The International Journal Of Engineering And Science (IJES)
|| Volume || 3 || Issue || 7 || Pages || PP-49-57 || 2014 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Responsive Web Design inFluid Grid Concept Literature Survey

Abdulrehman A. Mohamed, Dr. Cheruiyot W.K, PhD, Dr. Richard Rimiru, PhD, & Collins Ondago

Jomo Kenyatta University of Agriculture and Technology, Mombasa Campus, Institute of Computer Science and Information Technology, P. O. Box94090–80107, Mombasa, Kenya,
Jomo Kenyatta University of Agriculture and Technology, Main Campus, Institute of Computer Science and Information Technology, P. O. Box 62000, Nairobi, Kenya,
Jomo Kenyatta University of Agriculture and Technology, Main Campus, Institute of Computer Science and Information Technology, P. O. Box 62000, Nairobi, Kenya
Jomo Kenyatta University of Agriculture and Technology, Mombasa Campus, Institute of Computer Science and Information Technology, P. O. Box94090–80107, Mombasa, Kenya,

---ABSTRACT---
Extending beyond the boundaries of science, art, and culture, Responsive Web Design (RWD) provides new paradigms and techniques to develop one single website which looks different for different screen sizes so that it is usable on every device. In this paper we survey some of the state-of-art technical aspects of Responsive Web Design. A number of other overviews on Responsive Web Design have been published. However, in this survey, an effort has been made to show the chronological growth in this field by presenting and in-depth survey of fluid grid concept.The review reveals a shift away from traditional web design towards Responsive Web Design and a need for an alternative enhanced approach to RWD in fluid grid implementation which has evolve to become an unavoidable good practice in web designing. After extensive exploration of the literature, we recommend an outcome of classification of three meaningful categories as; Frame-based Solution (FBS), Support-based Solution (SBS), and Algorithm-based Solutions (ABS) Approaches for Fluid Grid Concept implementation.

KEYWORDS: *Responsive Web Design (RWD), Fluid Grids, Frame-based Solution (FBS), Support-based Solution (SBS), and Algorithm-based Solution (ABS).*

Date of Submission: 19 July 2014 Date of Publication: 30 July 2014

I. INTRODUCTION

Since the first websites in the early 1990′s, designers have been experimenting with the way websites look. Early sites were entirely text-based, with minimal images and no real layout to speak of other than headings and paragraphs. However, the industry progressed, eventually bringing us table-based designs, Flash, then Cascading Style Sheets (CSS)-based designs and finally Responsive Web Design.

1.1 Table-based Designs

Table-based layouts gave web designers more options for creating websites. The original table markup in Hyper Text Markup Language (HTML) was meant for displaying tabular data, but designers quickly realized they could utilize it to give structure to their designs, and create more complicated, multi-column layouts than HTML was originally capable of. Table-based designs grew in complexity, incorporating sliced-up background images, often giving the illusion of a simpler structure than the actual table layout [6]

1.2 Flash-based Web Designs

Flash (originally known as Future Splash Animator, then Macromedia Flash, and currently as Adobe Flash) was developed in 1996. It started with very basic tools and a timeline, and progressed to have powerful tools to develop entire sites. Flash presented a ton of options beyond what was possible with HTML.

Around the same time as the introduction of Flash to the scene of web design (late 1990′s – early 2000′s), the popularization of Dynamic Hyper Text Markup Language (DHTML) techniques, which consisted of several web technologies such as the use of JavaScript and sometimes server-side scripting, for creating interactive/animated page elements. During this time, with the inception of Flash and the popularity of DHTML, the concept of interactive web pages that allow users to not only read static content, but also to interact with web content, began[6].

1.3 CSS-based Web Designs

CSS-based designs started gaining popularity after the dotcom boom in the early 2000′s. While CSS had been available long before then, there was limited support for it in major browsers and many designers were unfamiliar with it (and even intimidated by it).

CSS-based designs had many advantages over table-based or Flash designs. The first is that it separates design elements from content, which ultimately meant that there would be greater distinction from the visual aspect of a web layout and its content. CSS was also a best practice for laying out a web page, where table-based layouts were not. It also reduced markup clutter and made for cleaner and semantic web layouts. CSS also makes it easier to maintain sites, as the content and design elements are separated. One could change the entire look of a CSS-based site without ever having to touch the content.

The document sizes of CSS designs are generally smaller than table-based designs, which translated to an improvement in page response times. Although there would be an initial bandwidth hit when first downloading the style-sheets of a website one had never visited before, CSS was cached by the user's browser (by default) so that subsequent page views would be faster-loading[6].

