



Responsive Web Design

Evaluation of Techniques to Optimize Load Time

Joel Nandorf

Student
Spring 2013
Bachelor Thesis, 15 hp
Computer Science

Abstract

Responsive Web Design has, in short time, become a common method to create websites that automatically adapt the layout to different screen sizes. However, critique has been raised about poor performance due to large page size and high number of requests to the web server. The aim of this study is to evaluate techniques to optimize the load time of a responsive website. This is done by creating a prototype in the form of a responsive website, which is used as a base for the optimization. Tests are performed by measuring the page size and the number of requests when using four different optimization techniques. The results show that combining optimization techniques can dramatically reduce the page size and the number of requests. Consequently, this has a positive impact on the load time of the website. Furthermore, the Mobile First approach is important in responsive web design as it prioritizes the use of mobile devices and as a result highlights the significance of web performance. It is also suggested to set a Performance Budget early in web development projects in order to avoid slow websites and spread awareness about the importance of performance.

Keywords: responsive web design, load time, optimization techniques, mobile first, performance budget

Table of Contents

1. Introduction	1
2. Background	2
2.1 The Core of Responsive Web Design.....	2
2.2 Page Load Time.....	3
2.3 Selected Techniques to Optimize Load Time	5
2.3.1 Image Optimizer.....	5
2.3.2 Responsive Images	5
2.3.3 Optimizing Javascript and CSS	5
2.3.4 Lazy Loading.....	6
3. Design of Prototype	6
3.1 Development using Mobile First.....	6
3.2 Creating a Responsive Prototype.....	6
4. Result of the Optimization.....	7
4.1 Image Optimizer.....	7
4.2 Responsive Images	8
4.3 Optimizing Javascript and CSS	8
4.4 Lazy Loading.....	9
4.5 Combination of Techniques.....	9
5. Discussion	10
6. References.....	13

1. Introduction

Throughout human history, innovations have emerged with existing knowledge about the world as a foundation. In a similar way, the web has been strongly influenced and shaped by the medium of printing (Allsopp, 2000). Conventions that were applicable to printing have been transferred to the web, for example the idea that the page has a fixed size and the belief that designers are in total control of the layout. As the landscape of the web is quickly changing, these ideas are nowadays being questioned.

The amount of mobile devices with web access is constantly increasing and screen sizes are becoming both smaller and larger at the same time (Marcotte, 2011). New smartphones and tablet computers are constantly entering the market and it is only a matter of years before devices with small screen sizes are estimated to surpass desktop computers in numbers. Many companies and organizations have responded to this development by creating mobile-tailored versions of their websites. However, the sheer amount of devices makes it difficult to design for all the available screen sizes. There is, therefore, a need for websites that are *adaptive* and *accessible* regardless of the device used.

One way to achieve this adaptability is called *Responsive Web Design (RWD)*, a term coined by Ethan Marcotte in his seminal article (2010). Ethan's contribution was to combine already existing techniques and to propose a new mindset that embraces the flexibility of the web instead of trying to achieve pixel-perfect control of the design. A responsive website can adapt itself and change the layout to fit the screen size of the device. Consequently, the same website can be viewed using any kind of device, from small smartphones to large TVs.

The introduction of RWD has drawn much interest among web developers from all over the world. The new method has been praised among many in the web design community as a viable approach to handle the increasing diversity of devices. However, it has also received criticism that needs to be addressed in to take fully advantage of the new method. Critique brought up is the issue with poor performance and long load times in responsive websites (Grigsby, 2010; Podjarny, 2012). The root of the problem is that the same content is downloaded on both desktop computers with fast broadband connections as well as on mobile phones with slower connections.

The load time of a website can noticeably affect the user experience. Almost 60 % of the web users expect a website to load on their mobile phones in 3 seconds or less and 74 % are only willing to wait 5 seconds or less before leaving the site (Gomez, 2011). This has important consequences for, among others, e-commerce sites, which face the risk of losing customers as well as affecting the reputation of the company negatively (Google, 2012b).

