Adaptive Layout Template for Effective Web Content Presentation in Large-Screen Contexts

Michael Nebeling, Fabrice Matulic, Lucas Streit, and Moira C. Norrie
{nebeling|matulic|norrie}@inf.ethz.ch
Institute of Information Systems, ETH Zurich
CH-8092 Zurich, Switzerland

ABSTRACT

Despite the fact that average screen size and resolution have dramatically increased, many of today's web sites still do not scale well in larger viewing contexts. The upcoming HTML5 and CSS3 standards propose features that can be used to build more flexible web page layouts, but their potential to accommodate a wider range of display environments is currently relatively unexplored. We examine the proposed standards to identify the most promising features and report on experiments with a number of adaptive layout mechanisms that support the required forms of adaptation to take advantage of greater screen real estates, such as automated scaling of text and media. Special attention is given to the effective use of multi-column layout, a brand new feature for web design that contributes to optimising the space occupied by text, but at the same time still poses problems in predominantly continuous vertical-scrolling browsing behaviours. The proposed solutions were integrated in a flexible layout template that was then applied to an existing news web site and tested on users to identify the adaptive features that best support reading comfort and efficiency.

Keywords

Adaptive layout, large display settings, new web standards

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Screen design*

General Terms

Design, Human Factors

1. INTRODUCTION

Since the early days of the web and the realisation that a static presentation of web content was overly limiting, researchers, web developers and designers alike have sought to come up with methods to adapt content to a variety of constraints and limitations imposed by a number of factors, in particular the viewing context. Considering the wide variety of browsing devices, types of content to be delivered and consuming conditions, the task of providing a solution that tackles all these different concerns at the same time is formidable. This complexity has led software engineers and researchers to develop comprehensive frameworks with the flexibility to distribute content in a broad range of contexts as well as concentrate on specific subsets of these issues in order to solve them more effectively.

In the past decade, the booming emergence of small-form factor devices such as PDAs, smartphones and tablets have caused much research effort to be directed towards addressing the particular problems associated with this category of appliances, i.e. small screen size, reduced resources and particular interaction models. However, recent years have also seen an increase at the other end of the spectrum, namely that of large-size viewing terminals, which are also becoming more widespread[1]. Despite this clear trend towards higher resolutions and screen sizes, this segment has surprisingly received little attention from the web engineering community.

Studies show that large displays bring productivity benefits if the interfaces are suitably adapted to make use of the available screen space [6, 18]. For web sites, pages that present a greater degree of adaptability to larger screen dimensions can increase the user's experience and overall comfort when engaging with web content, as we will demonstrate in this paper. Yet, an investigation on news web sites that we conducted recently [17] shows that the vast majority of them fail to fully utilise available screen real estate on large displays, as the layouts of web pages prove to be mostly fixed and static. The result is the appearance of wide margins surrounding the main content as well as smaller text and multimedia elements. To some extent, the latter can be remedied using zoom features of web browsers, but this does not always yield satisfactory results and, more importantly, it does not solve the layout problem. This failure to adapt affects the browsing experience and to a greater or lesser degree detracts from the web site's general usability.

The reasons for using static layout have mostly to do with design issues (it is easier to design for fixed dimensions) and the perceived added complexity and costs that a system supporting highly adaptive and flexible page structuring would incur. The increasing diversity of the web-browsing device landscape is real, however, and so instead of devel-

[1] http://www.w3schools.com/browsers/browsers_display.asp

DocEng'11, September 19–22, 2011, Mountain View, California, USA.

oping adaptive presentation engines for their content, publishers have opted to produce several versions of their web sites dedicated to specific device categories, mainly desktops and smartphones. There is currently no sign that content providers will address the needs of consumers with large and very large displays and so we feel that work needs to be done in that area. As a starting point, we believe that if the problem of web page layout adaptation could be posed purely in terms of web standards and dealt with in a relatively straightforward manner using native web technologies, it would provide an incentive to designers and publishers to consider more flexible layout strategies.

Based on the above philosophy, we show in this paper how layout adaptation can be achieved using standard web technologies and especially new features of HTML5 and CSS3 that lend themselves well to tackling this problem. By adopting this approach that is likely to be attractive to web developers and designers, we differentiate ourselves from prior work, which is mostly based on complex layout-generating systems implemented in proprietary browser software or requiring expensive server-side computations. The main technical contribution of this paper is an adaptive layout template that can accommodate a range of viewing situations and, because it is based on only native web technologies, can easily be applied to existing web sites (see Figure 3 at the end of the paper). In the latter part of the paper we report on the results of a preliminary user study conducted on such adapted news articles to evaluate the perception and impact of the different layouts that we create in this manner.

Our paper is structured as follows: we begin by discussing related work and relevant features of HTML5 and CSS3. This is then followed by an analysis of the required adaptation techniques and an implementation of these in the form of an adaptive layout template. Finally, we present and discuss the results of our user evaluation and conclude with an assessment of the offered functionality and its potential to improve the user experience.

2. RELATED WORK

As stated above, the adaptation of web sites to meet the constraints imposed by a heterogeneous array of consuming devices has been the subject of extensive research. In this area, one can generally distinguish two different categories of adaptation schemes: generational and transformational methods. The first type of techniques seek to build or generate dynamic, adaptive web pages usually using a dedicated server-based framework to retrieve raw content, often from a database, and then, based on a number of rules and settings, create a web page accordingly. Further adaptations can then also be performed with the help of client-side scripts. AMACONT [7] is an example of such a framework that explicitly addresses presentation and adaptive formatting of content based on the Hera web design methodology to generate reusable adaptive web documents. In the broader context of adaptive content delivery, there have also been a number of systems that address different aspects of context-specific content consumption such as multimedia content [15], web services [11] and user interfaces in general [3]. At the heart of these approaches, is the aim to provide device-independent access to content, often with a desire to separate the different architectural concerns, i.e. data management, application logic and presentation (see [4] and [8] for examples).