1.4 Responsive Web Design

Responsive Web Design (RWD) was founded by Ethan Marcotte who is a developer and a web designer as well. He had a particular interest in architecture and wanted to apply architectural concepts in web design. Inspired by this way of thinking, he applied architectural principals to web design; that would adapt itself to the users various devices. As a result of which, the idea of Responsive Web Design was coined [16]

RWD is a web design approach aimed at crafting sites to provide an optimal viewing experience; easy reading and navigation with a minimum of resizing, panning, and scrolling; across a wide range of devices (from mobile phones to desktop computer monitors). A site designed with RWD adapts the layout to the viewing environment by using fluid, proportion-based grids, flexible images, and CSS media queries, an extension of the @media rule [7]. The following are the core concepts of RWD:

a. The fluid grid concept – which calls for page element sizing to be in relative units like percentages, rather than absolute units like pixels or points.
b. Flexible images – which calls also for sized in relative units, so as to prevent them from displaying outside their containing element.
c. Media queries – which allows the page to use different CSS style rules based on characteristics of the device the site is being displayed on, most commonly the width of the browser.
d. Server-side components – which in conjunction with client-side ones such as media queries can produce faster-loading sites for access over cellular networks and also deliver richer functionality/usability avoiding some of the pitfalls of device-side-only solutions.

Another name used to describe this set of techniques is Adaptive Web Design (AWD). According to [7], this name would match more since the website really adapts to the device, rather than responding continuously to changes in its environment.

II. THEORETICAL LITERATURE REVIEW

In recent times, more and more people surf through the internet using mobile devices compared to a desktop computer. Consequently, mobile devices and computer screens designers have been trying to provide users with quality web-browsing but this hasn't been able to afford adequately users' needs that are exposed to traditional website layouts. Therefore, there is a need to switch to Responsive Web design which is capable to reshaping itself depending on various screen sizes and resolutions from largest screen sizes to smallest on mobile devices.

The field of Web design and development is quickly getting to the point of being unable to keep up with the endless new resolutions and devices. For many websites, creating a website version for each resolution and new device would be impossible, or at least impractical. Should designers and developers just suffer the consequences of losing visitors from one device, for the benefit of gaining visitors from another? Or is there another option?

It is asserted by [16] that, RWD stems from the notion of responsive architectural design, whereby a room or space automatically adjusts to the number and flow of people within it. Therefore, by transplanting this concept onto Web design, and we have a similar yet whole new idea. As a result of this then, why should we create a custom Web design for each group of users; after all, architects don't design a building for each group size and type that passes through it? Like responsive architecture, Web design should automatically adjust. It shouldn't require countless custom-made solutions for each new category of users.
According to [20], responsive web design is the term given to the concept of designing and developing a website so that the layout changes depending on the device/viewport on which the website is being viewed. By device, this could be a mobile phone, tablet, laptop, desktop computer, or even a smart TV.

According to [9], responsive web design is an approach that suggests design and development should respond to the user's behavior and environment based on screen size, platform and orientation. The practice consists of a mix of flexible grids and layouts, images and an intelligent use of CSS media queries. As the user switches from their laptop to iPad, the website should automatically switch to accommodate for resolution, image size and scripting abilities. In other words, the website should have the technology to automatically *respond* to the user's preferences. This would eliminate the need for a different design and development phase for each new gadget on the market.

It is stated by [15], that; responsive design is not a single technology but a set of techniques that allow web pages to serve the needs of both mobile and desktop users. The core components are:

- CSS @media queries
- Fluid images and video
- JavaScript, often triggered by window match Media
- Server-side solutions
- Scalable Vector Graphics (SVG) to create resolution-free images

A responsive site may utilize one, some, or all of these technologies, depending on the intentions of its designers. Web page text is fluid by default: as the browser window narrows, text reflows to occupy the remaining space. Images are not naturally fluid: they remain the same size and orientation at all configurations of the viewport, and will be cropped if they become too large for their container. This creates a conundrum when displaying images in a mobile browser: because they remain at their native size, images may be cut off or displayed out-of-scale compared to the surrounding text content as the browser narrows.

But responsive web design is not only about adjustable screen resolutions and automatically resizable images, but rather about a whole new way of thinking about design. A website designed with RWD adapts the layout to the viewing environment by using fluid, proportion-based grids, flexible images, and CSS3 media queries, an extension of the @media rule. These properties make the core concepts of responsive web design technologies as elaborated below.

2.1 Technologies of RWD

Several alternative technologies to RWD had been in existence such as AWD and Tableless Web Design (TWD) but had their challenges and limitation. According to [10], tableless web design (or tableless web layout) is a web design philosophy eschewing the use of HTML tables for page layout control purposes. Instead of HTML tables, style-sheet languages such as CSS are used to arrange elements and text on a web page.
Cascading style sheets was introduced in 1996 by the W3C to improve web accessibility and to make HTML code purely semantic rather than presentational. Around the same time, in the late 1990s, as the dot-com boom led to a rapid growth in the 'new media' of web page creation and design, there began a trend of using HTML tables, and their rows, columns and cells, to control the layout of whole web pages. This was due to several reasons:

- the limitations at the time of CSS support in browsers;
- the new web designers' lack of familiarity with CSS;
- the lack of knowledge of, or concern for the reasons (including HTML semantics and web accessibility) to use CSS instead of what was perceived as an easier way to quickly achieve the intended layouts,
- and a new breed of WYSIWYG web design tools that encouraged this practice.

