MAKE WEB CONTENT WORK IN RESPONSIVE DESIGN

[1]INDU JOSEPH, [2]V.S. FELIX ENIGO

[1]SSN College of Engineering, Department of Computer Science & Engineering, Chennai-603110
[2]Post Graduate Student, Associate Professor
Email: indjos@gmail.com, felixvs@ssn.edu.in

Abstract— Responsive web design is delivering web content across wide variety of display devices in an elegant way. The generic way of achieving responsiveness using relative width approach does not work well for all HTML elements. This paper deals with handling various non-responsive elements which can break the responsive layout. Different approaches for making them responsive using HTML5 and CSS3 web technologies are discussed. Furthermore, a case study on different approaches for making responsive site is carried out and its level of support across different browsers has been analyzed.

Index Terms— CSS3, HTML5, Relative width, Responsive web design.

I. INTRODUCTION

Responsive web design is a kind of device context aware computing [4] where the contents get adapted based on the screen size of device such as desktop, tablet, mobile etc. Screen size is the most obvious difference between the mobile browsers and desktop ones. Mobile browsers display significantly less of a desktop-optimized website than desktop browsers either by zooming out until the text is too small to read, or by showing only the small part of the site that fits in the screen.

The *viewport* meta element in HTML is what turns a regular website page into a responsive page. This is included in the head of the HTML document that tells the browser how the page should be rendered. Presentations can be tailored to a specific range of output devices without changing the content itself by using *media queries*. A media query [17] consists of a media type and zero or more expressions that check for the conditions of particular media features. Fig. 1 shows an example of simple media query. Media queries look at the capability of the device, and check width and height of the browser window, width and height of the device, orientation and resolution.

```
@media screen and (max-width: 768px) {
    .mobile {
      font-size: 11px;
    }
}
```

Fig. 1. An example of a simple media query

A. RESPONSIVE WEB DESIGN

Responsive web design is not just adjustable screen resolutions or automatically re-sizable images [5]. Rather allowing the user to easily read and navigate through the contents of the Website on different devices, with the minimum amount of manipulation on user part. In some cases, mobile-specific websites and native applications [1] can offer rich experiences than responsive design, but they aren't always ideally suited due to the difficulty in maintaining a separate desktop and mobile version.

As with all design patterns, responsive design also has both advantages and disadvantages. Organizations need to think about their actual needs before making the decision to implement it or not [2].

1) Advantages

Responsive design has a single code base and provides future proofing. Responsive sites use only one set of code for all devices, so there is only one site to maintain and hence less cost over time. Responsive web design makes it easy to adjust layouts for new devices and screen resolutions. Rather than redesigning the site to fit new device, existing or new layout rules can be applied to optimize content. This can optimize content to any window size even if the browser window is set to a smaller size. Also, using one URL eliminates the need for complicated redirects and eliminates the possibility of mobile users accidentally being sent to the desktop version of the site. One link is easily sharable for all devices.

2) Disadvantages

Development cost and Cross Browser Compatibility are the major challenges faced by the responsive designs. These take a little longer to be put together, but they survive longer and management, support and upgrades only need be applied to one place. This saves time and money. The site should be tested across different browsers and different versions of browsers to ensure the support of media queries. Another challenge for RWD is the bandwidth. Mobile users may have limited data plans. Designs that hide content for display on small screens can lead to bandwidth wastage and higher time to load the page. To effectively utilize the available bandwidth, conditional loading of JavaScript is done. Pairing of CSS and JavaScript allow only the necessary scripts and content to be loaded, hence minimizing the bandwidth usage. Some elements in the web page such as iframes, images, embeds, videos, tables can break the responsive layout. Such non responsive elements should be handled separately to develop a responsive site.

In this paper, an analysis of various non-responsive elements in a site and approaches to make them adapt

to various devices are considered. The browser compatibility, which is a major technical challenges faced by responsive sites is also analyzed.

II. RELATED WORK

An intelligent device independent UI adaption for heterogeneous devices using XHTML and CSS based code is proposed [3]. This aims at providing a reasonable browsing experience to mobile users. For porting applications between heterogeneous devices, approaches of Dissimilar content-sets based UI across terminals, CSS Support, non-uniform support for structural elements (tables, lists), XHTML based device specific markup are followed. Context-Aware adaptive rendering [6] adjust the quality of visualization, adapting to the current device context. It dynamically adjusts the quality of presentation on client devices according to the current device context.