There is, therefore, a need to reconsider the importance of performance in web development. The purpose of this study is to evaluate different techniques to optimize the load time of a responsive website. This is done by creating a responsive prototype and using it as a base to test different optimization techniques. The prototype is

developed in cooperation with Sogeti Umeå and implemented in SharePoint 2013 (Microsoft, 2013). Quantitative methods are used to measure the optimization techniques and the metrics used are the number of requests and the page size in kilobyte. Finally, the results are discussed and the optimization techniques are evaluated.

2. Background

This part describes the background, including the core of RWD, factors influencing load time and techniques to optimize responsive websites.

2.1 The Core of Responsive Web Design

Responsive Web Design is built on a combination of three core parts: *flexible grid*, *flexible images* and *media queries* (Marcotte, 2011).

Flexible Grid

The first part is a flexible grid. Grid-based layout in web design is used to divide the layout into a specified number of columns. The original grid-based layouts were created using columns with fixed widths in pixels. However, in order to create a responsive website the columns need to be specified using a relative value, which is done by setting the width of a column as a percentage of the width of its containing element (Figure 1). This creates a grid that is flexible, which automatically resizes itself according to the size of the window.

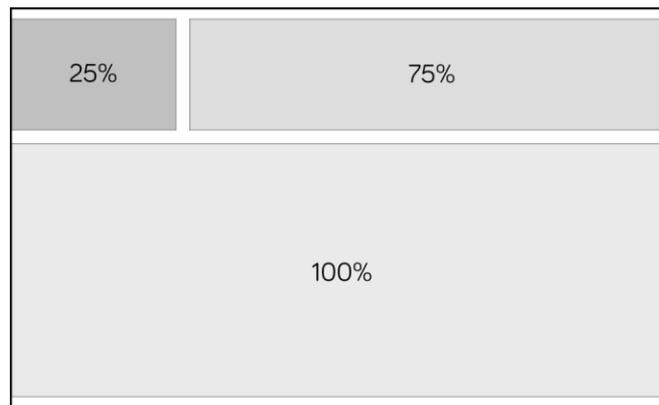


Figure 1. Illustration of a flexible grid. The widths of the columns are relative and resize depending on the size of the window.

Flexible Images

Flexible images allow the browser to scale images depending on the size of the browser. Consequently, the file size of an image is the same regardless of the device that is used. Furthermore, the image needs to contain enough pixels to be rendered on large screens with maintained quality. An alternative to scaling is cropping, which removes a part of

the image when the size of the container changes. This is useful if the image contains many details that need to be distinguishable on small screens. However, a negative aspect is that information is removed from the image, which could affect the perceived meaning of the picture.

Media Queries

Media queries allow the browser to use different styling depending on certain conditions. This functionality is used in RWD to modify the layout at specific widths in the browser, also called breakpoints. Marcotte (2011) has identified a set of breakpoints for different types of devices but these definitions have already become outdated as new devices are constantly emerging. Another approach is to let the content decide where to set the breakpoints, a method that is independent of specific resolutions (Kadlec, 2013a).

2.2 Page Load Time

The load time of a website is challenging to measure accurately due to changeable aspects in network performance, such as bandwidth, routes to the server and network congestion. There are, however, factors influencing the page load time that are possible to control.

Today's websites are complex ecosystems containing different types of content from several sources. Several factors influencing the load time of websites have been identified in a study by Butkiewicz et al. (2011). The top three factors found are, beginning with the most influential: 1) *the number of requests*, 2) *the number of objects that are Javascript* and 3) *the total page size*. Notable here is that the total page size is the third item in the list and that number of requests is deemed more important. The authors reason that since a typical website in the study is not that large the round-trip time to the server is more important than the number of bytes transferred.

The situation, however, is quickly changing since the total page size and the number of requests on an average website is constantly increasing (HTTP Archive, 2013). The average page size has increased dramatically by 88 % in merely two years and is now peaking at around 1.45 megabyte (Figure 3). The average number of requests has, at the same time, increased by 14 % and is now 92 requests per page (Figure 4). The consequences of this are that an average website takes a longer time to download due to larger page size, but also the increased number of requests will result in longer load times. This is especially the case for mobile networks where latency is a critical bottleneck (Grigorik, 2012). Latency is an expression of how much time it takes for a data packet to travel from the web browser to the server and back. Therefore, higher latency will result in a longer page load time. Also, the higher latency the more impact will the number of requests have on the load time.

