ADABYTE

The Fundamentals of RESPONSIVE WEB DESIGN

HTML5 AJAX CSS3 JS

WASIM HABIB

Table of Contents

Introduction

During the last few years the usage of smart devices has increased remarkably, throughout that period the demand of web content for such devices has been escalating equally. The enhanced hardware capabilities accompanied by improved web access, and wider web standard support, has amplified internet accessibility. Furthermore, specifically tailored web versions suitable for these smart devices have been shown to increase user efficiency [1]. This encourages web developers to design and deploy web content suitable for such devices.

Nevertheless, the dissimilarities between these devices range from various programming languages, operating systems and platform standards. There are also several physical attributes. One that is certainly noticeable and is of particular interest for the web developers is the display size. The size of the display affects the content disposition and has an impact on the user utilizing such a device [2, 3]. Clearly large display tailored web content will not perform similarly on a smaller display independently. The persistent release of smart devices with ranging displays leaves specific display design content inadequate for multi-display usage.

Recently the World Wide Web Consortium (W3C) introduced a new recommendation allowing the detection of the display size. This in conjunction with other techniques has introduced a term called Responsive Web Design (RWD) which is a way to provide optimal user viewing experience, regardless of display size and device. The main goal is to minimize: scrolling, panning and resizing, to provide enhanced web content consumption and improved task processing [3]. The design emphasizes common device characteristics, but equally provides enhanced features for devices with greater capability. This principle is referred to as progressive enhancement [4], and is a substantial part of the design technique.

Problem Statement

The wide range of devices and platforms available makes it challenging for developers producing content targeting multiple entities. Each of these platforms has specific technical requirements and a set of physical attributes. The majority of these devices have a browser with the ability to render web content which makes the development of multi-platform applications evident [5]. However, the web content should be optimized for the different platforms to satisfy the user of such a device [1, 6]. The work aspires to investigate the technical possibilities and provide an overview of this subject.

This report aims primarily to answer the following questions:

- What choices are there today to show web content on smart devices?
- What techniques are used for responsive web design?
- What are the significant technical parts to keep in mind when developing with responsive web design?

Delimitations

This work aims to highlight the methodological segments of responsive web design and does not extensively cover the human-computer interaction aspects. The design was recently made public and is under extensive development; and thus as with other design principles, may be subject to change as the requirements of the petitioners alter, and new standards are introduced. The report entails current research and development achieved within and adjacent fields. The significant technical parts of the term refer to the foremost present elements utilized to enable the design. The objective is to emphasize a general overview of the practice, and thus neglect other minor and content specific technical properties.

Background

This section clarifies and defines the rudimentary prerequisites of the underlying work. The clarifications revolve around the related fundamental areas of the surrounding fields; but also provide to the essential technical scope of the forthcoming implementation.

Smart Device

A smart device can refer to a range of devices such as smartphones and tablet computers, and come with numerous sizes and shapes. See figure 1. The characteristics of such devices are mainly: an individualized and easy-to-use device with the capability of obtaining information through a wireless or wired network [7]. These devices are used at a constantly increasing degree in several areas, some of which are government, education and financial institutions [8]. There are currently five big platform providers each competing for positioning and market shares [9]. Each one of these providers has a distribution system and platform development recommendation which challenges the developers targeting a wide range of these platforms. There are currently no widely recognized physical design standards. However, in recent years touch based smart devices have been increasing. This type of interaction method is the preferred way to associate with documents on the web. Seemingly, web content optimized for these devices does increase user productivity [1].

Figure 1: Smart devices in various shapes and sizes.

Applications

Application or just “app” in the context of smart devices refers to a piece of software executed by the operating system on the device. These applications manage a wide range of tasks and contribute in a major way to the devices’ usefulness [10]. The interchangeable applications similarly act as the sole component for the devices’ interaction and personal customization. Ever since the introduction of the smart device, the number of applications has been growing, and is expected to continue to grow for the foreseeable future [11]. Each user may install and utilize any available application by choice which contributes to the device’s subjective definition of usability.

The applications are written at an ever-increasing rate by various kinds of developers ranging from hobbyists to experienced professionals [11, 12]. Each application can be categorized into three separate types: native, web or hybrid [5, 13]. There are numerous differences between these three types, including but not limited to the technical aspects of the given types, such as programming languages, application programming interface (API) and platform standards.

Native

A native application, which is mainly referred to when using the term “app”, is an application created specifically for a device [14]. This type of application takes advantage of the devices hardware capabilities, some of which are the camera and Global Positioning System (GPS). It follows the platform providers’ guidelines regarding design conventions, and is usually downloaded from the application marketplace managed by the platform provider [15, 16]. Compared to the web application type, the native application has certain advantages: better performance, visibility through the platform provider’s application marketplace, and extensive access to device hardware and platform features [5, 13].

Web

The web applications exist in two dissimilar forms [13]. The most frequently occurring is the mobile website, where the application logic resides on the server, or on the client side as JavaScript code. These websites are accessed by the users through the devices’ web browser. The websites are preferably adjusted to function efficiently with cellular network speeds, accommodate for portable navigation control, and fit consistently on mobile screens [15].

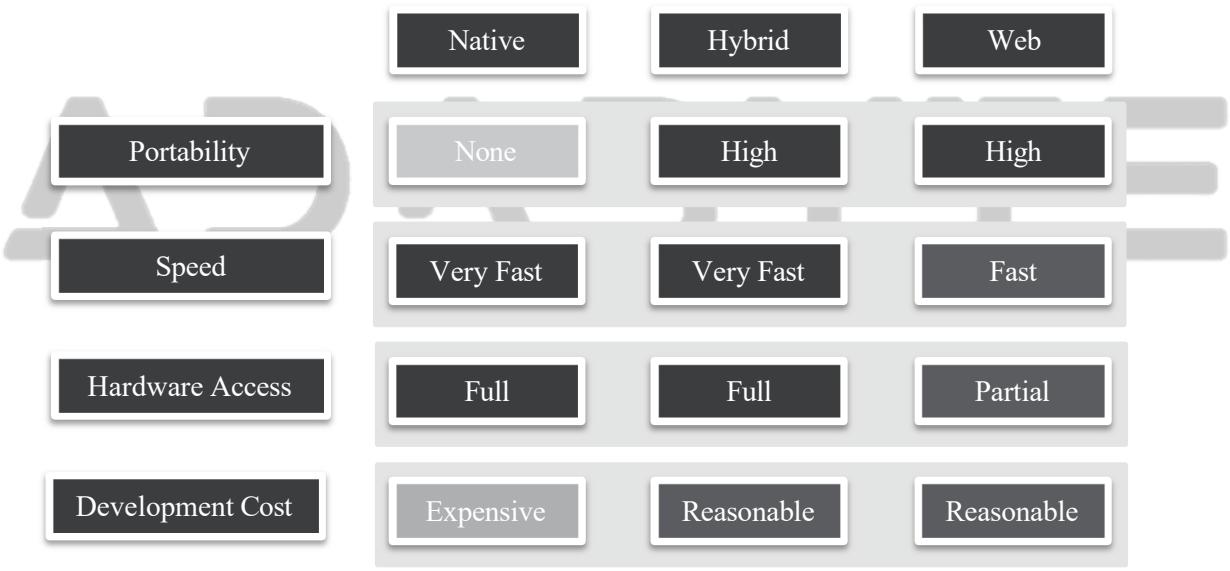
The other form is the offline web application or mobile widget. This form is an offline version of the website. The user downloads the application which exists in a browser stripped down of its components with the exception of its viewport. The benefits of using a web type over the native are: the unsophisticated multi-platform development process, and most importantly, the wide platform and audience accessibility [5].

Web technologies have been used to produce web content during a prolonged period, in contrast to smart device development process. There have been many improvements to the web process during the period, but the principle remains the same. This encourages developers within the field to rapidly produce content for these devices.

Hybrid

Hybrid application is a fusion between the native and web application by adapting certain parts from each preceding type. The aggregation consists of the native application's wide hardware access in combination with the web type uniform content across multiple devices and platforms [5]. Although the capabilities of both types are accessible, the practical implementation remains optional. Subsequently, device hardware access is linked with platform specific code. Certain tools exist to help with the developing processes of such application [17]. There are advantages to deploying hybrid applications. However, the benefits are associated with the requirements of the application. A generic comparison between all three types can be seen in table 1.

