Chandra Shekhar Aryal

Design Principles for Responsive Web

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

05 December 2014

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Chandra Shekhar Aryal Design Principles for Responsive Web 40 pages + 6 pages appendices 28 October 2014
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Supervisor	Olli Hämaläinen , Senior Lecturer

The purpose of this bachelor's thesis project was to study the responsive design paradigms and development approaches for creating web pages that are optimised for adaptive web design. The additional goals were to analyse the design principles and implement prototype to find out whether it is feasible to achieve responsive design for various screen resolution of devices.

The theoretical part of the thesis work explains more details of primary development approaches and design consideration for creating responsive web pages that work on the latest smartphones to desktop computers. The thesis also analysed the best design principles that help to optimize web performance. For practical purpose, simple web page as a prototype was built, tested on various screen resolution devices and evaluated in terms of adaptive design with the mobile first principle.

As a result of this project, a functional web page prototype was built using Dreamweaver, web application design software. HTML5, CSS3 and JavaScript were used for the structure, presentation and interaction of web functionality. The web page contains videos and text as web content. A mobile compatible menu was implemented using third parties library.

The development still needs further work for better user experience and more adaptive solutions with browser compatibility. The main challenges for developing project were designing web user interface across multiple devices with various browser supports. Understanding the responsive design paradigm, the mobile first approach, content planning, navigation and web performance were benefits gained from this study.

Keywords	Responsive web design, adaptive design, viewport, screen resolution

Contents

Appendices

Appendix 1. CSS Style Sheet Used for Presentation of Web Page

Appendix 2. CSS Media Query Used for Media Screen Conditions

Abbreviations and Terms

API	Application Programming Interface
CSS	Cascading Style Sheets
CSSOM	Cascading Objet Model
DOM	Document Object Model
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
MathML	Mathematical Markup Language
OS	Operating System
RWD	Responsive Web Design
SVG	Scalable Vector Graphics
URL	Uniform Resource Locator
UI	User Interface
UX	User Experience
W3C	World Wide Web Consortium
WWW	World Wide Web
XHTML	eXtensible Hypertext Markup Language

1 Introduction

Browsing a website via a mobile client, in general a smartphone or a tablet has been increasing in recent years. This phenomenon is mainly due to an increase in the use of smartphones and low data and the device cost compared to early stages of the smartphone. According to the Global Mobile Media Consumption research conducted by InMobi, mobile ad network 60 percent of Internet access is mostly mobile. This research was carried out in 14 different countries, including the USA and the UK among 14,000 respondents [1]. With this pace of growth it is likely predictable that within 3 to 5 years mobile browsing will outpace desktop-based access. So developing web content that are rendered smoothly and enhances user experience is the need of today's businesses and web service providers.

Traditionally websites were designed with a fixed grid layout. This design paradigm of a fixed grid layout was popularized by graphic designers with fixed rows and columns, which create modules where web content could be placed. The inflexibility of these fixed designed layout led to problems because websites were also browsed from a mobile device where user experience was poor compared to desktop browsing. To address the need of a mobile-friendly website, different versions of the same website were created based on the user's device. The solution for delivering a website based on the device request is done by redirecting to a separately designed mobile websites for mobiles, tablet websites for tablets and desktop websites for desktops. However, the solution is not appropriate for the latest context, where smartphones have gained much popularity as devices to browse websites. Different smartphones have different screen resolutions. Browsing a website that is designed for a particular device leads to poor user experiences. This new circumstance makes it difficult to design websites for all the devices whose resolution and screen sizes are predetermined by the developer.

The need of today's websites or web applications should be adaptive and accessible regardless of the device used. This approach of developing websites is popularly known as responsive website design. This thesis aims to explain the current trend of responsive website design to develop a website that does not affect the user experience on different devices. The purpose of this study is to understand the best practises used in responsive website design and its importance. A prototype website is created to demonstrate how responsive design can be accomplished.

2 Theoretical Background

2.1 Responsive Web Design

Designing a website for flexibility and adaptability is called Responsive Web Design (RWD). The RWD term was coined by Ethan Marcotte,a freelance web designer, inMay 2010 in an article “A list Apart”. RWD is a contemporary website design and development paradigm. It is a principal that focuses on optimizing a website so that it is flexible, adaptive and provides superior user experience. It enhance a websites with best possible viewing experience, smooth content reading and navigation with reduced panning, scrolling and resizing regardless of screen sizes on various devices. This approach leads to design that is simple and responds to the user’s behaviour and environment based on the screen size, platform and orientation. [2]

There are various advantages of choosing a responsive design approach instead of the traditional design approach where different versions are created for different devices. Developing a responsive website is expensive in the beginning but it eliminates the need to create several versions for mobile and other devices. This is rewarding in the long term financially and in terms of time spent to manage the website. Use of mobile devices to access the internet is rising exponentially which implies it is a wise approach to design websites with RWD. Since RWD enhances user experience and the users can easily find the content that they are looking for, the users will not be deviated from the website. [2]

Figure 1.Responsive web page for various screen resolution devices. Reprinted from DDSHOSTING [3].

In figure 1, a responsive web page for different devices is shown. The web page is adaptive for desktops, tablets, and for mobile devices. It is common for only important features to be present on a small screen device. Similarly the design is adaptive in a different viewing mode, viewed as landscape and portrait on mobile devices.

2.2 Elements of Responsive Design

In order to design a responsive website, a developer has to take different design aspects into consideration. A proposal by Ethan Marcotte suggests three main elements in responsive design namely Fluid grid, Flexible media and Media queries which are described in detail in the sections 2.2.1, 2.2.2 and 2.2.3 below. [2]

2.2.1 Fluid Grid

Before responsive design was popular, a typographic grid was used to design websites. Typography, a rational system of columns and rows upon which modules of content could be placed, was a popular web implementation method. The lowest screen resolution to support was chosen most commonly 1024x768 pixels for designing a website. This fixed unit, e.g. pixels, points, inches, used to define the grid size was a major problem towards supporting multiple devices. A desktop browser of today can have more than 1024 pixels, while a smartphone can be as small as 320 pixels. This problem is overcome by using a relative unit such as percentage to CSS property's value so that the viewport, browser display area, is relative to the parent container and its child. [4, 23-31]

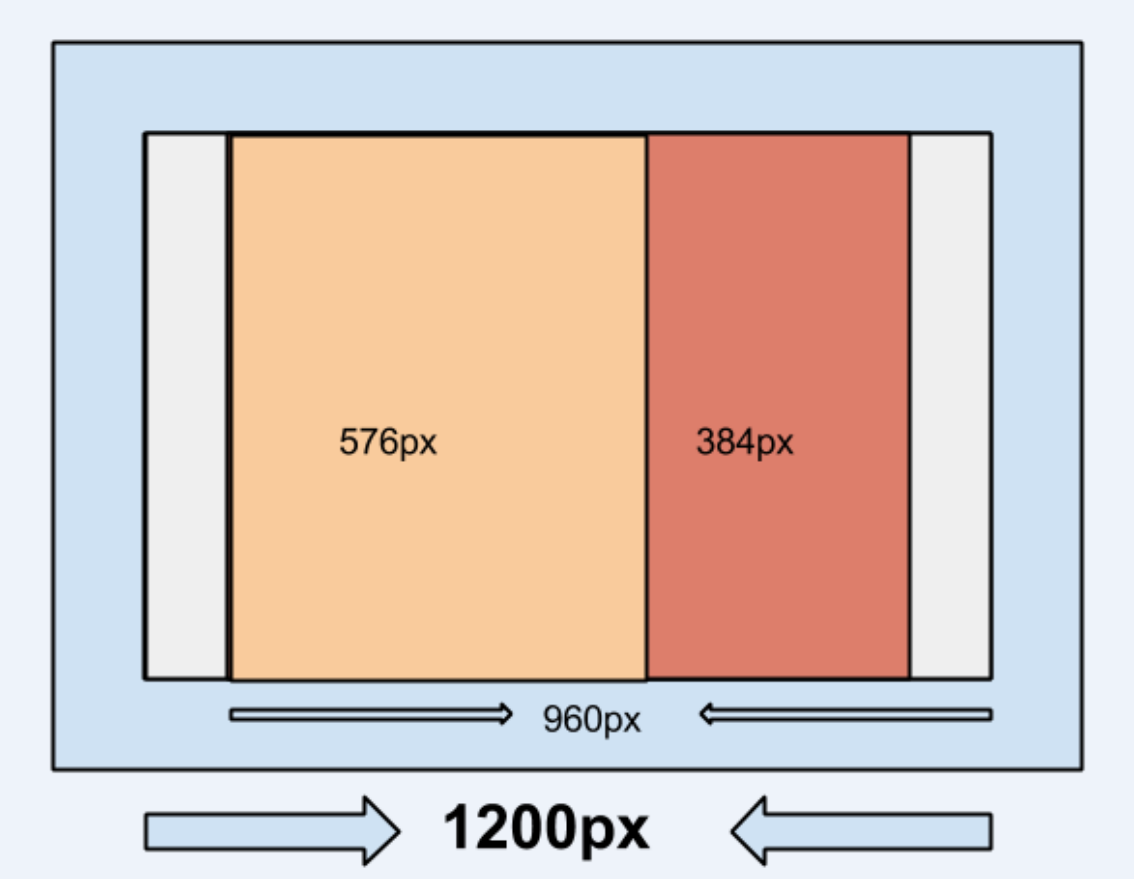

Figure 2.Typographic grid layout with fixed unit.

In figure 2, the main container width of a website is 960 pixels while the screen resolution of device is 1200 pixels. The main container of a website has two child containers with 576 pixels and 384 pixels respectively. The website design is based on fixed typographic grid and best suited for the screen with a width 1200 pixels. Viewing the fixed typographic grid of a website in figure 2, from small screen devices such as mobile, tablet is problematic. Before mobile websites were popular, this kind of design was widely used for desktop browsers.

The relative unit can be achieved as a percentage unit by dividing the target with context formula. [4, 23]

Context Formula

$$\frac{Target}{Context} \times 100\% = Result$$

In the formula, the target is the child container whereas the context is the parent container. [4, page 30, 31] In figure 2, the web page has 960 pixels width which is a child container of 1200 pixels parent container, which contains all the child containers. By using the context formula, the result is 80 percent for the parent container. Similarly, the parent container has two child containers with 576 pixels and 384 pixels widths which results in the percentage value of 60 percent and 40 percent for each child respectively as shown in figure 3.

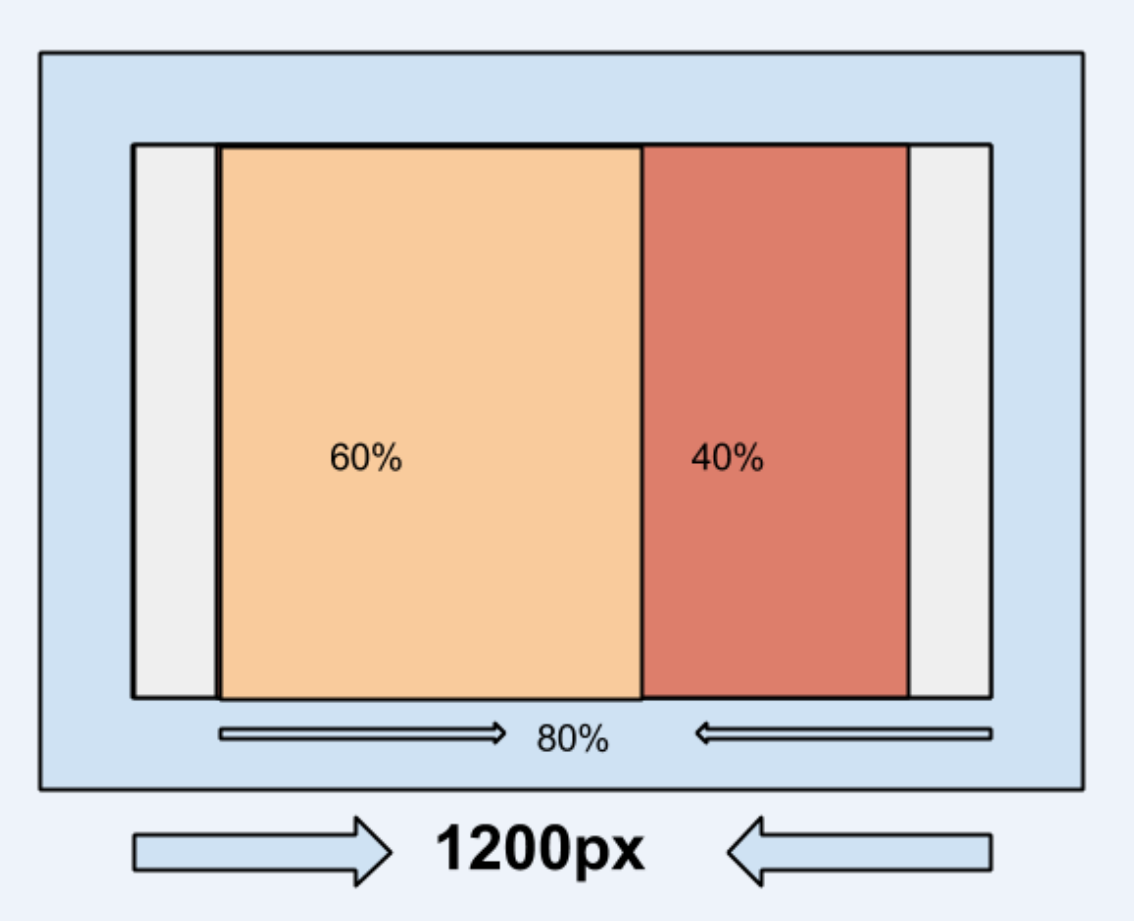

Figure 3. Flexible grid with percentage unit

Figure 3 shows the flexible grid of the same website. So instead of defining the web container with fixed width as shown in figure 2, the flexible grid takes the percentage approach by defining width in the percentage value. It is possible to replace pixels with percentage values in the width attribute of the container in CSS. This approach will solve the problem of today's devices with various screen resolutions. When a website is viewed on a modern web browser, it interpret the style that is in percentage and renders accordingly.