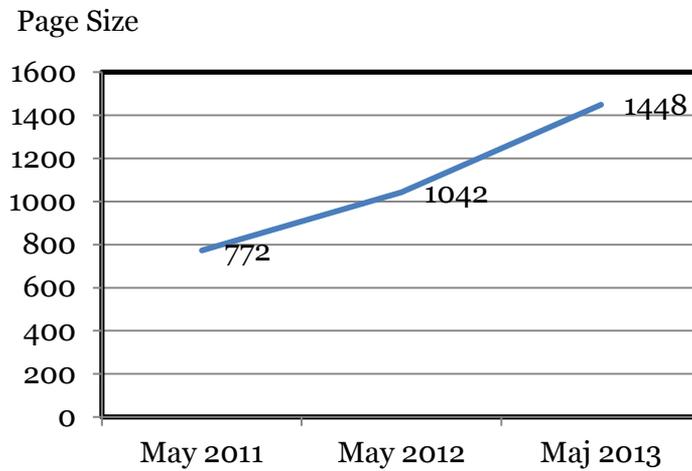


Figure 3. The total page size of an average website has increased by 88 % between 2011 and 2013.

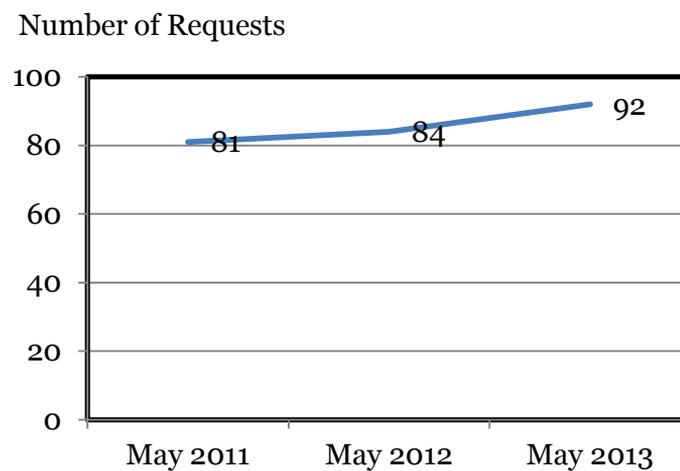


Figure 4. The number of requests on an average website has increased by 14 % between 2011 and 2013.

Other descriptions of factors affecting load time come from the IT industry. Google (2012a) have published a list of recommendations for making the web faster and better adapted for mobile devices. They conclude that in order to avoid a dramatic increase in load time it is important to reduce the number of requests to the server. Yahoo (2013a) have also compiled best practices for speeding up websites and they too highlight the reduction of requests to the web server as a key factor to optimize load time.

The load time affects the user's experience when browsing the web. It has been shown that users are willing to wait for only about 2 seconds for simple information retrieval tasks on the web (Nah, 2004). Within these 2 seconds, a response is needed to ensure a smooth interaction between the user and the web. Furthermore, the availability of feedback has been found to prolong the tolerable waiting time among the users. More recently, a survey from the IT-industry on mobile user's expectations (Gomez, 2011) reports that nearly 60 % of web users expect a website to load on a mobile phone in 3 seconds or less and that 74 % are only willing to wait 5 seconds or

less before leaving the site. Moreover, nearly half of the mobile web users are unlikely to return to a website that they had trouble accessing. For these reasons, it is essential to optimize the load time to ensure a good user experience.

2.3 Selected Techniques to Optimize Load Time

There are many techniques to optimize performance in responsive websites and a selection has therefore been done. The criteria used for the selection are based on the most influential factors affecting load time that was previously found: *the number of requests*, *the number of Javascript-files* and *the total page size*.

2.3.1 Image Optimizer

Images account for the majority of the total size of an average web page (HTTP Archive, 2013) and techniques to minimize them are therefore essential in order to optimize load time. Images often contain additional information, which is unnecessary and make the file size larger. The images can be run through an optimizer, for example Kraken Image Optimizer (2013) or Smush.it from Yahoo (2013b), in order to remove redundant information without affecting the quality of the image.

