Universal Access in the Information Society
https://doi.org/10.1007/s10209-023-01066-y

LONG PAPER

To wrap or not to wrap? A study of how long words are split when reflowed on magnified web pages

Frode Eika Sandnes[1]

Accepted: 26 October 2023

Abstract
Many low-vision users adjust the browser zoom level to make text more comfortable to read. Responsive websites attempt to fit the content within the viewport width, but several types of problems can potentially occur; long words may be wider than the viewport and thus partially hidden, they may cause large vertical space breaks, and words may be unnecessarily split or incorrectly split. This study set out to get insight into such word wrapping problems on responsive web pages. To help identify word wrapping issues, the tool HYPHERSPACE was developed. Experiments run on 91 websites suggest that hyphenation-related problems are prevalent. Excessive wrapping of words and overflowing words were the most notable problem on about 90% of the websites. One implication of this study is that web designers need to explicitly design for narrow viewports. The proposed tool can help web designers identify hyphenation problems on responsive web pages viewed with high magnification.

Keywords Accessibility · Low vision · Magnification · Hyphenation · Word wrapping · Reflow

1 Introduction

Word hyphenation is the breaking up of long words across two lines with a hyphen to signal the connection between the two-word parts. Consequently, more text can fit on a page, or it becomes easier to achieve justified right margins. Traditionally, word hyphenation was used for printed text with fixed layouts where a language competent proof-reader could check that the hyphenation adhered to formal hyphenation rules.

The concept of word wrapping has emerged with modern web technology where layouts are rendered dynamically. By default, words on web pages are not hyphenated, and if a word is too long, it may be pushed down the page and thus disconnected from its textual context, or even disappear from the viewport. Clearly, this can disrupt the reading process for low-vision users. Although hyphenation can slow down reading, intentional and careful use of soft hyphenation can prevent text from breaking up.

✉ Frode Eika Sandnes
Frodes@oslomet.no

[1] Department of Computer Science, Faculty of Technology, Art and Design, Oslo Metropolitan University, St. Olavs Plass, P.O. Box 4, 0130 Oslo, Norway

Word wrap is similar to word hyphenation but without the insertion of the hyphen-character and without the adherence to language specific rules. The two terms will be used interchangeably herein for simplicity.

There are situations where a lack of hyphenation can cause problems on responsive web pages with narrow viewports. Viewports may become narrow if browsing content on mobile devices with limited screen real estate, with small browser windows, or when browsing with zoom magnification to make reading more comfortable. The magnification means that the size of text and other elements are scaled up. Consequently, less text can fit in the viewport. The effective viewport width becomes narrower and can hold less content (although the number of physical pixels remains constant). Even though the visual appearance of the text is larger, its defined size is unchanged. The Web Content Accessibility Guidelines (WCAG) 2.1 recommends that web pages should support reflow with narrow viewport widths of 320 CSS pixels (criterion 1.4.10). This is equivalent to a display width of 1280 CSS pixels viewed with a 400% browser magnification.

Figure 1 shows an example web page with a text box and a paragraph of text with one very long word with a 150% browser magnification. Figure 2 shows the same page with a browser magnification of 175%. The increased zoom level leads to a narrower viewport width, which means that both

Published online: 11 November 2023

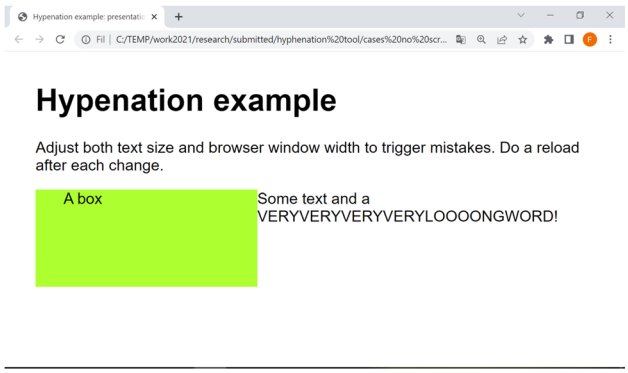

Fig. 1 The text snippet flows continuously with a wide viewport (150% magnification)

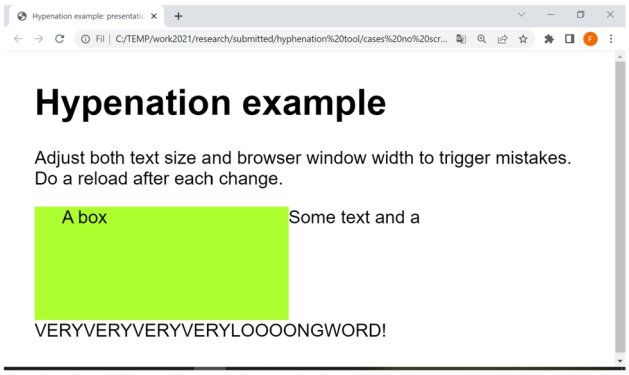

Fig. 2 The long word is separated from the text snippet with a narrow viewport (175% magnification)

the green text box and the long word cannot fit on the same horizontal line. The browser flows the word below the box. Consequently, the long word becomes disconnected from the other words of the same text paragraph.

Clearly, such vertical spacing can disrupt the reading process, especially if the text box is tall and the word flows outside the viewport. Figure 3 shows the same page with a browser magnification of 500%. In this case, the long word is wider than the viewport and a part of the word becomes hidden outside the viewport.

It must also be noted that the word character count is an unsuitable measure of word length in the context of layout on the web. It is therefore challenging to analyze potential hyphenation problems offline based on the text alone. This is because most websites employ proportional fonts where narrow letters, such as "l," occupy less horizontal space than the wide letter "m." For example, the two-character word "me" is wider than the three-character word "ill." How wide a word becomes therefore depends on the typeface use (glyph design), its font (specific glyph set, text size, and intra-word character spacing). Consequently, word length in the context

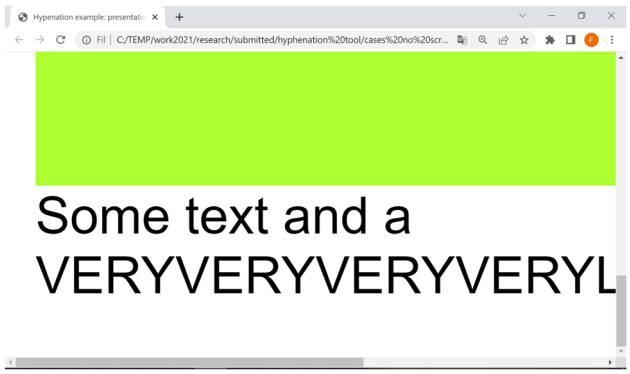

Fig. 3 Words without hyphenation are wider than the viewport and become partially occluded (500% magnification)

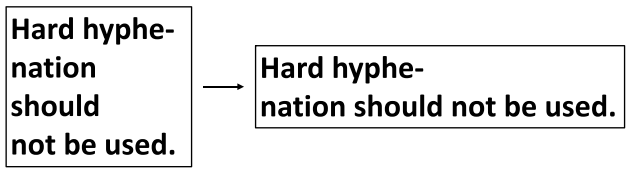

Fig. 4 The hard-coded hyphenations can cause lines to be broken in the middle of a text line if text is reflowed

of layout should be based on the actual width of words as rendered in the browser.