It is asserted by [3] that, an alternative to RWD is the method of Adaptive Web Delivery that is adopted by consumer brands worldwide. Although it is very similar to Responsive Web Design, with adaptive delivery the most significant difference is that the server hosting the website detects the devices making requests to it, and uses this information to deliver different batches of HTML and CSS code based on the characteristics of the device that have been detected. While the conceptualization of RWD since 2010 has resulted in development of new and enhances technologies in the field of web design.

Fluid Grid Concept - The fluid grid concept calls for page element sizing to be in relative units like percentages, rather than absolute units like pixels or points.So, the main idea of flexible grids is to create a layout where all elements are based on the calculated percentage width and so all elements in the layout are resizable in relation to one another [9].
Fluid Images Concept - Flexible images are also sized, in relative units, so as to prevent them from displaying outside their containing element. It is a concept that allows developers to adapt images or other media to load differently depending on the device, either by scaling or by using the CSS overflow property [8].
Media Queries Technique - Media queries allow the page to use different CSS style rules based on characteristics of the device the site is being displayed on, most commonly the width of the browser. With

media queries, designers can build multiple layouts using single HTML documents and selectively provide style-sheets based on different features such as browser size, orientation, resolution or color [11].

Responsive Typography Technology - It is the use of fonts which adapts to different resolutions so they are still viewable, with the overall layout still intact. Unlike using a simple fonts for a separate mobile site, you are using a fonts which are complex as you like, hence stretch or shrink according to the screen's need [19].

RESS Technology (Responsive Web Design + Server Side Components) – The RESS in conjunction with client-side ones such as media queries can produce faster-loading sites for access over cellular networks and also deliver richer functionality/usability avoiding some of the pitfalls of device-side-only solutions [22].

Responsive E-Mail System - Responsive design is a situation where website, email or blog adjusts to the size of the window viewing it. So the email will look slightly differently if it's on a desktop, a tablet or a mobile phone [2].

2.2 Fluid Grid Concept

Margins, page widths and indentation are all aspects of page design which can aid readability. The web presents difficulties for the designer with each of these. Browser windows can be resized, thereby changing the page size. Different web devices (such as web TV, high resolution monitors, PDAs) have different minimum and maximum window sizes. As with fixed font sizes, fixed page layout can lead to accessibility problems on the web [1]. Therefore, to understand Fluid Grid Concept, then an in-depth critic is needed

The first basic difference between the fixed-width type of layout and liquid layouts is the measurements of their size. The fixed-width layouts are measured in pixels, but for liquid or fluid layouts, dimensions are defined in percentages, and as you might expect, this affords greater malleability and fluidity. In other words, by setting a percentage, you won't have to think about device size or screen width, and consequently, you can find a reasonable solution for each case because your design's size will adapt to the size of the device used. Liquid layouts are closely linked to media queries and special styles for optimization. Percentage-based widths alone will likely not be enough to accommodate your design for a large variety of display sizes [21].

A flexible grid-based layout is one of the cornerstones of responsive design. The term "grid" is used rather freely and doesn't imply a requirement to implement any of the available grid frameworks. What it means here is using CSS for positioning and for laying out margins and spacing, and for implementing various web layout types in a new way. Layouts and text sizes are typically expressed in pixels. But a pixel can be one dot on one device and eight dots on another. So how do designers approach responsive web design if everything is pixel-based? The answer is: to stop using pixel-based layouts and start using percentages or the em for sizing.

By basing text sizes, widths and margins on percentages or on the em, a unit of measurement based on a font's point size, one can turn a fixed size into a relative size. This means that, one need to do a little math to achieve a flexible grid and text size system. It is suggested by [16] in his book a simple and consistent formula for converting fixed width pixels into proportional percentages, where the following formula is applied.

Target ÷ Context = Result

In order to calculate the proportions for each page element, one needs to divide the target element by its context. Currently, the best way to do this is to first create a high fidelity mockup in a pixel based imaged editor, like Photoshop. With your high fidelity mockup in hand, you can measure a page element and divide it by the full width of the page. For example, if one's layout is a typical size like 960 pixels across, then this would be one's "container" value. Then, let's say that one's target element is some arbitrary value, like 300 pixels wide. If one multiplies the result by 100, one gets the percentage value of 31.25% which one can apply to the target element. Here's the math in figure 1 below [17].

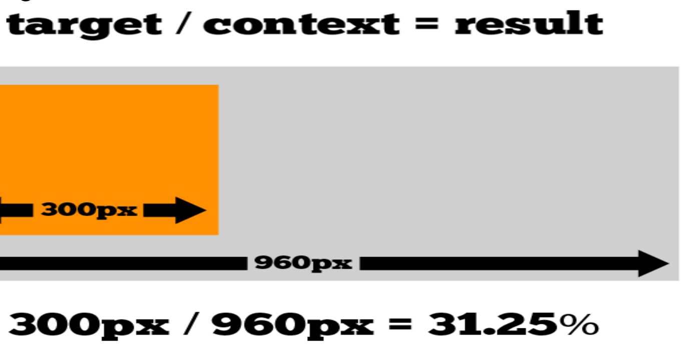


Figure 1: Proportion of a 300px element.Adapted from [17]

III. RELATED WORK

After extensive exploration of the related work, the study categorizes the existing solution for conversion from Fixed Grid Layout to Fluid Grid Layout in to three categories; Frame-based solutions, support-based solutions, and Algorithm-based solutions approaches.