Table 1: Depicting generic comparison between the specified application types.



HTML5

Introduced in 1992, the Hyper-Text Markup language, or HTML, defines the basic structure of the World Wide Web. HTML consists of an assembly of instructions called 'tags'. These tags are responsible for how information within a document is organized, and how the data should be styled and formatted [18]. The W3C develops and maintains several set of standards, including but not limited to the Hyper-Text Markup language. The newest revision proposed by the W3C is HTML5, which contains numerous new features and capabilities [19]. When developing HTML5 the W3C had certain goals: the ability for web applications to perform like native applications, separating content from

presentation by changing the syntax, using open standard multimedia plug-ins instead of propriety, and including location-based services [20].

While HTML5 has not yet been finalized, certain progress has been made regarding its specification and language accessibility. Some of the language properties are: native support for video and audio playback, file management, local storage, geolocation and syntax clarifications [20]. HTML5 similarly presented additional semantically expressive tags. Web documents currently share many common properties, such as header and navigation sections. The lack of concise tags in the previous version of HTML forced developers to use less content describing tags, some of which are *span* and *div* [21]. These tags where and are in some cases still used to define many sections of the web content. Countless new tags has been presented with the new recommendation, some of which are the *section* tag to designate a block of the document, *nav* for navigation block and *aside* tag for side elements. Since not all current active browsers provide the new semantic elements, one should try to avoid these if backward compatibility is preferred. Yet, the support for the language traits can be expected to elevate, as the browser vendors adopt the web standards.

In the following years to come the language is expected to grow and cover a wider range of features. The new capabilities of HTML5 blur the line between native and web applications, allowing developers to achieve native functionality with web technologies. One such notable feature is the socket API made possible through the WebSocket interface. This web technology will enable bidirectional TCP/IP sockets between the application and its originating server [22]. However, there are currently some issues to be addressed regarding certain language attributes [23], but the W3C organization actively addresses new concerns and attempts to find solutions or alternatives regarding the standard.

CSS3

CSS designates Cascading Style Sheets, which is a set of formatting properties referred to as ‘rules’, associated with specific tags in a HTML document [18]. Each rule starts with a single or multiple tags designated as ‘selectors’. The selector specifies the section of the document to associate the formatting with. W3C manages CSS standards with the initial version providing approximately 50 properties to format an HTML document. This was later superseded by CSS2 in 1998, with roughly 120 properties.

CSS3 is the latest proposed standard recommendation candidate. Unlike its predecessors, the new candidate consists of numerous modules which can be developed independently [22]. CSS3 entails an extensive range of features, of which many modern browsers support several. Some of these features include image effects, multimedia control and changeable text appearance. This functionality enables developers to produce rich web content inside the renderer, and avoids dependency on external

software or services. The modularity of the new proposed standard permits browser vendors to implement the components incrementally. But it also enables them to focus primarily on the furthermost requested features. And similarly to focus on the furthermost vital modules, in order to maximize browser functionality.

Media Queries

As stated previously, CSS3 consists of several modules, whereas one of these components is the media queries module. But unlike other parts of the specification, the media queries unit is a separate W3C standard recommendation. By adding expressions, developers can check for certain conditions with the module, these include screen width, height, orientation and resolution [13, 17]. This allows the web content creators to set different document styles depending on if the conditions hold.

The CSS media queries are sensitive to client-side events, such as resizing the browser viewport. Nevertheless, this does not require event handling through a script language, but is instead managed by the browser [21]. Consequently, media queries can be used to adjust the presentation and layout of the document automatically. For instance, one can choose to modify the layout when the viewport of the browser exceeds a specific size, or when the orientation of the device is altered. Similarly, this enables the option to set a certain layout for a specific set of devices, depending on the size or other attributes.
Although the module is extensively supported by a wide range of browsers, some obsolete but still in use clients do not recognise the feature. Preferably, content aimed for these clients should be complemented with other techniques, such as client-side script languages.

JavaScript

JavaScript is a client-side, object-based scripting programming language, and is frequently used in conjunction with HTML. The language remains a vital technology in providing rich and dynamic web content [24]. The JavaScript language can interact with the Document Object Model (DOM) through numerous API libraries based on user-triggered events [22]. This allows the developer to attach certain event-based logic to the underlying document including but not limited to: hiding and showing content, responding to a user interaction and dynamically adding elements.

Although the JavaScript language is cross-platform and works across several entities, there are certain limitations imposed by the extent of interpreting engines. Due to the diversity of engine vendors, the code interpreted and executed by the user's client may result in diverse performances and security implications. There might also be particular features absent or added to each engine by the vendors, which need to be address in order for the code to work correctly [24]. However, the European Computer Manufacturers Association (ECMA) International standardizes ECMAScript, in which

JavaScript is formalized and developed. This sets a goal for the browser vendors, in which code compatibility can be achieved.

Ajax

Ajax stands for Asynchronous JavaScript and XML, and is the currently dominant technique for implementing asynchronous services. The term Ajax covers a wide range of technologies that enables dynamic page updates in web applications [25]. Web application utilizing this technology uses it frequently to receive data from the server and present it to the client, without refreshing the web document. This functionality leads the web in the direction of desktop application, and is seen as the foundation in Web 2.0 development [26].

In order to create a web standard for the technique, W3C created a specification 2006, regarding XMLHttpRequest. This is an API, which is a part of Ajax, and is extensively used when producing responsive and dynamic web applications. It is stated [25] that the popularity of Ajax has made JavaScript the dominant language for web browser, but similar to JavaScript, Ajax suffers from browser incompatibilities. Although in recent years, the support for the technique has been improved.

SVG

Scalable Vector Graphics (SVG) is an XML-Based language to describe two-dimensional vector graphics [22]. Benefits of using SVG over raster graphics, is that the former allows magnification without the loss of quality. This is especially valuable when managing 2D images and shapes on transformable content. With the widespread usage of smart devices, SVG graphics can be used for images and icons with size dissimilarity but without the decrease in resolution. This method is faster at shape rendering and resizing, compared to bitmaps. SVG has been a W3C recommendation standard since 2001, but was recently made available for web content through HTML5 [13]. Since then, SVG based content has been utilized at an increasing rate.

Responsive Web Design

A relatively new web technology within web development is RWD, or Responsive Web Design which at a rudimentary level accommodates the dimensions of the devices on which the web content is being accessed in a fluid manner [4]. Primarily to create device-independent web content, classifying common characteristics and neglecting devices specific elements such as physical attributes, browsers and operating systems. As previously stated, there are two forms of web applications; offline web application and the more frequently occurring mobile website, RWD is a technology mainly used for the latter. The distribution of such a system is enabled from a single source. According to what conditions are specified, each device requesting the web information views an altered and appropriate design for the petitioners attributes, see figure 2. Some of the attributes that generally are used to trigger the mechanism are: display dimension limits and device orientation.

There are several advantages of using this technique, whereas one is making the content available on multiple devices from a single system. If the audience is located across multiple devices, specifically adjusted device content might get obsolete when a new device is made available with dissimilar attributes. Another benefit is user satisfaction, it has been shown [3] that screen size impacts user behaviours. As well as different levels depth and breadth of information structures effects user fulfilment. This can straightforwardly be solved from a central source and made available across current and future entities.