One way of enabling soft hyphenation is to use &shy, within long words in the html markup. This markup instructs the browser about where to hyphenate a word if it needs to be divided. If there is enough space words will be presented in one piece. Hard-coded hyphenation, that is, splitting up words into smaller parts with hard returns and the insertion of hyphens, is not a recommended, as such hard-coded hyphenations will cause line breaks at arbitrary positions within a line if the text is reflowed with different viewport widths (see Fig. 4).

Some browsers provide support for automatic hyphenation where the developer can set the language of the text explicitly so that the corresponding browser hyphenation dictionaries are used and specify the minimum number of characters in a hyphenated segment (hyphenate-limit-chars), or a web page can be configured to hyphenate words according to the given layout. Another approach supported by most browsers is word wrapping where the browser will wrap words that do not fit within a given space, but without the hyphens. Wrapped words do not provide readers any cues about words that have been split across lines besides the word parts.

A challenge with automatic hyphenation and word wrapping is that too many words may be hyphenated to make text fit a given width (see Fig. 5). It is argued herein that in the

Fig. 5 Example of how automatic word wrap can reduce readability due to excessive word wrapping in narrow viewports with an excessive margin

context of narrow viewport widths and or large texts, words should not be hyphenated unless they cause sentence splits or are wider than the viewport.

This study was therefore designed to address the following research questions:

RQ1: To what extent are soft hyphenation and word wrap mechanisms used on websites?

RQ2: How prevalent are text splits, word occlusions, and unnecessary word wrapping on websites viewed with narrow viewports (high magnification)?

RQ3: What hyphenation-related problems occur with typical page reflow implementations?

This work rests on a premise that many websites either do not hyphenate long words, or overuse word wrapping or auto-hyphenation. In addition to answering the research questions, an additional objective of this work was to provide a tool that can help identify similar problems on other websites. The HYPHERSPACE tool can help developers identify over- or under-use of hyphenation and identify words that should be hyphenated to prevent accessibility problems, while keeping the amount of hyphenation to a minimum.

2 Related work

Word hyphenation has been the focus of attention throughout the history of computer science from the early years [1] to the very current day [2]. Most work has focused on methods for automatic hyphenation of long words.

Hyphenation may be less important in languages with shorter words such as English with an alleged average word length of 5 characters [3]. However, many languages have a much longer average word length. For instance, Germanic languages such as German, Norwegian, Danish, and Swedish employ word concatenations where language rules dictate how several shorter words should be combined into long words. For example, according to the Guinness book of records in 2001 the longest Norwegian word with 60 characters is *minoritetsladningsbærerdiffusjonskoeffisientmålingsapparatur*. When broken into parts this word is built from the six words minoritets (minority), ladningsbærer (charge carrier), diffusjons (diffusion) koeffisient (coefficient), målings (measurement), and apparatur (equipment).

The longest alleged German word used to be *rindfleischetikettierungsüberwachungsaufgabenübertragungsgesetz* (63 characters), which can be translated as "the law concerning the delegation of duties for the supervision of cattle marking and the labeling of beef."

The English language also has some long words such as the 45 characters medical compound term *pneumonoultramicroscopicsilicovolcanoconiosis* meaning "a lung disease caused by inhalation of very fine silicate or quartz dust" (Encyclopedia Britannica); however, such compound words occur infrequently.

A ranking of 25 languages according to word lengths [4] lists Mongolian as the language with the longest words (mode 13 characters), German and Greek, (mode 12 characters), Magyar, Danish, Belarusian, French, and Russian (mode 11 characters). This list shows that some of these languages have occurrences of very long words (distribution tails), for example, German, Danish, and Slovenian listed with 25-character words. Based on this, it is argued herein that the issues of long words are relevant in many languages around the world.

The literature on word hyphenation in different languages support this view, such as Spanish [5], Greek [6], Turkish [7], Czechoslovak [2], and Norwegian [8, 9]. Differences in typical word lengths across languages are also known to cause challenges with general layout templates [10]. Long compound words also cause other human computer interaction challenges, such as prefix word prediction in text entry [11]. Except for wrapping long URLs in reflowed content, hyphenation and word wrapping issues are not explicitly addressed in the W3C Web Content Accessibility Guidelines (WCAG2.1).

Clearly, hyphenation is language specific as its rules are tied to languages (see, for instance, approaches documented for Spanish [5], Greek [6], Turkish [7], Czechoslovak [2], and Norwegian [8, 9]. Typical approaches involve rules and patterns [12] and hyphenation dictionaries [13]. Most work focuses on static layouts, but some have also addressed dynamic word hyphenation in documents with reflow [14].

Word hyphenation has been demonstrated to reduce reading speed [15, 16]. For example, one experiment showed that the reading speed was reduced as the ratio of hyphenated words approached 50% [8]. It has also been observed that word hyphenation can reduce accessibility for dyslexic readers [17]. With exceptions, such as speech

bubbles in comic strips [18], one could thus argue that it is preferable to avoid hyphenation at the cost of esthetical qualities such as block justification of text where both the left and right margins are vertically aligned.

Hyphenation is especially an issue on responsive web pages. Responsiveness can be understood as the fitting of contents within the viewport width through reflow [19, 20], adhering to users' color scheme preferences [21], etc. The goal of reflow is to avoid horizontal scrollbars to prevent the cognitive load of having to navigate the contents in two dimensions [19, 20]. With responsiveness and text reflow the need for hyphenation will depend on where words are rendered in the viewport [22]. Words with hard-coded hyphenation will appear hyphenated even if they appear at the beginning, or in the middle of a text line. Obviously, such hard-coded hyphenations can disrupt the reading process. Hard-coded hyphenation has also shown to be problematic for automatic processing of web contents [23].

Except for studies addressing web readability [24–26], the search for the literature did not uncover any past work on detecting problematic words on the web. However, a vast body of the literature addresses automatic tools for checking web pages. Many of these are intended for checking the visual layout of graphical user interfaces [27–30], cross browser rendering differences [31–34], responsive layouts rendering failures [35–37], and smartphone display orientation rendering failures [38]. Some tools address general accessibility [39–44], and others are specific to certain issues such as lack of color contrast [45–49] and inaccessible layouts [50].

3 Hypherspace

This section details the HYPHERSPACE accessibility tool. Checks and validations are performed within the browser using JavaScript to capture how the content is rendered. Relevant information about the rendered page is extracted by querying the document object model (DOM) and the browser.