The second type of adaptation techniques take existing, already structured web pages and reformat them to fit a set of new constraints, typically those associated with a mobile device with smaller screen size and processing resources. Here also the literature abounds with research efforts which tackle this problem. Most of the more advanced techniques attempt one way or another to infer the semantic structure of the page using segmentation or parsing algorithms and alter it or build a new document from extracted page components so that the returned web page meets the constraints of the target client [5, 10]. Methods that analyse the content of web pages including multimedia elements such as images in order to remove extraneous items or portions thereof have also been proposed [14].

Looking at the layout-producing problem itself, a popular approach consists of determining layout requirements for a document (web or otherwise) to be dynamically generated as a set of constraint-based relationships or rules between the different components [12, 13, 19]. These constraints are typically defined in one or more templates that, when fed with input values representing target viewing conditions, generate with the help of an appropriate solver a suitable layout for the given content. One of the main difficulties here lies in the specification and characterisation of those design templates that have to abide by a number of aesthetic and content-driven rules while also allowing for a great degree of flexibility in order to adapt to a wide range of rendering contexts.

The problem with most of the techniques developed so far is that they either target a very specific category of terminals, namely small-form factor devices, or they are relatively complex to implement with pure web standards and technologies, which are usually favoured by web designers and developers. As a matter of fact, some authors of adaptive layout algorithms have even suggested extensions to CSS to enable constraint-based layout generation in a more web-compliant manner [1]. While these propositions may not have made it yet to the W3C consortium, the standards continue to evolve and expand with new specifications that address a number of problems including that of web page layout and adaptation. Specifically, the new HTML5 and CSS3 specifications bring features such as multi-column layout and advanced media queries to the table, which can be used to afford web sites a certain degree of presentation adaptivity and flexibility. We examine these new features in more detail in the next section.

3. REVIEW OF WEB STANDARDS

The HTML5 standard is currently still in working draft state[2], but many parts of the specification are considered stable and already implemented in some web browsers. At the syntactical level, HTML5 introduces two kinds of new elements. First, elements such as <nav> for a navigation block, <header> for the top and <aside> for side bar elements of a web page aim at replacing the generic block elements and with semantically more meaningful ones. Second, elements such as the <video> or the <svg> tag provide a standardised way of embedding multimedia elements such as videos or vector-based graphics directly in a web page.

With respect to adaptive layout techniques, the new semantic elements of HTML5 are useful for annotating page

[2]http://www.w3.org/TR/2010/WD-html5-20101019/

elements, which may open up new and simpler ways for the automatic adaptation of web page layouts based on document analysis. However, due to the poor backwards compatibility, it will probably take some time before web developers adopt these new standards for this purpose and refrain from using or elements with custom id or class attributes. Furthermore, the <video> tag introduces a new possibility to embed videos as regular DOM elements and thus allow them to be styled with CSS and manipulated using JavaScript. For example, the dimensions of a video could be changed when viewed on a large screen and, additionally, its content could be replaced with a high-definition version—this only using web standards.

The development of CSS3 is modularised and consists of various separate recommendations which are developed independently. An important module is the CSS3 Values and Units module[3] which defines a number of new length units that can be used to specify the dimensions of page elements now also relative to the browser viewport. These new units, most notably vw and vh referring to the width and height of the viewport, respectively, seem in theory very useful for constructing flexible layouts that dynamically scale with the size of the viewport, but unfortunately current browser support is minimal to non-existent. The most practical unit for creating adaptive layouts therefore remains em which always refers to the font size of the current element. If the dimensions of an element are specified using this unit, the element keeps its proportions to the text when the font size is changed. This is desirable if users are allowed to adjust the font size via the web interface or the browser. For pixel-based layouts, it is still a problem to match these to the corresponding physical units, as the browser would need to know the pixel density of the display, typically measured in dots per inch (DPI), in order to convert them. As it turns out, our experiments have shown that all major browsers currently work with the system DPI value, e.g. 96 on Windows systems, which can vastly differ from the actual physical value. As a consequence, absolute length units are not suitable in the screen context unless corrected using additional methods discussed later in the paper.

Since CSS2, the scope of a style sheet can be limited with respect to certain media types. For instance, different layouts can be specified for on-screen reading or printing. CSS3 expands this concept by providing access, not only to the media type, but also to media-specific features[4]. By querying these properties, certain styles can be included or excluded depending on various factors such as the size of the browser viewport or the whole screen, the orientation of the device, the aspect ratio, or the resolution of the output device, i.e. the pixel density mentioned before. Equally important is the fact that media queries are re-evaluated whenever a relevant variable is changed at runtime. This means that CSS media queries are sensitive to client-side events, but do not require event handling via JavaScript. As a result, media queries can be used to automatically switch the layout of a web page, for example, when the device orientation changes or the size of the browser window exceeds a certain value. The Media Queries module is currently in the Candidate Recommendation stage and so all major browsers in their newest versions support most parts of the specification.