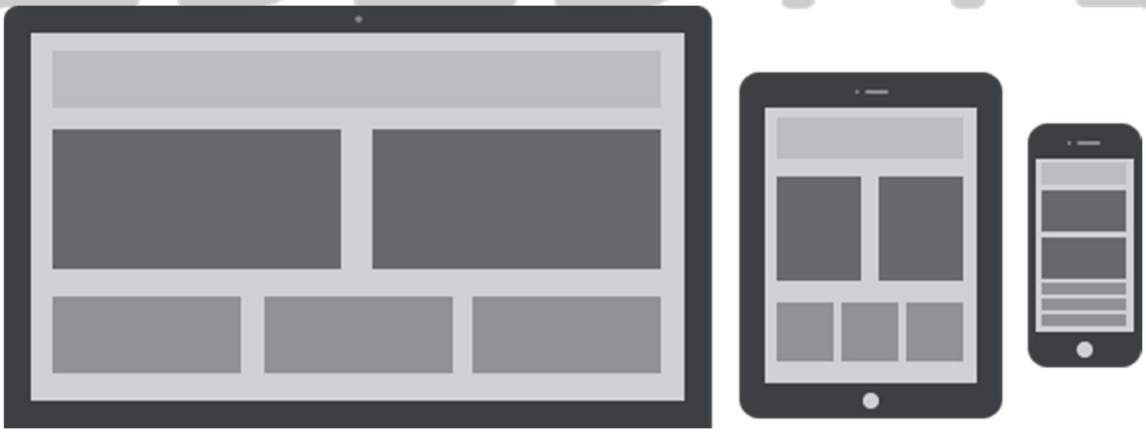


Figure 2: Responsive web design content transformation on different devices.

Compensation for the multiple platforms is generally attained through the elimination of all non-essential information, and instead focusing on the critical content [27]. The detached elements can despite the fact, be re-enabled later on devices with the capability to display them. Nevertheless, this might have an impact on the user utilizing the same content on different devices. The content on the minimalistic version might lack certain required features, and the user might get the impression of unfamiliarity with the

supplementary version. This encourages the content creators to emphasize the goal of the provided service, and supply relevant and familiar attributes in both versions.
Content layout and organisation is a significant part of web development and mobile web services. One of the foremost obstacles regarding mobile web interaction is considered to be bad interface design [28].

The wide adaptation of touch screen for smart devices has changed how users interact with the web. The number of websites optimized for such devices have been increasing and will continue to increase in the foreseeable future [15]. Many users also prefer touch screen as the foremost interaction method [1]. However, there are other techniques used, such as computer mouse, touchpad and stylus. In order to attain a wide audience with RWD, one should avoid associating critical functionality with a device dependent interaction method. One such example is the hover pseudo-class in CSS, which is used when a user designates an element, but does not activate it. This may work for a device with mouse pointer capability but might not under other conditions, and in some situations, may undermine the user efficiency.

Responsive web design can be achieved by using methods under HTML5 and CSS3 properties [4, 27, 29]. As previously stated, HTML5 presents countless new features, but also additional semantically expressive tags. For instance, new tags may introduce new and simpler ways for automatic adaption of web page layouts through document analysis [21]. One commonly practised web layout technique is the grid system, which is used to setup the document structure. By applying relative lengths units to each grid element, rather than absolute lengths, the grid element is subsequently resized by the browser subject to the viewport's dimensions. The grid adapting such a technique is referred to as "fluid grid", "flexible grid" or "adaptive grid", and is an essential part of RWD [4]. Contemporary web content frequently employ images to enhance the user interaction and enrich the interface. Each image is placed within a grid element, separate or in conjunction with relative content. However, in order to not exceed the parent size restrictions, certain properties are applied in order to enforce the parent's boundaries. When the browser imposes drastic size alterations, certain web elements may require modification in pursuance of reflecting the attributes of the device [17].
The aforementioned CSS3's media queries unit provides the necessary technique to achieve the detection of such an adaptation. The module does additionally provide straightforward methods to exercise the layout and style transformation.

The forthcoming subchapters give an expanded description, regarding the preceding abridged significant technical parts of responsive web design.

Grid System

Grid based design has been reused for a long period of time, particularly in published newspapers and magazines. The grid system is an ordering scheme for structuring a

page, composed of intersecting vertical and horizontal axes, or occasionally simplified to rows and columns. The system primarily aids in content placement such as images and text, and provides an overall element structuring. In recent years, this technique has been adapted for organizing web documents. The web structure can be fixed, similar to a printed page, but also adaptive or fluid, to fill available screen space [21]. With regards to RWD, the latter is more relevant.

The amount of columns and rows used for any given web document is specifically design and content dependent. The layout is a subjective practice, in which the same web document may be altered in direction of user satisfaction, or in attempt to achieve self-contentment. Nevertheless, when producing or utilizing an already existing adaptive grid, the column quantity is restricted and subjected to the browsers view port [21]. This is mainly due to the aggregation of columns, which will force certain elements to fall outside of the browsers viewport, but equally that scrolling on utmost sites occur vertically. Generally this increases the amount of rows applied to an information-rich web document in order to compensate for the column constraint. The content of the page might then increase vertically, and at the same time remain relatively fixed on the horizontal axis. Hence, the width predication of such a grid system is apparent, due it is primarily dependent on the applied information.

Fixed Grid

In a fixed grid system, absolute length units are applied, resulting in the same width uptake, regardless of the viewport's dimensions. While the viewport can hold the major column, the background of the page will be exposed. Subsequently, when the same element does not fit within the viewport, a horizontal scrollbar is revealed, allowing the user to view the hidden content, see figure 3. The latter should be avoided in order to increase user productivity, and is especially important for smaller screens [3].
Nevertheless, information dense websites employ multiple rows in order to supply the necessary space for the content. When the browser is unable to fit all the rows on the same page, a vertical scrollbar is revealed. This allows the user to navigate within the page and expose the hidden content.

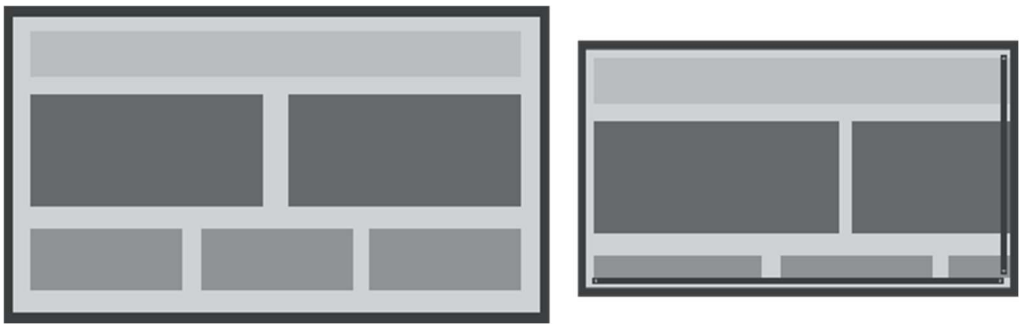


Figure 3: Fixed grid system size comparison, right window is resized vertically and horizontally to 80% of the original left window.

Adaptive Grid

Adaptive grid system is the primary choice when developing with RWD. It differentiates itself from the previous grid by using relative length units. Each column has the width specified in percentages, to mirror the viewport's breadth. When the window is reduced, each column reflects the change by decreasing the width uptake. This avoids revealing the horizontal scrollbar, until the actual content within the grid element exceeds the presented space.

A column occupying an entire row will require a width of 100 %, two columns will each have a width of 50 % three columns will have 33.3 % and so forth. The percentages are then translated into viewport pixels. For instance, if 100 % corresponds to 1180 pixels of the viewport, then 50 % is 580 pixels with a separation of 20 pixels, 33.3 % is 380 pixels with two separations of 20 pixels each. The separations are neither mandatory nor required, but do provide content distinction. When the width of the viewport is subsequently reduced, the column percentages are not changed but the equivalent pixel uptake is, see figure 4.

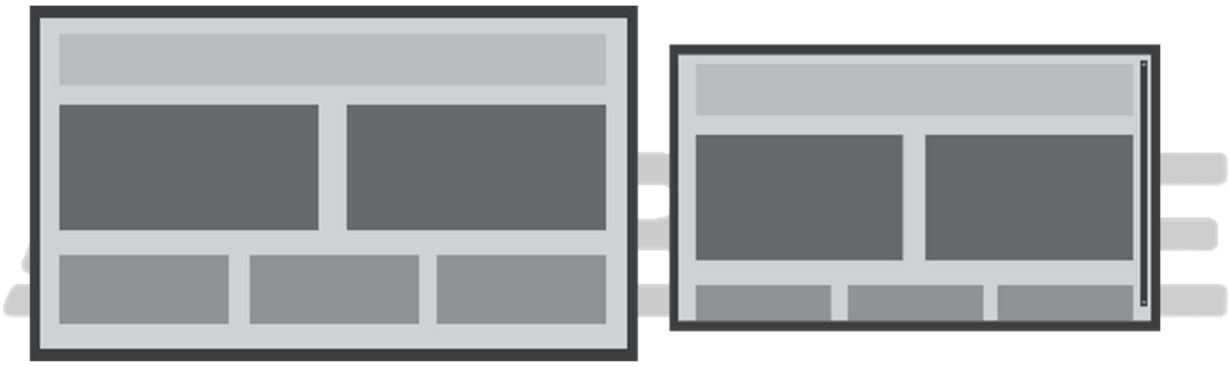


Figure 4: Adaptive grid system size comparison, showing the horizontal content alteration, right window is resized vertically and horizontally to 80% of the original left window.