Note that when increasing the magnification in the browser it is the ratio of CSS pixels in the magnified viewport width to unmagnified viewport width changed while the font size remains unchanged. Elements scaled in CSS pixels, rem, and em are scaled accordingly. Since the number of CSS pixels that can fit the viewport depends on this scaling factor, it is the actual width (and height) of the viewport in CSS pixels that changes, hence resulting in larger or smaller rendered text. Elements sized and positioned with reference to the viewport dimensions are not affected by changes in zoom magnification.

```
<p>
This is a text example.
</p>
```

Fig. 6 Original html source

```
<p>
      <span id="passage1">
            <span
      id="word1.1">This</span>
      <span id="word1.2">is</span>
      <span id="word1.3">a</span>
      <span id="word1.4">text
      example.</span>
      </span>
</p>
```

Fig. 7 Text nodes injected with span elements

3.1 Pre-processing

It is not possible to query a browser about where and how a particular word or letter is rendered because each word is not a separate DOM element. Text snippets are simply defined as *text nodes* of their parent DOM elements. To overcome this challenge, a pre-processing step is performed where span elements (inline containers) are dynamically inserted into the DOM such that they embrace one word of text each. The position of each word can then be determined by querying the bounds of its corresponding span element.

To achieve this, the DOM elements in the document body are traversed. For each element, its children are inspected to see if there are any text nodes. The content of each text node is split into words, and a span element is created for each word, together with an identifier, as well as a parent span to indicate a paragraph of words. The text node is then replaced with the list of span nodes containing the words (see Figs. 6 and 7).

All spaces, tabs, and newline characters within the text nodes are discarded. All elements with non-rendered text contents are also discarded, such as the title, style, and script elements. This approach ensures that nested structures with combinations of elements and text nodes are preserved.

3.2 Analysis

The analysis of how words are rendered can start once the coordinates of the bounding boxes of each word on the page are identified. The rendered width of each word is first compared to the viewport width. If its width is larger than the viewport width, the word is flagged as overflowing and partially occluded (Fig. 8).

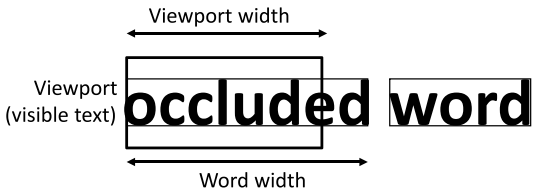

Fig. 8 Partially occluded words are wider than the viewport

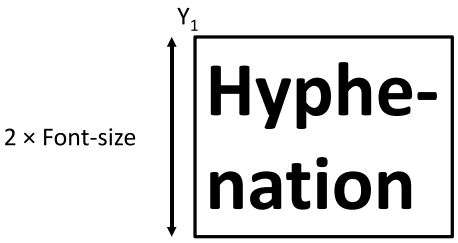

Fig. 9 Hyphenation or word wrap since the bounding box height is twice the font size

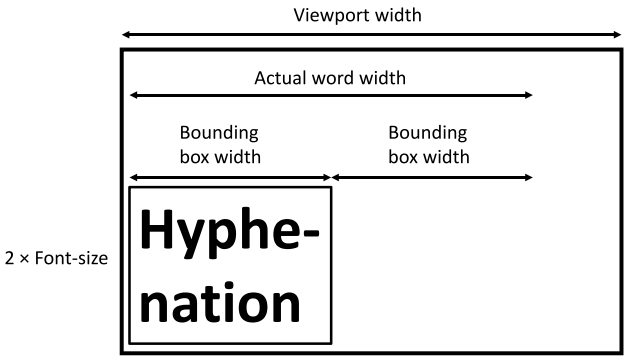

Fig. 10 Unnecessarily wrapped word

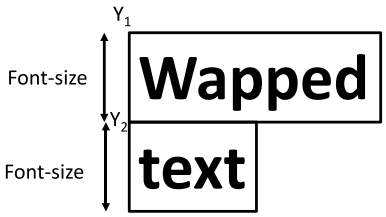

Fig. 11 Line break where two consecutive words have different *y*-coordinates

Fig. 12 No hyphenation (word bounding box height equals font size), no line breaks (same *y*-coordinate)

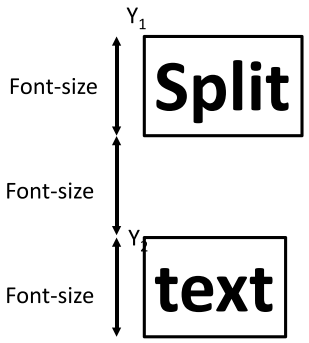

Fig. 13 Text splits are detected if the vertical distance between two consecutive words is larger than the font size

First, the rendered font size (letter height) for each text passage is determined by querying the DOM. Words with a bounding box height larger than its font size height are considered wrapped due to hyphenation or word wrapping (see Fig. 9). If a word is wrapped although its width is narrower than the viewport width, the word is flagged as unnecessarily wrapped (see Fig. 10).

The wrapping of full words with line breaks between words is simply detected by checking if the top positions (*y*-coordinates) of two consecutive words are different (see Fig. 11).

To detect whether any long words break up text, the set of words making up each text segment are analyzed collectively. First, the differences in the top position and bottom position (vertical) of consecutive words on the page are computed (the vertical difference between where the first word ends, and the next word begins). If these differences are negative, the words occur on the same line (see Fig. 12). Positive differences are compared to the font height. If any of these are larger than the font height, the difference is considered a text split (see Fig. 13).

The effective left text margin is found by analyzing the horizontal position of the bounding box of all the words. A frequency histogram is then computed, and the most frequent horizontal position is considered the most probable actual left margin. With magnified views, the actual left margin is likely to occur most frequently as the horizontal position of other words are likely to be dispersed throughout the viewport. This approach also eliminates negative margins, which can be observed on some websites when viewed with high magnifications.

The tool also counts the hard-coded soft hyphenation instances as their presence indicates that the developers have explicitly considered specific hyphenation issues. To trigger these errors, the browser must be configured by either

increasing the zoom level, reducing the viewport width, or both.

3.3 Reporting and visualization

The results of the tests are reported both textually (in the console) and visually. Visualizations are useful for seeing problems in context, and highlighting the problematic words is trivial once these are accessible via the respective span elements. However, visualizations are intended for human inspection and are therefore not useful in automatic test environments. The problems are therefore also reported textually, demonstrating that it is possible to configure automatic tests.

Figures 14, 15, 16, and 17 illustrate the tool in practice in the previous example in Figs. 1, 2, 3, and 4. Figure 14 shows the page in a wide viewport with no issues reported. Figure 15 shows the page in a narrower viewport and the text split is reported both visually with the yellow–red highlight and textually in the console. Figure 16 shows the page in a narrow viewport with the two overflowing words reported visually and in the console log. Figure 17 shows the words unnecessarily wrapped.