[3]http://www.w3.org/TR/2006/WD-css3-values-20060919/

[4]http://www.w3.org/TR/2010/CR-css3-mediaqueries-20100727/

Finally, the CSS3 Multi-column Layout module[5] introduces new CSS properties to lay out the content of web page elements in multiple columns. Elements inside a multi-column container are thereby treated the same way as content in regular containers, with the difference that alignment and positioning of elements now concern only the column in which they appear. However, content elements with an overflow inside a multi-column container will always be clipped at the centre of the column gap, which is unlike other container elements where overflow can be explicitly controlled using the corresponding CSS properties. The specification also foresees basic means to control the column-breaking behaviour, i.e. elements can be forced to remain together as one unit in the same column, which can be useful to avoid the separation of an image and its caption into two different columns. This mechanism, however, is not yet supported by browsers.

4. LARGE-SCREEN ADAPTATION

In order to assess the potential of the aforementioned features to form the basis of an adaptive layout solution, we now take a requirements perspective and analyse the technical and design challenges of adapting web page designs depending on the screen context, with a particular view towards large-display viewing situations.

One of the main differences with large screens is that they usually come with a significantly higher DPI value compared to smaller screens, i.e. the physical size of a pixel becomes smaller on such screens. Now the problem is that many web sites use fixed pixel values to specify the font size [17]. For example, a 15" screen running on a 1024x768 resolution has a DPI value of 85. Text displayed at a font size of 12 pixels has therefore a physical height of 3.59mm on such a screen. In contrast, a 30" screen running on a 2560x1600 resolution has a DPI value of 101 with the result that text using the same font size has a physical height of 3.02mm, which is considerably smaller. Ideally, font sizes should be specified using physical units such as millimetres or inches to guarantee the same font size independently of the screen context. However, as mentioned in the previous section, absolute units in CSS often do not match their physical counterparts as the browsers usually do not work with the real DPI value of the screen. So, to counteract this discrepancy, the font in the large screen setting would need to be increased from 12 to 14 pixels. Many web sites offer a function to manually change the font size, but this requires user intervention. Other sites rely on the browser's default font size and specify relative values using percentage units. Still, this approach is based on the assumption that the default size is appropriately configured for the user's screen context, which may not always be the case since many users are not even aware of this browser setting. A potentially better solution that we investigate in this work, is to use the new media features of CSS3 to adapt the font size relative to the increase in screen width.

As shown in our previous study [17], the majority of existing web sites use static layouts very often with fixed column widths. This is partly understandable if one considers the alternative of fluid or liquid layouts, which while allowing content to flow freely to fill the available screen space also

[5]http://www.w3.org/TR/2009/CR-css3-multicol-20091217/

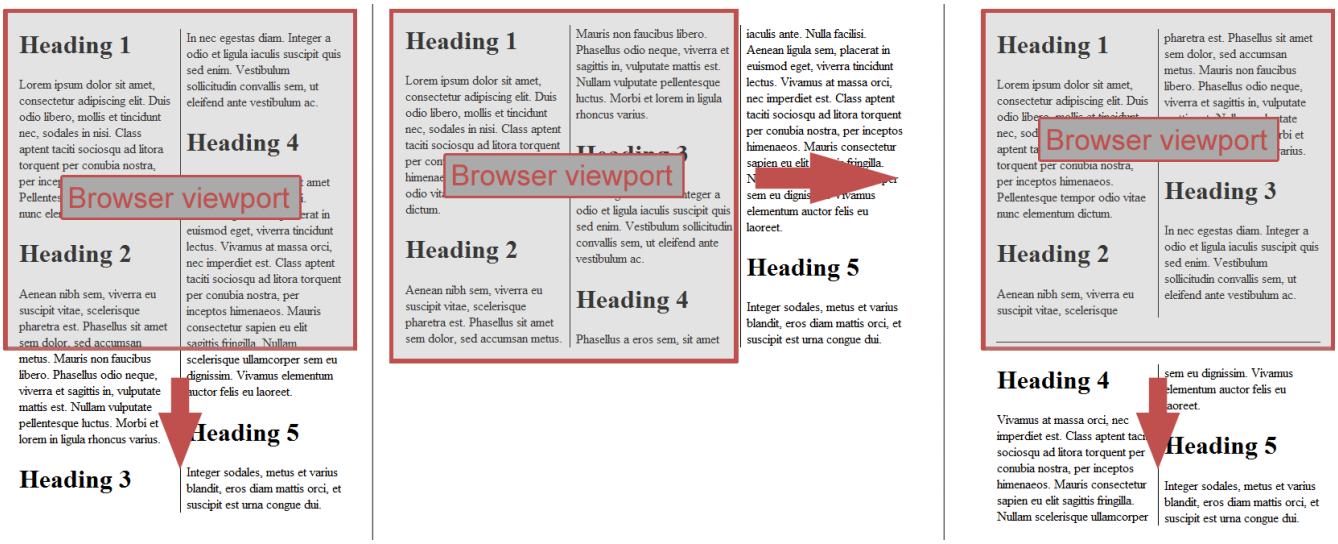

Figure 1: Comparison of different scroll models in combination with multi-column layout: vertical (left), horizontal (center) and paged, vertical scroll model (right).

lead to excessively long lines of text on large widescreens. We have therefore explored different ways of tackling this problem using the new Multi-column Layout module, since multiple columns divide the screen into smaller portions that can be filled with paragraphs of appropriate width. However, there are two important remaining problems due to the current limitations of the CSS3 module. First, the number of columns and their widths need to be constrained properly so that the problem of long text lines does not occur. Second, while multiple columns work fine for content that fits the browser viewport, problems arise when the height of the columns extends beyond the fold, i.e. exceeds the height of the window, which is typically the case for long, text-intensive content such as news articles or blog entries.