An analysis of impact of image-heavy responsive websites on the experience of mobile users is done. In a survey conducted [7], the user experience for image-heavy responsive websites was assessed. Only 21% of the responsive websites responded with acceptable load time. The book [8] gives an elaborate description of responsive web design using CSS3 and HTML5. All the concepts of Responsive Design with fluid layouts, viewports and CSS options are described. A white paper published by equator [9], discusses why and how to design a responsive website. The process is discussed in detail highlighting the importance of understanding and clearly defining the reasons for building a responsive site and the various steps in designing it. Responsive Web Design Best Practices Guide by IBM [10] discusses a checklist of RWD problems and the practices.

Various case studies on developing responsive sites are discussed. Web Representation of Vector based on Canvas API of HTML5, for developing interactive maps [11], makes a higher compatible, faster rendering speed, lower network latency website and considers the responsive layout. A responsive website is designed for college [12] using HTML5, CSS and JavaScript. Few other case studies and projects done on HTML5 with responsive layout are mentioned in [13], [14]. The design and source code analysis have been explained in Building Responsive Layouts [15]. It focuses on the responsive layouts like Fluid or Liquid layout that uses percentage width to adapt to the size of viewport.

The existing approaches do not provide responsiveness for the content embedded in a webpage whereas in the proposed system embedded content takes different display based on device width, has better wrapping and folding of data.

III. NON RESPONSIVE ELEMENTS

Responsive web design is the practice of building a website which can work on every device and every screen size, no matter how large or small, mobile or desktop. Percentages are the prevalent method to achieve the fluid layout in responsive designs. Such responsive features are added to the elements within the CSS. But all elements in the web page do not act responsively with these features. Iframes, embeds, images, videos and tables are such elements which break the responsive layout in HTML. The non-responsive behavior of these elements and approaches to make them responsive are discussed in this section.

B. Iframes

The HTML <iframe> Element (or *HTML inline frame element*) represents a nested browsing context, effectively embedding another HTML page into the current page. An example iframe element is shown in Fig. 2. The embedded webpage doesn't look like a separate web page it looks like a content that's part of the main page.

Iframe includes width and height attributes and URL that points to the source of the streamed content. On removing these, iframe disappear because it would have no dimensions. But having a fixed size breaks the layout and do not adapt to the device. In the example shown in Fig. 2, the width attribute means that, on a screen narrower than *420* pixels, the embedded content will protrude outside of its containing element, breaking the layout.

To make these elements responsive, the iframe can be placed in a wrapper which is a responsive element. This approach ensures that the iframe is not truncated and come within the visible region of device. But the content within the iframe might not show the responsiveness. To ensure content adapts to the dimension of the device separate CSS and JavaScript has to be added to calculate dynamically the size of the container and load the contents accordingly.

If not configured correctly these elements will begin to download their contents even though the users might not ever want them. To avoid this, a Lazy Load approach can be taken. Lazy load serve up an image heavy website without having to suffer with all the pre-fetching and loading of images that may never be seen by the user.

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/BznCRFuNpMc"
frameborder="0" allowfullscreen=""></iframe>
```

Fig. 2 An example of HTML iframe element

C. Images

Images are given fixed size and position when created and users expect it to be displayed in the position where they are. Consequently, when the page is resized, images should resize and stay where they are. Images have been a major obstacle in implementing truly adaptable and responsive pages. The images should retain its aspect ratio based on the parent/container width, setting the height to the correct

proportional size when resized.

An adaptive image needs to be able to render crisply at different device-pixel-ratios. High-resolution screens should get high-resolution images, but need not want to send those images to users who wouldn't see all of those extra pixels. If the layout is fluid (responsive), then images will need to squish and stretch to fit it. Another scenario for adaptive images is different browsers support different image formats like sending a new format such as WebP to browsers that can render it, and fall back to trusty old JPEGs in browsers that don't.

The <*picture*> specification includes features for all of these cases. It is a markup pattern that allows developers to declare multiple sources for an image. By using media queries, it gives developers control as to when and if those images are presented to the user.

When implementing responsive images (different images in HTML for different situations), switching between different versions of the same image is done with the help of 'srcset'. The *srcset* and *size* attributes extend the *image* and source elements to provide a list of available image sources and their sizes. Browsers can use this information to pick the best image source. Both srcset and sizes are part of the HTML specification and can be used separately or in conjunction with the picture element as shown in Fig. 3.