3.4 Deployment

Security mechanisms in modern browsers limit the possibilities on how to analyze web content to eliminate issues such as cross-site scripting. Two practical approaches were implemented. The first approach involves importing

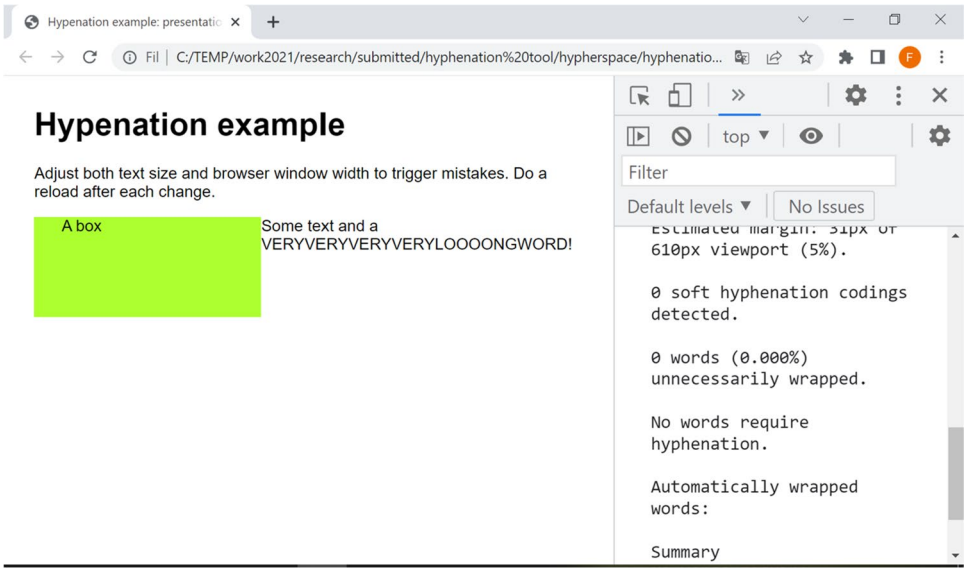

Fig. 14 Test tool reports no issues

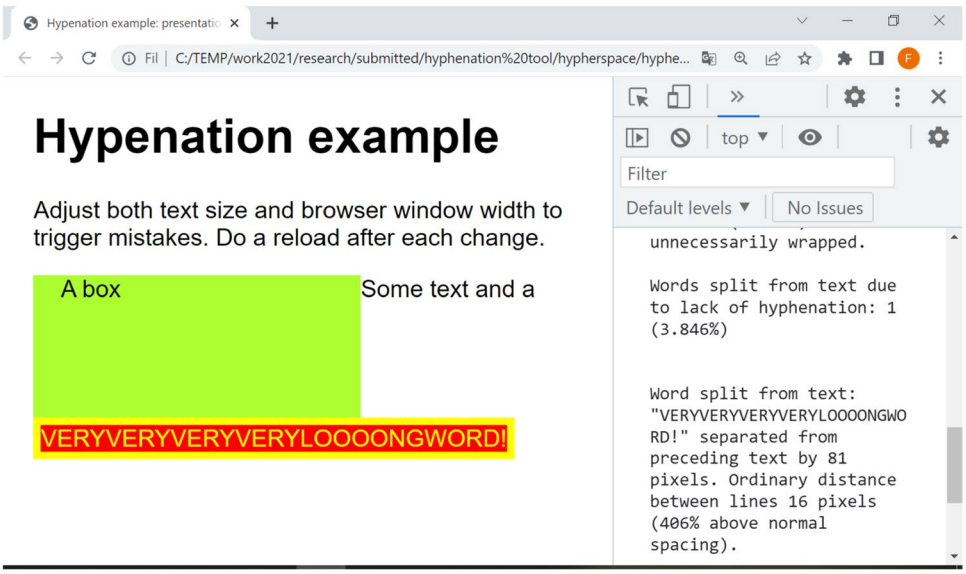

Fig. 15 Test tool reports one text split

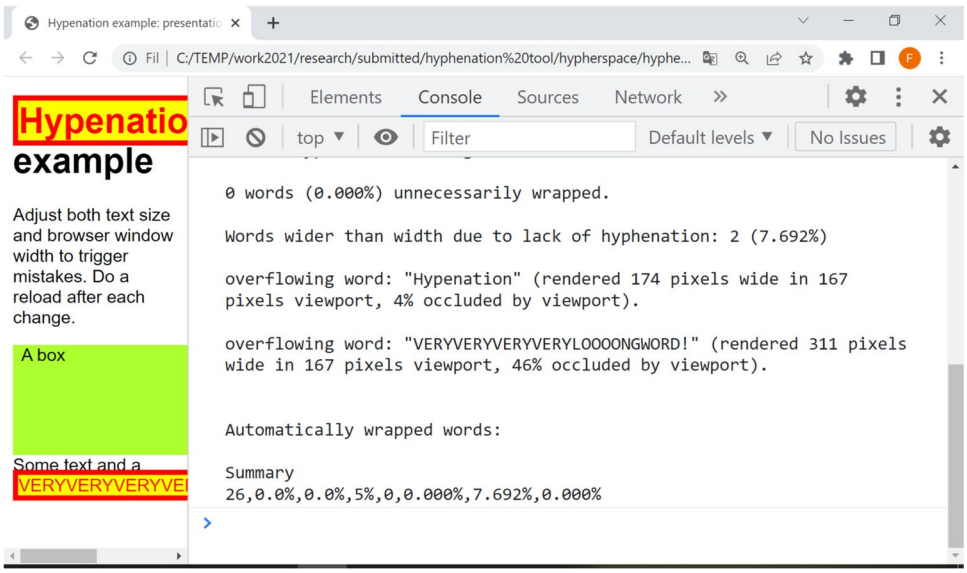

Fig. 16 Test tool reports two overflowing words

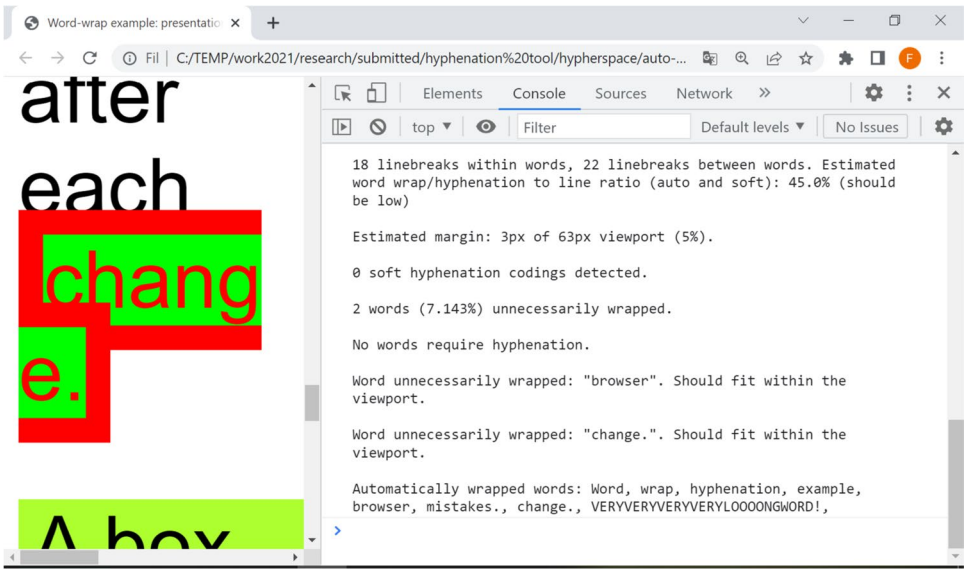

Fig. 17 Test tool reports unnecessarily wrapped words

the tool as script on the web pages. This may be useful when setting up automatic test scripts.