With respect to the first problem, suggested line lengths average around 70 characters [16]; however, the CSS3 specification does not foresee a way of specifying the column width in terms of characters per line. We have explored a possible way that uses approximated em values for specifying the column width. The idea behind this approach is that the width of a character is related to the font height, which em refers to. Having measured the width of an example text (which is basically a lower-case alphabet but taking into account the frequency of letters) in different font sizes and from this calculated the average width of a single character to compare the average character width with the given font size, we found that the ratio between font size and character width is often close to 2 (Table 1). Therefore, by setting the column width to 35em, the line length will average around 70 characters per line. We note that this ratio is relatively stable for many font types, e.g. Arial or Times New Roman, but, for example, a monospace font like Courier New has a slightly lower ratio of approximately 1.67, which may require additional balancing.

Table 1: Relation of the font size to the average width of a character for the Arial font.

Font size	Avg. character width	Font size/width ratio
10px	5.01	2.00
12px	5.89	2.04
14px	6.72	2.08
16px	7.35	2.18
18px	8.48	2.12
20px	9.22	2.17
30px	14.08	2.13

The second problem that needs to be tackled is when content laid out in multiple columns does not fit the screen so that users have to scroll down and up again to go from the end of one column to the beginning of the next one. This problem can be attributed to the fact that web documents are treated as continuous media and typically grow in the vertical direction (Figure 1, left). Some recent studies [2, 9] suggest adopting a horizontal scroll model for multiple-column designs where the height of the web site is constrained to the height of the browser window and additional content is instead added in the horizontal direction, i.e. usually from left to right (Figure 1, center). Scrolling is then only required for the horizontal axis of the viewport; however, as their study also indicates, we believe that this horizontal scrolling model is very uncommon and unfamiliar to most web users as they have become used to the dominant vertical layout used in the vast majority of web sites. This is further compounded by the fact that mouse wheels, often used for scrolling, only exist in vertical configurations, so a complete change to horizontal scrolling would be rather radical and require many users to change their browsing habits [20].

As an alternative, we have explored an approach based on a paged, vertical scrolling model (Figure 1, right) where content is laid out in fixed-sized virtual pages that fit the viewport so that multiple columns can be used in combination with vertical scrolling. We have decided to position these pages one on top of the other within the same document rather than use a dynamic scroll-less paging model, where content is laid out in a single frame and replaced upon activating associated links for the next or previous page. This is to avoid additional navigation levels and to maintain a continuous reading experience by favouring scrolling over paging. Still, the main principles of our layout adaptation

guidelines could also be applied to a dynamic paging model since this would only change how pages are embedded in the document, but not how they are constructed.

Finally, larger viewing sizes may not only require adjusting the font settings and text layout, but also scaling embedded media accordingly. However, media resources such as images and videos are often only available in one size and typically embedded using fixed dimensions [17]. As a result, they often do not adapt well to widescreen viewing contexts. Particularly in the case of ads, which make extensive use of animated GIFs or Flash animations to draw the user's attention, a failure to scale well with the page's remaining content may be to the disadvantage of the advertiser. Modern browsers provide basic support for media scaling which is used when a media resource is embedded in a web page at a smaller or larger size than the original one; however, this can lead to a loss of quality unless the content is available in a vector-based graphics format, e.g. SVG. For raster graphics, we have explored different approaches, again based on the new features of media queries, to replace pictures and videos with high-definition versions as soon as the automatic scaling factor exceeds particular threshold values.

5. ADAPTIVE LAYOUT TEMPLATE

The reported experiments with the aforementioned features and limitations of current web standards have driven us to consider the following features for adaptive web templates to accommodate a range of display settings:

- Automatic font scaling to maintain the physical size of text in different screen contexts.
- Controlled switching between single and multi-column layouts to produce text layouts and line lengths in proportion to the viewing conditions.
- Automatic pagination of content to support multi-column designs in combination with the vertical scrolling model.
- Automatic media scaling and substitution to adjust the size and detail of pictures and videos if available in different resolutions.

The benefit of working with templates for web designers and developers is that, like a framework or a programming library, they can provide general instructions and additional functions, in this case concerning web page layout, which can be tailored to particular application contexts. Technically, the template consists of a number of CSS3-based style enhancements and optimisations, predefined layout classes as well as JavaScript functions to leverage the more advanced features. Target web sites will need to include it in the HTML header definition as well as activate some of the optional functionality directly on selected content elements, as discussed below. The complete source is available for download from our website[6]. Before we move on to show how existing web sites can benefit from building on our template, we first discuss some of the implementation details and the role of the new HTML5 and CSS3 features for supporting the required web site adaptations.

[6]http://dev.globis.ethz.ch/adaptiveguardian

5.1 Adaptation of Text and Media

A primary feature of the template is to automatically scale the font size of text elements, i.e. headings, paragraph elements, etc., according to the screen width. The underlying method is based on CSS3 media queries and the assumption that a higher screen resolution usually also means a higher screen DPI value. The implemented countermeasure is therefore to increase the number of pixels used for the font in intervals at larger screen widths. The template automatically applies the new font size to the body of the document. This also sets the base value for all values specified in em (as explained earlier) and therefore automatically applies to headings and paragraph elements unless these define other sizes explicitly.

