RESPONSIVE IMAGES
Comparing techniques for implementing responsive images to websites

Bachelor Degree Project in Computer Science
30 ECTS
Spring term 2014

Jafet Bjarkar Björnsson

Supervisor: Jana Rambusch
Examiner: Henrik Gustavsson
Abstract

This thesis explores responsive images for websites, i.e., content images that respond to users display size, pixel density and bandwidth. We look into responsive web design and how it is related to responsive images. Different responsive image techniques are discussed and various use cases for responsive images are explored along with technologies that are needed to implement responsive images. A testing website is built and responsive image techniques are applied one by one, tested and compared and its effect on the websites load time is measured. The process of implementing the techniques is described with code examples and images. The results indicate that responsive images can certainly have positive effect on loading time of a website. Weather or not responsive images should be implemented depends on how much image data the website has and how fast the Internet connection is. Websites with more images can certainly profit from using responsive image techniques, especially if only slower bandwidths are available.

Keywords: responsive, images, mobile, performance, HTML5, CSS3
Table of Contents

1 Introduction..............................................................................................................................................1

2 Background..............................................................................................................................................2
  2.1 Responsive web design.........................................................................................................................3
  2.2 Responsive images.................................................................................................................................4
  2.3 Approaches.............................................................................................................................................5
  2.4 Use cases................................................................................................................................................6
  2.5 Techniques.............................................................................................................................................9
    2.5.1 HTML...............................................................................................................................................9
    2.5.2 JavaScript......................................................................................................................................10
    2.5.3 Server side components................................................................................................................11
  2.6 Responsive image techniques .............................................................................................................12
    2.6.1 Adaptive Images...........................................................................................................................12
    2.6.2 HiSRC...........................................................................................................................................12
    2.6.3 Picturefill .....................................................................................................................................12

3 Problem ..................................................................................................................................................14
  3.1 Method and approach..........................................................................................................................14
  3.2 Ethics .................................................................................................................................................16
  3.3 Previous research...............................................................................................................................16

4 Implementation .....................................................................................................................................17
  4.1 Finding the right testing environment ...............................................................................................17
  4.2 Testing the media queries .................................................................................................................18
  4.3 Implementing website template ........................................................................................................19
  4.4 Implementing Adaptive Images........................................................................................................21
    4.4.1 JPEG compression.......................................................................................................................21
    4.4.2 Adaptive Image cache on server .................................................................................................22
  4.5 Implementing Picturefill ....................................................................................................................22
  4.6 Controlling the bandwidth ...............................................................................................................23
  4.7 Considerations ...................................................................................................................................24
  4.8 Software overview .............................................................................................................................25
  4.9 Monitors .............................................................................................................................................25

5 Pilot study ..............................................................................................................................................27
  5.1 Responsive images characteristics .....................................................................................................27
  5.2 Website loading time ..........................................................................................................................28

6 Progression ..........................................................................................................................................29
  6.1 Changes in implementation ..............................................................................................................30

7 Evaluation .............................................................................................................................................32
  7.1 Before testing .....................................................................................................................................32
  7.2 The experiment ................................................................................................................................32
  7.3 Analysis .............................................................................................................................................35
  7.4 Result summary ................................................................................................................................39

8 Conclusion ............................................................................................................................................40
  8.1 Discussion .........................................................................................................................................40
8.1.1 Social and ethical issues ............................................................................................ 40
8.1.2 Critical perspective .................................................................................................. 41
8.1.3 Benefits to society .................................................................................................... 41
8.2 Future Work .................................................................................................................. 41

References ....................................................................................................................... 43

Appendix ............................................................................................................................. I
Testing website ..................................................................................................................... I
  index.html ....................................................................................................................... I
  style.css ......................................................................................................................... I
Adaptive Images .................................................................................................................... II
  index.html ....................................................................................................................... II
  adaptive-images.php ....................................................................................................... II
  .htaccess ....................................................................................................................... II
Picturefill ............................................................................................................................. III
  index.html ....................................................................................................................... III
1 Introduction

Since the launch of the first iPhone in 2007 the mobile marked has grown enormously. More and more people are starting to brows the web using their handheld devices such as smartphones or tablets. Web developers have been adapting to the smaller screen sizes by using responsive web design (RWD). RWD is a technique that allows websites to change layout based on the browser size the web page is being displayed in (Mohorovicic, 2013).

RWD certainly does solve a big problem but even though the layout has changed, the size of the page does not necessarily change. Handheld devices often have slower network connection so webpages can take awhile to load compared to loading them on desktop computer. One important factor that makes the webpages heavier is images. CSS media queries provide a way to change layout and images depending on browser size but the problem remains of how to handle content images in the HTML markup.

Responsive images are a technique that loads different image size depending on the capabilities of the device being used. Users viewing images on small screen usually do not need as big images as the ones using desktop computer. Instead of loading the full size image and shrinking it down for the handheld device, responsive images allows you to load a smaller, lighter version of the same image, saving you the trouble of loading unused pixels.

As later pointed out in this thesis the loading time of a website is a big concern and the web developer community is responding to this need. W3C (World Wide Web Consortium) has made first draft documentation (W3C Working Group, 2014) to try to standardize a solution for responsive images. Their proposal is in an early stage and the documentation could change without warning. While waiting for good standard for responsive images, developers have to use other responsive image techniques that have been created by individuals who try to solve the problem with existing web development technologies, such as JavaScript and PHP.

In this thesis some of the available responsive image techniques are tested. The techniques will then be applied to a website so we will be able to measure the load time between the difference, and document advantages and disadvantages of each technique. Size reduction, HTTP requests and loading time on different platforms will be compared for each responsive image technique and then other factors will be discussed such as legacy code, semantics, validity, ease of use and art direction.

By the end of this experiment we will be able to put together data to show the effect responsive images will have on a website, the benchmark results and characteristics. With all the collected data we can get closer to finding the best technique for handling responsive images in the future.

Images and graphics used in this thesis are my own unless stated otherwise.
2 Background

The Internet has become a global tool for accessing information. The technique is a new phenomenon in human history and it is evolving fast along with other technology. With each change in the environment the developers need to adapt to meet constantly changing standards and expectations. In recent years we have seen more and more users start to use handheld devices with smaller screens such as smartphones (Li, Ortiz, Browne, Franklin, Oliver, Geyer, Zhou, Chong, 2010). Apple presented the first smartphone in 2007 (Macworld San Francisco 2007 Keynote Address, 2007) and since then many devices have been introduced such as tablet computers, notebook computers and e-book readers.