```
<picture>
   <source media="(min-width: 36em)"
      srcset="large.jpg  1024w,
         medium.jpg 640w,
         small.jpg  320w"
      sizes="33.3vw" />
   <source srcset="cropped-large.jpg 2x,
         cropped-small.jpg 1x" />
   <img src="small.jpg" alt="A rad wolf" />
</picture>
```

Fig. 3. An example of HTML picture element

D. Video

If HTML5 video is used, it fits the width of the container. It's important that height declaration of video is removed, so that the aspect ratio of the video is maintained as it grows and shrinks. Declaring static widths isn't a good idea in fluid width environments. Videos can be made responsive via CSS by setting the width to 100%. But this trick does not work when dealing with video that is delivered via iframe. Setting a height is required or browsers will render the iframe at a static height of 150px.

One technique to overcome this is by wrapping the video in another element which has an intrinsic aspect ratio and then provide an absolute position to the video within that. That gives a fluid width with a reasonable height. But due to this need of an additional wrapper element, copy-and-pasting the video url directly from YouTube is not possible. When there are legacy content to be redesigned as fluid, all old videos need HTML adjustments. All videos need to be the same aspect ratio. Otherwise they'll be forced into a different aspect ratio

If either of these limitations applies, JavaScript solution should be considered. When the page loads, all videos are looked at and their aspect ratio is saved. On first load as well as when the window is resized, all the videos are resized to fill the available width and maintain their aspect ratio.

The 'preload' attribute has number of states that specifies the load properties of the video. If preload is set to none, the user does not expect any video and minimize unnecessary traffic. If the fetching of metadata is desirable and not the video, preload is set with metadata. For optimistically downloading the entire video, state of preload attribute can be set to 'auto'.

E. Tables

As the viewport gets smaller, the table width shrinks. However depending on how many columns the table has, it reaches a minimum size. This will probably be wider than a mobile viewport, and the layout gets broken. A JavaScript-CSS combo let the tables adapt to small device screens. For making such fluid table structure the rigid 'table' tag cannot be used. The table has to be reconstructed with 'div' which makes the page light weight as well.

One technique is to provide a horizontal scroll which allows the user to swipe left and right to view the entire table, without breaking the layout. But this technique reduces the user experience due to excessive scrolls. Another trick works by taking the first column and "pinning" it to the left of the table, allowing scrolling the other columns under it.

An alternate mobile layout for the mobile can be considered. When the browser is resized or site is viewed on a mobile device, a third layout of single column is constructed. This allows the user to have minimum data on the screen on load and view only the necessary information from corresponding rows.

IV. CASE STUDY

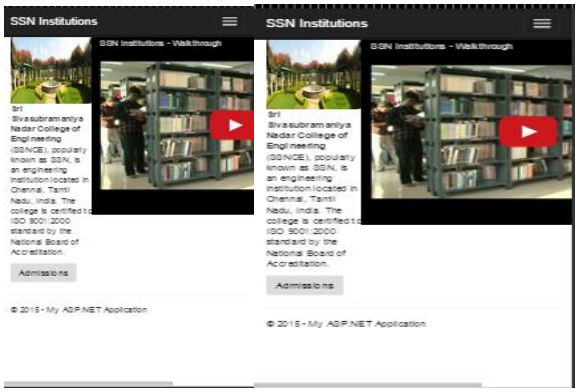
To analyze the non-responsive elements discussed in the previous section, a sample application with text and video are considered which is embedded into another site using iframes. It uses breakpoints to determine how the layout of a site will appear: one design is used above a breakpoint and another design is applied below that breakpoint based on the width of the browser. The same HTML is served to all devices, using CSS (which determines the layout of webpage) to change the appearance of the page. The page elements reshuffle as the viewport grows or shrinks.

F. Markup for Embedded Content

To embed content from an external source, the code includes an *iframe* as shown in Fig. 2. External

Services such as YouTube, Google Maps provide code that can be used in the website to embed content. However, iframe includes width and height attributes. On removing these and the iframe will disappear because it would have no dimensions. And having a fixed size breaks the layout.

The screen shots are taken from different devices showing how the content get rendered. The text in the page is shrunk so that the embedded content fits the screen. But as shown in Fig. 4, the video gets truncated due to its fixed width. Fig. (4a) is the screenshot taken from Apple iPhone 6 (375x667) and Fig. (4b) is from Samsung Galaxy Note (400x640).



(4a) (4b)

Fig. 4 Video gets truncated in small screen

G. Making the Site Responsive

To make embedded content responsive, a container is added to the contents as a wrapper. This wrapper and the iframe in it are styled based on the intrinsic ratio and positioned. Based on the intrinsic ratio [9], the following styles are applied on the wrapper and the iframe:

```
.responsive-container {
    position: relative;
    padding-bottom: 75%;
    padding-top: 35px;
    height: 0;
    overflow: hidden;
}
.responsive-container iframe {
    position: absolute;
    top:0;
    left: 0;
    width: 100%;
    height: 100%;
}
```

Fig. 5. Styles for wrapper and iframe

The style applied performs the following actions:

1. Setting the position to relative lets the contents load based on the absolute position of the iframe. Absolute positioning is used because the containing element has a height of 0.

2. The padding-bottom value is calculated out of the aspect ratio of the video. In this case, the aspect ratio is 4:3 16:9, which means that the height will be 75% of the width. For a video with 6:9 aspect ratio, padding-bottom is set to 56.25%. The padding-top value is set to 30 pixels to allow space for the browser.

4. The height is set to 0 because padding-bottom gives the element the height it needs. Width need not be set because it will automatically resize with the responsive element that contains this div.

5. Setting overflow to hidden ensures that any content protruding outside of this element will be hidden from view.

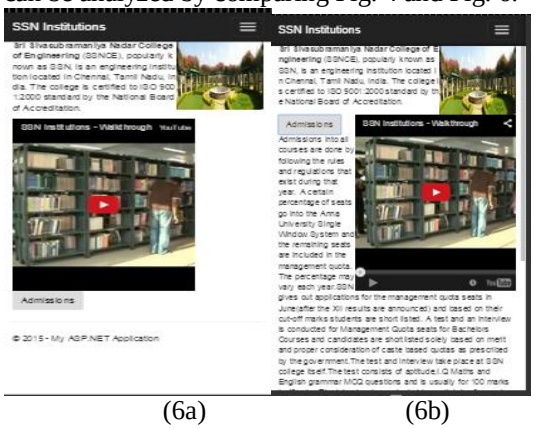
6. The width and height properties ensure that the video takes up 100% of the space given by the containing element.

Having done this, the video will now resize with the screen's width. To change the layout and prioritize the contents of the site, CSS Media queries are used. Media queries are the filters that are applied to CSS styles. They make it easy to change styles based on the characteristics of the device rendering the content, including the display type, width, height, orientation and even resolution.

The media query check for the width of the device and execute special styles accordingly. This includes hiding functionalities that do not make sense on mobile or take too much bandwidth, while it can also be used to change the site's grid from horizontally oriented to vertically oriented; by placing a sidebar not right of the main content, but below it.

Media queries are the ideal way of adapting the design to different screen resolutions. Still, they are not the ultimate of making the website mobile-friendly. Media queries do not stop the browser from downloading assets that will not be used on a mobile phone. And with regard to bandwidth this is a serious problem. A solution for this can be given by using JavaScript, that read the media queries and uses the data to decide whether to download the script, whether to download the low-source or the full-source images, or none.

On applying these styles and scripts, the site behavior in Apple iPhone 6 (375x667) and Samsung Galaxy Note (400x640) are shown in Fig. (6a) and Fig. (6b) respectively. The responsive behavior of the site can be analyzed by comparing Fig. 4 and Fig. 6.



(6a) (6b)

Fig. 6 Responsive site

V. BROWSER COMPATIBILITY

One of the major technical challenges of Responsive Design is to enable the site to work across multiple devices and browsers. Based on the data shared from StatCounter GlobalStats[16], Fig. 7 shows the statistics of market share percent of browsers in the period Feb 2014 to Feb 2015 for the desktop, mobile and tablet. It is necessary that the developed site is tested across the top browsers to check for compatibility.

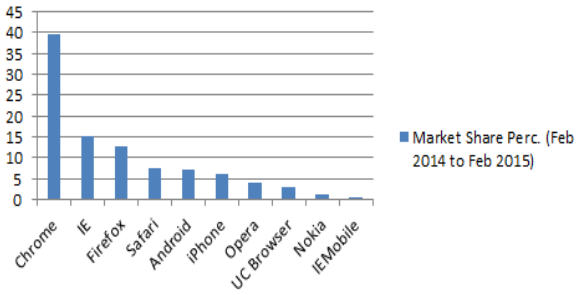


Fig. 7 Market Share Percent of Browsers

Modernizr is a JavaScript library that detects HTML5 and CSS3 features in the user's browser. It makes the developer easy to write conditional JavaScript and CSS to handle each situation, whether a browser supports a feature or not. It runs quickly on page load to detect features; then creates a JavaScript object with the results, and adds classes to the html element to key the CSS on.