As for media, it is more difficult to provide fully automatic means using CSS3 media queries only. We have implemented a number of controlled techniques based on using HTML placeholder elements, e.g. for low and high quality, and then using CSS to toggle their visibility according to the screen context. However, this approach is not optimal since all versions of the element would still be loaded by the client browser, including high-definition versions of media resources which may be rather large. Another possible approach is to add the images as background images to a container element. Again, CSS and media queries could be used to switch between the different versions; however, this solution has the drawback that designers would need to mix content and presentation when they define the rules in CSS. We have therefore implemented a technique based on JavaScript that uses only one HTML placeholder element with default content and a customisable threshold to indicate when content of higher quality should be used instead. Our solution automatically checks for alternative versions of pictures and videos based on naming conventions, namely suffixes '-small' and '-large', and replaces the source attribute of the respective DOM element accordingly. The relevant script is called at page-loading time and each time the size of the browser window changes.

5.2 Multi-column Layout

Unlike many of the transformational techniques discussed earlier, our approach is not concerned with content analysis to automatically determine suitable web page elements that could benefit from multi-column layout. Rather, we provide a collection of layout classes that designers can choose from and apply in their web sites in a controlled way. In CSS3, multi-column layout can be specified using a combination of values for the column-width and column-count properties. If only the number of columns is specified, then these are distributed evenly and the column width is computed from the viewport width; if the width is specified, then additional columns will be inserted as long as there is enough space available and the maximum column count (if provided) is not exceeded. Our template defines several multi-column layout classes based on popular grid layouts. For example, we provide 2col-layout and 3col-layout classes that, based on the rules discussed earlier, use an approximate column width of 30em and a maximum of two and three columns, respectively. The number of columns actually used thus varies depending on the size of the browser viewport. This means that if the available width is less than 60em (plus the column gap), the content will be laid out in a single column, otherwise the browser will add, depending on the layout class,

one or two additional columns of the same width so long as they can fit. The resulting line lengths for elements of this class therefore range roughly between 60 and 120 characters per column. Some of the classes provided by our template also make use of media queries to provide optimisations with respect to the screen size. This can be especially helpful for main content and side bar elements for which we give two examples in the next section. Finally, our template also provides a special paged-layout class that can be combined with other multi-column classes provided in order to invoke the pagination of content that is described below.

5.3 Pagination of Content

For multi-column layouts to be effective with the vertical scrolling model, we implemented a pagination algorithm that splits content into multiple smaller chunks that each fit into the viewport. We refer to these viewport-sized chunks as pages. The process of splitting content into multiple such pages goes beyond the capabilities of CSS3, where the default behaviour is to grow vertically until a maximum height is reached and then add new columns in the horizontal direction. In particular, there is currently no feature allowing a new line of columns to be added in the vertical direction and overflowing content to be automatically placed in the next row. We have therefore implemented a solution using client-side scripting to segment content into pages, while making sure that the height of each page is smaller than the height of the browser viewport and that new pages are appended in the vertical direction. Special measures can be taken for the first page that often starts at a certain offset below the web site's header elements such as the navigation bar, menus, top banners etc. By setting the height of the first page to the remaining viewport height, we can make sure that all pages have the appropriate dimensions. Additionally, developers can specify a minimum page height, which can be useful to prevent an oversegmentation of content at smaller viewing sizes. The underlying algorithm takes a DOM element's container associated with multi-column layout and performs the following steps in order to paginate its content.

1. If the height of the container is smaller than the viewport height or the container only has one column, nothing is done.

2. The width of one column is determined using the CSS3 column-width property, unless set explicitly. This is done by temporarily appending an element of 100% width to the container and then measuring its width in pixels.

3. Copies of the content elements of the original container are added to an invisible, temporary container whose width is set to the column width determined in the preceding step. The use of clones of the content elements is necessary in order to avoid browser repaints after moving each element to the temporary container.

4. The height of each copied element is measured. As the width of the temporary container is equal to the width of a single column, the height can be determined accurately. In particular, the problem of content elements spanning multiple columns and therefore having an incorrect height does not arise since we do not manipulate the original multi-column container.

5. New pages are created. Each page is filled up with the copied content elements as long as there is enough space available, i.e. the accumulated height of the elements added so far is still smaller than the maximum page height.

6. Only for the first page: the maximum height of the current page is reduced by the offset of the original container element. If the resulting maximum height is smaller than the minimum height, the maximum height is set to the minimum height.

7. The original container is replaced with its new paged version.

After executing these steps, the original container including all of its content elements are still available as objects in the DOM tree. This is required so that the above algorithm can run multiple times, which is necessary to be able to rebuild the pages from the original content when the user changes the size of the browser window. Our script also appends additional container elements between subsequent pages that primarily function as links to scroll to the next page, but can also be styled using CSS, e.g. to draw a line between pages. Such visual enhancements may be appreciated by users to guide their reading flow.

5.4 Page-by-page Scrolling

To take the proposed paged layout even further, our template provides an optional feature that allows users to scroll directly from one page to another with a single action. This page-by-page scrolling mechanism was inspired by a similar function available in common PDF readers and various other document viewing software. Using this kind of scrolling model, the vertical offset of the browser viewport moves directly to the beginning of the next or the previous page created from the pagination algorithm described before. To provide visual feedback to users, scrolling is done with a smooth animation rather than a sudden jump. We implemented three methods to activate page-by-page scrolling in our templates: when clicking a "next page" link inserted between pages, when pressing the Page Up or Page Down key and by using the mouse wheel.