Although web statistic sites should only be used as a rough guide, the results from gs.statcounter.com are quite interesting. According to StatCounter, mobile web traffic raises from 8.49% in January 2012 to 22.4% in January 2014. This is a 264% increase in only two years. The tablet use is also increasing and is 5.68% in January 2014. In those two years the overall desktop browsing has decreased in response to the increase in handheld devices (see Figure 2.1). These numbers hint that people are adopting the new technology quite fast and that we already have large number of users who already use it.

Before the mobile browsing became popular websites where usually built around static 800-960 pixels wide. This width worked great on most screen sizes used before and larger screens would just get a little extra margin at the sides. Screen sizes where pretty predictable before but today there has never been as much difference between the smallest and largest screens that are commonly used for browsing (Kadlec 2013).

At first when mobile browsing started to gain popularity developers started to create different versions of the same website, each version built to serve certain range of devices. This could be one version for desktop, one for tablets and one for mobile phones (Mohorovicic, 2013). This approach allowed developers to create a lightweight version for mobiles, which is great but the problem with this approach is that now the developers had to maintain two or more websites and it is usually hard enough to maintain one. The number of devices also grew and the problem grew as tablet computers, bigger screens and high-
resolution screens began to appear. Ethan Marcotte suggested another approach to this problem in an article for A List Apart (2010). The articles title was “Responsive Web Design”.

2.1 Responsive web design

Responsive web design (RWD) is a technique that allows developers to create websites in a way that they will adjust to browser size using only HTML5 and CSS3 (Frain, 2012). RWD is a combination of three techniques:

- Flexible grid layout
- Flexible images
- Media queries

Flexible grid layout enables website layout to resize and rearrange depending on percentage-based width of the website. Flexible image allows developers to scale, change layout and crop images depending on browser viewport (browser window size). Media queries allow us to target specific CSS styles depending display capabilities of the device rendering the page. All these features should be included in a website to have responsive web design.

RWD is a useful technique that allows developers to create one website that will respond to the browser window size the site is being displayed in and change layout accordingly. RWD is a new approach to web design to create flexible websites that can adapt to devices the developer does not even know that exist yet.

One problem with RWD is that although the technique has been applied to a website, the website is not necessarily any smaller (in size) than the desktop version of the website, because the same HTML, CSS, scripts and media files are being loaded. With CSS3 it is possible to display different images depending on the size of the browser, which can reduce size, but those images are thought of as being part of the layout of the website but not the content. Content-based images are images in the HTML, applied using the `<img>` tag and are often used in the main content of websites. RWD can be used to scale the content images but the full image is still being loaded (Baturay and Birtane, 2013) Making those content-based images responsive is a technique called responsive images and is one of the unsolved problems in responsive web design (Mohorovicic, 2013).

To see better the connection between RWD and responsive images we have put together a simplified graph (see Figure 2.2). In this graph RWD has two factors, size and layout. The layout part changes the appearances of the website while the size part has to do with how much smaller the websites becomes when the layout is changed for smaller screens. Reducing the amount of code loaded and reducing the footprint of the images can reduce the size.
2.2 Responsive images

*Responsive images* are a way to deliver the right image at the right occasion, depending on device capabilities (Matsudaira, 2013; Nandorf, 2013). The choices of image can depend on the users display size and orientation, browser window width and bandwidth. Without responsive images users browsing on a mobile device have to download images that are often big enough to be viewed on desktop computers. What happens normally on smaller screens is that the full image is downloaded and then the image is scaled down to fit on the smaller screen. This can be wasteful practice, as the user has to download many pixels that will never be seen. On slower networks such as 3G or EDGE this makes a lot of difference for users. Using the responsive images technique users can be provided with image that will fit better with the device they are browsing on. When using smartphone for example, the user current settings will be detected and the server in response, sends smaller version of the image to the smartphone, saving otherwise wasted space. The user could also be browsing on a paid network connection, which is another reason to use responsive images.

Lets say you have created a web page with many pictures. In the development process you decide that you want the images to look good on desktop size displays. User that accesses the website on his desktop using his home Wi-Fi connection will probably not have any difficulties loading the site. Another user visits the site but is sitting in a coffee house downtown and has only access to 3G on his smartphone. For him the site can take a painfully long time to load. Developers should take this seriously as studies show that every millisecond lost in loading time has negative effect on user experience (Wang, Balasubramanian, Krishnamurth, Wetherall, 2013; Butkiewicz, Madhyastha & Sekar, 2011) and consequently the websites revenue. Images are a big factor of website load time and responsive images might just be the solution.

There has not jet been established a standard for responsive images. As mentioned in the introduction the W3C published a *working group note* on responsive images in November 2013 where they “capture the use cases and requirements for standardizing a solution for responsive images” (Usecases.responsiveimages.org, 2013). Being a working group note means that there is no longer a plan to continue working on this requirement or that they cannot productively carry the work any further (W3.org, 2014). Responsive images are discussed a lot in the web community and W3C has created a community group ([http://www.w3.org/community/respimg](http://www.w3.org/community/respimg)) that works on finding a standard for responsive images.
Since there is no browser support for responsive images developers have been relying on other techniques to implement responsive images. Apple for example loads to begin with a low-resolution images before swapping them out if you are browsing on a high-resolution (retina) display. As a result you can see the images pixelated at first until they are replaced with higher resolution images. When apple.com is opened in a non-retina display the images will not change to the higher resolution images. The bigger images have four times as many pixels as the smaller ones and get “_2x” added to their name, e.g. “usa.png” becomes “usa_2x.png” (see Figure 2.3).

![Figure 2.3 Low and high resolution images from apple.com. The image with fewer pixels become pixelated compared to the higher resolution images](image)

### 2.3 Approaches
In his book about responsive design Kadlec (2013) talks about three categories all responsive image solutions fit into. The three categories are:

- Fighting the browser
- Resignation
- Going to the server

**Fighting the browser**
Browsers are designed to be fast. They try everything they can to load as many assets of a webpage in as small amount of time as possible. This is usually a good thing unless you want to be faster than the browser. Some responsive image techniques approach the problem by using JavaScript to change the image source before the browser downloads the wrong image. This is called a race condition when you are trying to make modifications before the image request starts. Few responsive images techniques are designed to use this method.