As previously mentioned, the *div* tag in CSS is used for multiple purposes. By providing specific identifications, it becomes possible to specify the role for each *div* tag within the web document [22]. To achieve maximum compatibility with browsers not supporting HTML5 semantic elements, the *div* tag is commonly used when producing grid systems. In the event of adaptive grids, suitable names are given for each *div* tag to compensate for the lack of linguistic attributes.

The emphasis of vertical layout is extensive within web design, and has become the dominant layout mechanism. Furthermore, horizontal scrolling is both uncommon and unfamiliar for most web users [21]. The latter is in many conditions avoided, in order to satisfy the predominant browsing habits, but also increase user efficiency [2].

Images

The majority of web documents today contain one or multiple images. These graphical units are generally perceived as supplements, and help enrich the document. For instance, they might enhance the functionality by equipping images for navigation or site operations, in conjunction with text description. With the introduction of CSS3, image operations features have increased [24], which contributes to their adoption. This functionality helps manipulate the image directly in the browser with the usage of CSS3.

Starting with HTML5, the *canvas* tag enables developers a wide range of two-dimensional features, including but not limited to: graphics, text and animations, which makes the tag appropriate for game application. However, the canvas element should be avoided when other existing element are sufficient [17]. One of the reasons to evade the *canvas* tag is the lack of support in certain present browsers. If the desired outcome is to display an image, then the more adapted *img* tag should be applied instead.

Similar to the early web content, the use of images on the initial smart devices was infrequent. But as the hardware capabilities increased, so did the use of the graphical units. Today, many of the images that were only viable on desktop computers are now accessible through handheld devices. Nevertheless, large raster graphics is still a challenge for these devices, especially when the browser is forced to resize the image in order to be fitted inside the screen. In many situations, it is preferred to provide two different resolutions, one for smaller screens and another for the larger, where the proper version will be selected and used [5]. Usually it is favoured to utilize SVG graphic over bitmaps on smaller devices, which enables faster shape and image rendering capabilities [13]. This is especially true when the images are used in RWD; since the same image might be accessed with multiple devices, each requiring resolution alteration in order to fit within the devices screen. Ideal candidates for SVG are the supplementary images or icons provided in the web document, such as the ones applied for navigation and readability proposes. A large size reduction might render the icons indistinguishable, and likely hinder the functionality of websites depending on such enhancements.

When employing adaptive grids, it is fairly significant to make certain that the content is retained within the designated grid elements. Disproportional elements may likely overflow other content, or get entirely hidden by the renderer. This arises generally when the enclosed content requires more space than what is announced by the grid element. Large graphical entities, such as images, are usually affected by this phenomenon. Due the flexible nature of the adaptive grid, the grid elements may be in constant size change. In order to encounter this event, one may choose to permit the graphical objects to adjust their size accordingly. The aforementioned *img* tag provides the necessary functionality to embed graphical content [20]; but when mutually

employed with RWD, one avoids defining the image dimensions in the HTML code, thus to achieve maximum device compatibility. However, without the size declaration, the image requires directions to account for the parent element's size restrictions. By setting the CSS *max-width* attribute to 100 %, the image acquires an upper boundary in which not to exceed. This enables the renderer to adjust the image but simultaneously satisfy the constraints of the grid.

Content Transformation

With the previous two techniques, an adequate foundation has been laid for structuring and modelling adaptive web content. Yet, this might not be sufficient for extensive fluctuations. The physical variations between the target devices might be severe, and thus require addition adjustment in order to function according to the intended design. To detect the device specific differences, such as screen width, height and orientation, one utilizes the W3C recommended, CSS3 *media queries* module [13, 29]. The transformation occurs when the specified conditions defined within the style sheet are valid. One may then choose to change the layout entirely, or apply a more definite modification.

The adaptive grid system has an outstanding transformation method; by adjusting the grids' dimensions when the viewport's size alters, and increasing or decreasing the background space accordingly. However, once a drastic transformation occurs, the applied grid layout might not be appropriate. For instance, when a multi-column grid experiences a radical width reduction, the enclosed content might subsequently not receive the required space with the current layout. A layout modification might be required in order for the content to be perceived correctly, see figure 5.

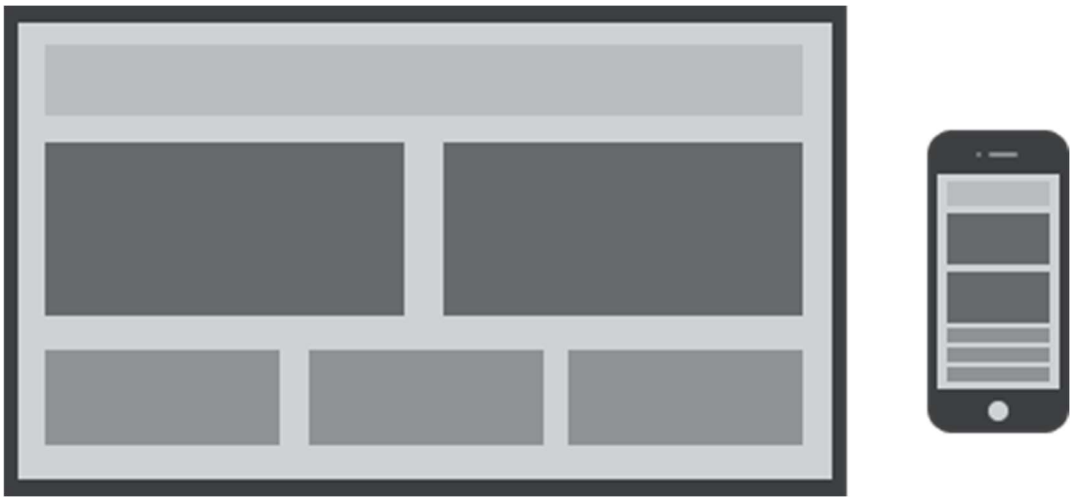


Figure 5: Depicting drastic size alteration and the corresponding response, by shifting the grid layout.

To detect such a change, one may instruct the media query to be valid for the screen type, together with a certain display *max-width* or *min-width*, and then apply the desired alterations.

The font size of web documents is a crucial element, an incorrect value might force the user to unnecessary scrolling, zooming and might render the text useless [30]. By using the media query module, one can change the size of the font depending on the physical attributes of the device. The media queries can similarly be employed for other design enhancing functionality. For instance, certain content might not be desirable for a specific device. The designer might then choose to replace certain elements with more appropriate ones, or remove them completely. Appropriate in this case might be to optimized icon for visibility or provide larger ones for touch based interaction methods.

When observing a webpage on a smart device, the browser usually employs an enlarged viewport by default. The assumption made by the browser is that the website is not optimized for smart devices. This enables the users to straightforwardly perceive the web document. Nevertheless, in order to instruct the browser the contrary, one utilizes the viewport *meta* tag, sets the *initial-scale* to 1.0 and disallows user scaling [13]. This will inform the browser of the desired proportions and adjust the viewport accordingly.

As previously stated, the CSS3 media queries module provides an excellent layout and design functionality, and is an essential and enormous part of responsive web design.

Implementation

The first part of this section embodies a general practical prototype, concerning the formerly mentioned responsive web design components to emphasize the essentials of the technique. The second part includes an analysis of a business platform utilizing the design, in conjunction with a brief description of commonly utilized responsive frameworks to provide a conceptual overview.