6. APPLICATION

To evaluate the proposed features, we tested our template on a number of existing web pages, in particular The Guardian's news web site[7], which we used for our user study, as we felt it was fairly representative of text-centric web sites with statically laid out columns for navigation and content [17]. Figure 3 (top) shows a screenshot of the web site's original design viewed on a 30" screen at a resolution of 2560x1600 pixels. The content comprises the typical web page elements, such as the header containing the main navigation bar and a slot for advertisements at the top, followed by the main content and the footer at the very bottom of the page. The main part is presented in three columns. The leftmost column is the largest and contains the news article content which typically consists of a header, a picture and the article text itself. A smaller column in the middle of the layout provides various functions to the user, e.g. to

[7]http://www.guardian.co.uk/

manually adjust the font size, as well as links to related information. The far right column contains a combination of advertisements, related pages and various services.

As is evident from the screenshot, the web site uses a fixed layout which does not adapt well to the example viewing situation. The main problem is that the overall design is contained in a fixed-width wrapper of 940 pixels which is placed in the middle of the browser viewport, leaving a considerable amount of unused space on both sides when viewed on screens with higher resolutions. Also critical is the default font size of the web site, which is set to 14 pixels. While a function to manually change the font size is available, due to the static layout the column widths do not adjust if the font size is increased, which leads to gradually shorter line lengths. This problem is also observed in many other online news web sites that provide such a function.

The first step in providing a more flexible design was therefore to remove the fixed-size constraints of the web page's layout. We used our template in combination with a fluid layout shown in the middle of Figure 3, where the width of the wrapper was set to 95%so that the document fills most of the available viewport width while still leaving some space at both sides where peripheral vision would set in. As can be seen, the font size automatically adapts to the new viewing context thanks to our template and in order to keep the line lengths proportional to the font size, the width of the three main columns was changed to em values, which also allows those columns to grow if the font size is manually increased. To ensure that the page does not get too narrow, a minimum width was additionally used. We enabled also CSS3 multi-column layout for both the main content area and the sidebar in the third column using features of our template, but we restricted the column numbers to two at this stage in order not to produce a too fragmented web site appearance. To support multi-column layout in combination with a vertical flow of the page, pagination was enabled for the main article container and pages were visually separated by a horizontal line as well as a "Read on" link at the bottom of each page to to scroll directly to the next page.

The second step then focused on adapting the media. As can be seen best when comparing with the bottom of Figure 3, the adaptations concerned both the article image, which now spans the full width of the very first column, as well as the advertisements in order to make sure that the proportions of text and media, and in particular the visibility of ads, are preserved given the relative increase of the spatial coverage of content.

7. EVALUATION

So far, the paper has focused on which adaptations are actually possible based on the new web standards and how they can be complemented technically with the help of a template. Our implementation is based on a lightweight and efficient scripting solution that is carried out only on the client and hence the fluidity of the web page rendering in the web browser is not impacted. Considering performance was not an issue (it might however become one for large web pages viewed on slow machines), we focused our efforts on assessing the effectiveness of our layouts in terms of reading comfort and efficiency for the viewer as well as the perception of the overall look-and-feel of the adaptive web site in comparison with the original version. We therefore conducted a user study with the goal of finding out which of the adaptations contributed to improving the users' overall reading experience on the web.

7.1 Method

Twelve people were asked to read three different news articles taken from The Guardian's web site using different layouts and comment on various aspects of the design. Most participants were at the age of 20 to 29 (two at the age of 30 to 39) with normal or corrected vision and proficient readers of English. Eight of the participants reported reading online news 5-10 hours per week, three declared reading more and one less. When asked which media and devices they regularly used for reading news, ten participants reported using printed newspapers as well as desktop screens and five participants said they also used smartphones.

The three articles that were chosen for the study were *Wimbledon 2010: Ruthless Andy Murray gives Sam Querrey no quarter*[8] (905 words), *Chilean miners: A typical day in the life of a subterranean miner*[9] (1835 words) and *High-speed rail link gets £800m more in state funding*[10] (939 words).

The articles were presented on a 30" desktop screen with a 2560x1600 resolution using three different layouts—one without adaptations and two with different levels of adaptations. The order of the articles and the layouts was rotated for each participant in order to ensure that all articles and layouts were used the same number of times in the whole study. The participants were told that they could freely resize the browser window (initially set at an intermediate size of about 1300x1100 pixels) and change the font size to suit their needs. After reading, the participants were asked simple comprehension questions (e.g. "In which round did Andy Murray win against Sam Querrey?" or "Who will be his next opponent?") for which they were allowed to refer back to the article, if needed, to find the answers. Furthermore, participants were asked to fill in a post-task questionnaire after each article, in which they had to rate reading comfort, positioning of the elements on the web page, image alignment, font size and scrolling behaviour on a five-point Likert scale. At the end of the sessions, participants were asked to compare the three articles in terms of the layout.

The three layouts used for the study were the following:

- **Original.** The original layout of the article with standard scrolling behaviour. Text is presented in a single column at a default font size of 14 pixels (Fig. 3, top).
- **Adaptive 1.** This layout included most of the adaptations described above, i.e. multiple columns and pages, page-by-page scrolling using keyboard or "read on" links and automatic font size adaptation (which evaluated to 16 pixels in this case). Adaptations not available in this layout were page-by-page scrolling using the mouse wheel and the alignment and replacement of images (Fig. 3, middle).
- **Adaptive 2.** Adaptive 1 layout with the addition of page-by-page mouse-wheel scrolling and automatic image scaling and substitution (Fig. 3, bottom).