2.3.2 Responsive Images

Images in RWD need to be able to adapt to the size of the device. The method proposed by Ethan Marcotte (2011) was to let the browser automatically resize the original image, but this will result in a situation where the same image is used regardless of the type of device. Consequently, the page size will be the same on a mobile phone as well as on a desktop computer. There is a need for truly responsive images, which are served in different sizes depending on the width of the window. Many solutions have been created by enthusiasts in order to provide this functionality, see Grigsby (2011) for a compilation of available solutions. Since the prototype is implemented in SharePoint 2013, a functionality called *Image Renditions* (Microsoft, 2012) will be used together with a plugin called *Responsive Image Renditions* (Mavention, 2013). The result is a solution where the browser creates a cookie with information about the current size of the window. This information is then used by Responsive Image Rendition to serve an image with the appropriate size to the user.

2.3.3 Optimizing Javascript and CSS

Javascript and CSS-files contribute to both an increased page size and number of requests and optimizing them are therefore essential in order to reduce the load time. Two methods can be used to achieve this; *minification* to remove unnecessary information such as white space, line breaks and indentation and *concatenation* to combine several files (Olsen, 2013). Minification makes the file size smaller and can be done using, for example, YUI Compressor, which is an online tool provided by Yahoo (2013c). Concatenation, on the other hand, reduces the number of requests and YUI Compressor can too be used for this purpose.

2.3.4 Lazy Loading

Lazy Loading is originally a design pattern in computer programming that defers loading of content until it is needed. The technique can be used in web development to load any type of content, for example images, html-documents and scripts. The result is a reduction of both page size and number of requests. The method has been adapted for RWD by loading certain content depending on the screen size (Keith, 2011). Another usage is to delay loading of images outside the visible part of the screen until the user scrolls down and they are downloaded (Tuupola, 2013).

3. Design of Prototype

The following part describes the methods that were used in the design of the prototype.

3.1 Development using Mobile First

Mobile First is a philosophy in web development created by Luke Wroblewski (2009), which highlights the need to prioritize the use of mobile devices. By using Mobile First the designer is forced to focus and prioritize the most important content by considering the constraints in mobile design, for example limited screen size and bandwidth. This practice was followed when the prototype was created by designing for mobile devices first and then adapting the design for tablet and desktop computers.

3.2 Creating a Responsive Prototype

The prototype is a responsive version of an existing website, belonging to a company in the forest industry. The website was selected by Sogeti Umeå with the purpose of studying how SharePoint 2013 (Microsoft, 2013) can be used for implementing responsive websites. The Bootstrap framework (Twitter, 2013) was used in order to simplify the process of designing the prototype. Figure 5 shows the wireframe of the prototype viewed on a large screen. The layout is using the available space to show images in full size. The menu at the top is also using the width of the page to present the menu items in full size. Figure 6 demonstrates the wireframe of the prototype viewed on a small screen, for example a mobile phone. The content is stacked on top of each other to avoid scrolling horizontally. The menu is initially hidden but is shown when the button on the top right corner is pressed.