The second option is to run the tool as a bookmarklet. A bookmarklet is JavaScript code applied to the current web page when selecting a given bookmark. Bookmarklets are supported by common browsers such as Chrome, Firefox, Edge, and Safari. Bookmarklets have been used for resource sharing [51] and accessibility [52]. The bookmarklet implementation is useful for rapid visual inspection. The tool is available at https://github.com/frode-sandnes/HYPHERSPACE.

4 Experimental evaluation

4.1 Objective

An experimental evaluation was conducted with the tool to document the prevalence of the hyphenation-related problems discussed herein.

4.2 Materials

A small test suite of basic test cases was designed to trigger the hyphenation situation discussed herein (overflowing words, text splitting, and excessive hyphenation). However, simple toy examples are quite different from actual deployed websites in production. It was therefore decided to perform tests on a set of representative and frequently used websites.

A total of 91 Norwegian websites were selected for this evaluation. The inclusion criteria were high traffic or content of official importance. Moreover, the content had to be primarily in Norwegian. The list was compiled based on several top list of websites, including Kantar's list of most frequently visited websites in Norway (https://kantar.no/medier/offisielle-digital-tall-for-1q-2022/), and Bonzer's list of most popular online retail websites (https://bonzer.no/topp-50-norske-nettbutikker/).

The list was also supplemented with online banks, the major public governmental websites, and key higher education institution (HEI) websites. Popular non-Norwegian websites, such as Facebook, were discarded. The total list of 91 websites were classified into six categories with 20 retail websites, 17 news websites, 17 public/government websites, 16 HEI websites, 15 online banks, and 6 miscellaneous/other popular websites.

4.3 Procedure

The tests were performed using the bookmarklet implementation of HYPHERSPACE using the 64-bit Google Chrome version 103.0.5060.134 on a Lenovo laptop computer running Windows 10 with the display configured to 1680×1050 pixels with an operating system element scaling set to 175% (resulting in a display width of 960 CSS pixels). According to the Institute for Disability Research, Policy, and Practice (WebAIM) [53] Google Chrome was the most common web browser (37.5%) among low-vision users in 2018. The tests were conducted from July 29 to August 4, 2022.

In this study, the maximum browser magnification of 500% was used. The effective viewport width was set to 141 CSS pixels, less than half of the minimum specified in WCAG2.1 (320 CSS pixels typically found on many smartphones). The WCAG limit (assuming 400% magnification of a display width of 1280 CSS pixels) seems to serve as a compromise that meets the needs of most users with no corrected vision to moderately reduced vision. The study by WebAIM [53] suggests that the magnification needs of low-vision users are diverse. Of the 248 low-vision participants, 44% utilized browser magnification and 36.7% adjusted browser text size configurations. Of these, 30.7% needed less than 200% magnification, 29.7% needed 200–400% magnification, 17.9% needed 400% magnification or more, while 8% needed a magnification of 600% or more. These magnification needs can be corroborated by inspecting the technical specifications of commercial screen magnifiers that offer even higher magnification rates. For example, ZoomText, MAGic, and Supernova can each magnify contents by 3600%, 6000%, and even 6400%.

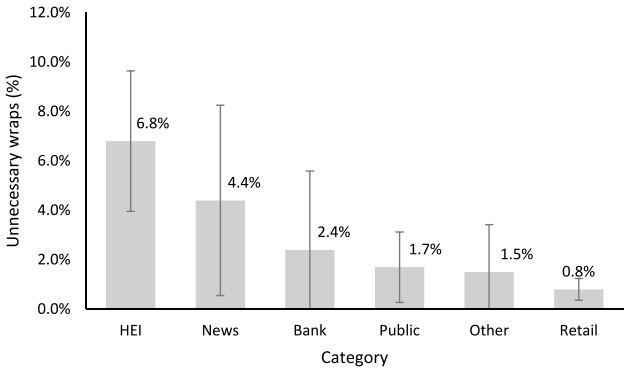

Fig. 18 Percentage of words that were unnecessarily wrapped. Error bars show 95% confidence intervals

The viewing conditions were configured as follows. First, DevTools were activated. Note that the DevTools window consumes some of the browser real estate. The viewport width of 141 CSS pixels was achieved by maximizing the browser window, increasing the magnification to 500%, and moving the DevTools divider as far right as possible (see Fig. 17). The resulting browser view comprised 141 CSS pixels mapped to 961 physical pixels.

The main landing pages for the 91 websites were loaded with the respective configuration before the HYPHERSPACE bookmarklet was applied and the results recorded.

4.4 Measurements

The hyphenation statistics for the landing page of each website were computed using HYPHERSPACE. The statistics for each website were copied into a spreadsheet for aggregated analysis. JASP was used for statistical analysis. Observations were also noted down during the testing.

5 Results

5.1 Use of hyphenation-related mechanism (RQ1)

Explicit coding of soft hyphenation was only observed on 14.3% of the sites, that is, nine news sites (more than 50%), two banks, one HEI, one of the other websites. No soft hyphenations were detected on any of the public or retail websites.

5.2 Prevalence of hyphenation-related problems (RQ2)

Overall, 90.1% of the website landing pages wrapped words unnecessarily (see Fig. 5), that is, the width of these words would fit within the viewport, but the words were split across lines, instead being moved to the next line. Figure 18 shows the percentage of words on the website categories that were unnecessarily wrapped with the given viewport. To calculate these aggregate statistics, the percentage of problematic words out of the total number of words on each landing page were computed. HEIs had the overall highest rate of wrapped words and retail the lowest rate.

The news category had the largest spread in the use of unnecessary word wrappings. A Kruskal–Wallis test detected a statistical difference between the categories ($H(5) = 15.424$, $p = 0.009$). Bonferroni post hoc testing showed that the difference between the HEI and retail websites was statistically significant ($p = 0.013$).

Overall, 89.0% of the websites triggered overflows. Figure 19 shows the percentage of overflowing words with the given viewport. Again, these percentages represent the portion of problematic words out of the total number of words. Here, public websites yielded the highest rate of word overflows, followed by HEIs. The other category and retail had the lowest percentage of overflows. A Kruskal–Wallis test detected a statistical difference between the categories ($H(5) = 29.154$, $p < 0.001$). Bonferroni post hoc testing flagged differences between public and retail ($p < 0.001$), news ($p = 0.004$), other ($p = 0.011$), bank ($p = 0.038$), as well as between HEI and retail websites ($p = 0.05$).

The results show that 17.6% of the websites triggered text splits (illustrated in Fig. 2) across the six categories, although there were few occurrences. Overall, the HEIs exhibited the highest rate of text passage splits.