**Resignation**
In this case you accept a defeat by the browser and instead you take a mobile-first approach and load the smallest image first. After the browser has detected the settings of the device, a larger image can replace the smaller image if that is appropriate for current device. The detection whether to send higher resolution images can be done in two ways, by detecting display or viewport size and by detecting the bandwidth.
**Going to the server**
Techniques using this approach will use the server to detect users device and all decisions on which image to choose are done on the server. The server uses the user agent string or a cookie to determine what image size is suitable for current device. This method removes the race condition problem but there are some problems that are presented instead. Solutions in this category often use client side scripting to set cookie with device information so that the server can decide what image size is suitable. This creates a *first time load problem* when user has to wait a little longer the first time he enters a website while information about his device is being sent to server to be processed.

### 2.4 Use cases
In this chapter different use cases for responsive images is discussed. These use cases will give us a better idea of when and how responsive images would be used in real life and what challenges can arise with this technique. This list of use cases is built on W3C’s document on responsive images (Usecases.responsiveimages.org. 2013).

**Image selection based on screen resolution**
Developers sometimes want to be able to control the size of the image based on the device users are using. This provides users browsing on desktop computers with optimum image resolution while users on smaller screens get lower resolution images, saving bandwidth in the process (see Figure 2.4).

![Figure 2.4 Different screen sizes in pixels. Comparison between desktop monitor, iPad 2 and iPhone 3](image)

**Image selection based on viewport**
Sometimes it might be appropriate to make image sizes respond to browser viewport (the visible part of the website in the browser window) rather than the screen resolution. Responsive layouts change on different viewport sizes. In certain cases this could mean that smaller viewport uses bigger images (see Figure 2.5). In that case it could be better to base image size on viewport rather than screen size.
Pixel per inch (PPI) based selection
Monitors come with different pixel density. A recent trend in the computer business is high-resolution displays that can have pixel density close to the resolution of fine print. For example: MacBook Pro 15” had traditionally 1440*900 pixel displays which equals about 113 PPI. The more recent high-resolution displays have 2880*1800, which equals to about 226 PPI. Displays with higher PPI need higher resolution images for them to look good. This applies also to icons used on websites.

Art direction
In some cases developers want to have control over how the image is made smaller for smaller screens. For example you would sometimes rather want to crop the image rather than just scale it down. Some images can have lots of details and on smaller screens the purpose of the image can loose its mark when details in the image become too small. By “art directing” the developer can control how image is presented in different screen sizes (see Figure 2.6).
Breakpoints

Breakpoints are used to control changes in layout based on certain media features. Breakpoints are commonly used in CSS3 in responsive web design.

Example:

```css
@media screen and (min-width: 900px) {...}
```

Instead of “...” the developer could add all the CSS styles he would want to apply when the browser viewport is larger than 900px wide. The `<picture>` element is one of the solutions proposed in the W3C document and it uses this breakpoint technique.

Different media types

It can be appropriate to serve different images depending on which media format the image is supposed to be displayed on. Printers can usually pack more dots per inch (dpi) than are needed to display the picture on displays. It can therefore be good idea to serve high-resolution images for printed media.

Displaying images on monochrome media types also present its own type of challenges. Images where color is important can be useless when displayed on monochrome devices such as e-ink displays, used for example in the Kindle devices from Amazon (see Figure 2.7).

![Pie chart in color versus monochrome](image)

**Figure 2.7 Pie chart in color versus monochrome**

Image formats and control over sources

Developers use different image formats in web design for different purposes. Certain image format can fit better than others in certain situations. There are four main image formats used on the Internet, JPEG, GIF, PNG and SVG (Niederst Robbins, 2012). The first three are pixel based and the SVG is a vector image format. They all have their place on the Internet.

JPEG is good for photographs and images with smooth color blend. GIF images are good for flat colors; it has one level of transparency and can be animated (like short videos). PNG is similar to GIF but cannot be animated. It has multiple levels of transparency. Often used for layout graphics and symbols on websites. SVG is vector based which means the graphics are based on numbers that are translated into points, lines and curves. Vector images can be scaled up without loosing sharpness and without getting larger in size.

When users are browsing on a limited or expensive bandwidth users could be served lower quality images or even no images at all. The user could then choose to display the images or not.
2.5 Techniques
There exist a number of techniques to implement responsive images into a website. All the solutions fit into Kadec’s three categories for responsive images: *Fighting the browser, resignation* and *going to the server*. The rest of this chapter will be dedicated to technologies that are used in many of the responsive image solutions. These technologies are commonly used in web development. There will be given short introduction of each technique, what it is used for, what makes it useful and how it fits into Kadec’s categories. Lastly there will be a discussion how this technology can be used in context with responsive images.

2.5.1 HTML
It is important to note that HTML (Hyper Text Markup Language) is designed for presenting content while CSS is designed to control the layout of the website (Pfaffenberger, 2004). HTML and CSS along with graphics and scripting make the structure of webpages on the Internet. All responsive image techniques have something to do with HTML. Some require you to change the markup, and some can effect the semantics or validity of your website. These subjects will be discussed further so we can have better understanding on what we are dealing with when trying to find suitable solution for responsive images.

Markup
Most of the existing responsive images techniques require you to change the markup (the HTML) in some way. In some cases this would not be a problem but in cases where the website already has a large amount of legacy content it can be time consuming to go back and change all the markup to implement responsive images for the whole website. It could be possible to leave the markup unchanged in older content and just use the new markup rules on newer content. Then the responsive images technique would only work on new content but the old content would be left unchanged.

A problem with changing the markup is that while the solution you are using is not a standard it is quite likely that a standard will be set or a better solution will be discovered. In that case you will have to go back and change the markup again if you want to apply this new solution to your website. In other words, you can make your markup more future proof by not changing the markup.

Semantics
The semantic web is a collection of design patterns and technology standards defined by W3C to share data and metadata on the web (Yu, 2011). The semantic web tries to bridge the gap between human-readable and machine-readable information on the web. It is not separate from the existing web but rather an extension to the current one to allow humans and computers to work in cooperation (Berners-Lee, Hendler, Lassila, 2001).