2.2.2 Flexible Media

Images and videos are not naturally adaptive in web pages. The initial setup and orientation stay the same at all configurations of the viewport of the devices. The screen where web contents are rendered is called the viewport of the device. The scaling of media does not fit to the various screen resolutions of devices with initial setups. This results in media that either stretches horizontally or vertically or is cropped.

The shortcoming of rendering media can be overcome by using a relative unit, percentage in CSS. The media inherits the same properties as the grid layout. However this is not a reliable solution all the time since scaling of media might not take place according to the container scaling. Richar Rutter, a user interface designer of websites, has experimented and discovered the technique for scaling of the media with its container by using “max-width: 100%” attribute in the CSS properties of media elements. It is possible to overcome the scaling of media with respect to its container and also media will not occupy more than the 100% width of the container.[4, 43-47]

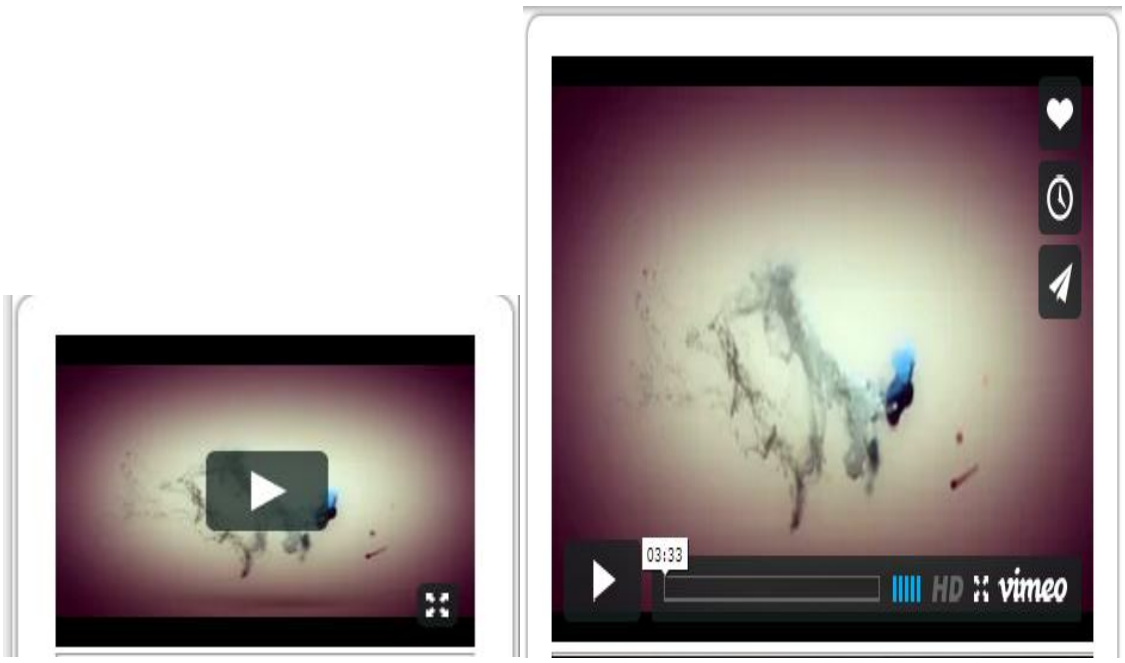

Figure 4. Snapshot of flexible media viewed in smartphone and tablet device.

In figure 4, the snapshot on the left is from a smartphone and one on the right is from a tablet, the flexible media was experimented on mobile and tablet devices. The videos were experimented by using max-width: 100 percent attribute in the CSS properties of the media element. As a result, the video is scaled properly with respect to media containers and devices viewport.

2.2.3 CSS Media Queries

The third element in addition to flexible grid and flexible media needed for responsive design is CSS Media Queries. Responsive design is incomplete without CSS Media Queries. The percentage and proportional design may not be feasible to adjust with the requirements of different viewport dimensions of devices such as mobile phones to huge screen desktop browsers. The layout can result in a squeezed format in small screen devices. [4, 64-69]

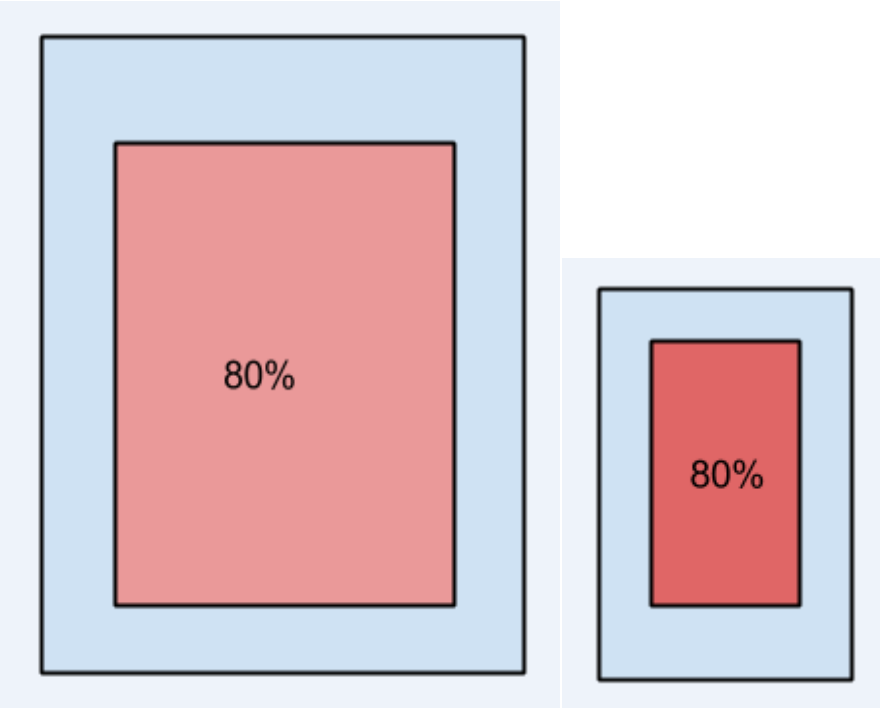

Figure 5.Web container using percentage unit.

In desktop browsers, the layout of the website works perfectly by using 80 percent of width. However on the small screen devices such as mobile phone, the layout of the website is squeezed to narrow and not feasible for the web contents as shown in figure

5. On the mobile phone the viewport of the screen is too small to fit all the contents. Using all space will be appropriate for the responsive approach. For this reason using media query as a break point, which refers to the devices viewport where certain attributes apply, for defining new attributes of CSS properties will help to overcome the problem , as shown in figure 6.

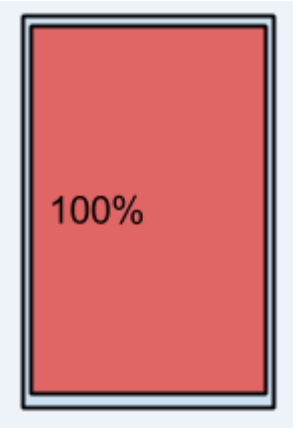

Figure 6.Web container with 100% width.

The World Wide Web (W3C) introduced the media query as a part of the CSS3 standard in June 2012, which solved this problem of not being able to define a particular CSS for a particular range of viewport and give way to designing responsive web pages. [5]

Media Query is a CSS3 module allowing different media types and features to adapt different styles for individual device viewport, device orientation, and many more. It consists of different media types, aural, braille, handheld, print, projection, screen, tv. Each media type is followed by different expressions that check conditions of particular media features such as: width, height, device-width, device-height, orientation, resolution, grid, color. It uses logical expression true or false. When the media features and values allocated is true then the given style is applied. The expression and value allocated is false, the style is ignored. [5]

Table 1. Common screen resolutions of devices for defining break points. Reprinted from Marcotte (2010) [4, 114].

320 pixels	For small screen devices, like phones, held in portrait mode.
480 pixels	For small screen devices, like phones, held in landscape mode.
600 pixels	Smaller tablets, like the Amazon Kindle (600×800) and Barnes & Noble Nook (600×1024), held in portrait mode.
768 pixels	Ten-inch tablets like the iPad (768×1024) held in portrait mode.
1024 pixels	Tablets like the iPad (1024×768) held in landscape mode, as well as certain laptop, netbook, and desktop displays.
1200 pixels	For widescreen displays, primarily laptop and desktop browsers.

In the example listing 1, the screen is used as the media type. The common screen resolution is implemented as a breakpoint to check the conditions. In table 1, the common screen resolution of the devices is shown. The given screen resolution can be used as a breakpoint for various devices to check the condition of the media query and apply appropriate CSS style.

```
@media only screen and (min-width: 320px) and (max-width: 480px)
{
          // CSS style for given screen resolution devices.
}
```

Listing 1.CSS media query condition for mobile devices

For example in listing 1, when the browser window width matches the condition of the given media query where the screen's minimum resolution is 320 pixels and maximum 480 pixels, the given CSS style is applied. This design will be appropriate for mobile

phones in the portrait and landscape modes. Similarly, by defining the appropriate media query condition it is possible to achieve adaptive and responsive websites.

2.3 Web Technologies

The Hyper Text Mark-up Language (HTML), Cascading Style Sheets (CSS) and JavaScript are three main technologies for creating web pages. In the web pages there are mainly three components to name structure, presentation and interaction. The HTML is used for structuring the web content, the CSS is used for presentation and JavaScript is used for all interactive functionality. These technologies are the basic components for designing web pages.

2.3.1 Hyper Text Mark-up Language 5

HTML5 is a mark-up language used for structuring and presenting the content for web pages. On 28 October 2014 the World Wide Web and Web Hypertext Application Technology Working Group standardized the fifth revision of the HTML. The core aims is to support the latest multimedia, subsume HTML4, XHTML and DOM Level 2 HTML and to make it easily readable elements. [6]

The new features of HTML5 in comparison to HTML4 are the following:
New features were added while old features such as <img> were deprecated in HTML5. W3C added additional elements to support multimedia and dynamic graphics in HTML5. This included the <audio>, <video> and <canvas> .Using these elements makes it possible to play sound, video and dynamic graphics in the web browser without relying upon on external plug-in or software. The new media element enables designers and developers to provide modern browsers to create rich experiences and dynamic graphics reliably across all platforms. [7]

In addition to multimedia, HTML5 allows for MathML and SVG vector graphics elements to be linked or embedded inside the content of a single HTML file. By using JavaScript it is possible to interact with these elements for any changes without relying upon any third-party platforms. [7]

Semantic tags <article>, <section>, <header>, <footer>, <aside> and <figure> tags, are used to create self-describing HTML documents. It helps in organizing the web contents, describes the clear meaning to both developer and browsers, easy navigation for search engine, facilitating for device-to-device communication. [7]

New types of form input elements such as telephone number, email, date and time were introduced in HTML5. These form elements enable high speed data capture within a native paradigm without using any further scripts libraries. [7]

In addition to these features, HTML5 content uses, XML- based formatting rules. It is a simple DOCTYPE syntax <! DOCTYPE HTML> compare to previous SGML (Standard Generalized Markup Language) used in HTML4, which requires specific DTDs (Document Type Definitions) to validate document structure. Also, HTML5 specifies several new application programming interfaces (APIs) that can be used with JavaScript. [7]

Major browsers, all of the leading smartphones, tablets and multi-screen devices support HTML5. It is intended to bring rich media, dynamic graphics, high user experience, and complex user interface, well-structured and semantic contents, supporting many APIs, and having compatibility with modern web browsers. Also it supports all existing HTML 4 based content with backwards compatibility.