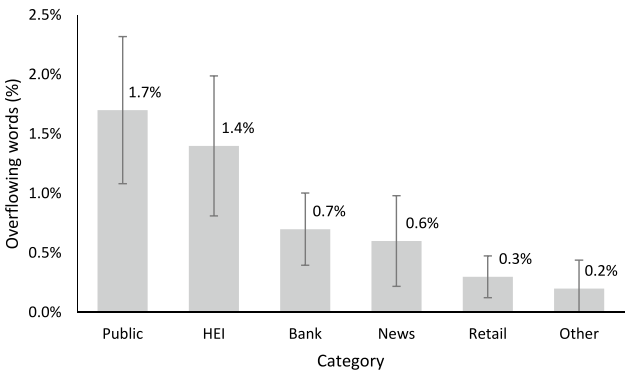

Fig. 19 Percentage of words that were overflowing. Error bars show 95% confidence intervals

5.3 Hyphenation-related page reflow problems (RQ3)

Some of the websites (5%) had a margin larger than 25% of the viewport (with some completely out of view). In these cases, the text was horizontally pushed partially or fully out of view (see Fig. 20). In several cases, this was due to multiple column layouts.

In several instances, words narrower than the viewport widths were pushed out of view because of magnification-growing margins. Although these words were only partially visible, they were not reported by the tool as needing hyphenation. Instead, these margins were reported as too wide.

One news site had several instances with photographs on the left with the text squeezed to the right. This is similar to the issue of magnification-growing margins as shown in Fig. 20 but with magnification-growing images instead of growing margins (see Fig. 21).

The testing revealed one website (Norwegian PayPal) with mechanisms that prevented the execution of bookmarklet scripts.

6 Discussion

6.1 Use of hyphenation-related mechanism (RQ1)

News sites stood out as about 50% of the websites contained explicitly coded soft hyphenation, yet they did not exhibit notably fewer wrapping and overflow errors. Perhaps, news

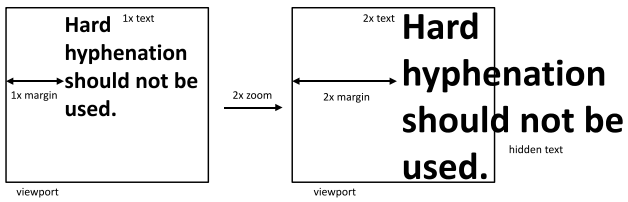

Fig. 20 Magnified margins push text out of view

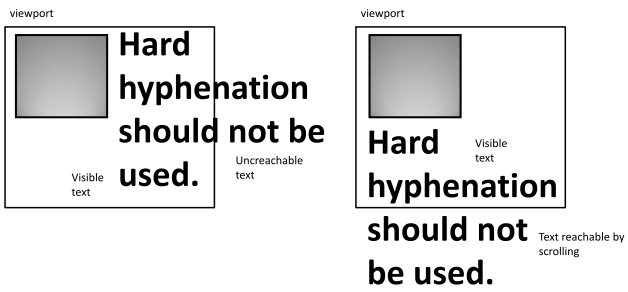

Fig. 21 Horizontal (left) versus vertical (right) stacking of elements

agencies have a tradition with hyphenation from printed publications which have transferred to online publications? Could it be that the soft hyphenation was intended as an esthetic device to achieve a certain layout for the default viewport width? Still, the main causes of wrapping and overflow errors, despite the soft hyphenation coding, were large magnification-growing margins and images leaving too little space even for the hyphenated words.

The absence of soft hyphenation coding among the other website categories could indicate too little awareness about soft hyphenation mechanisms. A lack of awareness may also explain the lack of soft hyphenation in non-news websites. Another explanation may be a lack of willingness to prioritize the explicit effort needed to implement soft hyphenation.

Manual coding of soft hyphenation may place an undesirable burden on the content creators. Therefore, a more pragmatic approach may be for content management systems to automatically assist with soft hyphenation coding using in-house hyphenation rules. Alternatively, CSS auto-hyphenation could be employed. Note that the auto-hyphenation is a relatively new addition to the CSS standard (CSS Text Module Level 3, 2023) and was only recently supported by most browsers. It is therefore unlikely that auto-hyphenation has been widely implemented on existing websites. A substantial portion of websites implemented word wrap. This can suggest that the designers considering the visual side effects of text growing beyond their containers. Yet, it also suggests that formal hyphenation rules were not prioritized. A notable portion of websites did not even implement word wrap, suggesting that growing text scenarios were not explicitly considered.

CSS word wrap can effectively be used to only break words that are wider than the viewport width even though the result does not follow formal hyphenation rules. One potential challenge with all the word-splitting mechanisms is that too many words may be hyphenated unnecessarily (as illustrated in Fig. 5) due to magnification-growing margins, images, and other elements. In many instances keeping the margin fixed, and vertically stacking elements may be more effective solutions.

The six classes of websites exhibited quite different characteristics, such as typical lengths and amounts of content on their landing pages, technical expertise, and development budgets. The content on the news sites changes frequently, sometimes every minute, while other websites have a longer interval between changes (typically days). It is indeed relevant to observe how owners of rapid content minimize accessibility issues while operating under time constraints. Websites of public organizations such as HEIs and government bodies are subject to stronger regulation, for instance the web accessibility directive (WAD), than private organizations such as retail and banking. However, retail and banking may be motivated by a goal to increase the customer base and number of visitors.

6.2 Prevalence of hyphenation-related problems (RQ2)

The results indicate that hyphenation and wrapping problems were indeed prevalent. Unnecessary word wraps and overflowing words seem frequent. Text passage splits were less frequent.

It appears that the public and HEI websites had more excessive word wrapping challenges (see Fig. 5) than retail, banks, and the *other* category. One possibility is that banks, retail, and *others* were indeed targeting a broader audience and therefore have resources allocated toward web design and development.

The goal is to keep hyphenation and within-word wrapping to a minimum, only to be used when a word is wider than the widths of its content. It seems that current web technology not yet provides sufficiently simple markup mechanisms for handling long words. Still, the results suggest that many of the hyphenation-related problems are cascading side effects of other layout problems. Certain word wrapping problems may be avoided if other layout issues are resolved first. For example, if a multiple column layout is used in a narrow viewport, there is a larger chance of word wrapping problems occurring since the two columns compete for the horizontal space. Instead, if these columns are stacked vertically in a single column exclusively utilizing the horizontal space, the risk of word wrapping problems is reduced.

6.3 Hyphenation-related page reflow problems (RQ3)

It was observed that a handful of websites had large left margins under the magnified viewing condition. It indicates that these pages have not been tested with a high magnification as these issues would have been very noticeable with simple testing.

The margins do not need to respond to magnification changes. Therefore, margins could be specified relative to the viewport width, instead of being specified in CSS pixels. At least, margins should be set such that they do not exceed a small fraction of the viewport width.

One may question whether large margins qualify as an accessibility problem. If the user is viewing a large margin page with a wide viewport and a large text size, a large margin should not pose a notable problem, unless the margin is scaled in relation to the text size (with rem or em). Problems start to occur with narrow viewports.

One possible coping strategy is to use the horizontal scrollbar to align the text margin with the left viewport border, if sideways scrolling is enabled. If a large left margin

causes words to be obstructed by the right viewport border, or words are wrapped although not wider than the viewport, one may argue that the margin constitutes an accessibility problem. Regardless, web developers should ensure that margins are dimensioned in relation to the viewport width to avoid margins occupying most of the view. Another argument is that one should not compensate for large margin problems using hyphenations.