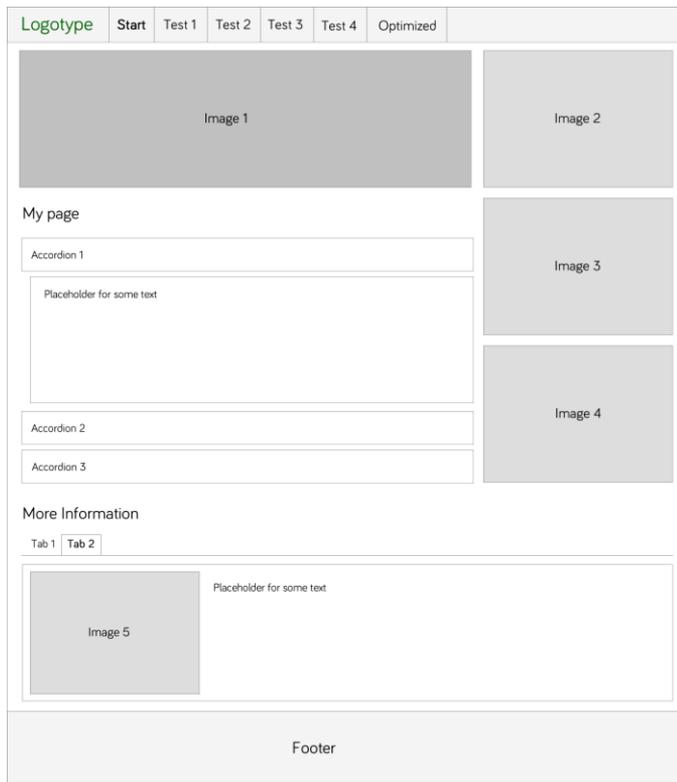


Figure 5: Wireframe viewed on a large screen.

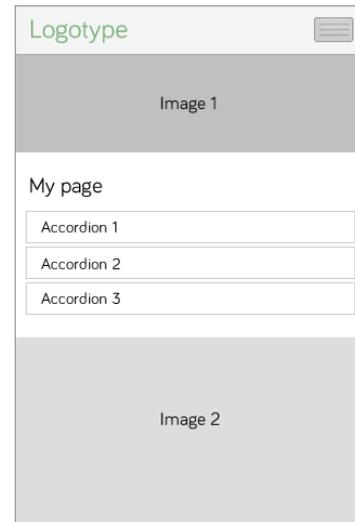


Figure 6. Wireframe viewed on a small screen.

4. Result of the Optimization

This part presents the result of the optimization. Five versions of the prototype were created in order to test different optimization techniques. The start page is used as a base for the testing, with a total page size of 1082 kb and 27 requests. The tool for measuring is Firebug (2013) and the metrics are the total page size and the number of requests.

4.1 Image Optimizer

The file size of the images is minimized using Kraken Image Optimizer (2013). Five images in the file format JPG are uploaded to the website and downloaded after the optimization. The results in Table 1 demonstrate a reduction of the page size by 194 kb and the optimization is performed without any visible degradation of the image quality.

Table 1: Results of the optimization using an image optimizer.

Technique	Requests difference	Req. diff. percentage	Page size difference	Page size diff. percentage
Minimizing Images Using an Image Optimizer	0	0 %	-194 kb	-18 %

4.2 Responsive Images

Image Rendition in SharePoint is used together with the Responsive Image Rendition plug-in in order to deliver different versions of the same image. Tests are done using the predefined breakpoints in Bootstrap. The results in Table 2 show a reduction of the page size in the range between 553 kb (phone) and 280 kb (large display). Although the page size is reduced, the number of requests increases at the same time in the range between 6 (phone) and 0 (large display). The reason behind this phenomenon is found by analyzing the network traffic between the browser and the web server. When the browser requests an image from the server a HTTP 302 error message is sent back, which tells the browser that the file has been moved temporarily and can be found on a new URL, that is included in the message. The Responsive Image Rendition plugin is accountable for this behavior, which increases the download time by causing an extra round trip to the web server for each image.

Table 2: Results of the optimization using responsive images.

Technique	Requests difference	Req. diff. percentage	Page size difference	Page size diff. percentage
Responsive Images				
Phones < 480 pixels	+6	+22.2 %	-553 kb	-48.9 %
Phones to portrait tablets 481 - 767 pixels	+2	+7.4 %	-316 kb	-29.2 %
Landscape tablets 768 - 979 pixels	+2	+7.4 %	-298 kb	-27.5 %
Default 980 - 1199 pixels	+1	+3.7 %	-310 kb	-28.7 %
Large display > 1200 pixels	0	0 %	-280 kb	-25.9 %

4.3 Optimizing Javascript and CSS

Table 3 demonstrates the results of optimizing Javascript and CSS-files. SharePoint automatically adds Javascript and CSS-files that are used for the top menu (Figure 7), which is visible when a user has logged into the system. The files added increase the page size by 203.5 kb. When a user visits the page without logging in, the menu is not present and the files are therefore not necessary to include. However, it is difficult to remove the files due to dependency issues in SharePoint and as a result, only some of the files could be excluded. The removal of the files results in a reduction of the number of requests by 4 and the page size by 125 kb.



Figure 7: The top menu is automatically inserted by SharePoint.