To make a website semantic you have to follow certain design patterns in your markup. The `<img>` tag should include the `rel` and `src` attributes (Pfaffenberger, 2004). The `src` attribute should include the path to the image and the `rel` attribute should include description of the image. Computers cannot recognize and create semantic patters from images (Badr, 2010) so if the `rel` attribute is missing the markup cannot be semantic. When these patterns are followed in websites, data becomes more available in other places and the data can be read and interpreted with other data existing on the Internet. By making the web semantic the data becomes more structured, more like data in databases. Some responsive
Images techniques require abnormal use of HTML markup that can have negative effects on the semantics of the website.

**Validity**
You generally want to follow the current Internet standards when you create your website. No two browsers provide the same support to HTML and CSS standards (Pfaffenberger, 2004: 549). The browsers can handle HTML and CSS in different ways so if your markup does not meet the standards the users are more likely to run into problems you have not anticipated.

You can run websites through validator tools such as the W3C validator (validator.w3.org). The validator will check the grammar, vocabulary and syntax and return errors and warnings if there are any rules broken in the code. A valid website is not necessarily a good website but it is generally considered a good practice to have your website valid. Some responsive image techniques use special HTML syntax that in some cases will not validate.

### 2.5.2 JavaScript
JavaScript is a really important component to make interactive websites and it has become the most ubiquitous programming language in the world (Flanagan, 2011). It is used on most websites on the web (around 97%) and is included in all modern web browsers on PCs, phones, tablets and consoles (Richards, Lebresne, Burg and Vitek, 2010). JavaScript runs client side so content on a webpage can be changed without having to reload the page.

Most of the available responsive images technique use JavaScript in one way or another. JavaScript can be used in each of Kadec's categories for responsive images. The following is examples of how JavaScript could be used:

- **Fighting the browser**: After the HTML is downloaded JavaScript swaps out the image source before the image is requested.
- **Resignation**: The browser downloads small versions of the image first then JavaScript will swap them out if necessary.
- **Going to the server**: JavaScript sets a cookie with information about the client device and sends to server to process.

Relying on JavaScript to display images could cause problems if the user has JavaScript turned off in the browser. That could lead to no images showing up at all on the website. Some techniques use `<noscript>` and place image tag inside as a fallback if JavaScript is not available.

JavaScript provides powerful tools to be able to run on users devices. This can create security holes for the users and unwanted code can run on client site without the user knowing. Browser vendors have worked hard to create a powerful API for JavaScript while trying to prevent malicious code from be able to run on users devices (Pfaffenberger, 2004).

**jQuery**
jQuery is a JavaScript library. It is built on JavaScript in the purpose of enhancing its functionality and to make it easier to use (Dayley, 2013). To use jQuery libraries it is necessary to import the jQuery code to the website, either by storing the code on your server and linking it, or by linking to other servers that store the code. When the jQuery library has
been loaded to the website it can be used like any other JavaScript (.js) file. With jQuery you can do more by writing less code. Underneath it’s all just a plain JavaScript.

The problem with using jQuery dependent responsive images solutions is that first of all the user has to load the jQuery library. If you are not otherwise using jQuery this can increase the loading time of your website. The other reason jQuery could be problem is that if you are using other JavaScript libraries there could be some compatibility issues that could arise between the libraries.

**Polyfills**

As mentioned earlier, browsers can handle HTML and CSS differently. This applies specially to older browsers who are more limited in their support for markup standards. Polyfills are add-ons to browsers written in JavaScript to add or modify functionality of browsers (Niederst Robbins, 2012). Polyfills are used for example to add support for HTML5 elements or CSS3 selectors to Internet Explorer 8 and older.

Polyfills can be used to add support to special markup rules for HTML and functionality to browser to implement responsive image solutions, like the responsive image technology Picturefill does (discussed later).

### 2.5.3 Server side components

Some responsive images rely also on server side components like PHP and Apache’s .htaccess file. These techniques utilize the server so more work can be done on the server instead of the client machine. These techniques are mostly used in Kadlec’s third category, *going to the server*.

**PHP**

PHP is a programming language designed to create dynamic web content (Tatroe, Lerdorf, Macintyre, 2013). It runs on all major operating systems and can be used on most servers, including Apache. PHP runs on the server and the webpages won’t update on client side like JavaScript does. Instead PHP can perform functions on the server and generate HTML to send to user.

According to Tatroe, Lerdorf and Macintyre (2013), 78% of websites use PHP so the programming language has pretty wide adoption. Others might not use any server side programming language at all and others can also use other languages such as Ruby or Python. For those the responsive images solutions using PHP might not work.

**Apache**

Some responsible images solutions make use of the .htaccess file. The .htaccess file is a per directory configuration file where developers can change configuration of the webserver without an access to the main configuration file (Bowen and Liska, 2002). The .htaccess file is used in some responsive images solutions. The file is supported by the Apache webserver software that runs on more than 60% of all websites on the Internet. Other webserver software might support .htaccess but the file might have to be modified for current webserver software.
2.6 Responsive image techniques

There exist a number of available techniques to implement responsive images. In this chapter we will introduce and discuss few responsive image techniques. We will make a subset of solutions to test because there exist too many techniques for us to be able to test and document them all. We will select techniques that have gained most popularity on the web and are most mentioned in scientific papers. I will also select techniques that are different from each other in their approach to responsive images.

Some of the information about individual responsive image techniques written below is acquired from the techniques documentations. The information from the documentations is summarized here but the information should be taken with caution, as some of the material provides no proof of being accurate. However, the selected techniques all use tested and approved web technologies discussed earlier in this thesis. More details on implementation and characteristics the techniques will be discussed in chapter 4.

2.6.1 Adaptive Images

Adaptive Images is a server side solution for delivering responsive images (Nandorf, 2013). The solution uses JavaScript, .htaccess and PHP. JavaScript is used to create cookie with information about device screen size (in pixels) to be sent to server for future reference. When the server gets an image request from the browser the screen size is determined and smaller version of the image is created, cached on server and sent back to browser (Kadlec, 2013).