3.1 Frame-based solution

The frame-based solution (FBS) is framework approach that uses a predefined layout which needs installation, training, and heavy customization prior to its use. There are several solution developed using this technique. Two of the most popular responsive frameworks are Twitter Bootstrap and Foundation. Twitter has open-sourced Bootstrap, a framework they use on Twitter. Foundation is a similar responsive framework created by ZURB. Both frameworks are fully responsive, allowing developers to use well-documented and tested components that will work on a large variety of screens and devices [12].

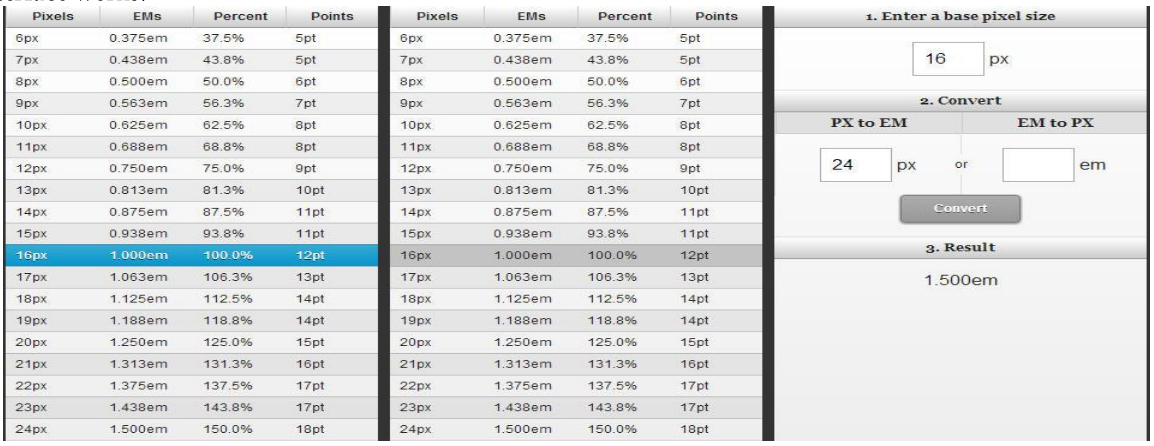
Creating a website to have a responsive, fluid layout can be a lot of work. There are a few responsive CSS frameworks that do a lot of the heavy lifting. CSS frameworks are libraries that package CSS and JavaScript components for commonly used UI components and interactions such as grids, buttons, or carousels. For example, if one needs converting an existing website, that has a design for horizontal navigation. Both Bootstrap and Foundation provide this component, however the default styles for the Foundation one is simple and plain, whereas the Bootstrap version is more polished. Foundation provides more control while Bootstrap does more work for you.

It seems clear from the above discussion that, FBS approach have address the implementation of Fluid Grid Layout from Fixed Grid. However, web designers have to spend more man-hours to customize the framework hence prolonging development time.

3.2 Support-based solution

The support-based solution (SBS) approach uses standalone tools which help a web designer to make some calculations to achieve responsive web design. The tools uses mathematical process associated with converting fixed-width design work to a fluid layout is converting absolute units of measurements (i.e. px and pt) into relative units of measurement such as ems and percent (%) for typography, spacing, container widths, etc.

There are several tools available to date. The most commonly used is PXtoEM at http://pxtoem.com/[5]. It is a tool that provides web designers with a simple conversion tool to help them with the entire math. The site also allows web designers the ability to quickly and easily change the base font size of their layout to something that leads to more manageable math. Figure 2 below demonstrate how the PXtoEM interface works.



Pixels	EMs	Percent	Points
6px	0.375em	37.5%	5pt
7px	0.438em	43.8%	5pt
8px	0.500em	50.0%	6pt
9px	0.563em	56.3%	7pt
10px	0.625em	62.5%	8pt
11px	0.688em	68.8%	8pt
12px	0.750em	75.0%	9pt
13px	0.813em	81.3%	10pt
14px	0.875em	87.5%	11pt
15px	0.938em	93.8%	11pt
16px	1.000em	100.0%	12pt
17px	1.063em	106.3%	13pt
18px	1.125em	112.5%	14pt
19px	1.188em	118.8%	14pt
20px	1.250em	125.0%	15pt
21px	1.313em	131.3%	16pt
22px	1.375em	137.5%	17pt
23px	1.438em	143.8%	17pt
24px	1.500em	150.0%	18pt