Components

The foundation of Responsive Web Design (RWD) consists of the adaptive grid system, and is the initial step of the design. Separated into copious rows, the columns of the grid are usually finite and fixed to even integers. Although absolute length units are avoided in RWD, the rows in such a system frequently have predefined widths. This enables the information to remain relatively centred and avoids stretched out content on large screens. The length of the width is highly subjective and adjusted after the document information, however commonly set to a frequent occurring screen resolution. The row similarly receives a max-width of 100 % to adjust accordingly. The column of an adaptive grid has no defined length units; instead the element can be floated and given a minimum space uptake to not get hidden when containing no child elements. There may be various quantities of columns; however it is important that the widest column has a maximum width of 100 %, and that the sum of the lesser columns have equally the span of the same length, see figure 6.

```
    /* 12 column system */
.one    { width: 8.33333%;  }
.two    { width: 16.66667%; }
.three  { width: 25%;       }
.four   { width: 33.33333%; }
.five   { width: 41.66667%; }
.six    { width: 50%;       }
.seven  { width: 58.33333%; }
.eight  { width: 66.66667%; }
.nine   { width: 75%;       }
.ten    { width: 83.33333%; }
.eleven { width: 91.66667%; }
.twelve { width: 100%;      }
```

Figure 6: A CSS 12 columns system definition within an adaptive grid.

Each child element within the grid system may be subject to style modifications such as paddings and borders. This type fashioning might cause the element to exceed the predefined width set by the parent, due to such styles are made available outside the element. By applying the CSS3 *box-sizing* tag, accompanied by the border-box

attribute, all equivalent styles are applied within the elements boundaries. This functionality is vital in order to attain an evenly structured grid system.

While the adaptive grid is highly flexible and adjusting, the grid might be observed on a small screen, such as a handheld device. In this case, the 12 grid system from previous might not be feasible. The information within the grid system endures major size contraction and the result is usually incomprehensible content. One can subdue this effect by changing the structure of the grid. For instance, it is possible to change the grid from 12 columns per row to single column-row arrangement. This can be realized by setting a breakpoint at specific screen size and altering the grid style. The CSS media query module is an excellent tool to detect such a change, and therefore simultaneously evade the usage of script languages.

Images are vital to the presentation of web documents and their utilization is shared by many websites. When employed in conjuncture with adaptive grids, one avoids setting absolute length units on the graphical entity. This is due to the flexibility of the grid; an image with fixed size might overflow other information on the page. Nevertheless, to force the image to respect the given space, one sets the maximum space boundary of the image. In CSS, this can be achieved with the *max-width* tag set to 100%. The graphical unit will then resize according to the available space.

When a drastic change occurs, as in previous screen size scenario, grid structure alteration might not be sufficient. In some situations the entire document style may require modifications, or certain elements need to be toggled. As in the grid system detection and adjustment, the media query module can be applied. In figure 7, one can recognize the breakpoints of 768 pixels and 767 pixels applied for the screen. When the viewport is larger than 768 pixels then mobile-only elements are hidden, and when the opposite occurs, desktop-only elements are concealed and mobile elements are displayed.

```
@media only screen and (min-width: 768px) {
  .mobile-only {
    display: none !important;
  }
}

@media only screen and (max-width: 767px) {
  .desktop-only {
    display: none !important;
  }
}
```

Figure 7: Screen specific breakpoints and corresponding style alteration.

As previously stated, when a browser renders a webpage, the default assumption made is that the content is not tailored for small screens, and thus a large viewport is provided

for the petitioner. By setting the viewport's initial-scale to 1, the browser zooms in the content and thus provides a satisfactory viewport scale. This functionality is produced by the *meta* tag, where other properties can be supplemented, such as minimum-scale and maximum-scale.

A comparison can be seen in Appendix A between different screen sizes in a responsive system. In Appendix B, the corresponding HTML, CSS and JavaScript code can be attained.

Frameworks

Developing and maintaining a responsive system can be challenging, especially if the RWD service is required to be general and reusable. The system additionally requires extensive testing in order to assure the desired functionality across multiple devices, this appends to the development complexity.

Nevertheless, there are alternatives to internal produced responsive systems, such as the ZURB Foundation and Bootstrap. These frameworks contain a wide set of tools for website and web application development. The components provided are highly practical, and range from navigation bars to alerts and progress bars, which make the reusability of such entities evident. The web technologies employed by the frameworks are HTML, CSS and JavaScript and the utilization of RWD for multi-device application. The vendors equally aspire to support many of the major browsers, which also contribute to the cross-platform nature of the frameworks. Many of these platforms are open source with licences enabling the usage of such units in both personal and commercial projects without charge. This, including the extensive component and device support makes them ideal candidates for production development. While the main goal of the frameworks is similar, the differences range from diverse component availability to the practical implementation of such a system.

Mediusflow

Mediusflow is a multi-tenant enterprise platform, divided into a variety of workflow applications each with various functionalities, such as invoice management and contract administration. The platform is highly flexible, and provides necessary API to enable the development of individualized and personalized applications to meet the businesses' needs.

Separated into two parts, the platform consists of a frontend and backend, while the communication between them occurs through web services. The latter fragment is the sole component for the user interaction. When this part of the framework was conceived, one of the main goals was to provide wide service availability, and furthermore avoiding the need to download or install any external software. Therefore,

web technologies are employed for the frontend unit such as CSS and Ajax, enabling the user access to the service on any device with web and browser capabilities.
However, in recent years, full-scale webpages has been made available to devices with various shape and sizes. This has encouraged the developers to adapt Mediusflow to these changes in order to satisfy its wide service accessibility goal. To secure current and future device-independence, responsive web design has been chosen as the main technique. Other portions of the technique has also made an impact, such as reducing the development time, and distributing the system from a single source with minimal changes to current information. To simultaneously receive reusable and tested components, the framework Bootstrap has been utilized.

By applying the Bootstrap framework, Mediusflow adopted a generalized responsive system. However, in some situations this might not be enough, some elements may need to be adjusted in order to fit within the guidelines of the host. This is true for Mediusflow, due to the data extensive nature of the platform. For instance, one major part of the frontend is the inbox HTML table in which to represent the application fields. This table does not have a finite column limit, but can grow according to the number of units defined by each application. The challenge with such a feature is to fit the information on a small screen without the loss of functionality. This type of obstacles may occur frequently for niched content employing general responsive system.

Figure 8 and 9 demonstrate the user perspective of the frontend unit, presenting responsive web design and content adaptation on different screen sizes.

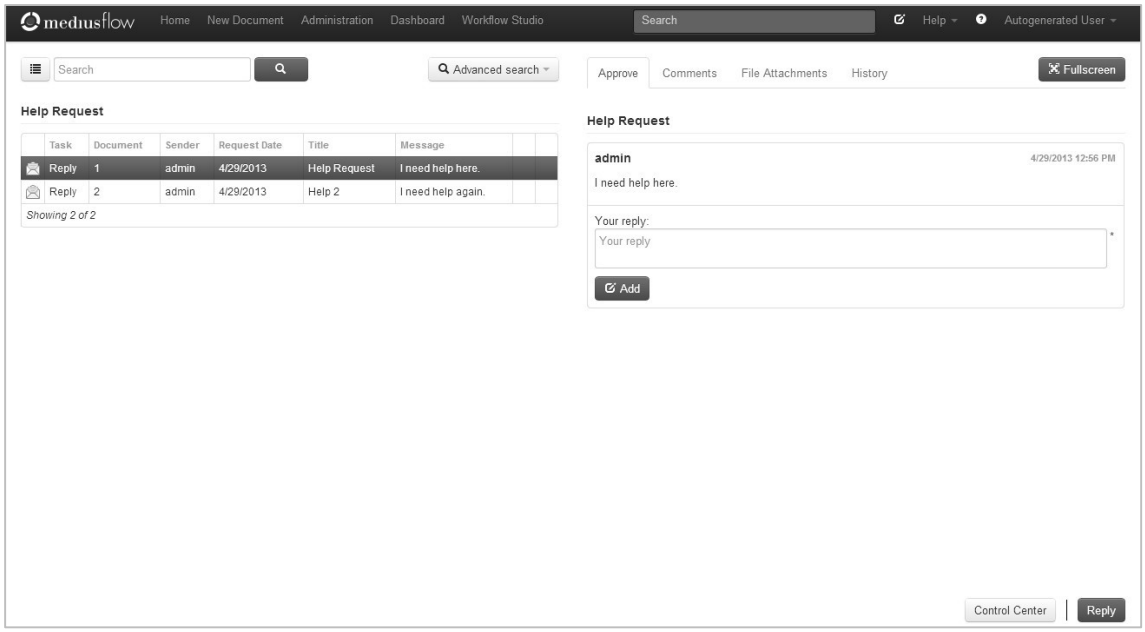


Figure 8: Mediusflow platform with help request application on a desktop sized screen.