2.3.2 Cascading Style Sheet

On 17 December 1996, CSS (Cascading Style Sheets) level 1 was first released by HäkonWium Lie and Bert Bos with the W3C recommendation [14]. Until today, It has several CSS levels such as: CSS Level 1, CSS Level 2, and CSS Level 3. CSS Level 3 is a large single specification defining various new features including several modules with new capabilities or extends the features defined in CSS2. [8]

The CSS is the language for describing the presentation of web pages such as colours, layout and fonts. It is one of the core technologies for building web pages with visual and aural layout of web pages for a variety of devices. It can be used with any XML-based mark-up language and independent of HTML. The style can be inserted in three different ways: External style sheet, Internal style sheet, Inline style. [8]

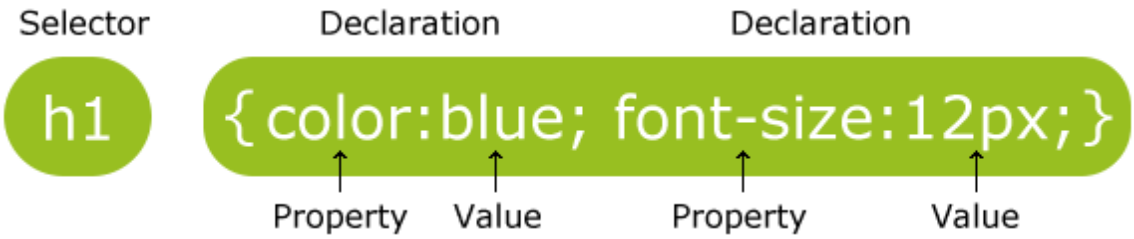

Figure 7.CSS syntax declaration, Reprinted from w3schools website [9]

The CSS style consists of list of rules with selectors and declarations as shown in figure 7. The selectors are the elements of the HTML and the declarations are the attributes of the CSS style. As shown in figure 7, selector “h1” is the header element of HTML. By using the CSS attribute color and the font-size it is possible to give style with colour blue and font size12 pixels.

2.3.3 JavaScript

JavaScript was created in 1995 by Brendan Eich and was released with Netscape 2 in 1996. It is an object-oriented and loosely written client-side scripting language. However the list of areas where it is used is rising up every day. With the introduction of node.js, we can use JavaScript for server side scripting as well. Today JavaScript interpreters can be found on other platform such as, Adobe Acrobat and Photoshop. [10]

Defining the content with HTML and presenting the content with the CSS design is incomplete without adding user interaction achieved by the use of JavaScript. Creating interactive content, responding the behaviour of user selection, adding functionality, validating the input form, loading and submitting data from server are created by using JavaScript. JavaScript is embedded inside the HTML pages and interacts with document object model (DOM), organizes objects of HTML in the tree structure of the page. The script runs locally in web browser to provide quick response to user actions. [10]

3 Modern Design Paradigms

3.1 Mobile First Principle

Starting from 2010 there have been tremendous changes in the field of web development. Initially web contents were designed so that they were best fitted for desktop browsing. After the introduction of the first iPhone in 2007 by Apple, the mobile web crossed several milestones, including the latest paradigm popularly known as the Mobile First Principle.

Before 2007, accessing web content using a mobile web browser was very cumbersome, due to technological constraints such as slower mobile Internet, unreliable connection, not so advanced mobile web browsers and unpleasant user interface experience. A number of people accessing web via a mobile device was very low. Now, with the advancement in technology, the smartphones become a need for most people, Today the trend is towards mobile web. [11, 1-3]

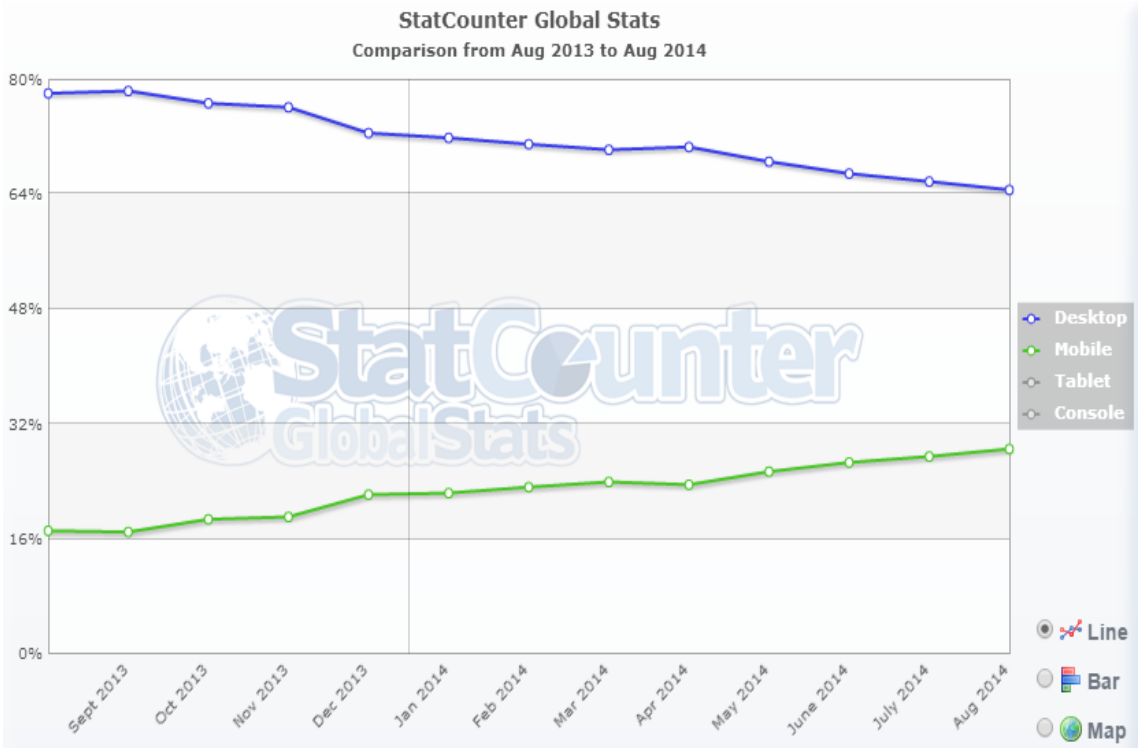

Figure 8.Worldwide Internet usage through desktop and mobile platforms. Reprinted from StatCounter [12]

Figure 8, shows the Internet usage of desktops and mobiles from August 2013 to 2014 through web browsers. In August 2013, the Internet usage of mobiles was nearly 17 percent of total internet usage. After 12 months, the Internet usages increased to around 28 percent of the total Internet usage. Whereas Internet access to mobile devices increased by 67 percent. On the other hand, the Internet access through desktop browsers decreased by 21 percent, from 77 percent to 64 percent. These statistics suggest that accessing web pages through a mobile browser is increasing rapidly.

Table 2. Shipment of Worldwide mobile devices. Reprinted from Canalys [13]

Worldwide mobile device shipments

Category	2012 shipments (millions)	2016 shipments (millions)	CAGR
Total	**1,936.2**	**2,614.2**	**7.8%**
Basic phone	122.0	58.0	-17.0%
Feature phone	770.8	660.9	-3.8%
Smart phone	694.8	1,342.5	17.9%
Tablet	114.6	383.5	35.3%
Notebook	215.7	169.1	-5.9%
Netbook	18.3	0.3	-65.4%

Source: Canalys estimates and forecast, © Canalys 2013

Similarly, as can be seen from table 2, worldwide shipment of mobile devices (notebooks, tablets, smartphones and phones) will reach around 2.6 billion units by 2016 according to the forecasts from Canalys. The tablet and smartphone are the fastest growing categories with a compound annual growth rate (CAGR) of 35 percent and 17 percent respectively. So, it is clear that in the coming future, there will be huge numbers of smartphones and tablets used for various purposes.

Based on data from figure 8 and table 2, we can conclude that there will be a huge surge of mobile devices and mobile web access by 2016. Developers, organizations and governments should focus on creating web pages with Mobile First Principle. Designing for mobile first now creates new opportunities for growth and future, and also it

can lead to better solutions and user experience for websites or applications. It is highly recommended for developers to start everything from mobile-based designed.

When planning mobile design for web experiences there are many factors that developers should focus on. Understanding the current needs of the user at the right time and providing right solutions is a primary need for today's web developers. Designing responsive web pages for small to large screens is a very challenging approach. Organizing content first, focusing on priority issues, providing good navigation, designing good user interface and maintaining clarity are a good start for mobile first design. Aligning the important uses, understanding mobile behaviors, providing relevant exploration and pivoting options are also important factors that should be considered in mobile first principle. [11, 7-14]

3.2 Content First Principle

"There is no Mobile Web. There is only The Web, which we view in different ways. There is also no Desktop Web. Or Tablet Web. Thank you." [14]

Today, web pages are not built for a specific viewport. However the excess of these web pages is increasing rapidly through different viewports that include handheld devices. In these cases, presentation of the content first approach is important. However the intrinsic designs of the desktop web are only possible for mobile devices via zooming, which is not the today's target for responsive design. [14]

Responsive design is about displaying the source of relevant content across countless viewports, using media queries and a flexible grid to ensure that design is adaptive and accessible as needed [4,107]. Since the same layout is not possible to be implemented across different viewports, it is important to focus on providing the content as it is in different viewports because users care about the easiness of getting the right content. [11, 52]

The primary goal of a web page is to provide content to a user. The content is the soul of any website. The content can be anything: text, image, video, music or application. The visitors are the users of web pages. Users need the desired content immediately when they visit a web page, which means that content should come first. So designing the responsive web is not only about the flexible grid and media queries. It's more

about content first, which serves the right content at the right time to the user regardless of different viewports from where the webpage is viewed. [11, 52-58]

"The main goal of content strategy is to use words and data to create unambiguous content that supports meaningful, interactive experiences. We have to be experts in all aspects of communication in order to do this effectively."[15]

As explained in the above quote, the primary use of a website is to get content. A user expects it to be presented beautifully and in a very interactive manner but when situations are not favorable, such as when there is a slow Internet connection or when the user is browsing via a mobile device, what the user really cares about is to get the content. Different functionalities such as animation, color and easy navigation are increasingly used in websites to engage users, but they are not appropriate when what a user really cares about is getting the right content. [15] For example in a situation when there is a slow internet connection and the user is browsing the BBC website on a mobile device, the user care about getting the meaningful content and not how beautifully it is designed or how the buttons animate when the user clicks on them. So content first is a good approach when content is more important than presentation and functionalities.

3.3 Optimization of Web Performance

The other important principle when going toward responsiveness is performance. Performance of web pages is an important factor for engaging a user. According to research done at Google, delay and slower performance reduces the user activities by 0.2% to 0.6% within half a second [16]. Today, mobile phones are less efficient and rely on mobile data connections whereas traditional desktop computers and laptops in general, have a steady and persistent Internet connection. Mobile users can easily deviate from web pages if it takes a longer time to load and slow down the web performance. Performance should be focused as a priority before starting development. [11, 22-24]

At present broadband speed is getting faster which implies that content can be easily fetched from the Internet, but different countries have different Internet infrastructures to facilitate faster connection. For example a Northern European country like Finland

enjoys a faster connection while developing countries like Nepal have a very slow Internet connection. So the target audience of the website should be taken into consideration before implementing. A website that performs well in one part of the world might not do the same in other part because of the Internet speed factor. Optimization is possible with eliminating unnecessary download, leverage browser caching, reducing HTTP Requests, image optimization, enabling compression techniques and minimizing uses of HTML, CSS and JavaScript. Using asynchronous scripts is essential to consider for optimization of web development. Optimization possibilities are further discussed in the chapter 3.3.1 to follow. [17, 4-5]

3.3.1 Reducing Web Page Weight

Web pages continue to grow day by day with their own ambition and functionality as well as by the number of users. The downloaded data for a web page also increases at a steady pace as shown in figure 9. To provide better performance on both desktop and mobile platforms the downloaded data must be taken into consideration.