Pixels	EMs	Percent	Points
6px	0.375em	37.5%	5pt
7px	0.438em	43.8%	5pt
8px	0.500em	50.0%	6pt
9px	0.563em	56.3%	7pt
10px	0.625em	62.5%	8pt
11px	0.688em	68.8%	8pt
12px	0.750em	75.0%	9pt
13px	0.813em	81.3%	10pt
14px	0.875em	87.5%	11pt
15px	0.938em	93.8%	11pt
16px	1.000em	100.0%	12pt
17px	1.063em	106.3%	13pt
18px	1.125em	112.5%	14pt
19px	1.188em	118.8%	14pt
20px	1.250em	125.0%	15pt
21px	1.313em	131.3%	16pt
22px	1.375em	137.5%	17pt
23px	1.438em	143.8%	17pt
24px	1.500em	150.0%	18pt

1. Enter a base pixel size

16 px

2. Convert

PX to EM		EM to PX
24 px	or	em

Convert

3. Result

1.500em

Figure 2: PX to EM Conversion made simple.Adapted from [6]

PX to EM Demonstration. Assuming you aren't using PXtoEM.com to keep it simple, here are the formulas PXtoEM.com uses. **Note:** 16px is used as the body text size in all conversions because that is the browser default. You will change 16px to your base text size.
PX to EM - Formula: size in pixels / parent size in pixels**.** Example: 12px / 16px = .75em
PX to % - Formula: size in pixels / parent size in pixels * 100**.** Example: 12px / 16px * 100 = 75%
PX to PT - Formula: size in pixels * (points per inch / pixels per inch)**.** Example: 16px * (72pt / 96px) = 12pt
EM to PX - Formula: size in EMs * parent size in pixels**.** Example: .75em * 16px = 12px
EM to % - Formula: size in EMs * 100**.** Example: .75em * 100 = 75% [6].

It seems clear from the above discussion that, SBS approach have address the implementation gap by converting pixel to em of Fixed Grid Layout. However, web designers are burden by the use of a calculator as a result prolonging development time.

3.3 Algorithm-based solution

The algorithm-based solution (ABS) is an approach of creating algorithms which automatically convert fixed grid layout to fluid grid layout. There exist several algorithms to implement fluid grid concept. The most commonly used is BlockIt.js [13]. It is a jQuery plugin for creating dynamic grid layout. It is used to convert HTML elements into '*blocks*' and position them in well-arranged grid layout. It allows joining of two or more blocks into a big block element. It is licensed under the GNU General Public License. The plugin can be access through the following function interface for customization and creation of dynamic grid layout shown in Figure 3 below.

```
$(document).ready(function() {
                $('#container').BlocksIt({
                                numOfCol: 4,
                                offsetX: 8,
                                offsetY: 8,
                                blockElement: '.block'
                });
});
```

Figure 3: BlockIt.js Interface. Adapted from [13]

Where:
numOfCol: *Type: Int (Default: 5)* // The number of columns to be created.
offsetX:*Type: Int (Default: 5)* // Margin left and right for each block.
offsetY:*Type: Int (Default: 5)* //Margin top and bottom for each block.
blockElement:*Type: String (Default: div)* //Targeted child element, which will converted into blocks.

The figure 4 below demonstrates how BlockIt.js can randomly generate dynamic blocks. The source code algorithm for BlockIt.js is shown in Appendices A of the study.

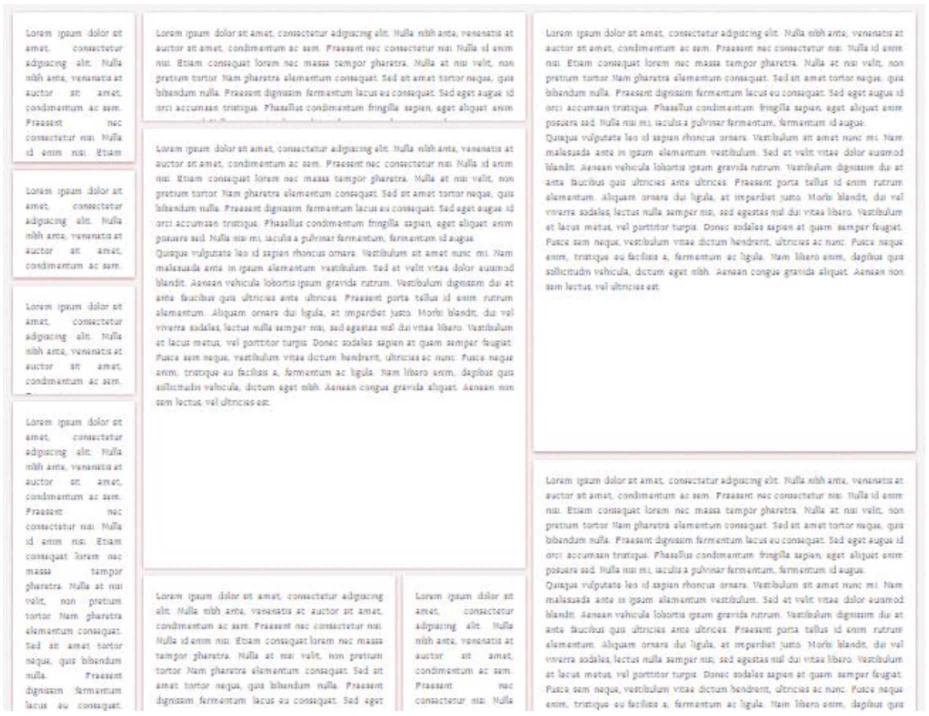


Figure 4: Dynamic grid layout with random generated blocks [13]