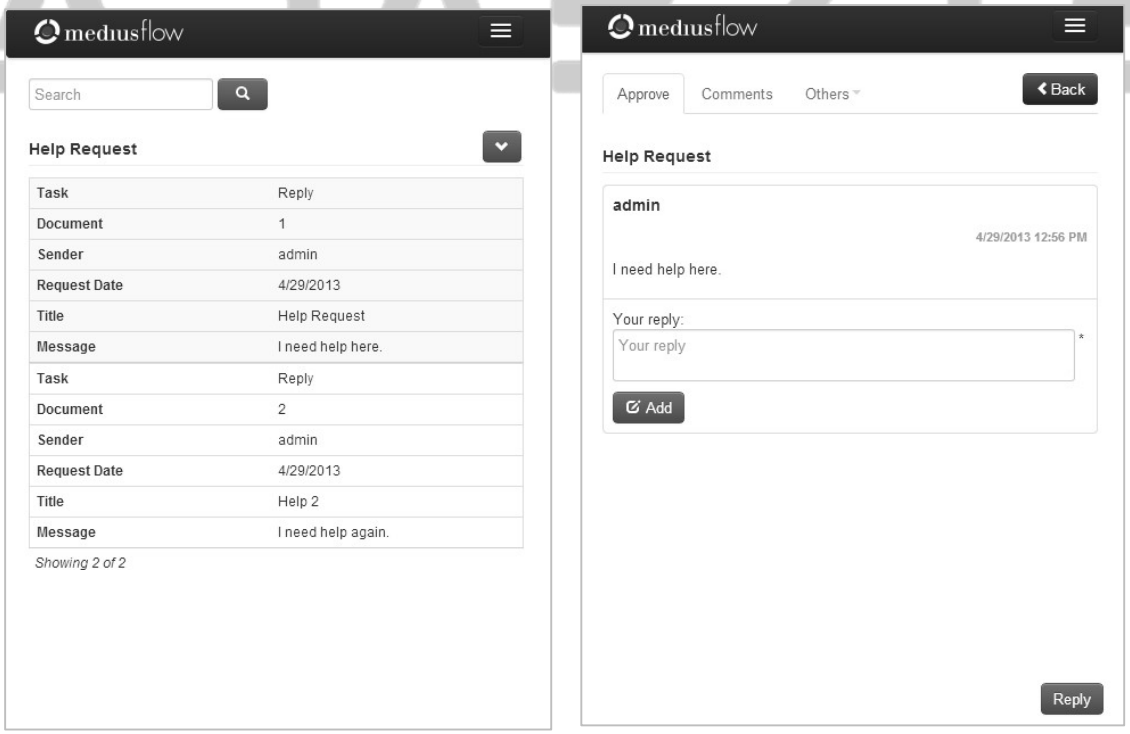


Figure 9: Mediusflow platform with help request application on a tablet or smartphone screen. The left side depicts the inbox view and the right side presents the managing view.

Due to the lack of space, the modification applied is foremost to the separation of the inbox and managing view on the smaller screens. As an alternative, when a task is selected in the inbox, the view is toggled between these two entities. This together with the collapsing and expanding of the application specific tasks provides the user a wide overview of the inbox. The inbox table layout alteration similarly contributes to the readability enhancement, and assists the information dens attributes to fit seemingly within the provided space. Another large aspect of the transformation is the progressive enhancement achieved within the platform by the application of responsive web design. By comparing the two previous figures, one can determine the elements supplied only in one of the two versions. For instance, the advanced search and control centre is only available on the version with the capability to display the entire functionality. Equally, certain elements may be included only for a specific device group.

Discussion

Although responsive web design can be employed to produce cross-platform content, in some scenarios it is not the best option. For instance, when graphical intensive operations are required, or when device specific features are vital to the functionality of the product. A more practical approach in this case would be to utilize platform specific functions to achieve the desired outcome. The aforementioned native or hybrid application is adequate as substitutes for this purpose. The media queries module can be used to toggle the view of an element depending on the requesting device. Nevertheless, this does not prevent the element from getting loaded by the browser; on information dense web pages this does increase the load time, especially on devices with less hardware capabilities. The web server can however detect this and thus avoid supplying the unused information. Another challenge of the media query application is the definition of the breakpoints applied to the web page. It is possible to define a set of specific breakpoints depending on the content provided; or according to most common screen resolutions, whereas the latter is subject to change due to the constant release of devices with numerous sizes and shapes.

One might similarly argue that the incorporation of diverse versions, into a single web page may undermine the support and development process. This is mainly due to the additional code required for the adjustment of the web content. While it is true, that supplementary code is mandatory, the reusability factor of the technique is relatively high compared to the substitutes. With responsive web design, the emphasis of common device attributes is endorsed. This is mainly due the multi-platform nature of the technique, and thus the implication is the avoidance of device specific properties, in order to target an extensive amount of users. However, this may undermine the user experience of such a product. Users accustomed to interact with the platform's native elements, may find themselves disoriented with the replacements. One way to encounter this is to produce elements that emulate native controls, or employ existing libraries to achieve the same effect.

An alternative to responsive web design could be to separate each version of the web page depending on the petitioners device. This can be achieved through web server techniques, but this technique has disadvantages. For instance, this requires the development and maintenance of separate versions, but equally the support of wide amount of devices. Nevertheless, in some cases, the separation of web versions might prove to be more feasible and applicable. Lastly, as with every technique, the application of responsive web design depends on the prearranged requirements and goals.

Future Work

The responsive web design technique is relatively new technique, and has recently acquired a large amount of attention. The practice has been shown to fill a certain void within the web development branch. Nevertheless, as with every new method, it may be prone to impending complications. Future applications and testing such as case studies may provide necessary scenarios to further demonstrate the technique's effectiveness or limitations. This will either establish the method as the de facto standard of web development, or demise in favour of other methods. Additional investigations may be required for the user centric implications, the performance factor and alternative practical approaches of the technique.

Conclusions

The recent amplification of smart devices has expanded the demand for content suitable for such entities. These devices are also expected to surpass the personal computer market within the foreseeable future, which makes tailored content development significant in order to appeal to its wide and growing user base. The utilities of the device are mostly packaged into applications or apps, which consists of three different types: native, hybrid, and web. There is however no globally common design and development standard, each device vendor may require a set of tools and programming languages. The web has in contrast been available for a long period of time, and is a key component on these devices. This, together with the recent web technologies introduced by standard organizations simplifies for developers aiming to target various platforms.

The technique of responsive web design provides an excellent and inexpensive alternative to cross-platform development and content distribution. This is especially true for developers already familiar with web technologies, or for already existing web pages to increase their accessibility. By providing content that alters depending on the conditions set to reflect the petitioner's device, the content can be distributed from a single source, but also be reused on different platforms. There are several frameworks where responsive web design is a major component. The employments of these libraries decrease the development process and simultaneously acquire proven and documented functionality. The approach makes the technique both commercially viable and effective.

The major techniques employed by responsive web design are the adaptive grid, adaptive images and media queries. The adaptive grid usually expands vertically while remaining relatively fixed horizontally. This is made possible by regulating the width of the grid. In order for the grid to alter according to the given space, relative length units are utilized. When graphical units occur in conjunction with adaptive grid, certain attributes must be applied to restrict these usually large entities. The attributes set an upper boundary to which the graphical unit is not to exceed. Finally, when a web document endures a dramatic size change, the layout might need to be altered in order to reflect this event. The detection and transformation can be achieved by the application of media queries. These three techniques make up the building blocks of responsive web design.