Another method to optimize Javascript and CSS is to remove unnecessary components in the Bootstrap framework and to use minified versions of the scripts. This reduces the page weight by 10 kb but the number of requests is still the same.

Table 3: Results of optimizing Javascript and CSS.

Technique	Requests difference	Req. diff. percentage	Page size difference	Page size diff. percentage
Optimizing Javascript and CSS				
Remove files	-4	-14.8 %	-125 kb	-11.6 %
Optimize Bootstrap-files	0	0 %	-10 kb	-0.9 %

4.4 Lazy Loading

Lazy Loading is used to prevent the browser from downloading the whole page at once and instead partially load data when requested by the user. The method is used to load content in the *More Information* section of the prototype (Figure 5). The results in Table 4 show a reduction of the number of requests by 2 and the page size by 132 kb. Two images account for the reduction of requests and the majority of the saved page size.

Table 4: Results of the optimization using Lazy Loading.

Technique	Requests difference	Req. diff. percentage	Page size difference	Page size diff. percentage
Lazy Loading	-2	-7.4 %	-132 kb	-12.2 %

4.5 Combination of Techniques

Table 5 shows the results of a combination of all optimization techniques. The total reduction of the page size varies between 699 kb (phone) and 440 kb (large display) and the number of requests between 1 (phone) and 6 (large display). The page size is reduced 41-65 % and the number of requests 4-22 %.

Table 5: Results of the optimization using a combination of techniques.

Technique	Requests difference	Req. diff. percentage	Page size difference	Page size diff. percentage
Combination of Techniques				
Phones < 480 pixels	-1	-3.7 %	-699 kb	-64.6 %
Phones to portrait tablets 481 - 767 pixels	-4	-14.8 %	-488 kb	-45.1 %
Landscape tablets 768 - 979 pixels	-4	-14.8 %	-468	-43.3 %
Default 980 – 1199 pixels	-5	-18.5 %	-481 kb	-44.4 %
Large display > 1200 pixels	-6	-22.2 %	-440 kb	-40.6 %

5. Discussion

The emergence of mobile browsing has profoundly changed the way websites are made today compared to only a few years ago. The web design community is starting to embrace the flexibility of the web and letting go of previous limitations. RWD has been successful as a method to create websites that adapt to different type of devices. However, the importance of web performance needs to be considered in order to create websites that are truly accessible, also on mobile devices with slow network connections.

The prototype was created with the Mobile First approach. The mindset supports thinking about the design from a mobile perspective and to consider constrains such as limited screen size and low bandwidth. With a Mobile First perspective, it is natural to think about web performance and therefore it is an essential addition to RWD.

Brad Frost (2013) makes the case that performance should be treated as an essential design feature. It should be as important to design for good performance, as it is to design for an attractive visual appearance. This idea is further developed by Tim Kadlec (2013b), as he suggests setting a *Performance Budget* early in a development project in order to avoid ending up with a slow website. The performance budget can be included in the requirement specifications, in the form of a specific load time or broken down into page size or number of requests. BBC News (Malsen, 2012) used this concept when they designed a responsive version of their website. They determined that each page should be usable within 10 seconds on a GPRS connection and they based the goals for page size and number of requests upon that. Throughout the project, the performance budget guided them in design decisions, for example whether they should add certain functionality or not.

The results of the optimization show that the use of responsive images is successful by reducing the page size by almost half of the original size when viewed on a phone. The decreased size will considerably reduce the load time but the progress is counteracted by the increased number of requests. Latency is a critical issue in mobile web performance and the addition of one roundtrip to the server per image adds unnecessary waiting time. The prototype is only delivering five images using this technique but on a real website, the number of requests is considerably higher. Consequently, this has negative consequences on the load time. The Responsive Image Rendition plug-in to SharePoint is accountable for this behavior by sending an error message instead of directly delivering the image and therefore it needs to be improved in order to avoid the drawbacks of duplicate requests. If this is adjusted, the plug-in is a useful addition to SharePoint, which enables the use of responsive images and facilitates the implementation of responsive websites on the platform.