[8]http://www.guardian.co.uk/sport/2010/jun/28/wimbledon-2010-andy-murray

[9]http://www.guardian.co.uk/world/2010/sep/09/chilean-miners-typical-day

[10]http://www.guardian.co.uk/uk/2010/oct/03/high-speed-rail-network-transport

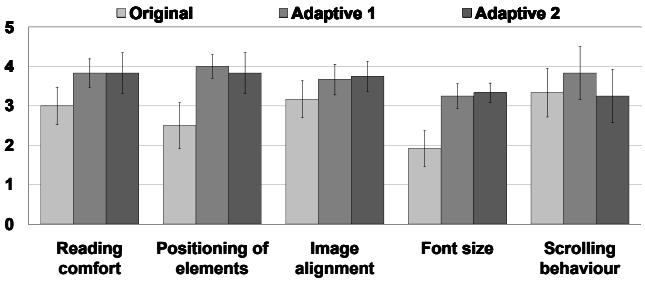

Figure 2: User ratings of various aspects of the layouts on a five-point Likert scale.

The reason for creating two versions of the adaptive layout was that we wanted to test various degrees of adaptations. As we did not want the sessions to last too long, we decided to test only three versions of each article. In all three layouts, a function to change the font size was available at the top navigation as well as in the sidebar. This function had the same behaviour as on the original Guardian web site.

7.2 Results

Since we selected articles of moderate length to keep the study participants interested while attending to the task, we expected to see greater differences in terms of perception and experience than actual reading speed. However, while users took roughly the same amount of time to finish reading the articles regardless of the layout, they were in general faster answering the comprehension questions for articles with adapted layouts, because they could find the relevant passages more easily, thanks to larger fonts and a clearer organisation of the text. Overall, reading comfort was rated higher for the adapted layouts than for the original ones, although a one-way ANOVA statistical analysis reveals that only the difference between the first adapted layout and the original is significant (P=0.031 vs. P=0.193 for the second adapted layout). Indeed, more than half of the participants (seven people) preferred the Adaptive 2 layout, four favoured the Adaptive 1 layout and only one participant liked the original layout best.

Examining the results in more detail, we observe that the greatest rating differences between the original and the adapted layouts are those concerning font-size and positioning of the elements (statistical significance for those factors was confirmed by the ANOVA and pairwise comparisons performed on the data). Participants appreciated the fact that more content was visible on the screen, despite the larger fonts used for the text. For the font size, text on the original layout was perceived as too small by almost all participants (ten), while only two participants perceived it as optimal, or too large. As a consequence, more than half of the users increased the font size when reading with the original layout. With the adaptive layouts, the initial font size was changed less often. Only one person decreased the font size to 14 pixels and two persons increased it to 18 pixels when reading with the Adaptive 2 layout. The initial font size of the adaptive layouts was rated as optimal by half of the participants. Two persons perceived it as too large when reading with the Adaptive 1 layout and four persons stated the same when reading with the Adaptive 2 layout.

Multi-column and multi-page breaking was also rated highly by a majority of testers, who felt that it increased the visibility and clarity of the page. However, there were also a few users who preferred the single-column layout because text could be read in a continuous fashion. The fact that some people were confused about whether to jump to the next column on the same page or continue on the same column but on the next page below, shows that it is important to make an even stronger visual separation between the pages.

Regarding scrolling behaviour, no statistical significance was found in the rating results. When asked about their preferences, three out of four participants chose the standard scrolling behaviour and the rest favoured page-by-page scrolling with the mouse wheel. The other scrolling mechanisms were rarely used or, as in the case of the "Read on" link, never. Two participants did not like the page-by-page scrolling and used the arrow keys for scrolling instead. Another participant used the arrow keys throughout all layouts. Only one person used the adapted Page Up/Page Down keys for scrolling. While users who preferred the page-by-page scrolling appreciated the speed of this scrolling mechanism, those who did not remarked that the jump was too big and sudden and that they lost track of the context after moving to another page. One participant said that scrolling should give feedback if it was successful (e.g. by showing part of the previous page), another person suggested to scroll only by half a page in order to maintain reading context. We tried to counteract this problem when designing the page-by-page scrolling, by smoothing the scrolling with an animation on the one hand and by not making the pages take up the whole viewport height on the other hand. However, the result indicates that further measures are necessary. Another factor for preferring the standard scrolling behaviour is attributable to the reading behaviour of some participants. We observed that these people preferred to adopt a fixed gaze when reading and use gradual scrolling so that the text to be read remained roughly around a fixed area on the screen. The same behaviour was incidentally observed by Braganza et al. [2]. Page-by-page scrolling is inappropriate for this reading behaviour. The same problem applied to multiple columns, where the participants had to relocate their gaze to the right in order to read the second column. However, participants stated that moving their gaze in the horizontal direction is less tedious than moving it in the vertical direction. Another interesting aspect that was not investigated in our study performed in a mouse and keyboard-based desktop context is if and how the reading and scrolling behaviours change when content is viewed on a touch-based device, such as a tablet or an E-book reader. Many document reading applications on those devices are based on page-by-page scrolling models (particularly on E-book readers) and thus our web page-splitting algorithm with tappable "read-on" button would perhaps prove useful for the viewing of web content in such settings.

Concerning image alignment and scaling, the statistical difference between the means is also not significant to draw clear-cut conclusions, but it seems that the adaptive layouts have a slight edge. More importantly perhaps, is the fact that there was no perceived difference between the Adaptive 1 and Adaptive 2 layouts, where for the latter the image was larger and aligned with the column (see Fig. 3). A possible explanation for this result is the fact that images were not a central part of the study and did not contain important information that was necessary for the comprehension of the article. Nevertheless, we believe that having the arti-

cle image (or other illustrative figures such as charts or fact boxes) remaining in view while reading the article laid out in neighbouring columns is better than seeing it immediately disappear after scrolling down to get to the text.