Situations where text is pushed out of view to the right by images could have been remedied by specifying narrow viewpoint media query breakpoints where the image and the text are stacked vertically into a single column (see Fig. 21). Similarly, word wrapping is a problematic compensator for a lack of space caused by horizontal stacking of elements side-by-side within narrow widths. Again, media query breakpoints could be specified for narrow viewports to ensure that elements are vertically instead along a single column.

Another challenge with several of the websites was fixed (sticky) consent boxes that obstructed the entire viewport in the large text test case. Some websites provided no options for removing these fixed elements within the website interface. It has been argued that such consent boxes often are designed to nudge visitors into accepting conditions [54]. The sticky elements are not a hindrance for the tool, but they would hinder access for browser magnification users. For such situations, Brinkmann's modified Kill Sticky Headers bookmarklet could be used [55].

6.4 Limitations

The results reported herein only represent one scenario with one distinct viewport width. Clearly, different results would be produced with other viewport configurations. This may especially be the case for text splits. In fact, the intentional split test case does not trigger with the current experimental configuration. Perhaps, text splits would occur with other width viewport configurations for other websites. This could, for instance, occur on websites with viewport ranges where media queries are not clearly defined.

Testing of live websites is not replicable as the contents continuously change. One approach adopted by several researchers [35–37] is to capture local copies of websites. It can be challenging to replicate a dynamically generated website. Moreover, to make such websites available as test suites to the research community raises questions of copyright and intellectual property rights. The English language websites used in [35–37] were not suitable for the context of this study due to the lack of long compound words.

Only the landing pages for each website were included in the evaluation. The results may have been different if one had explored specific pages on these websites. Still, the landing pages are important as it is often a first point of entry to a website, unless the site is reached via a search engine.

A moral issue with such testing is that listing a website with test results can be interpreted as criticism and that these websites have certain problems. The 91 websites were therefore not listed explicitly, and the results have been presented at an aggregated level. It must be noted that the websites selected herein were not selected based on their flaws and are not problem cases; they were simply selected based on popularity and importance. These websites are likely to be representative of other sites not included.

7 Conclusion

Hyphenation problems on magnified reflowed web pages were studied. The results show that soft hyphenation is not frequently used. Most instances were found on news media websites. Moreover, text passage breaks due to unhyphenated long words is uncommon, while unnecessary splitting of words indeed is a common challenge. The HYPHERSPACE tool developed for this study can also be used by developers for identifying problems related to word wrapping and hyphenation on responsive web pages. The tool detects unnecessarily split words, and words that are not split but should have been. The implication of the results is that developers should design for narrow viewports where elements are stacked vertically so that words can span the entire viewport width, margins should not be specified in pixels, and developers should test their designs with narrow viewports. The tool is available at https://github.com/frode-sandnes/HYPHERSPACE. Future work includes investigating the effects CSS inter-word text justification (block justification) may have on reading due to the resulting large spacing between words.

Author contributions The single author did all the work associated with this study.

Funding Open access funding provided by OsloMet - Oslo Metropolitan University.

Declarations

Conflict of interest No funding was received for conducting this study. The author has no competing interests to declare that are relevant to the content of this article.

References

1. Rich, R.P., Stone, A.G.: Method for hyphenating at the end of a printed line. Commun. ACM. ACM **8**(7), 444–445 (1965)
2. Sojka, P., Sojka, O.: Towards New Czechoslovak Hyphenation Patterns. Zpravodaj Československého sdružení uživatelů TeXu **30**(3), 118–126 (2020)
3. Sandnes, F.E., Thorkildssen, H.W., Arvei, A., Buverad, J.O.: Techniques for fast and easy mobile text-entry with three-keys. In: 37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the (pp. 10-pp). (2004) IEEE
4. Parikh, R.S.: Distribution of Word Lengths in Various Languages. http://www.ravi.io/language-word-lengths (2015). Accessed 6 Dec 2022
5. Manas, J.A.: Word division in Spanish. Commun. ACM. ACM **30**(7), 612–616 (1987)
6. Noussia, T.I.: A rule-based hyphenator for Modern Greek. Comput. Linguist.. Linguist. **23**(3), 361–376 (1997)
7. MacKay, P.A.: Turkish hyphenations for TEX. TUGboat **9**(1), 12–14 (1988)
8. Kristensen, T.: A neural network approach to hyphenating Norwegian. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, vol. 2, pp. 148–153. IEEE (2000)
9. Kristensen, T., Langmyhr, D.: Two regimes of computer hyphenation-a comparison. In: IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222), vol. 2, pp. 1532–1535. IEEE (2001)
10. Mahajan, S., Alameer, A., McMinn, P., Halfond, W.G.: Effective automated repair of internationalization presentation failures in web applications using style similarity clustering and search-based techniques. Softw. Test. Verif. Reliab. **31**(1–2), e1746 (2021)
11. Sandnes, F.E.: Reflective text entry: a simple low effort predictive input method based on flexible abbreviations. Procedia Comput. Sci. **67**, 105–112 (2015)
12. Sojka, P., Sojka, O.: Towards Universal Hyphenation Patterns. In: RASLAN. pp. 63–68. (2019)
13. Peterson, J.L.: Use of Webster's Seventh New Collegiate Dictionary to construct a master hyphenation list. In: Proceedings of the June 7–10, 1982, National Computer Conference, pp. 665–670. (1982)
14. Pinkney, A.J., Bagley, S.R., Brailsford, D.F.: Reflowable documents composed from pre-rendered atomic components. In: Proceedings of the 11th ACM Symposium on Document Engineering, pp. 163–166. (2011)
15. Häikiö, T., Hyönä, J., Bertram, R.: The role of syllables in word recognition among beginning Finnish readers: evidence from eye movements during reading. J. Cogn. Psychol.Cogn. Psychol. **27**(5), 562–577 (2015)
16. Laarni, J.: Optimizing Text Layout for Small-screens: the Effect of Hyphenation and Centering. In: Human-Centered Computing, pp. 260–264. CRC Press (2019)
17. de Santana, V.F., de Oliveira, R., Almeida, L.D.A., Baranauskas, M.C.C.: Web accessibility and people with dyslexia: a survey on techniques and guidelines. In: Proceedings of the international cross-disciplinary conference on web accessibility, pp. 1–9. (2017)
18. Trubnikov, S.V., Denysiuk, O.R.: Implementation of a helper program for comics creation using text processing methodS. Comput. Model. Anal. Control Optim. **2020**(1), 64–69 (2020)
19. Dick, W.E.: Operational Overhead Caused by Horizontal Scrolling Text. Technical note. http://nosetothepage.org/Fitz/2dScroll.html(2017). Accessed 19 Mar 2022
20. Sandnes, F.E.: Lost in OCR-Translation. Pixel-based Text Reflow to the Rescue. Magnification of Archival Raster Image Documents in the Browser without Horizontal Scrolling. In: Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments, pp. 500–506. (2022)
21. Pedersen, L.A., Einarsson, S.S., Rikheim, F.A., Sandnes, F.E.: User interfaces in dark mode during daytime–improved productivity or just cool-looking? In: Universal Access in Human-Computer Interaction. Design Approaches and Supporting Technologies: 14th International Conference, UAHCI 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, Proceedings, Part I 22, pp. 178–187. Springer (2020)
22. W3C: CSS Text Module Level 3. Hyphenation (2022). https://www.w3.org/TR/css-text-3/#hyphenation. Accessed 6 Dec 2022
23. Bildhauer, F., Schäfer, R.: Token-level noise in large Web corpora and non-destructive normalization for linguistic applications. In: Proceedings of Corpus Analysis with Noise in the Signal (CANS 2013), (2013)
24. Eika, E.: Universally designed text on the web: towards readability criteria based on antipatterns. Stud. Health Technol. Inform **229**, 461–470 (2016)
25. Eika, E., Sandnes, F.E.: Authoring WCAG2. 0-compliant texts for the web through text readability visualization. In: Universal Access in Human-Computer Interaction. Methods, Techniques, and Best Practices: 10th International Conference, UAHCI, Held as Part of HCI International 2016, Toronto, ON, Canada, July 17–22, 2016, Proceedings, Part I 10, pp. 49–58. Springer (2016)
26. Eika, E., Sandnes, F.E.: Assessing the reading level of web texts for WCAG2. 0 compliance—can it be done automatically?. In: Advances in Design for Inclusion: Proceedings of the AHFE 2016 International Conference on Design for Inclusion, July 27–31, 2016, Walt Disney World®, Florida, USA, pp. 361–371. Springer (2016)
27. Chang, T.H., Yeh, T., Miller, R.C.: GUI testing using computer vision. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1535–1544. (2010)
28. Liu, Z., Chen, C., Wang, J., Huang, Y., Hu, J., Wang, Q.: Nighthawk: fully automated localizing UI display issues via visual understanding. IEEE Trans. Softw. Eng.Softw. Eng. **49**(1), 403–418 (2022)
29. Moran, K., Li, B., Bernal-Cárdenas, C., Jelf, D., Poshyvanyk, D.: Automated reporting of GUI design violations for mobile apps. In: Proceedings of the 40th International Conference on Software Engineering, pp. 165–175. (2018)
30. Tanno, H., Adachi, Y., Yoshimura, Y., Natsukawa, K., Iwasaki, H.: Region-based detection of essential differences in image-based visual regression testing. J. Inf. Process. **28**, 268–278 (2020)
31. Choudhary, S.R., Prasad, M.R., Orso, A.: X-PERT: accurate identification of cross-browser issues in web applications. In: 2013 35th International Conference on Software Engineering (ICSE), pp. 702–711. IEEE (2013)
32. Mahajan, S., Alameer, A., McMinn, P., Halfond, W. G.: Xfix: an automated tool for the repair of layout cross browser issues. In: Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 368–371. (2017)