Bibliography

[1] G. Schmiedl, M. Seidl, and K. Temper, "Mobile phone web browsing - A study on usage and usability of the mobile web," Bonn, 2009.

[2] M. Jones, G. Marsden, N. Mohd-Nasir, K. Boone, and G. Buchanan, "Improving Web interaction on small displays," *Computer Networks,* vol. 31, pp. 1129-1137, 1999.

[3] M. Chae and J. Kim, "Do size and structure matter to mobile users? An empirical study of the effects of screen size, information structure, and task complexity on user activities with standard web phones," *Behaviour and Information Technology,* vol. 23, pp. 165-181, 2004.

[4] R. Fox, "Being responsive," *OCLC Systems and Services,* vol. 28, pp. 119-125, 2012.

[5] A. Holzinger, P. Treitler, and W. Slany, "Making Apps Useable on Multiple Different Mobile Platforms: On Interoperability for Business Application Development on Smartphones," in *Multidisciplinary Research and Practice for Information Systems. International Cross-Domain Conference and Workshop on Availability, Reliability and Security (CD-ARES 2012),* pp. 176-89, 2012.

[6] C. C. Tossell, P. Kortum, A. Rahmati, C. Shepard, and L. Zhong, "Characterizing web use on smartphones," pp. 2769-2778, 2012.

[7] H. J. Lee, "A study on business opportunity for small smart devices in finance," *Mathematical and Computer Modelling,* 2012.

[8] S. Takahara and M. Yasaki, "Problems with using smart devices for business and efforts to resolve them," *Fujitsu Scientific and Technical Journal,* vol. 49, pp. 160-165, 2013.

[9] A. Holzer and J. Ondrus, "Mobile application market: A developer's perspective," *Telematics and Informatics,* vol. 28, pp. 22-31, 2011.

[10] J. Voas, J. B. Michael, and M. Van Genuchten, "The mobile software app takeover," *IEEE Software,* vol. 29, pp. 25-27, 2012.

[11] S. Tiarawut, "Mobile Technology: Opportunity for Entrepreneurship," *Wireless Personal Communications,* pp. 1-7, 2013.

[12] S. Agarwal, R. Mahajan, A. Zheng, and V. Bahl, "Diagnosing mobile applications in the wild," 2010.

[13] *Mobile Platforms and Development Environments*, Morgan & Claypool Publishers, 2012.

[14] E. Mullan, "A guide to mobilizing your content: Which app solution is right for you?," *EContent,* vol. 35, pp. 30-31, 2012.

[15] K. Buettner and A. M. Simmons, "Mobile web and native apps: How one team found a happy medium," in *1st International Conference on Design, User Experience and Usability: Theory, Methods, Tools and Practice, DUXU 2011, Held as Part of 14th*

International Conference on Human-Computer Interaction, HCI International 2011, July 9, 2011 - July 14, 2011, pp. 549-554, 2011.

[16] A. Charland and B. Leroux, "Mobile Application Development: Web vs. Native," *Communications of the Acm,* vol. 54, pp. 49-53, 2011.

[17] R. Gawley, J. Barr, and M. Barr, "Native to HTML5: A real-world mobile application case study," vol. 95 LNICST, pp. 188-206, 2012.

[18] J. K. Fugate and R. J. Vokurka, "Progress in separation of structure and style: HTML, XHTML, XML and cascading style sheets," *International Journal of Innovation and Learning,* vol. 2, pp. 425-433, 2005.

[19] G. Anthes, "HTML5 Leads a Web Revolution," *Communications of the Acm,* vol. 55, pp. 16-17, 2012.

[20] M. B. Hoy, "HTML5: A new standard for the web," *Medical Reference Services Quarterly,* vol. 30, pp. 50-55, 2011.

[21] M. Nebeling, F. Matulic, L. Streit, and M. C. Norrie, "Adaptive layout template for effective web content presentation in large-screen contexts", pp. 219-228, 2011.

[22] G. Jakus, M. Jekovec, S. Tomažič, and J. Sodnik, "New technologies for web development," *Elektrotehniski Vestnik/Electrotechnical Review,* vol. 77, pp. 273-280, 2010.

[23] J. A. Clark, "HTML5 Changing How You Use the Web," *Online,* vol. 34, pp. 12-14, 11 2010.

[24] E. Mitchell, "Standards, efficiency, and the evolution of web design," *Journal of Web Librarianship,* vol. 4, pp. 453-455, 2010.

[25] J. Kuuskeri and T. Mikkonen, "Partitioning web applications between the server and the client," pp. 647-652, 2009.

[26] E. A. Hernandez, "War of the mobile browsers," *IEEE Pervasive Computing,* vol. 8, pp. 82-85, 2009.

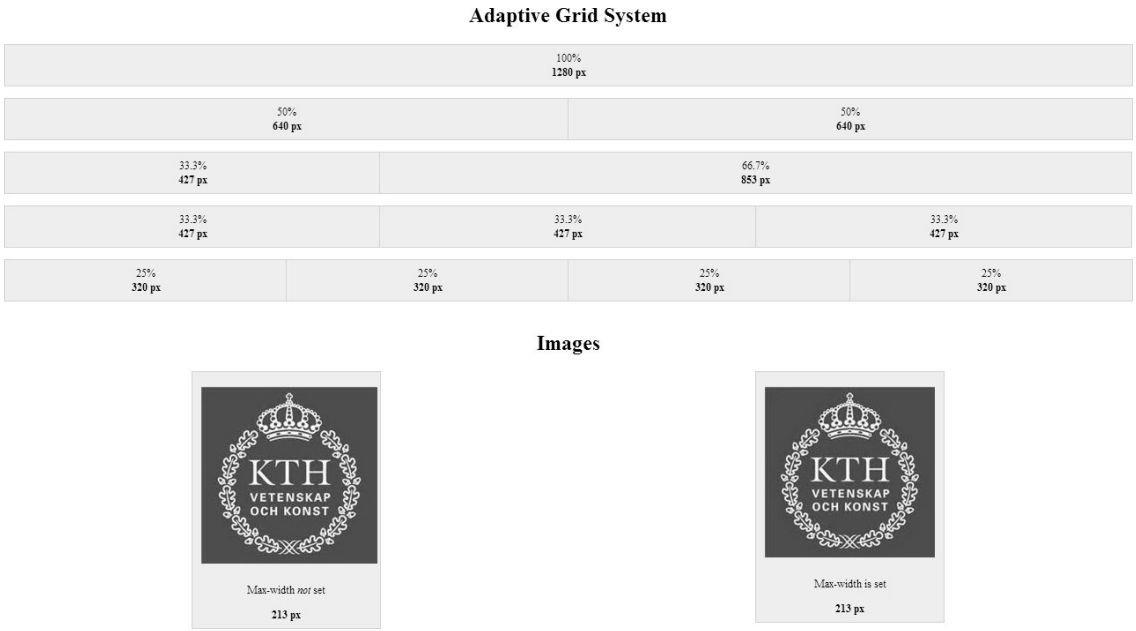
[27] J. Bengtson, "Scaling Into the Future With Smartlinks," *Journal of Hospital Librarianship,* vol. 12, pp. 378-383, 2012.

[28] Z. Qu and J. Sun, "Adaptive mobile web interface: User readiness in context," *International Journal of Mobile Communications,* vol. 10, pp. 132-149, 2012.

[29] S. Pastore, "Website development and web standards in the ubiquitous world: Where are we going?," *WSEAS Transactions on Computers,* vol. 11, pp. 309-318, 2012.