There are other techniques for using responsive images, outside the SharePoint platform, which avoid the drawback of duplicate requests. *Adaptive Images* (Wilcox, 2013) is a server-side solution that delivers responsive images without the additional roundtrip to the server. Another alternative is *Picturefill* (Jehl, 2012), which is a client-

side solution that uses links to images of different sizes in the HTML-file and Javascript to identify the screen size of the browser and send a request for an image in the appropriate size. These techniques are workarounds while waiting for a method to handle responsive images natively in the web browsers. The Responsive Image Community Group at W3C (RICG, 2013) is currently working on a specification of a *picture* element. The new element is a markup-based solution that enables responsive images in the web browsers without the need of server-side or Javascript support.

Optimization of Javascript and CSS is an important technique since it has the benefits of both reducing the page size and the number of requests. Javascript-files are the second largest contributor to the size of an average page on the web and it is important to keep them as few and as small as possible (HTTP Archive, 2013). The implemented prototype contains several Javascript and CSS-files that are automatically added by SharePoint. Historically, SharePoint has been associated with content management for intranets but the latest versions can also be used for external websites. However, it is obvious that the focus is still on intranet sites due to the complicated process of optimizing the code. The results show that it is difficult to remove the files without breaking functionality in SharePoint, for example the top menu. Therefore, only certain files are removed, which reduce the page size by 125 kb and the number of requests by 4. If all files were removed, however, the reduction would be higher. Optimizing Javascript and CSS-files is an important step in improving the web performance and SharePoint needs to provide an easier way for developers to remove files that are unnecessary in a public website.

Lazy Loading is used to defer loading of content until it is requested by the user. In the prototype, two images are loaded using this technique, which result in a reduction of the page size by 132 kb and the number of requests by 2. The amount of data saved is obviously depending on the type and size of content loaded. The more objects that are loaded using this technique the larger is the reduction of page size and the number of requests. Lazy Loading is a useful technique for optimizing load time because of the considerable amount of page size and requests that can be saved.

The combination of all optimization techniques demonstrates a considerable reduction in page size, between 41-65 %, and a smaller decrease in number of requests, between 4-22 %. Responsive images account for the majority of the reduction in page size. However, optimizing Javascript and CSS together with the use of Lazy Loading have the advantage that they reduce both page size and number of requests at the same time. Therefore, it is beneficial to use a combination of all the stated techniques in order to achieve the most effective optimization of the load time.

There are limitations in the study that need to be considered. Since the sample data is limited to the prototype any generalizations of the result should be done with caution. The result is dependent on how the prototype is created, for example, how many images are used, how the Bootstrap framework is customized, etcetera. Nevertheless, the results and the discussion can provide valuable information about

positive and negative aspects of the different techniques and support decisions about what method to use when optimizing the load time of a responsive website.

Another limitation is that screen resolutions have not been taken into consideration in the study. New devices with extremely high resolution are emerging, which require images with a larger amount of pixels than before in order to look sharp. However, serving larger images will result in an increased load time and a suggestion for future research is therefore to explore how to optimize performance when using high-resolution screens.