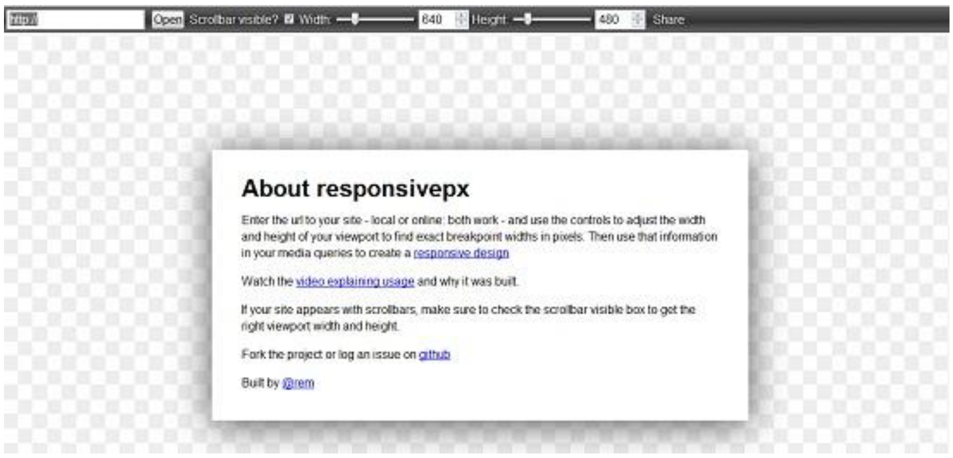
It seems clear from the above discussion that, ABS approach have address the implementation gap by developing an algorithm that generate dynamic grid layout but only after receiving an input through its interface. Therefore, the survey recommends an enhanced approach of ABS to be access by a single line of code.

IV. RWD ANALYSIS AND TESTING TOOLS

There exists various Analysis and testing tools which demonstrate how website responds to different screen and browser sizes implementing Responsive Web Designs. The two most commonly used are Responsivepx and Matt Kersley RWD Testing Tool.

4.1 Responsivepx

Responsivepx is an awesome tool for testing responsive website design. The main feature that distinguishes it from others is its capability to resize the website pixel-by-pixel. This awesome feature will identify the breakpoints and also test how the CSS media queries are working in a website. It is an online tool, which can be accessed at http://responsivepx.com/. The figure 5 below demonstrates how the interface of the Responsivepx works [18].



About responsivepx

Enter the url to your site - local or online: both work - and use the controls to adjust the width and height of your viewport to find exact breakpoint widths in pixels. Then use that information in your media queries to create a responsive design

Watch the video explaining usage and why it was built.

If your site appears with scrollbars, make sure to check the scrollbar visible box to get the right viewport width and height.

Fork the project or log an issue on github

Built by @rem

Figure 5: Interface of Responsivepx [18].

How responsivepx works. Simply, one enters the URL website - local or online (both works) and use the controls to adjust the width and height of ones viewport to find exact breakpoint widths in pixels. Then one uses that information in one's media queries to create a responsive design. If the website appears with scrollbars, then one has to make sure to check the scrollbar visible box to get the right viewport width and height. The figure 6 below demonstrates an example of a local host web site with a width of 700px and height of 567px.

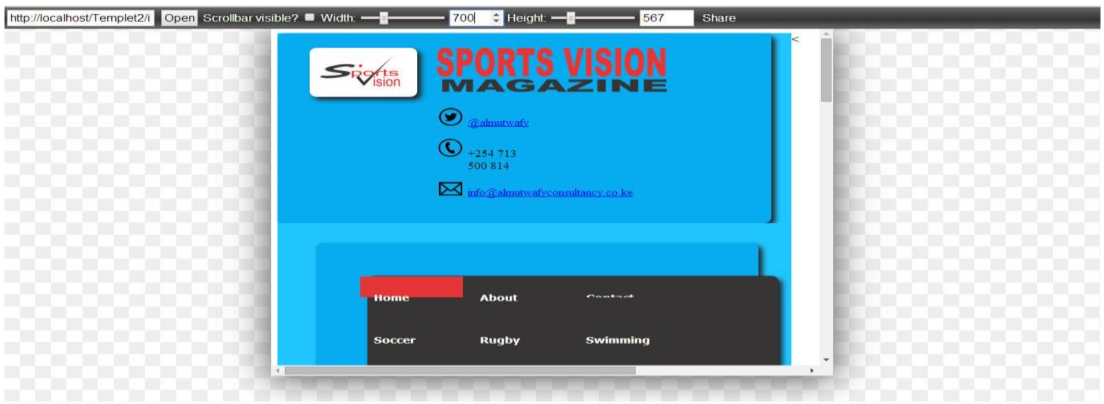


Figure 6: Local host website demonstration

One can find the exact breakpoints in pixels and update one's web page's media queries to create the responsive design one is looking for. One can also share the URL of the adjusted site viewport with one's colleagues to discuss one's design modifications online.