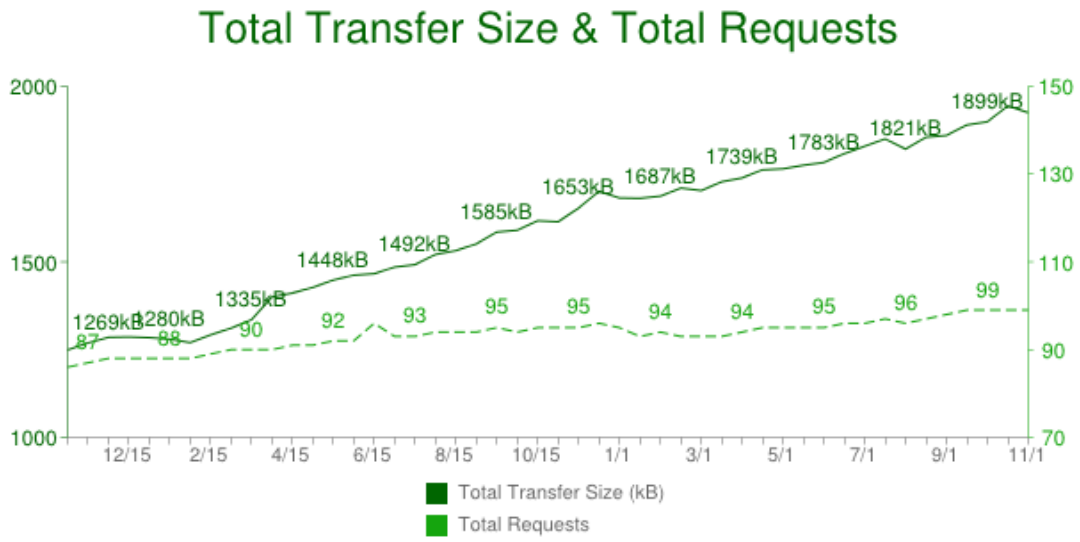

Figure 9. Data downloaded through Hypertext Transfer Protocol (HTTP) request, Reprinted from HTTP Archive-Trends [18]

Figure 9 shows the average downloaded data of all web pages worldwide by HTTP request from November 2012 to November 2014. The data was taken from the HTTP Archive web page and around 290,000 URLs were analyzed. According to the HTTP Archive, the average downloaded data for a web page was 1899 kilobytes (kB), compared to 1269 kilobytes in November 2012. This represents around 49 percent of growth rate in just two years. Similarly, the number of HTTP request is also rising up by a 13 percent growth rate. From this we can conclude that the number of HTTP request and download bytes have grown at a steady pace throughout the years which implicating the performance.

A Few simple steps while writing code for web pages include removing all unnecessary code comments from HTML, CSS and JavaScript and collapsing two style declarations into one without affecting each other and removing whitespace (space and tabs). This technique can be used also when implementing CSS frameworks and JavaScript libraries to minimize the size of download and optimize the performance. [19]

GZIP and Deflate are tools used for compression of web page files for all HTTP requests. Both are supported in all modern browsers. HTML documents, scripts and style sheets are common file types used for compression. Image and PDF files should not be compressed because they are already compressed. Using compression tools reduces the HTTP response by web pages on about 70%. Comparing both tools for compression, GZIP is typically used for compression because it reduces about 6% more compared to Deflate as shown in table 3. [17, 29-31]

Table 3.Compression file sizes using gzip and deflate. Adapted from Sounders (2007) [17, 31]

File type	Uncompressed size	Gzip size	Gzip savings	Deflate size	Deflate savings
Script	3,277 bytes	1076 bytes	67%	1112 bytes	66%
Script	39,713 bytes	14,488 bytes	64%	16,583 bytes	58%
Stylesheet	968 bytes	426 bytes	56%	463 bytes	52%
Stylesheet	14,122 bytes	3,748 bytes	73%	4,665 bytes	67%

Compression works in the following way:

- The browser sends http request header telling the server that it accepts the compressed content : `Accept-Encoding :gzip, deflate`
- The server sends an HTTP response: `Content-Encoding: gzip`

3.3.2 Image Adaptability

Adaptability of downloading images enhances performance. Images often occupy a significant amount of space on the web page. Downloading the actual size of the image slows the speed of the web page. Compression of the image can be one solution to save space. However for responsive design, the image has to be best fitted from a mobile to desktop browser. Choosing the right format of the image and its size with the respective screen resolution of devices is more essential. The fewer bytes the browser downloads, the less usage of the Internet and the faster the speed. [20]

By using CSS media queries it is possible to know the screen resolution of devices and apply an appropriate image. In addition to matching the screen resolution it also matches the device pixel ratio condition to choose the right format of image. The detail of using the media queries is explained in chapter 2 section 2.1.

Using JavaScript script, it is possible to have specific images. JavaScript allows checking the capabilities of the devices to determine device pixel ratio, get screen width, height via `window.devicePixelRatio` and `navigator.connection` for network connection. Once the collection information such as device pixel ratio, get screen, height and network connection known, then it will be easy to decide which image to load. [20]

The HTML5 picture element offers a declarative approach for an image loading solution for providing multiple version of an image based on the device viewport, device size, device resolution and orientation. Also, by defining the `srcset` attribute of HTML enhances the behavior of image selection. It allows the browser to choose the best image depending on the device behaviors. There will be no longer need for CSS or JavaScript to handle responsive images. The developer benefits the optimization of an image by

using picture elements for image resource loading, which only download the image necessary for the particular viewport. [20]

```
<picture>
<source media="(min-width: 1024px)" srcset="image-1x.jpg, image-
2x.jpg">
<imgsrc="head-fb.jpg" srcset="image-2x.jpg 2x" >
</picture>
```

Listing 2.Using picture element in HTML5.

In listing 2, if the browser width is at least 1024 pixels, then the image-1x.jpg or either image-2x.jpg will be used, depending on the device resolution. If the browser does not support the picture element, it will renders from <img> element and it should always include.

3.3.3 JavaScript Interactivity

Modification of web pages such as content, styling, user interaction and validations are all done in the HTML Document Object Model (DOM) tree and the CSS Object Model (CSSOM) with JavaScript. In a production environment, an adding an inline script or external file script it is possible to interact with DOM and CSSOM.JavaScript offers interactivity power to web pages for modification but dependencies between the DOM and CSSOM can lead to a significant delay. The bottleneck of using JavaScript comes with the price of slower performance over a low speed network. [21]

Execution of the script blocks DOM construction for an unnoticeable time in a high speed Internet connection but it is clearly visible on a slower connection. When the HTML tag encounters a script tag, it will immediately pause the construction of DOM and will yields control over to the JavaScript engine. Once the script has finished running, it will allow the DOM tree to pick up from where it left and resumes the construction. The waiting time for execution of the script, always interrupts the construction of the main HTML DOM tree. Similarly, the construction of DOM tree is also interrupted by CSSOM. Until the downloading and building of CSS is not accomplished the script interrupts the construction of HTML DOM tree. [21]

By default JavaScript's parser is blocking and the browser does not know the output of the script and what it is planning to do on the page. However there is a solution for it, by using the "async" in the script tag as shown in the code example below. It is possible that, the script does not need to be executed at the exact point where it is defined and referenced in the HTML tag. It tells the browser that it should allow the DOM construction to continue without any interrupt and wait for the script to become available. In this way it is possible to increase the performance of web pages without affecting the interactivity of JavaScript. The other way is to include the script tag in the end of the body, which ensures that the DOM is loaded first. [21]

```
<script src="ScriptFile.js" async></script>
```

3.3.4 Reducing HTTP Requests

The Hypertext Transfer Protocol (HTTP) is the client/server protocol which requests and responses for communicating with the browsers and servers over the Internet. Uniform Resource Locator (URL) is a webpage address for sending an HTTP request in the human readable form. The server hosting that URL sends back the response. GET, POST, HEAD, PUT, DELETE, OPTIONS and TRACE are types of requests with simple, plain text format. Figure 10 shows the architecture for an HTTP request and response. [17, 6]

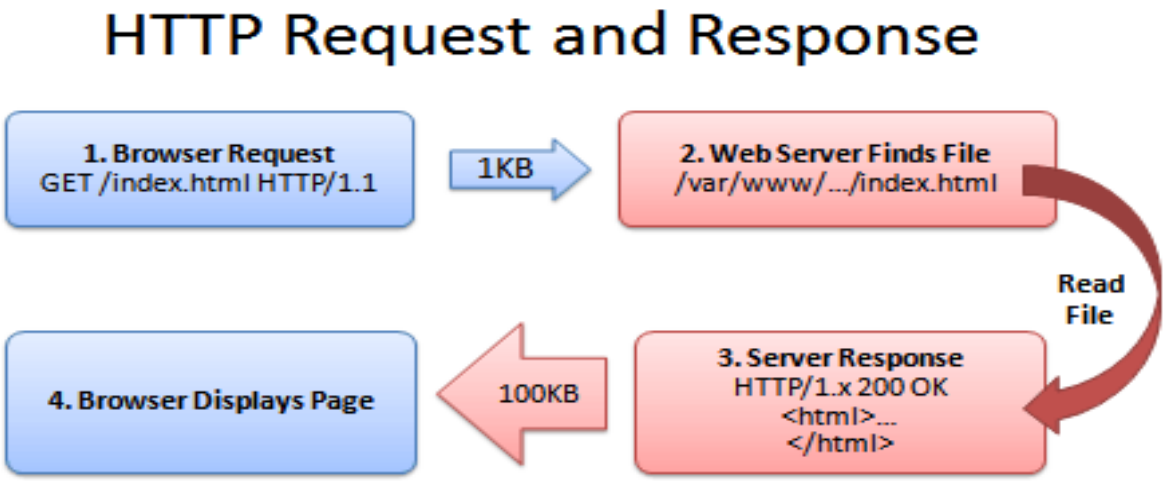

Figure 10, HTTP request and response architecture .Reprinted Web page [22].

In an HTTP request and response, end-users get the HTML document in just only 10-20 percent of the response time. However the rest of 80-90 percent of the response time is spent making other HTTP requests for components of web pages such as images, script file, style sheets and flash. Reducing the number of HTTP requests reduces the response time. It can be done by using techniques such as CSS sprites, Image maps , inline images and combined scripts and style sheets. These techniques can reduce the response time by 50% of total response time. [17, 10]

JavaScript and the CSS are most used in web pages. There are a number of scripts and style sheet files in web pages for specific purposes. These separate files require an additional HTTP request for downloading from a server. Combining multiple script files into a single script and multiple style sheets into a single style sheet reduces the number of HTTP requests. Steve Sounder, author of High Performance Web Sites, experimented this technique and his finding showed that a page with combined scripts loads 38 percent faster than the uncombined scripts files. [17, 15-16]

Another way to reduce the HTTP request is to reduce the number of image request in web pages. This can be done by using the CSS Sprites. In this technique all images are combined into a single image file. Using CSS properties, background-image and background-position, it is possible to know the location and position and easy to fetch the image for a required layout. [17, 11-13]

Similarly, there is another technique called Image maps. It is the same as the CSS sprites principle, by combining multiple images into a single image and it can fetch image by defining the coordinates of image maps. However there is one drawback in it: it only works for contiguous images in the pages, such as a navigation bar. [17, 10-11]

Additionally, an Inline image is another technique. It is a simple approach where it includes images in web pages without adding more HTTP requests. It can be done by using the data URL scheme. In this approach the image data is embedded in the actual page through a URL scheme. However, a drawback is there will be no cached of the image in the browser. To rectify this situation, a better way is to use inline images with the CSS rule. [17, 13-14]