6. References

- Allsopp, J. (2000). *A Dao of Web Design*. Retrieved May 1, 2013, from A List Apart: <http://alistapart.com/article/dao>
- Butkiewicz, M., Madhyastha, H., & Sekar, V. (2011). Understanding Website Complexity: Measurements, Metrics, and Implications. In *Proceedings of IMC'11*. Berlin, Germany: ACM.
- Frost, B. (2013). *Performance as Design*. Retrieved May 23, 2013, from <http://bradfrostweb.com/blog/post/performance-as-design>
- Gomez. (2011). *What Users Want from Mobile*. Retrieved April 24, 2013, from http://www.gomez.com/wp-content/downloads/19986_WhatMobileUsersWant_Wp.pdf
- Google. (2012a). *Make the Mobile Web Faster*. Retrieved April 24, 2013, from <https://developers.google.com/speed/articles/mobile?hl=sv>
- Google. (2012b). *What Users Want Most from Mobile Sites Today*. Retrieved April 19, 2013, from <http://www.google.com/think/research-studies/what-users-want-most-from-mobile-sites-today.html>
- Grigorik, J. (2012). *Latency: the New Web Performance Bottleneck*. Retrieved April 24, 2013, from <http://www.igvita.com/2012/07/19/latency-the-new-web-performance-bottleneck>
- Grigsby, J. (2010). *CSS Media Query for Mobile is Fools' Gold*. Retrieved April 19, 2013, from <http://blog.cloudfour.com/css-media-query-for-mobile-is-fools-gold>
- Grigsby, J. (2011). *Responsive IMGs part 2 - In Depth Look at Techniques*. Retrieved April 29, 2013, from <http://blog.cloudfour.com/responsive-imgs-part-2>
- HTTP Archive. (2013). Retrieved May 15, 2013, from <http://httparchive.org/interesting.php>
- Jehl, S. (2012). *Picturefill*. Retrieved May 24, 2013, from <https://github.com/scottjehl/picturefill>
- Kadlec, T. (2013a). *Implementing Responsive Design*. San Francisco: New Riders.
- Kadlec, T. (2013b). *Setting a Performance Budget*. Retrieved May 23, 2013, from <http://timkadlec.com/2013/01/setting-a-performance-budget>
- Keith, J. (2011). *Conditional Loading for Responsive Designs*. Retrieved April 30, 2013, from <http://24ways.org/2011/conditional-loading-for-responsive-designs>
- Kraken.io. (2013). *Kraken Image Optimizer*. Retrieved April 26, 2013, from <http://kraken.io>
- Malsen, T. (2012). *Moving Swiftly: The Story of How BBC News Fell in Love With Responsive Web Design*. Retrieved May 24, 2013, from <https://speakerdeck.com/tmaslen/moving-swiftly-the-story-of-how-bbc-news-fell-in-love-with-responsive-web-design>
- Marcotte, E. (2010). *Responsive Web Design*. Retrieved April 28, 2013, from <http://alistapart.com/article/responsive-web-design>

- Marcotte, E. (2011). *Responsive Web Design*. New York: A Book Apart.
- Mavention. (2013). *Responsive Image Renditions with SharePoint 2013*. Retrieved April 30, 2013, from <http://www.mavention.nl/blog/responsive-image-renditions-sharepoint-2013>
- Microsoft. (2012). *How to: Manage Image Renditions in SharePoint 2013*. Retrieved April 30, 2013, from <http://msdn.microsoft.com/en-us/library/jj720398.aspx>
- Microsoft. (2013). *SharePoint*. Retrieved June 5, 2013, from <http://office.microsoft.com/en-us/sharepoint>
- Mozilla. (2013). *Firebug*. Retrieved May 25, 2013, from <http://getfirebug.com>
- Nah, F. (2004). A Study on Tolerable Waiting Time: How Long are Web Users Willing To Wait? *Behaviour & Information Technology*, 23(3), 153-163.
- Olsen, D. (2013). Optimizing for Mobile. In *The Mobile Book* (pp. 175-222). Freiburg, Germany: Smashing Media.
- Podjarny, G. (2012). *Responsive Web Design Makes it Hard to Be Fast*. Retrieved April 19, 2013, from <http://www.guypo.com/technical/responsive-web-design-is-bad-for-performance-there-i-said-it>
- RICG. (2013). *Responsive Images Community Group*. Retrieved May 24, 2013, from <http://www.responsiveimages.org>
- Tuupola, M. (2013). *Lazy Load Plugin for jQuery*. Retrieved June 4, 2013, from <http://www.appelsiini.net/projects/lazyload>
- Twitter. (2013). *Bootstrap*. Retrieved April 14, 2013, from <http://twitter.github.io/bootstrap>
- Wilcox, M. (2013). *Adaptive Images*. Retrieved May 24, 2013, from <http://www.adaptive-images.com>
- Wroblewski, L. (2009). *Mobile First*. Retrieved April 25, 2013, from <http://www.lukew.com/ff/entry.asp?933>
- Yahoo. (2013a). *Best Practices for Speeding up Your Website*. Retrieved May 5, 2013, from <http://developer.yahoo.com/performance/rules.html>
- Yahoo. (2013b). *Smush.it*. Retrieved April 26, 2013, from <http://www.smushit.com/ysmush.it>
- Yahoo. (2013c). *YUI Compressor*. Retrieved June 4, 2013, from <http://refresh-sf.com/yui/>