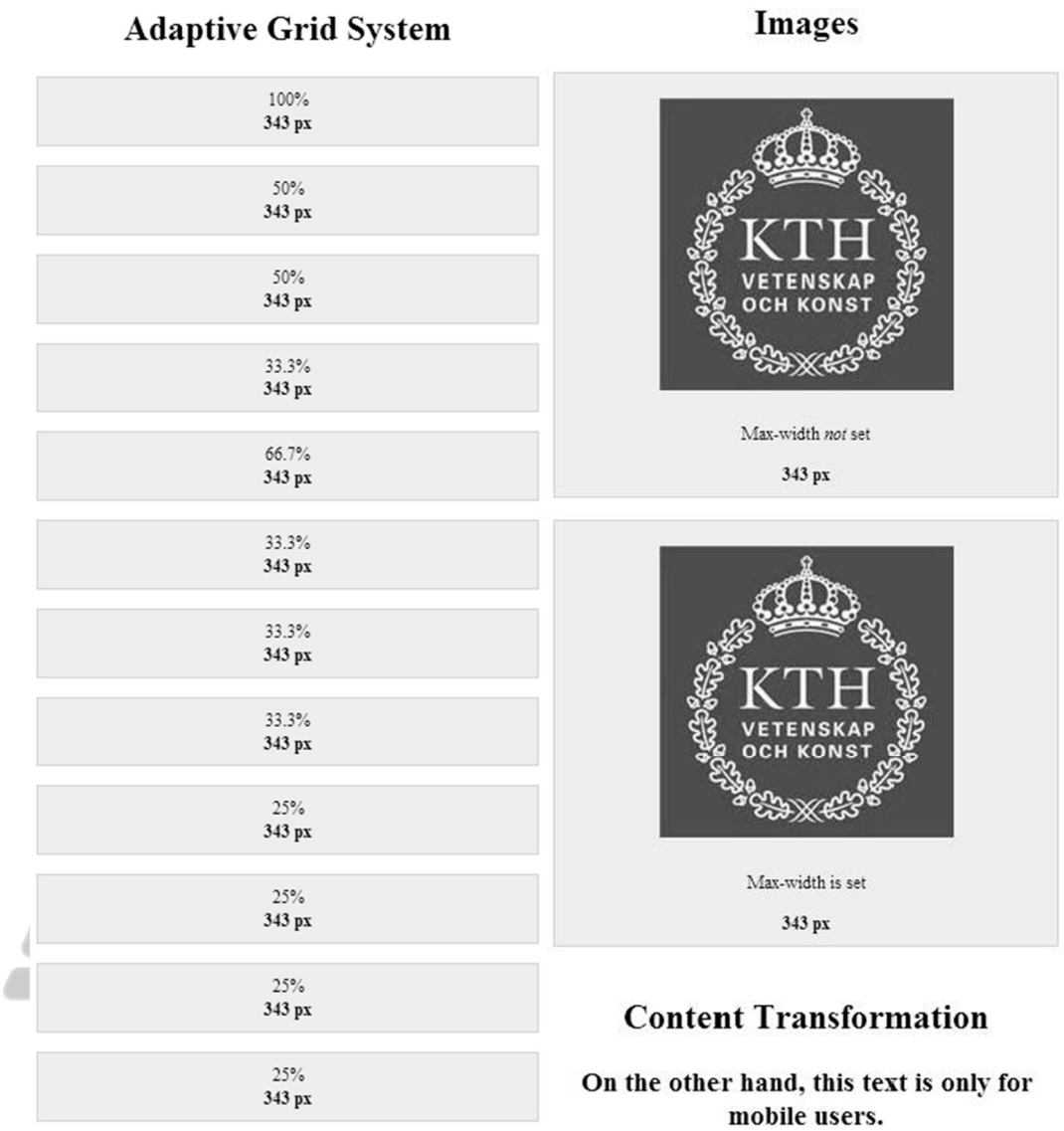
[30] B. Ryan, "Developing library websites optimized for mobile devices," *Reference Librarian,* vol. 52, pp. 128-135, 2011.

Appendix A



Responsive system perceived on a screen bigger than 1280 pixels in width.

Responsive system perceived on a screen smaller than 1280 pixels but bigger than 767 pixels.



Responsive system perceived on a screen smaller than or equal to 767 pixels.

Appendix B

```
/* Filename: grid.css */

/* Element box sizing */
* { -webkit-box-sizing: border-box; -moz-box-sizing: border-box; box-sizing: border-box; }

/* Standard row and column styles */
.row      { width: 1280px; max-width: 100%; margin: 0 auto;   }
.columns  { float: left; min-height: 1px; padding: 0 15px; position: relative;  }

/* Clear after row and columns */
.row:after, .columns:after { content: ""; display: block; height: 0; clear: both; }

/* 12 column system */
.one    { width: 8.33333%;  }
.two    { width: 16.66667%; }
.three  { width: 25%;       }
.four   { width: 33.33333%; }
.five   { width: 41.66667%; }
.six    { width: 50%;       }
.seven  { width: 58.33333%; }
.eight  { width: 66.66667%; }
.nine   { width: 75%;       }
.ten    { width: 83.33333%; }
.eleven { width: 91.66667%; }
.twelve { width: 100%;      }

/* Small screens */
@media only screen and (max-width: 767px) {

  /* Remove widths and let the browser decide */
  .row      { width: auto; min-width: 0; margin-left: 0; margin-right: 0; }
  .columns  { width: auto !important; float: none; }

}
```

```
<!-- Filename: index.htm -->

<html>
<head>
  <title>Responsive Web Design</title>
  <!-- Set the viewport scale -->
  <meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1, maximum-scale=1">
  <!-- Include the adaptive grid system -->
  <link rel="stylesheet" type="text/css" href="grid.css">
  <!-- Document Style -->
  <style>
    .sample-block {
      background: #eee;
      font-size: 12px;
      margin-bottom: 15px;
      padding: 7px 0;
      text-align: center;
      box-shadow: 0 0 0 1px #D4D4D4;
    }

    .pixel-width {
      display: block;
    }

    img {
      padding: 10px;
    }

    h2, h3 {
      text-align: center;
    }
  </style>
```

```
    <div class="row">
      <div class="four columns sample-block">33.3%<b class="pixel-width"></b></div>
      <div class="four columns sample-block">33.3%<b class="pixel-width"></b></div>
      <div class="four columns sample-block">33.3%<b class="pixel-width"></b></div>
    </div>
    <div class="row">
      <div class="three columns sample-block">25%<b class="pixel-width"></b></div>
      <div class="three columns sample-block">25%<b class="pixel-width"></b></div>
      <div class="three columns sample-block">25%<b class="pixel-width"></b></div>
      <div class="three columns sample-block">25%<b class="pixel-width"></b></div>
    </div>
    <h2>Images</h2>
    <div class="row">
      <div class="two columns"></div>
      <div class="two columns sample-block">
        <img src="kth.jpg">
        <p>Max-width <i>not</i> set</p><b class="pixel-width"></b>
      </div>
      <div class="two columns"></div>
      <div class="two columns"></div>
      <div class="two columns sample-block">
        <img src="kth.jpg" style="max-width:100%">
        <p>Max-width is set</p><b class="pixel-width"></b></div>
      <div class="two columns"></div>
    </div>
    <h2>Content Transformation</h2>
    <div class="row">
      <div class="twelve columns">
        <h3>
          <span class="desktop-only">This text is only shown for desktop users.</span>
          <span class="mobile-only">On the other hand, this text is only for mobile users.</span>
        <h3>
      </div>
    </div>
  </div>
</body>
  <!-- Content Transformation -->
  <style>
    @media only screen and (min-width: 768px) {
      .mobile-only {
        display: none !important;
      }
    }

    @media only screen and (max-width: 767px) {
      .desktop-only {
        display: none !important;
      }
    }
  </style>
  <!-- Document script -->
  <script>
    function updatePixelWidth() {
      var widthElements = document.getElementsByTagName('b');
      for(var i = 0; i < widthElements.length; i++) {
        widthElements[i].innerHTML = widthElements[i].offsetWidth + " px";
      }
    }
  </script>
</head>
<body onresize="updatePixelWidth()" onload="updatePixelWidth()">
  <div class="container">
    <h2>Adaptive Grid System</h2>
    <div class="row">
      <div class="twelve columns sample-block">100%<b class="pixel-width"></b></div>
    </div>
    <div class="row">
      <div class="six columns sample-block">50%<b class="pixel-width"></b></div>
      <div class="six columns sample-block">50%<b class="pixel-width"></b></div>
    </div>
    <div class="row">
      <div class="four columns sample-block">33.3%<b class="pixel-width"></b></div>
      <div class="eight columns sample-block">66.7%<b class="pixel-width"></b></div>
    </div>
```

ADABYTE