The application developed as part of case study is checked across the top browsers and ensured that the responsive behavior is achieved irrespective of the browser. The browser versions that support CSS media queries [18] and the currently used version are summarized in Table I.

Table I. Browser Support for CSS Media Queries

Browser	*Supporting Version*	*Current Version*
Chrome	38,39,40,41	41
IE	9,10,11	11
Firefox	35,36	36
Safari	7,8,8.1	8.1
Android	4.3, 4.4, 37	37
Opera	27	27

CONCLUSION

Few HTML elements have the habit of breaking responsive layouts, because it's contained in a container with a fixed width. In this paper, an approach of adding a containing wrapper, and some CSS, to ensure that all embedded content resizes with the browser's window. The most used browsers and its compatibility for media queries are analyzed. A major challenge faced in Responsive web design is the almost same weight webpage for desktop and mobile. Feature detection and including server side components overcome this with a faster loading time. But the problem of caching and server load has to be handled effectively.

REFERENCES

[1] WhitePaper by RapidValue Solutions, Responsive Web Design vs. Mobile Web App:Whats Best for Your Enterprise?, Web designer Depot,December 2012.
[2] S. Mohorovii, Implementing Responsive Web Design for Enhanced Web Presence, University of Rijeka, Faculty of Maritime Studies, Rijeka, Croatia, MIPRO 2013.
[3] Qamas Gul Khan Safi, Dr. Tabassam Nawaz, Syed M. Adnan Shah, Toqeer Mahmood, Intelligent Device Independent UI Adaption for Heterogeneous Ubiquitous Environments, IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.11, November 2011
[4] G.Kousalya, P.Narayanasamy Hetergogenous Device Context Aware Information Dissemination, International Journal of Multimedia and Ubiquitous Engineering, Vol. 6, No. 1, January, 2011
[5] Pallavi Yadav, Paras Nath Barwal, Designing Responsive Websites Using HTML And CSS, International Journal Of Scientific & Technology Research Volume 3, Issue 11, November 2014
[6] N.A.Nijdam, S. Han, B. Kevelham, N. Magnenat-ThalmannA Context-Aware Adaptive Rendering System for User-Centric Pervasive Computing Environments, MELECON 2010- 2010 15th IEEE Mediterranean Electrotechnical conference
[7] Trilibis Mobile Survey, Responsive Web Design: Why Image Optimization crucial for a Mobile-Friendly Customer experience , April 2014.
[8] Ben Frain, Responsive Web Design with HTML5 and CSS3, Packt Publishing, 2012.
[9] Equator White Paper, How to design a responsive website, 2013.
[10] IBM, Responsive Web Design (RWD) Best Practices Guide, Version: 2013.11.20.
[11] Beiqi SHI, Neng CHEN, Lin SHEN, China, Research on web vector representation of thematic map using HTML5, May 2012.
[12] Deanna Klein, Aleksandar Gubic, Responsive website design for higher education utilizing mobile centric features, Online Journal of International Institute for Applied Knowledge Management, Volume 2, Issue 1, 2014.
[13] Karl Andersson, Dan Johansson, Mobile e-Services Using HTML5, 6th IEEE Workshop On User Mobility and Vehicular Networks, Lule University of Technology, Sweden, 2012.
[14] Meltem Huri Baturay, Murat Birtane Responsive Web Design: a new type of design for web-based instructional content, 4th International Conference on New Horizons in Education, 106 (2013) 2275-2279.
[15] Zoe Mickley Gillenwater, Building Responsive Layouts, Responsive Web Design Summit, August 2012.
[16] http://gs.statcounter.com/#browser-ww-monthly-201402-201502
[17] http://www.w3.org/TR/css3-mediaqueries/
[18] http://caniuse.com/#feat=css-mediaqueries

★★★