It seems clear from the above discussion that responsivepx is a great tool but the survey finds some limitation such as manual adjusting of the width and the height, and luck of displaying of various viewports on a single screen simultaneously.

4.2 Matt Kersley RWD Testing Tool

Matt Kersley RWD Testing Tool is also an awesome testing tool that allows viewing responsive website but in various screen sizes simultaneously in a single screen, while building or designing them. The survey prefers this tool mainly because it shows all the screen resolutions side-by-side which makes it easier for debugging. It is an online tool, which can be accessed at http://mattkersley.com/responsive/. The figure 7 below demonstrates how the interface of the Testing Tool works [14].

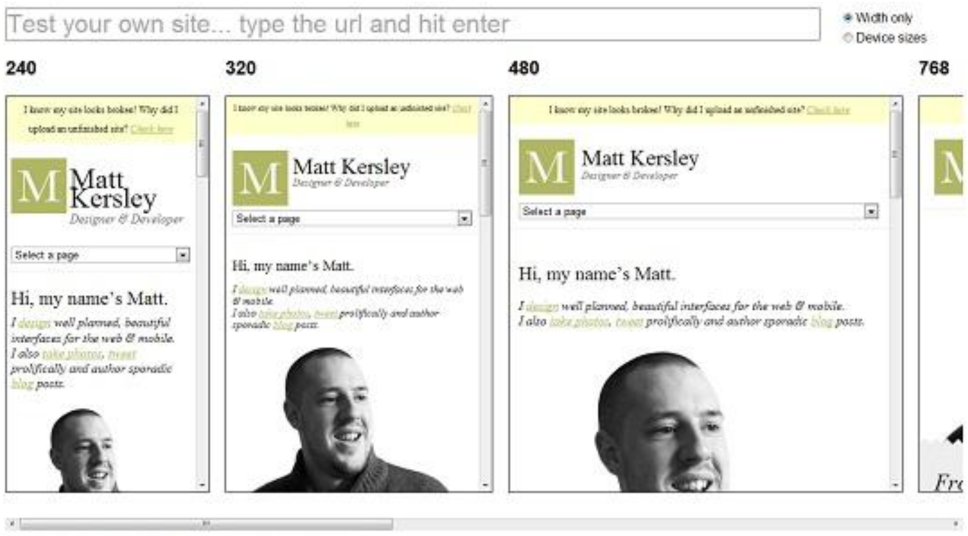


Figure 7: Matt Kersley Default Interface [14].

This tool has been built to help with testing responsive websites while designing and build them. One can enter website's URL into the address bar at the top of the page (not browser's address bar) to test a specific page. The figure 8 below demonstrates how a local host website is display in various screen sizes simultaneously in a single screen.

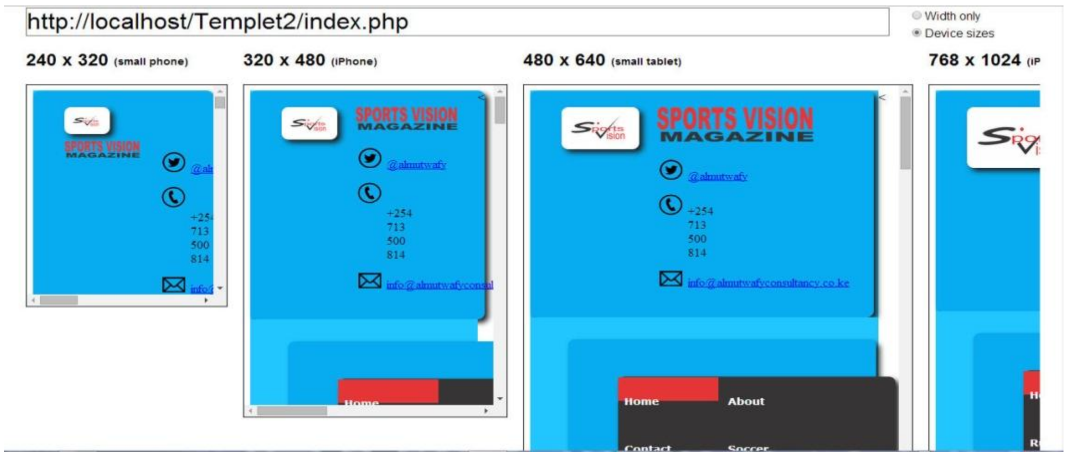


Figure 8: Local host website demonstration

Unfortunately, with the way browser security works, One is unable to navigate the website through the frames that website appears in. The only way this is possible is by hosting the testing tool on website's own host. Matt Kersley have provided at github repository for downloading and installation of the tool on any website.

It seems clear from the above discussion that Matt Kersley RWD Testing Tool has some limitation, however, its strength to display websites in various screen sizes simultaneously in a single screen make it suitable for fluid grid concept application and testing.

