. 2018. The Users Experience Quality of Responsive Web Design on Multiple Devices. In Proceedings of the 2nd International Conference on Future Networks and Distributed Systems (Amman, Jordan) (ICFNDS '18). Association for Computing Machinery, New York, NY, USA, Article 69, 6 pages. https://doi.org/10.1145/3231053.3234632

[2] Jay Patel, Gil Gershoni, Sanjay Krishnan, Matti Nelimarkka, Brandie Nonnecke, and Ken Goldberg. 2015. A Case Study in Mobile-Optimized vs. Responsive Web Application Design. In Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (Copenhagen, Denmark) (MobileHCI '15). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2786567.2787135

[3] Gilbert Louis Bernstein and Scott Klemmer. 2014. Towards Responsive Retargeting of Existing Websites. In Adjunct Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (Honolulu, Hawaii, USA) (UIST '14 Adjunct). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2658779.2658805

[4] Jari-Pekka Voutilainen, Jaakko Salonen, and Tommi Mikkonen. 2015. On the Design of a Responsive User Interface for a Multi-Device Web Service. In Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems (Florence, Italy) (MOBILESoft '15). IEEE Press. https://doi.org/10.5555/2825041.2825052

[5] Bohyun Kim. 2013. Chapter 4: Responsive Web Design, Discoverability, and Mobile Challenge. In Library Technology Reports. https://doi.org/10.5860/ltr.49n6

[6] Regine M. Gilbert. 2019. Accessibility, Content, HTML, JavaScript, CSS, and the Land of Accessible Rich Internet Applications. In Inclusive Design for a Digital World: Designing with Accessibility in Mind. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5016-7_2

[7] Jaehyun Park, Sung H. Han, Hyun K. Kim, Youngseok Cho, Wonkyu Park. 2013. Developing Elements of User Experience for Mobile Phones and Services: Survey, Interview, and Observation Approaches. In Human Factors and Ergonomics in Manufacturing & Service Industries. https://doi.org/10.1002/hfm.20316

[8] Hyesung Ji, Youdong Yun, Seolhwa Lee, Kuekyeng Kim, Heuiseok Lim. 2017. An adaptable UI/UX considering user's cognitive and behavior information in distributed environment. In Cluster Computing. https://doi.org/10.1007/s10586-017-0999-9

[9] World Wide Web Consortium. 2022. Introduction to Web Accessibility. Retrieved from https://www.w3.org/WAI/fundamentals/accessibility-intro/.

[10] World Wide Web Consortium. 2019. Designing for Web Accessibility. Retrieved from https://www.w3.org/WAI/tips/designing/.

[11] web.dev. Introduction. Retrieved from https://web.dev/learn/design/intro/.

[12] web.dev. Flexbox. Retrieved from https://web.dev/learn/css/flexbox/.

[13] web.dev. Macro layouts. Retrieved from https://web.dev/learn/design/macro-layouts/.

[14] web.dev. Micro layouts. Retrieved from https://web.dev/learn/design/micro-layouts/.

[15] web.dev. Typography. Retrieved from https://web.dev/learn/design/typography/.

[16] web.dev. Responsive images. Retrieved from https://web.dev/learn/design/responsive-images/.

[17] MDN. tabindex. Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/tabindex/.

[18] web.dev. Typography. Retrieved from https://web.dev/learn/accessibility/typography/.

[19] web.dev. Learn Responsive Design. Retrieved from https://web.dev/learn/design/.

[20] MDN. Web technology for developers. Retrieved from https://developer.mozilla.org/en-US/docs/Web.

[21] World Wide Web Consortium. Web Accessibility Initiative. Retrieved from https://www.w3.org/WAI/.