This happens through the use of .htaccess and PHP on the server. The .htaccess file is configured so when the server gets the image request the .htaccess file will point the request to adaptive-images.php (Wilcox, n.d). In this file you can configure whatever breakpoints you want to have for your images and the PHP code will create smaller version of the image and cache the image on server for future requests.

2.6.2 HiSRC

This solution uses the resignation approach and is a lot similar to the solution Apple uses on its website. Low-resolution image is loaded first by default and then jQuery is used to detect if the device is running on fast network or high-resolution display (Schmitt, 2013). If the network is fast enough and the browser is running on high-resolution display, higher resolution image will replace the original. Otherwise a “normal” resolution will replace the low-resolution image.

2.6.3 Picturefill

Picturefill is a solution that mimics the proposed <picture> tag solution in the W3C work note (Jehl, 2013). It uses <span> tag instead to add the images. Picturefill uses JavaScript to identify the viewport size (Nandorf, 2013) and then media-queries with the media attribute inside the HTML markup do decide which image should be chosen. Picturefill uses the machMedia polyfill to add support to the media attributes used in the markup. The
machMedia polyfill allows you to write media queries, with the same syntax as CSS media queries, in the HTML.

Example of the markup used for Picturefill:

```html
<span data-alt="Lion in a cage">
 <span data-src="small.jpg"></span>
 <span data-src="medium.jpg" data-media="(min-width: 400px)"></span>
 <span data-src="large.jpg" data-media="(min-width: 800px)"></span>
</span>
```

This will result in three different outcomes, depending on screen size. If for example the display is 800px or bigger, Picturefill will use JavaScript to generate:

```html
<img alt="Lion in a cage" src="large.jpg">
```
3 Problem

Our hypothesis is that using responsive images will decrease the loading time of a website compared with website without using responsive images. The responsive image techniques are discussed in chapter 2.6 but as far as we know there has not been done any formal tests on the performance gain of any technique. The goal of this thesis is to benchmark and evaluate how these techniques impact the performance of a website. Similar to the study presented in the paper *Performance Comparison of Dynamic Web Technologies* by Titchkosky, Arlitt, and Williamson (2003), this project aims to benchmark the techniques on a website to be able to see with quantitative results how responsive images affect the performance of a website and further how they compare against each other.

In this thesis we will do a test on three responsive image techniques, Adaptive images, HiSRC and Picturefill. These techniques are common solutions for many developers to implement responsive images to their website. The responsive image techniques use different technologies such as JavaScript, jQuery, polyfill, PHP and Apache and the techniques handle art direction, syntax rules and semantics differently (see Chapter 2.5). No solution has been officially standardized jet.

The three responsive image techniques we will be testing are very different from each other in approach and in the technologies they use. By testing the techniques on the same website, we can get better idea of what responsive image technique reduces the load time most, and if they could possibly be a suitable solution that could be standardized. A solution for responsive images is needed more every day that passes. New devices are created at a growing rate as they are becoming cheaper to manufacture (Kadlec, 2013). The users are also getting more confortable by using their mobiles and smaller screen devices to browse the Internet and we can probably see that number grow as more people modify their websites for better mobile experience.

3.1 Method and approach

We will be conducting *technology-oriented experiment* in this project to research responsive image techniques. Experiments are done in a controlled environment where the tested object can be manipulated precisely and systematically to compare outcomes (Wohlin, 2012). Experiments can be *technology-oriented* or *human-oriented*. In technology-oriented experiments different tools are typically applied to different objects, while in human-oriented approach humans apply different treatments to objects. Experiments are done in a controlled environment and the subject can be shaped in a way to simulate its use in the real world (Titchkosky, Arlitt, and Williamson, 2003)

Case study was ruled out as a feasible technique for this thesis. Case study is an empirical enquiry that is used to gather information about existing phenomenon to investigate a case or few cases in real-life context (Wohlin, 2012). Rather than researching the responsive image techniques being used on real websites we concluded that it would give us more concrete results by creating our own testing website where all responsive image techniques could be applied, one by one, to get precise measurements. After the tests we will compare the data to get a new perspective on the current situation.
Each of the discussed responsive image techniques will be applied to testing website and then the benchmark tests will be performed. The benchmark testing will give us quantitative data about each technique, which will give us quite good image of the performance of each technique. The size (in KB) of the website is also measured and HTTP requests are documented. After the benchmark tests we will document the characteristics of every technique, what technologies current technique uses and where the technique has advantages and disadvantages compared with others. The most important factor is the load time of the website. All other measurements can be useful to give us perspective on each responsive image solution. All data that is collected from this experiment will be quantitative which completely frees the experimenters from being able to make their own opinion effect the results.

After implementing, benchmarking and evaluating the responsive image techniques we put the data together in graphs and tables to help us analyse the data. From the data we hope to be able to create hypothesis about what standardized responsive image technique should include (see Figure 3.1).

![Diagram](image)

**Figure 3.1 The implementation process**

The implementation process of each responsive image technique will be documented so we will get a good sense of how each solution works. After implementation the benchmark tests will be conducted and the results will be discussed and put together in graphs to allow the readers to view the results in an easy and understandable way. The characteristics of each technique will be discussed along the way and then be put together in a table for the reader to get a good overview of each responsive image technique.
3.2 Ethics

There is no testing done on people in this thesis so we think it is unlikely to be any direct ethical issues arising from cultural or social needs. All the benchmarking tests and observation are done without the need to investigate any specific type of users or groups.

By making this study about responsive images we could possibly affect the course the development of responsive images will take. Bad study with inaccurate results we could change the development in a wrong direction that could harm and slow down the process of finding the best solution for responsive images.

Everything used to set up the experiment, browser version, website code, responsive image version and version of technologies will be provided in this thesis to preserve the repeatability of this study. Ethics in highly publicized scientific researches have received some attention in the past several years because of scientific misconduct (Frase, Laura, Priscilla and Kovac, 1997). We take it seriously to make this an accurate and fair study with good and fault free code.