3.3.5 HTTP Caching

HTTP Caching is a way to achieve the performance of web pages and to provide high end-user experiences. It is adopted universally through all modern browsers. Implementing the HTTP caching benefits the web page by improving the response time and reducing the server load. The web browser stores the copies of web resources. When the resources are required, browser will retrieve the resources without requesting the server, which is called HTTP caching. However, when the content of web pages changes frequently, this is not a good solution. To overcome problem, HTTP headers have ETags, tokens generated by the server typically a hash or fingerprint of web contents. Browsers automatically validate the ETags token and notify the client if there is modification done. Analyzing necessary ETags tokens from servers should be confirmed and ensured by the web developer. [23]

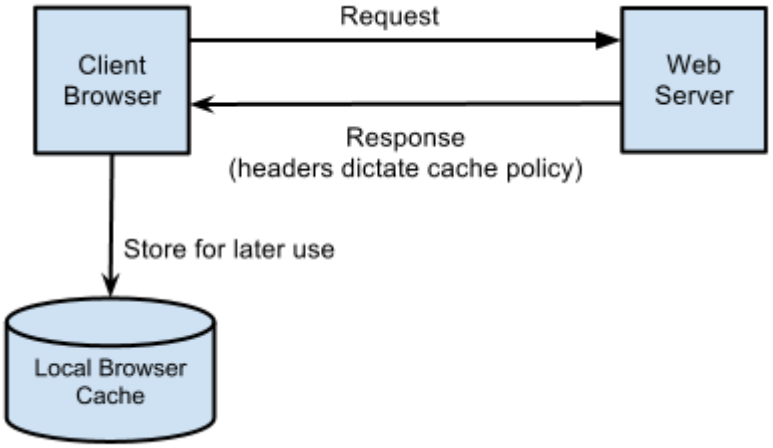

Figure 11.HTTP caching in local browser cache. Reprinted from Heroku Dev Center [23]

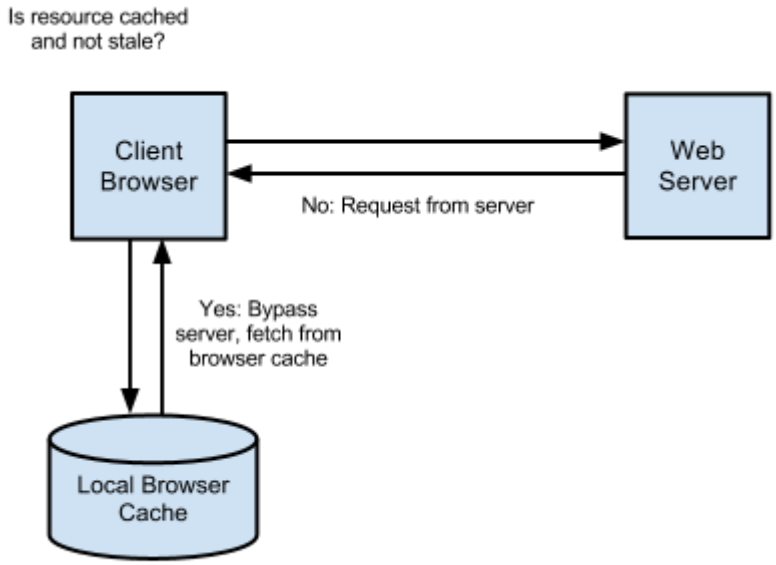

Figure 12.Browser cache alleviating resources without an HTTP request for the server. Reprinted from Heroku Dev Center [23]

Figure 11 illustrates the simple cache architecture of a browser. A browser requests an HTTP request for the first time, web server will response with all the required information. The required responses are stored for later use in a browser cache. Whenever the resource is requested again, as shown in figure 12, a web page using HTTP cache headers is able to fetch the required resources without alleviating the server-load again. This practice will increases response time.

4 Prototyping a Responsive Web Page

The primary purpose of the study was to examine the principle behind the responsive web and use that principle and implement a simple prototype. To find the best practical way is to design a simple web page and make it adaptive and responsive. This section discusses how the prototype was carried out while following the software life-cycle development phases: requirement analysis, overall design, implementation and testing the result.

4.1 Requirement Analysis

The requirement analysis is the initial phase of software life-cycle development. In this phase the web page prototype requirements were analysed and outlined. The requirements of the web page are listed below:

1. Responsive design approach
2. Uses the latest technologies HTML5 and CSS3
3. Following the design principles
4. Supporting cross browser / device testing

Each requirement is discussed in detail in the paragraphs below.

The First approach explains that the web page has to be responsive with mobile, tablet and desktop browsers. The three main elements of responsive web design, Fluid grid, Flexible media, and CSS media queries as discussed at the beginning of this thesis, should be used.

In the web the basic web technologies are HTML, CSS and JavaScript. As mentioned in the section 2.3.2, HTLM5 has more benefits compared to HTML4. Applying the new feature such as picture element, semantic tags and media element of HTML5 will enhance the design. Similarly, the web page has to use the CSS3 media query for the latest presentation and visual design of the web page. Rich user interface and better user experiences are valuable key points for web page success.

After analyzing the third requirement it was decided to follow the mobile first principle. This means the prototype was designed for a mobile initially. Combining JavaScript and CSS was done using a normal compression method, so that the script was minified for optimization.

Finally, the fourth requirement is about testing the design in the latest version of browsers: Chrome, Mozilla, Internet explorer, Safari. It is also important that the design is appropriate for each device desktop, laptop, tablets, and mobiles. It does not mean the design should be the same, it is about coherent experience across all devices.

4.2 Overall Design

After analyzing the basic requirements, the user interface sketch layout was designed before the real implementation of web pages. The desktop, tablets and mobile layout of the content was designed separately. This helped to establish the system design of the whole web page. Sketching the webpage layout before implementing helped to identify the breakpoints of responsive design.

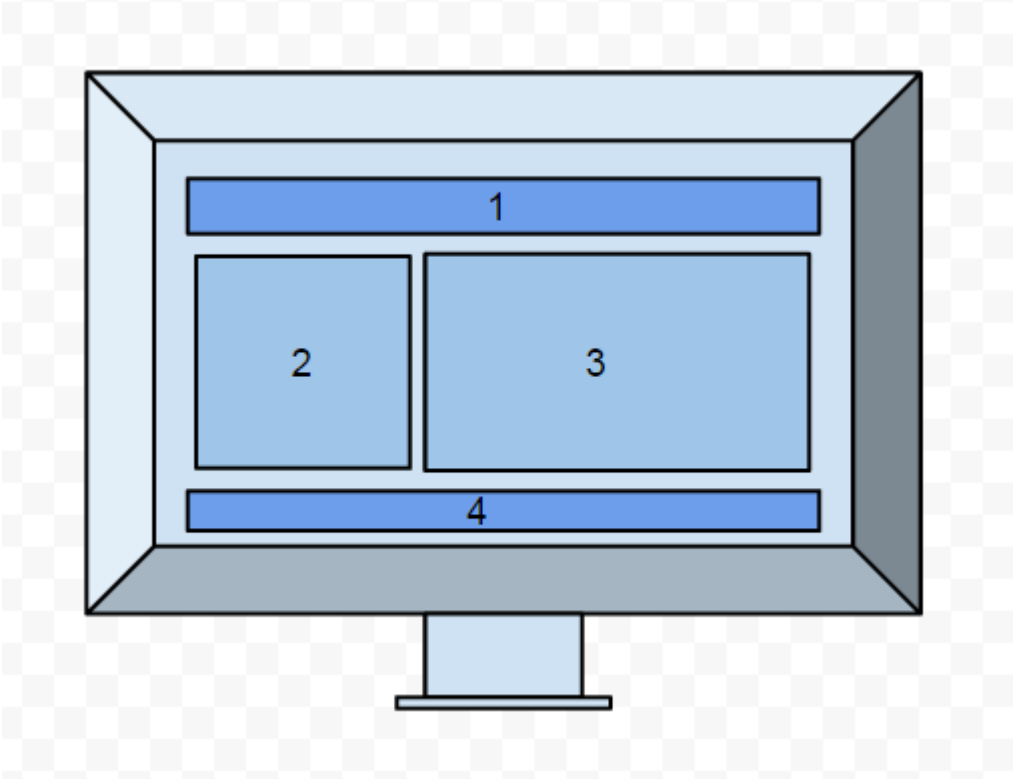

Figure 13.Sketch diagram for desktop layout.

The desktop browser's screen varies from 1024 to 1200 pixels in average. This range of screen resolution works for most laptops and notebooks. Designing the target resolution will be appropriate for all wide screen browsers. Figure 13 shows the layout primarily for a desktop browser with a higher than 1024 pixel screen resolution. Web page has well fitted web container with respective header, sidebar and container and footer which are represented with numbers 1, 2, 3 and 4 respectively. In the desktop design the web container has enough margin and padding on it.

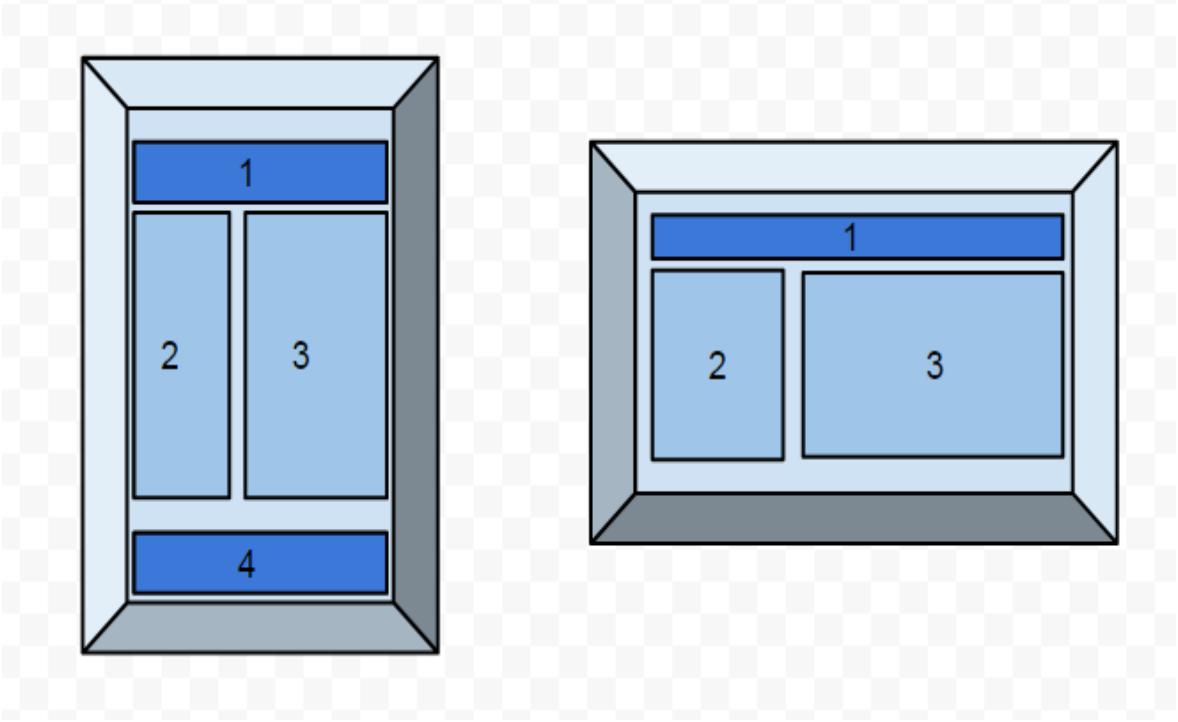

Figure 14 . Sketch diagram for a tablets layout in the portrait and landscape modes.

Tablets have a screen resolution that varies from 600 to 1024 pixels in average.
At present there are small tablets, 10-inch tablets from various vendors such as Apple, Google, and Microsoft. In figure 14, the sketch diagram for a tablet layout was designed. In the portrait mode the screen view ports support the entire layout by adjusting a dynamic web page container. Similarly this design is not supported in the landscape mode, so a separate design for the landscape mode was prepared which contained only container 1, 2 and 3 while the page loads. Upon scrolling down the user can see the fourth container.