33. Mesbah, A., Prasad, M. R.: Automated cross-browser compatibility testing. In: Proceedings of the 33rd International Conference on Software Engineering, pp. 561–570. (2011)
34. Saar, T., Dumas, M., Kaljuve, M., Semenenko, N.: Browserbite cross-browser testing via image processing. Softw. Pract. Exp. **46**(11), 1459–1477 (2016)
35. Althomali, I., Kapfhammer, G.M., McMinn, P.: Automated Repair of Responsive Web Page Layouts. In: 2022 IEEE Conference on Software Testing, Verification and Validation (ICST), pp. 140–150. IEEE (2022)
36. Ryou, Y., Ryu, S.: Automatic detection of visibility faults by layout changes in HTML5 web pages. In: 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST), pp. 182–192. IEEE (2018)
37. Walsh, T.A., Kapfhammer, G.M., McMinn, P.: Automatically identifying potential regressions in the layout of responsive web pages. Softw. Test. Verif. Reliab. **30**(6), e1748 (2020)
38. Amalfitano, D., Riccio, V., Paiva, A.C., Fasolino, A.R.: Why does the orientation change mess up my Android application? From GUI failures to code faults. Softw. Test. Verif. Reliab. **28**(1), e1654 (2018)
39. Abascal, J., Arrue, M., Fajardo, I., Garay, N., Tomás, J.: The use of guidelines to automatically verify web accessibility. Univ. Access Inf. Soc. **3**, 71–79 (2004)
40. Abascal, J., Arrue, M., Valencia, X.: Tools for web accessibility evaluation. Web accessibility: a foundation for research, pp. 479–503. (2019)
41. Abduganiev, S.G.: Towards automated web accessibility evaluation: a comparative study. Int. J. Inf. Technol. Comput. Sci. **9**(9), 18–44 (2017)
42. Alsaeedi, A.: Comparing web accessibility evaluation tools and evaluating the accessibility of webpages: proposed frameworks. Information **11**(1), 40 (2020)
43. Brajnik, G.: Comparing accessibility evaluation tools: a method for tool effectiveness. Univ. Access Inf. Soc. **3**, 252–263 (2004)
44. Kasday, L.R.: A tool to evaluate universal Web accessibility. In: Proceedings on the 2000 Conference on Universal Usability, pp. 161–162. (2000)
45. Hansen, F., Krivan, J.J., Sandnes, F.E.: Still not readable? An interactive tool for recommending color pairs with sufficient contrast based on existing visual designs. In: Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility, pp. 636–638. (2019)
46. Sandnes, F.E., Zhao, A.: An interactive color picker that ensures WCAG2. 0 compliant color contrast levels. Procedia Comput. Sci. **67**, 87–94 (2015)
47. Sandnes, F.E.: Understanding WCAG2. 0 color contrast requirements through 3D color space visualization. Stud. Health Technol. Inform **229**, 366–375 (2016)
48. Sandnes, F.E.: An image-based visual strategy for working with color contrasts during design. In: Computers Helping People with Special Needs: 16th International Conference, ICCHP 2018, Linz, Austria, July 11–13, 2018, Proceedings, Part I 16, pp. 35–42. Springer, (2018)
49. Sandnes, F.E.: Inverse Color Contrast Checker. Automatically Suggesting Color Adjustments that meet Contrast Requirements on the Web. In: Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility, pp. 1–4. (2018)
50. Panchekha, P., Geller, A.T., Ernst, M.D., Tatlock, Z., Kamil, S.: Verifying that web pages have accessible layout. ACM SIGPLAN Not. **53**(4), 1–14 (2018)
51. Ma, K., Zhang, L.: Bookmarklet-triggered unified literature sharing services in the cloud. Int. J. Grid Util. Comput.Comput. **5**(4), 217–226 (2014)
52. Mangiatordi, A., Sareen, H.S.: Farfalla project: browser-based accessibility solutions. In: Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, pp. 1–2. (2011)
53. WebAIM.: Survey of Users with Low Vision #2. Institute for Disability Research, Policy, and Practice. https://webaim.org/projects/lowvisionsurvey2/ (2018). Accessed 9 July 2023
54. Utz, C., Degeling, M., Fahl, S., Schaub, F., Holz, T.: (Un) informed consent: Studying GDPR consent notices in the field. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 973–990. (2019)
55. Brinkmann, B.: Remove anything that is sticky on websites. https://www.ghacks.net/2018/08/16/remove-anything-that-is-sticky-on-websites/ (2018). Accessed 19 Mar 2022