8. CONCLUSION

In this paper we presented ways of providing more flexible web page layouts which is necessary given the proliferation of small and large-screen devices used to access web content and the fact that many existing web sites still fail to adapt to the viewing context. Unlike prior work in this area, we focused on web standards, especially new features of HTML5 and CSS3, in order to demonstrate how web developers and designers can produce adaptive layouts using technologies they are familiar with.

To accommodate also large-screen contexts, we proposed balanced multi-column solutions with adapted media elements and addressed challenges of readability through automatic adjustment of the font size and line lengths. We also explored pagination of long content, which is necessary if multi-column layouts are to be used in conjunction with the vertical scrolling model employed by the majority of web sites today.

The main outcome of this work is an adaptive layout template that can be used by web developers as well as designers and applied in existing web projects. By means of formative evaluation, we demonstrated that especially text-centric web sites, such as online newspapers, but also wikis, blogs and forums, can directly benefit from the features offered by our template and improve the overall user experience. In future work we plan to extend the proposed template by considering also other types of web sites, such as web mail interfaces, online calendars or task managers, and investigate how we could make more effective use of larger screen real estates in order to support working with web-based applications. At the same time, we hope that the proposed CSS3 standard and in particular the CSS3 multi-column module will continue to evolve and provide more features in the future so that solving issues related to the vertical scroll model and pagination, as demonstrated in this paper, can be solved without the need for client-side scripting.

9. REFERENCES

[1] G. Badros, J. Tirtowidjojo, K. Marriott, B. Meyer, W. Portnoy, and A. Borning. A Constraint Extension to Scalable Vector Graphics. In *Proc. WWW*, 2001.

[2] C. Braganza, K. Marriott, P. Moulder, M. Wybrow, and T. Dwyer. Scrolling Behaviour with Single- and Multi-column Layout. In *Proc. WWW*, 2009.

[3] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt. A Unifying Reference Framework for Multi- Target User Interfaces. *IWC*, 15, 2003.

[4] S. Ceri, F. Daniel, M. Matera, and F. M. Facca. Model-driven Development of Context-Aware Web Applications. *TOIT*, 7(1), 2007.

[5] Y. Chen, W. Ma, and H. Zhang. Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices. In *Proc. WWW*, 2003.

[6] M. Czerwinski, G. Smith, T. Regan, B. Meyers, G. Robertson, and G. Starkweather. Toward Characterizing the Productivity Benefits of Very Large Displays. In *Proc. INTERACT*, 2003.

[7] Z. Fiala, F. Frasincar, M. Hinz, G.-J. Houben, P. Barna, and K. Meißner. Engineering the Presentation Layer of Adaptable Web Information Systems. In *Proc. ICWE*, 2004.

[8] F. Frasincar, G.-J. Houben, and P. Barna. Hypermedia presentation generation in Hera. *IS*, 35(1), 2010.

[9] J. H. Goldberg, J. I. Helfman, and L. Martin. Information Distance and Orientation in Liquid Layout. In *Proc. CHI*, 2008.

[10] G. Hattori, K. Hoashi, K. Matsumoto, and F. Sugaya. Robust Web Page Segmentation for Mobile Terminal Using Content-Distances and Page Layout Information. In *Proc. WWW*, 2007.

[11] J. He, T. Gao, W. Hao, I.-L. Yen, and F. B. Bastani. A Flexible Content Adaptation System Using a Rule-Based Approach. *KDE*, 19(1), 2007.

[12] N. Hurst, W. Li, and K. Marriott. Review of Automatic Document Formatting. In *Proc. DocEng*, 2009.

[13] C. Jacobs, W. Li, E. Schrier, D. Bargeron, and D. Salesin. Adaptive Grid-Based Document Layout. *TOG*, 22(3), 2003.

[14] S. Kopf, B. Guthier, H. Lemelson, and W. Effelsberg. Adaptation of Web Pages and Images for Mobile Applications. In *Proc. IS&T/SPIE*, 2009.

[15] R. Mohan, J. R. Smith, and C.-S. Li. Adapting Multimedia Internet Content for Universal Access. *TMM*, 1(1), 1999.

[16] A. Nanavati and R. Bias. Optimal Line Length in Reading-A Literature Review. *Visible Language*, 39(2), 2005.

[17] M. Nebeling, F. Matulic, and M. C. Norrie. Metrics for the Evaluation of News Site Content Layout in Large-Screen Contexts. In *Proc. CHI*, 2011.

[18] G. Robertson, M. Czerwinski, P. Baudisch, B. Meyers, D. Robbins, G. Smith, and D. Tan. The Large-Display User Experience. *CGA*, 25(4), 2005.

[19] E. Schrier, M. Dontcheva, C. Jacobs, G. Wade, and D. Salesin. Adaptive Layout for Dynamically Aggregated Documents. In *Proc. IUI*, 2008.

[20] H. Weinreich, H. Obendorf, E. Herder, and M. Mayer. Not Quite the Average: An Empirical Study of Web Use. *TWEB*, 2(1), 2008.

Figure 3: Different layouts compared in the study: original layout (top), adaptive layout using a multi-column design and paged, vertical scroll model (middle) and additionally using adaptive media for the article image and advertisements (bottom). All the above screenshots were captured at a resolution of 2560x1600 pixels and scaled down with a common factor for this figure.