V. RECOMMENDATION

The review reveals a shift away from traditional web design towards Responsive Web Design which has evolve to become an unavoidable good practice in web designing. Further exploration of both the theoretical and empirical literature review, seems clear that there an urgent need for an alternative enhanced approach to RWD for fluid grid implementation. Hence, survey recommends a classification of three meaningful categories of; Frame-based Solution, Support-based Solution, and Algorithm-based Solutions approaches for Fluid Grid Concept implementation. Therefore the survey recommends further research works on an alternative enhanced ABS approach for fluid grid implementation and testing.

REFERENCE

[1]. Allsopp, J. (2000, April 7). A Dao of Web Design. *A LIST APART*. Retrieved from http://alistapart.com/article/dao

[2]. Bastian, J. (2014, February 18). Responsive Email Templates Are On the Way! *VR Product Blog*. Retrieved from http://www.verticalresponse.com/blog/product/responsive-email-templates-are-on-the-way/

[3]. Chapman, C. (2009a). *The Evolution of Web Design*. Retrieved June 26, 2014, from http://sixrevisions.com/web_design/the-evolution-of-web-design/

[4]. Chapman, C. (2009b, November 28). *The Evolution of Web Design*. *Six Revisions*. Retrieved May 18, 2014, from http://sixrevisions.com/web_design/the-evolution-of-web-design/

[5]. Cray, B. (2012a, November 3). *PXtoEM.com: PX to EM conversion made simple*. Retrieved May 8, 2014, from http://pxtoem.com/

[6]. Cray, B. (2012b, November 3). *PXtoEM.com: PX to EM conversion made simple*. Retrieved May 3, 2014, from http://pxtoem.com/

[7]. Doyle, M. (2011, September 30). *Responsive Web Design Demystified*. Retrieved May 3, 2014, from http://www.elated.com/articles/responsive-web-design-demystified./

[8]. Graeve, K. D. (2012, January 30). Responsive Web Design. *SitePoint*. Retrieved from http://www.sitepoint.com/responsive-web-design/

[9]. Harb, E., Kapellari, P., Luong, S., & Spot, N. (2011). Litrature Study: Responsive Web Design. Retrieved from http://courses.iicm.tugraz.at/iaweb/surveys/ws2011/g3-survey-resp-web-design.pdf

[10]. Hazael-Massieux, D. (2005). *Tableless layout HOWTO*. Retrieved June 29, 2014, from http://www.w3.org/2002/03/csslayout-howto

[11]. Hurb, E., Kapellari, P., Luong, S., & Spot, N. (2011, December). Survey Responsive Web Design.pdf. Retrieved from courses.iicm.tugraz.at/iaweb/.../g3-survey... Graz University of Technology

[12]. Johal, P. (2012, August 20). *Converting Your Website to Responsive Design - Part I*. *OpenRoad*. Retrieved May 6, 2014, from http://www.openroad.ca/2012/08/20/converting-your-website-to-use-responsive-design-part-i/

[13]. Kenny. (2012, April 24). *BlocksIt.js - Dynamic Grid Layout jQuery Plugin*. Retrieved May 8, 2014, from http://www.inwebson.com/jquery/blocksit-js-dynamic-grid-layout-jquery-plugin/

[14]. Kersley, M. (n.d.). *Matt Kersley | Designer & Developer*. Retrieved May 12, 2014, from http://mattkersley.com/

[15]. Knight, K. (2011). *Responsive Web Design: What It Is and How To Use It*. *Smashing Magazine*. Retrieved June 29, 2014, from http://www.smashingmagazine.com/2011/01/12/guidelines-for-responsive-web-design/

[16]. Marcotte, E. (2010, May 25). Responsive Web Design. *A LIST APART*. Retrieved from http://alistapart.com/article/responsive-web-design

[17]. Pettit, N. (2012, August 8). Beginner's Guide to Responsive Web Design | Treehouse BlogTreehouse Blog. Retrieved from http://blog.teamtreehouse.com/beginners-guide-to-responsive-web-design

[18]. responsivepx. (n.d.). *responsivepx - find that tricky breakpoint*. Retrieved from http://responsivepx.com/?localhost%2FSportsVision%2Findex.php#665x632&scrollbars

[19]. Smarty, A. (2013, July 15). A Look Into Responsive Typography. *Internet Marketing Ninjas Blog*. Retrieved from http://www.internetmarketingninjas.com/blog/design/a-look-into-responsive-typography/

[20]. Solanki, K. (2012, May 17). Responsive Web Design - Ideas, Technology, and Examples. Retrieved from http://www.onextrapixel.com/2012/05/17/responsive-web-design-ideas-technology-and-examples/

[21]. Tranfici, A. (2013, January 29). Responsive Web Design: Fluid Layouts. *SitePoint*. Retrieved from http://www.sitepoint.com/responsive-web-design-fluid-layouts/

[22]. Wroblewski, L. (2012, February 29). *LukeW | Which One: Responsive Design, Device Experiences, or RESS?* Retrieved May 7, 2014, from http://www.lukew.com/ff/entry.asp?1509