3.3 Previous research

So far there has not been done lot of scientific research on responsive images. The most extensive text about responsive images is in Kadec’s book (2013) where he discusses responsive images as a concept of RWD. He talks about the three categories we mentioned in chapter 2.3 and then he talks about Adaptive Images and Sencha, how they work and (roughly) how they are implemented. Sencha (also called Sencha.io Src) is a third-party server side solution for responsive images that gets the closest to being a plug-and-play solution (Kadlec, 2013). Sencha uses user agent string to figure out the pixel width of the device display and resizes the image by default down to that width. Sencha is not tested in this thesis because it is a third party solution so all the work is performed on their server with a code that we can not view and test and that would prevent us from preserving the repeatability of this thesis.

Niederst Robbins has a subchapter about responsive images in his book Learning Web Design (2012). He describes the concept and how it is seems to be more debate about responsive images than solutions.

Responsive images are shortly defined in few other articles (Mohorovičić, 2013; Matsudaira, 2013; Nandorf, 2013) but it seems that there has not been any research done to measure and compare the available responsive image techniques, or to see how responsive images affect websites loading time.
4 Implementation

In this chapter the implementation of the testing environment before the pilot study is discussed. Testing website and responsive image techniques are implemented and the process that led to this setup is described. Some parts of this setup will change after the pilot study. Those changes will be described in chapter 6.

4.1 Finding the right testing environment

Finding the right testing environment was a tricky puzzle to solve. There has to be a way to accurately speed test websites running on different screen sizes and resolutions. All code for tools used in the benchmark tests needs to be accessible because I need to know that they do not cheat on benchmark tests (Klung & Shimpi, 2014). The responsive image techniques use different properties to resolve which image size should be used. Adaptive Images uses screen size while Picturefill uses browser size. Both techniques can also be configured to take pixel density into account.

The first solution we tried was to test the techniques using different devices like smartphones, tablets and desktops. There where couple of problems with this approach. First of all there is limited number of developer tools (especially for iPhone). Also even if we would find developer tools on all platforms the tools could give us different results depending on how the tool is built. We need to be able to test the techniques on different screen and viewport size on different pixel densities using single benchmarking tool. Next we tried using virtual machines but that approach was also ruled out as feasible solution because both Windows and OSX running as virtual machine did not provide necessary tools to control the pixel density. We also tried changing the output signal the computer would send the monitor using SwitchResX. This also did not work out well enough, as the software could not change the pixel density and the resolutions options where too limited.

At last I found a suitable solution. By using laptop computer with high resolution display and connecting low resolution monitor I could test the techniques on both low and high pixel density monitors, just by moving the browser window between the screens. To test the techniques that depend on viewport size I just had to resize the browser window. Adaptive Images uses the screen size to make decisions for image resizing so resizing the browser will have no effect. With Adaptive Images you have to create a cookie with information about users screen properties (more details in Chapter 4.4). By changing the cookie Adaptive Images creates I could trick the server to think the screen has certain screen size (see Table 4.1). With this setup we have total control of screen size and resolution.

<table>
<thead>
<tr>
<th>Original</th>
<th>Manipulated</th>
</tr>
</thead>
</table>

Table 4.1 - Manipulated script creates cookie that tells the server that the resolution is 480 pixels
4.2 Testing the media queries

As mentioned earlier in this thesis (Chapter 2.1) media queries in CSS3 is one of three main building blocks of RWD. They are used to control how content is presented on website depending on media properties such as screen size and resolution (Baturay & Birtane, 2013). Media queries are used to define breaking points for websites in RWD. Responsive web design and responsive images are closely related concepts so therefore it is appropriate to define breaking points for our project so we will have some kind of structured guidelines to follow when implementing the techniques to the website (see Table 4.2). With these guidelines in place we can implement the responsive image techniques so they return the same image sizes on same device properties.

@media only screen and (max-width: 959px) {
  body { background: url("img/calendula-480.jpg") top left no-repeat; }
}
@media only screen and (min-width: 960px),
  only screen and (min-device-pixel-ratio: 2), (-webkit-min-device-pixel-ratio: 2) {
  body { background: url("img/calendula-960.jpg") top left no-repeat; }
}
@media only screen and (min-width: 960px) and (min-device-pixel-ratio: 2),
  only screen and (min-width: 960px) and (-webkit-min-device-pixel-ratio: 2) {
  body { background: url("img/calendula-1920.jpg") top left no-repeat; }
}

Table 4.2 - CSS3 media queries define the breaking points for image

We are working with the image “calendula.jpg”. The image has been rendered in three different sizes:

- 480*300 px (small)
- 960*600 px (medium)
- 1920*1200 px (large)

The medium and large images are 2 and 4 times respectively the width and height of the smallest one. So with every increase in image size we have 4 times the amount of pixels. The width of the image is appended to the name of the file.

Every image is only displayed if certain requirements are met. Basically we want the smallest one to be displayed on low-resolution screens smaller than 960px and the biggest one to be displayed on big, high-resolution displays. Figure 4.1 helps visualise the code in Table 4.2:
Figure 4.1 - Media queries visualized

Following the color coded arrows you can see what image size will be displayed. For example if the screen width is less or equal to 959px and the screen has low dpi, image with 480px width is displayed. If the screen is bigger than or equal to 960px and has low dpi the 960px wide image is displayed.

These are the terms we want to use in our experiment (will change after pilot study). The responsive image techniques will be implemented to follow this form. Without responsive images the website will always display the same image version. Implementing responsive images to fit these terms will be discussed later in this chapter.

4.3 Implementing website template

To find out how responsive image techniques affect a website we need to create a website on which we can apply the chosen techniques (cf. Chapter 2.6) one by one to see how they affect the website’s loading time as described in chapter 3. To find out what this website should have we started by downloading Skeleton which is a RWD website template licensed under open source MIT license (Gamache, 2014). We added few JPEGs and PNGs from our personal collection to the template and started testing the responsive image techniques. Before applying responsive image technique to the website the whole website was duplicated so each technique is implemented to the template without affecting each other (see Figure 4.2).
The folder “basic” holds the website without responsive image technique. There is one folder for each technique (Adaptive Images, HiSRC and Picturefill). Some media query tests are done in the “test” folder that we discuss later.

After implementation of every technique we update the website to fit our requirements. After few iterations of implementing the chosen techniques to the website and updating the website, the idea of what the website should include became clearer. The Skeleton template had a lot of unnecessary CSS3 code. The Skeleton is a responsive template, which changes the layout on certain widths. Because the breakpoints in the Skeleton template did not match the breakpoints we wanted to test on our images there was no need to keep them so we changed the template from adaptive to liquid where the layout is percentage based (Davison, 2014). Nearly all of the code from the Skeleton template was removed from the website so we have a very basic website to test.