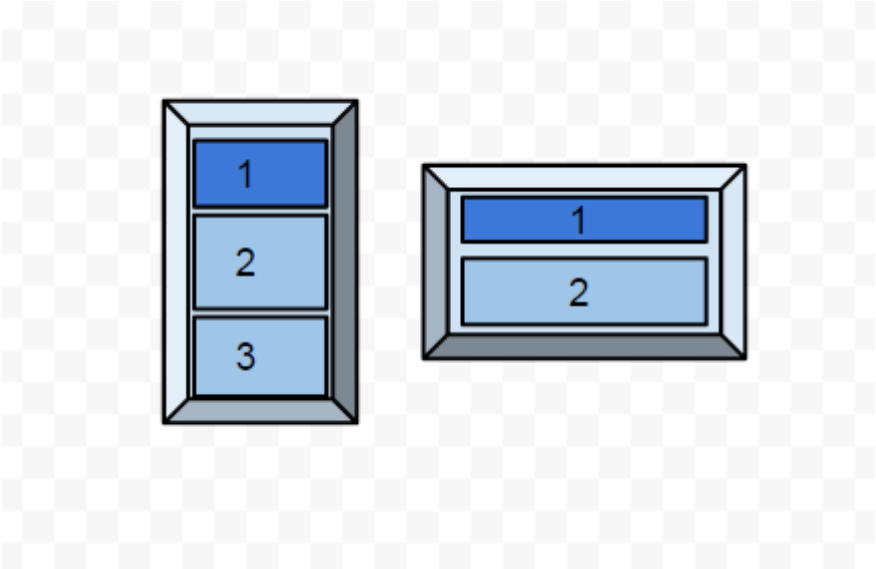

Figure 15. Sketch diagram for mobile layout in portrait and landscape modes.

In figure 15, the mobile sketch layout was designed. The screen widths of the mobile are nearly 80 percent smaller than the desktop width. The mobile screen resolution starts from 320 in the portrait mode to 480 in the landscape mode. The container of the mobile layout cannot fit the same layout as shown in the tablet and desktop layouts. So that the mobile design of the web page would fit in a single column. A single column layout that best fits the mobile screen was designed and containers 1, 2, 3 and 4 were aligning in a vertical order. By removing padding and margin and adjusting the entire container layout it is possible to have high user interface design for mobile users. Aligning these containers is based on the priority of a container. For example if a page has an important content in container 2 instead of container 1, then it is good practice to align in the top of the container stack.

4.3 Implementation

Once the layout and designed are finalized in the sketch which gives the basic idea of how the web page renders in different viewports, following the mobile first principle the first web page was designed for mobile using Dreamweaver CS5. Web development software, used for creating static and dynamic web applications and web pages. The created product was installed on a desktop local server and tested with latest browsers. The implementation procedures are briefly described below.

After the installation of Dreamweaver web platform, the new HTML5 project was created.

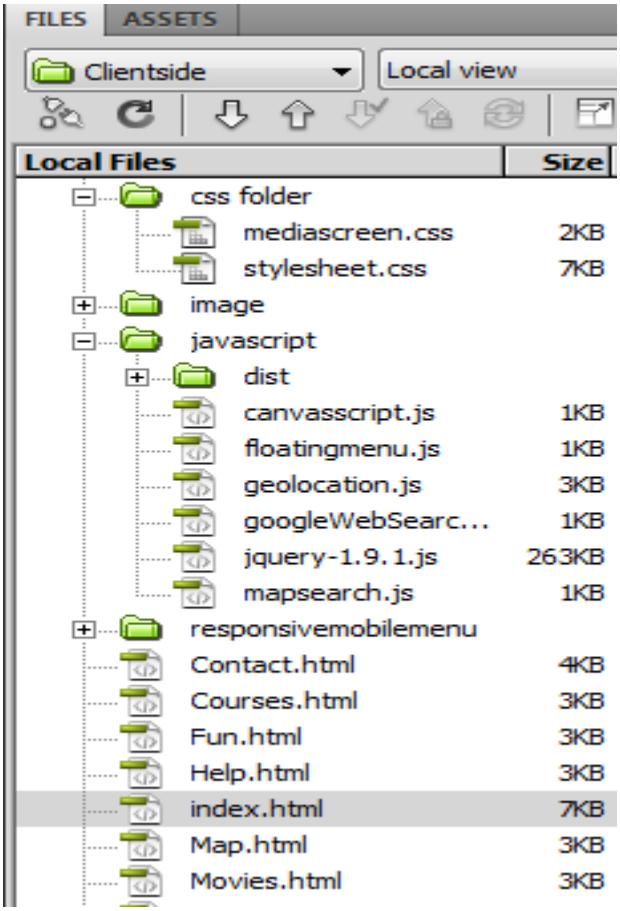

Figure 15. File structure of the prototype web page.

Figure 15 shows the folder structure of the website. The Clientside is the name of the folder that represents the `public_html` folder where a website's index page is normally stored. It contains the entire required folder for storing CSS, JavaScript and HTML5 files, which is good practice because it makes it easy to navigate between files and locate the files when needed. The actual implementation of the requirement was started after creating this folder structure. Necessary files for each type of document were placed in their respective folders. The prototype was designed for adaptive and responsive web layout. For this reason flexible grid, flexible media, and media queries were implemented.

For the flexible grid, the width of the container was calculated from the context formula.

```
#wrapper aside {
margin:1%;
float:left;
width:40%;
box-shadow:0px 0px 3px 1px #888;
-webkit-border-radius: 10px;
-moz-border-radius: 10px;
border-radius: 10px;
padding:2%;
min-height:10px;

}

#wrapper section{
margin:1%;
float:right;
width:56%;
box-shadow:0px 0px 3px 1px #888;
-webkit-border-radius: 10px;
-moz-border-radius: 10px;
border-radius: 10px;
padding:2%;
min-height:100px;

}
```

Listing 3. Source code for flexible grid: section and aside container.

As illustrated in listing 3, the web page has main two containers excluding the header and footer. These two containers width was defined in a percentage value so that the containers were flexible when the width of browser narrowed down. The width of both containers is 96 percent of the viewports width. The 4 percent of width was used for the margin of the container.

Similarly, the page has two embedded videos in the aside container. For the demonstration purpose the video container was used for flexible media. For the flexible media, the video container defines as maximum width 100 percent.

Finally, to achieve real adaptive and responsive design the CSS media queries were implemented inside the mediascreen.css file. The file has several breakpoint conditions to achieve different layouts for appropriate viewports. This is a simple prototype, so two breakpoint conditions were used. The breakpoint condition can vary according to the list of the supported viewport. If none of these conditions are matched then a default CSS would be applied for the web page. Those conditions that are not defined in this breakpoint are default.

```
@media only screen and (min-width :320px) and (max-width:600px){
          // CSS style
}

@media only screen and (min-width: 600px) and (max-width:
1024px){
// CSS style

}
```

Listing 4. Media queries breakpoint condition.

For example as shown in listing 4, if the media type is screen and the minimum width is 320 pixels to maximum width is 600 pixels, which is the range of a mobile and small tablet's viewport, the first condition will be true and all the CSS properties defined inside these conditions will be applied.

The mobile menu, which is usually represented by three line menus, is a standard way to tell a user that there are more items upon clicking the menu button. A third-party library was included to render this menu in mobile devices because of a small viewport in mobile devices. This represents the content first principle.

4.4 Result

The final result was obtained after successful implementation of the entire requirements. The web page was tested including the general testing and cross browser compatibility. For the general testing of the web page, the functionality of web pages was checked that it would works flawlessly. Also assuring that it would work in the entire latest browser such as Google Chrome, Mozilla Firefox and Microsoft Internet .In addition, for supporting legacy browsers extra code was added to the HTML5 document. This was necessary for backward compatibility of the webpage. During testing phases several bugs were noticed which were fixed later on. The final result for different view ports is described below.

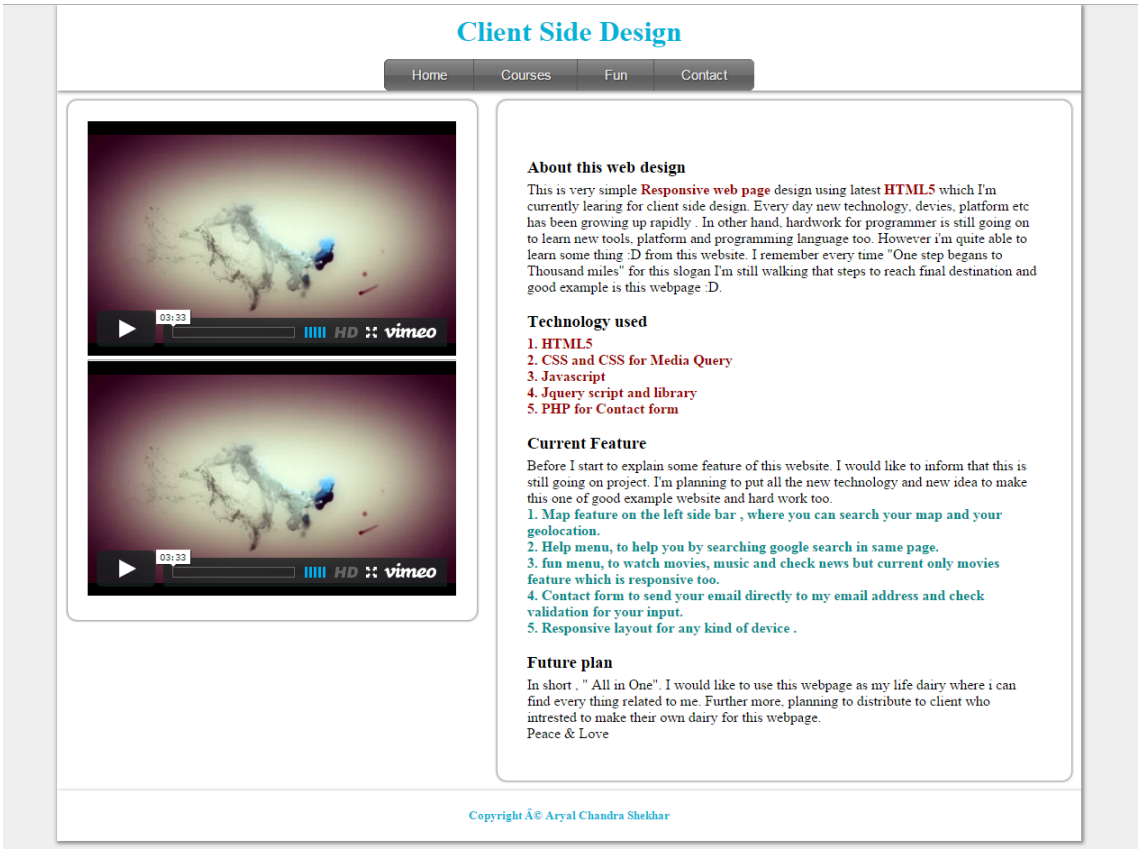

Figure: 16. Prototype web page shown in desktop browser.

Figure 16 is the final web page prototype for a desktop browser with a screen resolution higher than 1024 pixels. The web page has a simple design with two containers, and a simple menu on the top. Left side container contains two responsive videos. Videos are adaptive and responsive regardless of devices screen size. Similarly, right side

container contains the simple text. The web page follows the responsive design principles for high user interface and superior user experience.