The final version of the website was stripped down of all irrelevant code so the focus could be kept on the code that is relative to the experiment. The code for the template can be viewed in the appendix. The basic template (see Figure 4.3) has one 1920*1080 pixels JPEG image. Each responsive image technique can then be added one by one to this website.

The website has header, navigation and content. The only thing that will change, between each implementation excluding all the changes required to implement each responsive image technique, is the “BASIC” text in the <h1> tag. For example: “Responsive Images | BASIC” becomes “Responsive Images | ADIMG” for the website with Adaptive Images implemented. By keeping the same length of the titles we will not affect the amount of markup added by each technique (which will be a measured in our tests).
4.4 Implementing Adaptive Images

Adaptive Images is implemented into our testing website following the instructions provided on the Adaptive Images website, adaptive-images.com/details.htm. For the pilot study I used the alternative JavaScript provided on the website which allows the Adaptive Images plugin to detect high-resolution displays (see Table 4.3).

In “adaptive-images.php” I put in two breaking points in the $resolution array, 960 and 480 (see Table 4.3). Now images will be proportionally scaled down so the width of the image will match these values. Adaptive Images checks the screen size and if the size is smaller than one of the breaking points, the image will be scaled down to that size. If for example the screen is less than 480px wide, the image is scaled down to 480*300px.

```
$resolutions = array(960, 480); // the resolution break-points to use (screen widths, in pixels)
```

Table 4.3 - Breakpoints defined in adaptive-images.php

To finish implementing Adaptive Images you have to add code into the .htaccess file, set cache folder path and more. These instructions can be found on Adaptive Images website. All code can be viewed in the appendix.

4.4.1 JPEG compression

The images used in this experiment are scaled down and compressed in Photoshop CS6 using the “Save for web” option. Adaptive Images uses PHP to scale down and compress the images. Both Photoshop and PHP use compression scales from 0-100 but setting both to for example 70 will render two totally different results (Fotoforensics.com, 2014). The problem is that we will be measuring the loading time of each technique and different image sizes will affect the load time.

We did a little test to better understand how JPEG compression compares between Photoshop and PHP. Adaptive Images will use the biggest image (calendula-1920.jpg) to create two smaller versions. In adaptive-images.php we can change the JPEG compression quality PHP will use. We set the image quality to 92 and rendered the two smaller pictures. We then rendered the images from Photoshop and adjusted the compression so a file size that would fit as well as possible (see Table 4.4).

<table>
<thead>
<tr>
<th>Size (px)</th>
<th>File size (kb)</th>
<th>Compression</th>
<th>File size (kb)</th>
<th>Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>480*300</td>
<td>38</td>
<td>70</td>
<td>38</td>
<td>92</td>
</tr>
<tr>
<td>960*600</td>
<td>109</td>
<td>67</td>
<td>110</td>
<td>92</td>
</tr>
<tr>
<td>1920*1200</td>
<td>247</td>
<td>64</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4.4 - Comparison of compression and size between Photoshop and PHP JPEG compressors

Compressing image in PHP at 92 gives similar size image as compression in Photoshop at 70. The image sizes are not exactly the same as you can see in the comparison between the image sizes on the 960*600 image. **In this thesis I will replace the files Adaptive Images caches on the server with the images saved from Photoshop so I will have exactly the same file sizes to compare.**
4.4.2 Adaptive Image cache on server
When Adaptive Images shrinks an image it will create a cache folder for the image on the server. This only happens the first time the image is loaded and after that the cached image will be used. To find out how much performance impact this caching process would make, I did small test to compare the load speed with and without the images already cached on server (see Table 4.5).

<table>
<thead>
<tr>
<th>Adaptive Images</th>
<th>3G, Good connectivity (850 kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>960 image used</strong></td>
<td><strong>No cache</strong></td>
</tr>
<tr>
<td>Load 1</td>
<td>3.69</td>
</tr>
<tr>
<td>Load 2</td>
<td>3.54</td>
</tr>
<tr>
<td>Load 3</td>
<td>3.54</td>
</tr>
<tr>
<td><strong>Average time</strong></td>
<td><strong>3.59</strong></td>
</tr>
</tbody>
</table>

Table 4.5 - Comparing loading speed with and without server cache

The average difference between the images already cached versus not cached is 3.59 – 3.11 = 0.48 seconds. This, not surprisingly, proofs that scaling down and saving the image before sending it back to user is slower than just sending the image when it has already been cached. In our final time test (see Chapter 7) we will use be testing Adaptive Images with images already cached. We do this because this performance hit only happens once for each image. We want to test the technique in a more usual case, when the image has already been cached.

4.5 Implementing Picturefill
Picturefill is downloaded at Github (github.com/scottjehl/picturefill). To start using Picturefill you only need to add one JavaScript file (picturefill.js) to the website. To make Picturefill work for our calendula image we need to change the HTML markup (see Table 4.6).
Table 4.6 - Image implemented using with Picturefill

Picturefill uses `<span>` tag to insert images into website. There are three `<span>` tags and a `<noscript>` tag in a parent `<span>` tag. The first `<span>` is the default image (the smallest one) and the second and third have path to the two bigger images. The `data-media` attribute holds the media queries we defined in chapter 4.1.2. In the `<noscript>` tag we have the default image, which is automatically used if JavaScript is turned off.

4.6 Controlling the bandwidth

To get accurate results I also needed to control the network speed (bandwidth) because any changes in the Internet connection could affect the results. I will be measuring the loading time of the testing website which will be loaded from server through the Internet. Any fluctuations in network connection would affect the results. The solution adapted here is to cap the bandwidth way below the full bandwidth so any fluctuations are unlikely to affect the Internet connection. The most suitable tool we found is Network Link Conditioner, which is a tool from Xcode Developer Tools package (see Figure 4.4). With Network Link Conditioner you can accurately and consistently control the bandwidth of your Internet connection (Thompson, 2014). For the pilot study I will use preset that simulates 3G connections in good condition. The bandwidth for this preset is 850 kilobits per second (kbps).
To verify this tool is working well we did an Internet speed test using the popular Speedtest.net tool from Ookla. Speedtest.net is one of the best Internet speed testing tools and has a long list of bandwidth test servers located around the world (Fisher, 2014). We did some quick research on this using Google and indeed Speedtest.net seemed to be widely accepted network speed test tool. We applied the option “3G, Good Connectivity” (850 kbps) to Network Link Conditioner and compared the speed with the results from Speedtest.net (see Table 4.6).