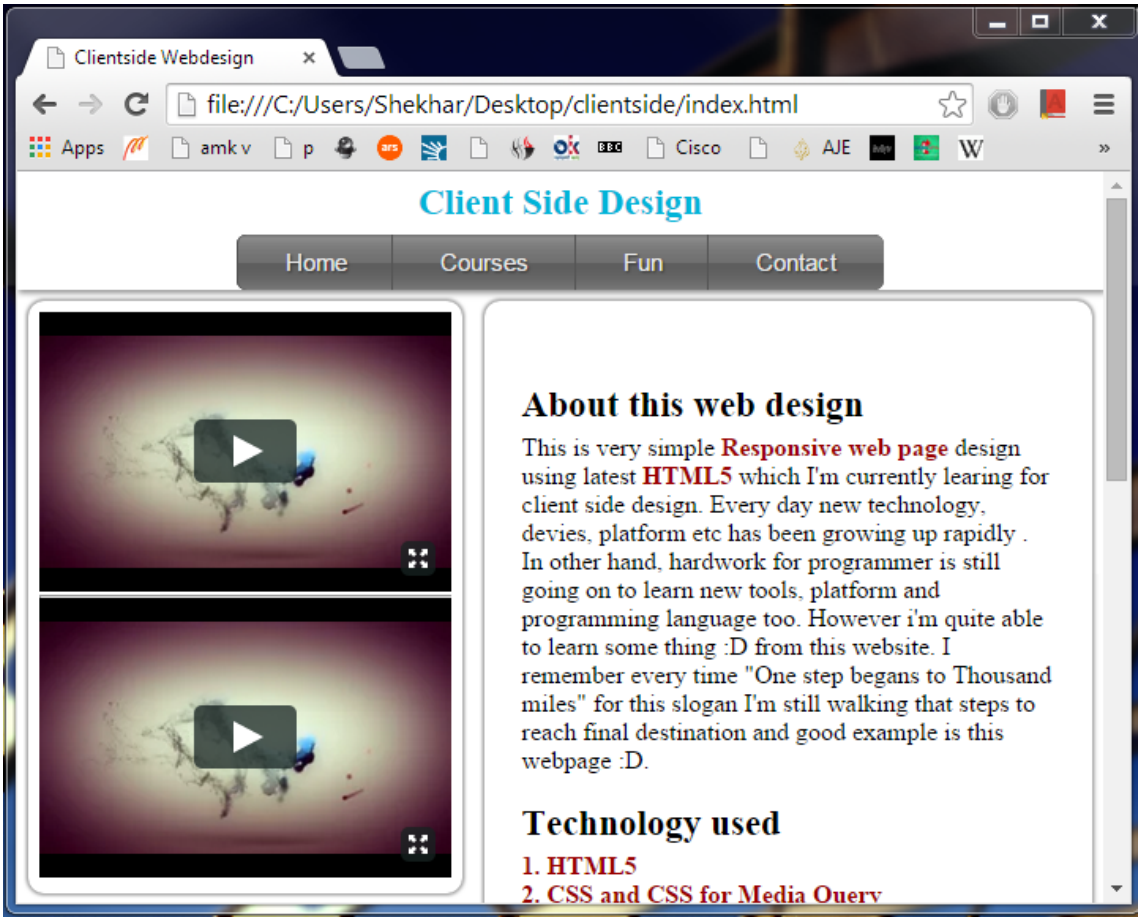

Figure 17.Web page in tablet web browser.

Figure 17 shows the tablet view of the web page. The web is designed for devices with more than 600 pixels to 1024 pixels. There can be various tablets available in the market. The design is also adaptive and responsive by implementing the required condition of the media queries CSS for this layout.

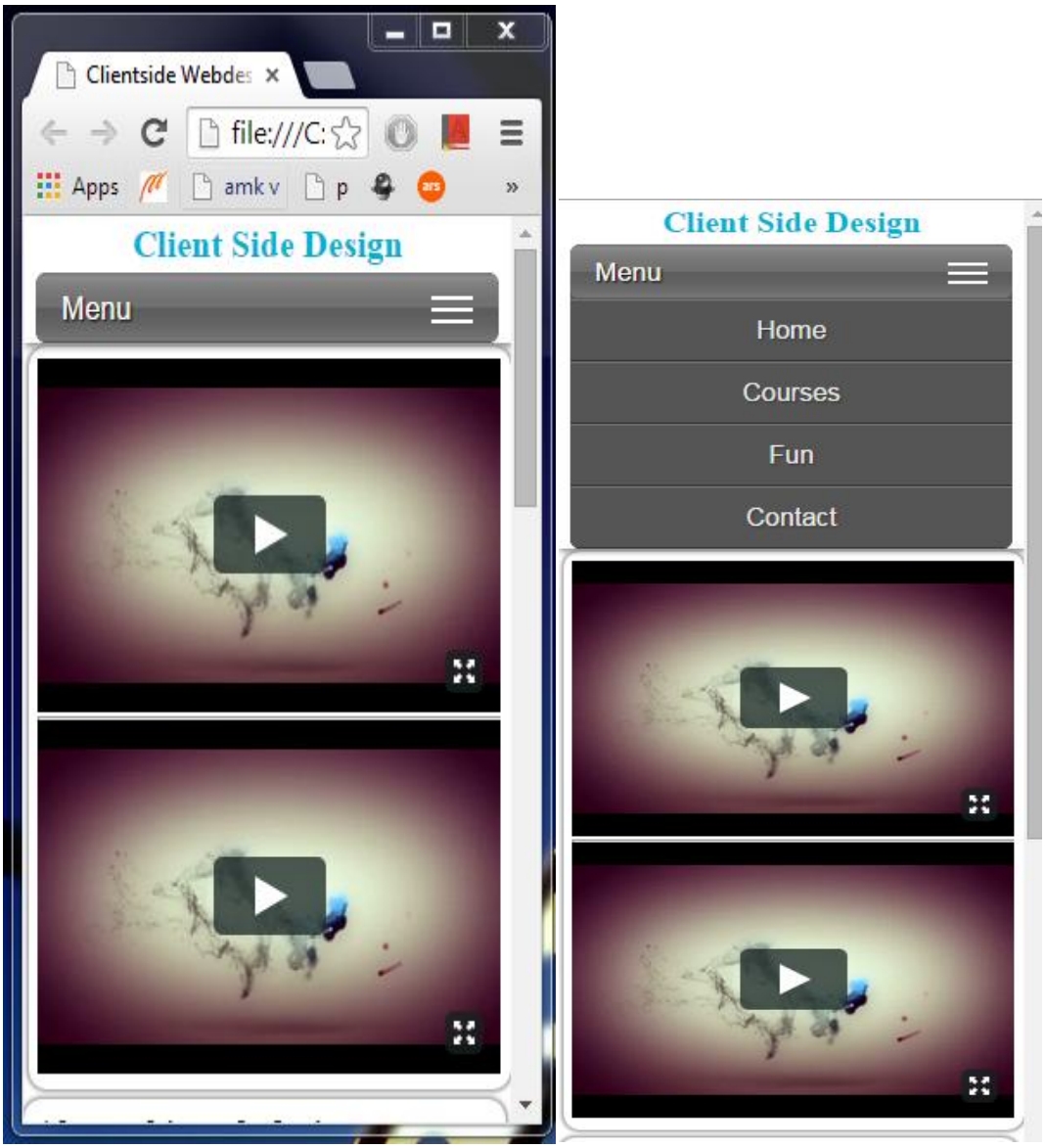

Figure 18.Web page in Mobile web browsers.

Figure 18 shows that the design is adaptive for mobile devices regardless of various viewports. In the mobile design all the containers are in a single column because the screen sizes of devices are small in comparison to desktops. The three line menus represent the menu bar for easy navigation. The menu items are hidden by default for mobile viewports and extended once clicked or touched.

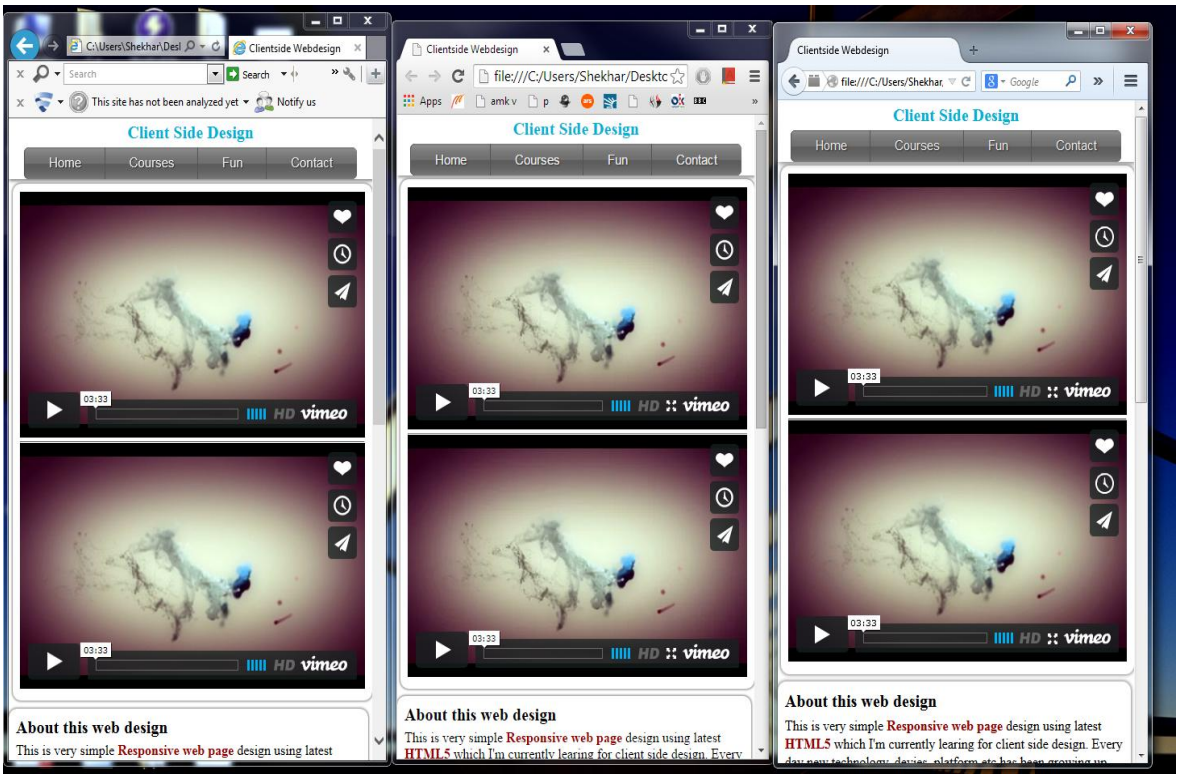

Figure 19. Web page tested for browser compatibility.

The web page was tested for the latest version of browsers Google Chrome version 38, Mozilla Firefox version 33, Internet Explorer version 11. As shown in figure 19 the prototype's design, CSS properties, JavaScript and HTML5 are supported across all the modern browsers in use.

5 Discussion

5.1 Benefits

Browsing web pages through smartphones has been on the rise with the increase of smartphones as discussed above in chapter 3.The main benefit of responsive web design is to provide web page for different devices that render without affecting user experience and content. The web content can be rendered flawlessly without zooming and with greater user experiences. In this way it is possible to reach a high number of users.

Flexibility to adaptation reduces the workload for creating different versions of websites. This eliminates the needs to create web solutions for particular devices. The cost for development also reduces automatically in the long run. There will be no extra effort for maintaining the website, since it can be done from a single file. Also it is easy to update, maintain the content. It also helps to save time and avoid an extra threshold for developing many versions of websites. Besides these benefits, it provides brand consistency. The web identity will be consistent across all the screen sizes which will helps the user to recognize the website. Also the web URLs will be the same. It is easy to remember for the user and helps the developer to optimize the content from one link.

5.2 Challenges

The responsive web page design is a principle that is tedious to implement in the beginning because it is still in a nascent stage. Designing the desktop version of the website is much less time-consuming but maintaining several versions for desktops and mobile versions proves to be expensive and complicated in the long run. In comparison to a responsive approach it is easy to implement a desktop version because it takes less resources and designing is faster and saves time. This advantage comes with the price of implementing different versions for different devices.

The users of each browser have specific needs and goals. In the desktop version of the webpage, there are not many constraints to provide all the required content as needed even with advertisement banners. However in the mobile or tablet devices which has

small screen and less content to fit. It is not possible to fit all the content. It is necessary to know the required goals and prioritize the content first principle.

The hardware capabilities of devices have many benefits to users. The interaction which works in one device may be irrelevant on another. In desktops and laptops, the interaction can be done with a mouse and a keyboard. For example the mouse pointer can be used to navigate through website links. Similarly by using keyboard short cut key, it is easy to navigate or interact with web page. However, this interactivity cannot be applied for finger touch screen devices.

The browser compatibility will be another challenge for a developer. In the webpage market there have been many available browsers with many versions. There are still users who stick to one browser without updating a newer one. The older versions of browsers does not support the latest web technology such as CSS, Media queries, HTML5 or even JavaScript. The responsive design may not work for older browsers. Each platform has its own browser which is unique to it. For example Safari is used as default browser in iOS devices while Internet Explorer is used for Windows Phones. The capabilities in Safari browsers can differ from the Internet Explorer. This means certain features might not be supported by all browsers. This should be taken into consideration while developing a responsive webpage. Customization is often a solution to deal with different browser capabilities, which will help to customize a webpage based on the browser used.

6 Conclusion

The thesis project was initiated as a study on how to design a responsive web page, what the main elements in responsive designs are, what the design paradigms are for the latest context and what the challenges and solutions for web pages are. As a result, the goal was achieved and the web page prototype was designed to test the output result of the study.

The Responsive web design principle is one of the new trends for the World Wide Web. It has many benefits and opportunities to provide appropriate solutions to present-day web pages. The adaptability of web pages to various sizes of devices with a respective screen resolution would have benefits such as easy maintenance, low cost and less work load. However this approach has also many challenges. Designing the web pages from mobile devices to big screen desktop computers creates problems for developers. The content strategy, mobile first design, interactivity of various devices in the market and performance of web pages can lead to poor user experience. Considering this fact beforehand and applying the best practices as a solution will provide better user experience.

Responsive design is an emerging field that is growing rapidly and it requires study for a better solution of responsive design principles. As a result of this study, the overall outcome was positive. The prototype was created from scratch for learning purposes however it can be implemented using a CSS framework such as Bootstrap, Foundation 3, Skeleton or 960 gird systems. The use of CSS framework can help fast development by reusing the code provided for the common structure. The simple approach of designing the solution in the initial phase can open the path of complex and better solutions of the future. Following the trends and using a new technology will always create new opportunities and challenges.

References

1 Greg Sterling Report. 60 Percent of Internet Access Is Mostly [online]. Mobile Marketing Land; 19 February 2014. URL:http://marketingland.com/outside-us-60-percent-internet-access-mostly-mobile-74498. Accessed 15July 2014.

2 Ethan Marcotte. Responsive Web Design [online]. A List Apart; 25 May 2010. URL:http://alistapart.com/article/responsive-web-design. Accessed 17 July 2014.

3 Lisa Morton. Responsive Web Design- Why it Matters, Why You Need it [online].DDSHOSTING, Website Design News; 21August2014. URL:http://www.ddshosting.com/responsive-web-design-matters-need/. Accessed 26 August2014.

4 Ethan Marcotte. Responsive Web Design. New York,U.S.A , A Book Apart; 2011.

5 Media Queries, W3CRecommendation [online]. W3C; 19June 2012. URL:http://www.w3.org/TR/css3-mediaqueries/. Accessed 5August2014.

6 HTML5, W3C Recommendation [online]. W3C; 28 October 2014 URL: http://www.w3.org/TR/html5/. Accessed 2November 2014.

7 HTML5 Differences from HTML4, W3C Recommendation [online].W3C; 18th September 2014. URL: http://www.w3.org/TR/html5-diff/. Accessed 20September 2014.

8 Selector Level 3, W3C Recommendation [online].W3C; 29 September 2011. URL: http://www.w3.org/TR/css3-selectors/Accessed 25September, 2014.

9 CSS Syntax,CSS [online].W3schools; 1999-2014. URL: http://www.w3schools.com/css/css_syntax.asp. Accessed 27September 2014.

10 Simon Willison. A re-introduction to JavaScript [online].Mozilla Developer Network;October 2014. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript. Accessed 2 October 2014.

11 Luke Wroblewski.MobileFirst. New York, U.S.A, A Book Apart;2011

12 Internet usage comparison [online]. StatCounterGlobalStats; 8October 2014. URL: http://gs.statcounter.com/#all-comparison-ww-monthly-201308-201408. Accessed 8October 2014.

13 Mobile device market to reach 2.6 billion units by 2016 [online]. Canalys; 22 February 2013. URL: http://www.canalys.com/newsroom/mobile-device-market-reach-26-billion-units-2016. Accessed 8October 2014

14 Stephen Hay. There is no Mobile Web [online]. The Haystack; 7January 2011. URL: http://www.the-haystack.com/2011/01/07/there-is-no-mobile-web/. Accessed 10 October, 2014

15 Rachel Lovinger. Content Strategy:The Philosophy of Data [online]. Boxes and Arrows; 27March 2007.
URL: http://boxesandarrows.com/content-strategy-the-philosophy-of-data/. Accessed 11 October 2014

16 Jake Brutlag. Speed Matters [online]. Google Research Blog; 23 June 2009. URL: http://googleresearch.blogspot.fi/2009/06/speed-matters.html .Accessed 12 October, 2014

17 Steve Sounders. High Performance Web Sites. First Edition, O'Reilly Media,Inc.; September 2007.

18 URLs Analyzed, HTTP Archive-Trends [online].
URL:http://httparchive.org/trends.php?s=All&minlabel=Nov+1+2012&maxlabel=Nov+1+2014#numurls. Accessed 2 November 2014.

19 IlyaGrigorik. Optimizing Content Efficiency, Optimizing Performance [online].Google Developers.
URL: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/. Accessed 14October 2014.

20 Pete LePage. Images in Markup, Web Fundamentals [online].Google Developers; 30 September, 2014.
URL: https://developers.google.com/web/fundamentals/media/images/images-in-markup. Accessed 16 October 2014.

21 IlyaGrigorik. Adding Interactivity with JavaScript, Critical Renderign path [online]. Web Fundamentals [online].Google Developers; 18 September2014.
URL: https://developers.google.com/web/fundamentals/media/images/images-in-markup. Accessed 18 October 2014.

22 Kalid Azad, How to Optimize Your Site with HTTP Caching [online].
URL: http://betterexplained.com/articles/how-to-optimize-your-site-with-http-caching/. Accessed 20 October 2014.

23 Increase Application Performance with HTTP Cache Headers [online].Herokudevcentre, 18 September 2014.
URL: https://devcenter.heroku.com/articles/increasing-application-performance-with-http-cache-headers. Accessed 22October 2014.

Appendix 1. CSS style sheet used for presentation of web page

```
/* CSS style sheet*/
* {
          padding: 0;
          margin: 0;
          -webkit-box-sizing: border-box;
          -moz-box-sizing: border-box;
          box-sizing: border-box;
}
body {
          background-color:#EFEFEF;
          margin:0;
          padding:0;
}
#wrapper {
          margin:0 5% 1% 5%;
          background-color:#FFF;
          box-shadow: 1.5px 1.5px 5px 1.8px #888;
          overflow:hidden;
}
header {
          width:100%;
          height:auto;
          overflow:hidden;
          box-shadow: 1.5px 1.5px 5px 0.8px #888;
}
header h1 {
          text-align: center;
          color:#00B4D8;
          padding:1%;
}
header nav {
          overflow:hidden;
          float:right;
          padding:0.7%;
          width:40%;
          background:#444;
          margin-right:5%;
          margin-bottom:2%;
          margin-right:5%;
          -webkit-border-radius: 12px;
          -moz-border-radius: 12px;
          border-radius: 12px;
}
header nav ul {
          list-style: none;
          overflow:hidden;
}
header nav li a {
          float:left;
          color:#00B4D8;
```

```
          display:inline-block;
          font:  400  17px/1.4  'Cutive',  Helvetica,  Verdana,
Arial, sans-serif;
          padding: 3px;
          text-align: center;
          text-decoration: none;
          width: 25%;
          -webkit-border-radius: 15px;
          -moz-border-radius: 15px;
          border-radius: 15px;
          /*TRANSISTIONS of hover effect*/
          -webkit-transition: background 0.5s ease;
          -moz-transition: background 0.5s ease;
          -o-transition: background 0.5s ease;
          -ms-transition: background 0.5s ease;
          transition: background 0.5s ease;
}
/*HOVER*/
header nav li a:hover {
          background:#EFEFEF;
}
/*BORDER FIX*/


/*
.fixed and .default for flauting the navigation bar
*/
.fixed {
          position: fixed;
          top: -5px;
}
.default {
          overflow:hidden;
          float:right;
          padding:0.7%;
          width:60%;
          background:#444;
          margin-right:5%;
          margin-bottom:2%;
          margin-right:5%;
          -webkit-border-radius: 12px;
          -moz-border-radius: 12px;
          border-radius: 12px;
}
#wrapper aside {
          margin:1%;
          float:left;
          width:40%;
          box-shadow:0px 0px 3px 1px #888;
          -webkit-border-radius: 10px;
          -moz-border-radius: 10px;
          border-radius: 10px;
          padding:2%;
```

```
          min-height:10px;
}
#wrapper aside nav ul {
          list-style: none;
}
#wrapper aside nav ul li a {
          color:#0076AE;
          display:inline-block;
          font:  600  15px/1.4  'Cutive',  Helvetica,  Verdana,
Arial, sans-serif;
          padding:3%;
          text-decoration: none;
          width:80%;
          -webkit-border-radius: 15px;
          -moz-border-radius: 15px;
          border-radius: 15px;
          /*TRANSISTIONS of hover effect*/
          -webkit-transition: background 0.5s ease;
          -moz-transition: background 0.5s ease;
          -o-transition: background 0.5s ease;
          -ms-transition: background 0.5s ease;
          transition: background 0.5s ease;
}
#wrapper nav li a:hover {
          background:#EFEFEF;
}
#wrapper section {
          margin:1%;
          float:right;
          width:56%;
          box-shadow:0px 0px 3px 1px #888;
          -webkit-border-radius: 10px;
          -moz-border-radius: 10px;
          border-radius: 10px;
          padding:2%;
          min-height:100px;
}
#wrapper section article {
          padding:10px;
          margin:0 0 10px 0;
}
#wrapper section article h1 {
          margin-bottom:5px;
}
footer {
          clear:both;
          min-height: 20px;
          background-color: white;
          padding: 20px;
          box-shadow: 1.5px 1.5px 5px 0.8px #888;
          overflow:hidden;
          width:100%;
}
```

```
footer h5 {
          text-align:center;
          color:#00B4D8;
}
#geolocation {
          float:right;
}
/*#main article button,#msg,#lat,#long{
          float:right;

}*/

button, input[type="button"] {
          display: inline-block;
          outline: none;
          cursor: pointer;
          text-align: center;
          text-decoration: none;
          font: 13px/100% Arial, Helvetica, sans-serif;
          padding: .4em 2em .5em;
          text-shadow: 0 1px 1px rgba(0, 0, 0, .3);
          -webkit-border-radius: .5em;
          -moz-border-radius: .5em;
          border-radius: .5em;
          -webkit-box-shadow: 0 1px 2px rgba(0, 0, 0, .2);
          -moz-box-shadow: 0 1px 2px rgba(0, 0, 0, .2);
          box-shadow: 0 1px 2px rgba(0, 0, 0, .2);
}
button:hover {
          text-decoration: none;
}
button:active {
          position: relative;
          top: 1px;
}
input[type="text"] {
          display: inline-block;
          outline: none;
          cursor: pointer;
          text-decoration: none;
          text-shadow: 0 1px 1px rgba(0, 0, 0, .3);
          -webkit-border-radius: .5em;
          -moz-border-radius: .5em;
          border-radius: .5em;
          -webkit-box-shadow: 0 1px 2px rgba(0, 0, 0, .2);
          -moz-box-shadow: 0 1px 2px rgba(0, 0, 0, .2);
          box-shadow: 0 1px 2px rgba(0, 0, 0, .2);
}
input:focus {
          outline:none;
          border-color:#CACACA;
}
.video-wrapper {
```

```
          max-width: 100%;
}
.video-container {
          position: relative;
          padding-bottom: 56.25%;
          padding-top: 30px;
          height: 0;
          overflow: hidden;
          margin-bottom:1%;
}
.video-container   iframe,   .video-container   object,   .video-
container embed {
          position: absolute;
          top: 0;
          left: 0;
          width: 100%;
          height: 100%;
}
#maincontact article {
          float:left;
          width:50%;
}
#getintouch {
          padding:20px;
          border:1px solid #ccc;
          -moz-border-radius: 10px;
          -webkit-border-radius: 10px;
          -khtml-border-radius: 10px;
          border-radius: 10px;
}
#getintouch legend, h2 {
          font-family : Arial, sans-serif;
          font-size: 1.3em;
          font-weight:bold;
          color:#333;
}
#getintouch p {
          color:#525252
}
#getintouch h1 {
          color:#573143;
}
#technology {
          color:#900;
          font-weight:bold;
}
#feature {
          color:#008C8C;
          font-weight:bold;
}
```

Appendix 2. CSS media query used for media screen conditions

```
/* CSS MEDIA QUERIES*/
@media only screen and (min-width :0px) and (max-width:600px) {
 #wrapper {
 margin:0%;
}
 #wrapper h1 {
 font-size:20px;
}
 #wrapper aside {
 width:99%;
}
 #wrapper section {
 width:99%;
}
 #wrapper section article {
 padding:0px;
 margin:0px;
}
 footer h5 {
 font-size:12px;
}
}
 @media only screen and (min-width :600px) and (max-
width:1024px) {
 #wrapper {
 margin:0%;
}
 #wrapper h1 {
 font-size:23px;
}
 #wrapper aside {
 margin:1%;
 padding:1%;
}
}
```