Note that Network Link Conditioner sets the bandwidth to 850 kbps but Speedtest.net returns results in mbps. 1 mbps equals 1024 kbps so 0.82 mbps = 839.68 kbps. This test indicates that Network Link Conditioner works in a way that we get fairly stable Internet connection. When Network Link Conditioner is turned off the network connection is the less stable (notice the drop in bandwidth speed on Speed test 3). The uncapped network speed is the speed of the Internet connection with no bandwidth limitation applied. The uncapped bandwidth speed is much faster than 0.82 mbps so we should not be affected by its limitations or fluctuations.

### 4.7 Considerations

**PNGs**

To begin with the website I would test had both JPEG and PNG images. As discussed in chapter 2.4 about use cases, PNG images are usually used for symbols and in cases where you need parts of the image to be transparent. Those images are often used as a part of the layout and the layout images can be controlled with the CSS3 media queries. Because of this I decided not to include them in our test and focus on JPEG images instead.
HiSRC
As discussed before I had chosen three responsive images techniques to apply to a website, Adaptive Images, Picturefill and HiSRC. All the techniques have different approaches to apply responsive images. We want to compare the loading time of website with the techniques applied and how much time it takes load each image size (see Chapter 3). The HiSRC works in a different way though. HiSRC always loads the smallest resolution image and then loads a bigger image depending on pixel density and bandwidth. This means that HiSRC always loads two images for every single image. If high-resolution (retina) display is detected, HiSRC will also do a bandwidth test be sending small image to client and measure the time. HiSRC’s approach is so radically different from the other techniques that I do not see any way to compare them fairly. The HiSRC will therefore be left out of the experiment.

4.8 Software overview
We will be comparing two browsers in this experiment, Chrome and Firefox, because we want to see if there is any difference between running the chosen techniques on different browsers. The browsers both come with installed developers tools, DevTools for Chrome and Firebug for Firefox. The developer tools provide us with information about page load time, HTTP requests and amount of transferred kilobits.

Here is a complete list of software used in this experiment:
- Chrome 34.0.1847.137
- Firefox 29.0
- OS X Version 10.9.2
- Network Link Conditioner 1.0
- PHP 5.4.26
- Apache 2.2.27 (Unix)
- JavaScript 1.8.5

4.9 Monitors
The pilot study is done on two monitors, 15” high-resolution monitor and 24” low-resolution monitor. See more details about monitors in Table 4.7. The 15” monitor is part of the laptop and the 24” is connected to the laptop so browser windows can be moved between the screens.

<table>
<thead>
<tr>
<th>Size</th>
<th>24”</th>
<th>15”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width (px)</td>
<td>1920</td>
<td>2880</td>
</tr>
<tr>
<td>Height (px)</td>
<td>1080</td>
<td>1800</td>
</tr>
<tr>
<td>Device Pixel Ratio</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.8 – Comparison between the low and high-resolution monitors
These properties along with viewport size can be viewed in the “window” object of the DOM. We accessed this information by calling the exact properties we needed and used `console.log()` function to output them (see Table 4.8). The output appears in the console in the browser development tools.

```javascript
console.log("devicePixelRatio: " + devicePixelRatio);
console.log("screen.width: " + screen.width);
console.log("screen.height: " + screen.height);
console.log("innerWidth: " + innerWidth);
console.log("innerHeight: " + innerHeight);
//console.log(window);
```

Table 4.9 - JavaScript code used to get screen information

devicePixelRatio gives us the pixel density. screen.width and screen.height returns the size of display in pixels. innerWidth and innerHeight returns viewport size in pixels.
5 Pilot study

For this thesis I will to do a pilot study. By doing pilot study I can evaluate how much time the final study will take and and if there are any details that should be changed before the final study. First an overview of the techniques tested is presented in a table and shortly discussed. Little test on cached files with Adaptive Images is discussed and finally we describe the testing process.

5.1 Responsive images characteristics

For overview of the responsive image technologies in this thesis I put together a table with the characteristics of the responsive image techniques compared with the website with no responsive images (see Table 5.1).

<table>
<thead>
<tr>
<th></th>
<th>Basic website</th>
<th>Adaptive Images</th>
<th>Picturefill</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Requests</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Plugin size (kb)</td>
<td>-</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>Needs change in markup</td>
<td>-</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Added markup (characters)</td>
<td>-</td>
<td>0</td>
<td>563</td>
</tr>
<tr>
<td>Validates</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Art direction</td>
<td>-</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Bandwidth testing</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Uses JavaScript</td>
<td>no</td>
<td>yes, to set cookie</td>
<td>yes</td>
</tr>
<tr>
<td>Uses jQuery</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>.hdaccess</td>
<td>-</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Uses PHP</td>
<td>-</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 5.1 - Difference between implementations

Both techniques add one extra HTTP request. Both techniques require you to add some files to the website. The Adaptive Images file is much bigger than Picturefill (12kb vs. 2).

Adaptive Images does not require any change in markup, which is a big advantage (see further discussion in Chapter 2.6.1). The markup added for Picturefill is a 635 character for only one image, which is quite a lot. To be fair the media queries are pretty detailed but even though we would remove the extra markup to detect high-resolution displays we would still have 316 characters. This quickly adds a lot of overhead when you add multiple images to your site.

Both techniques are valid when checked in the W3C validator (validator.w3.org). Art direction (see Chapter 2.4) is possible with Picturefill but not with Adaptive Images. Adaptive Images only uses JavaScript to set cookie that is sent to server to be processed by Apache and PHP. Picturefill relies on JavaScript to control image selection.
5.2 Website loading time

In the pilot study the loading time is recorded, with or without responsive images implemented. The variables in this study are:

- Monitors or viewport size
- Responsive image techniques
- Browsers

For each responsive image technique we are testing two browsers and three different image sizes. This is equal to 3(techniques)*2(browsers)*2(images) = 12 iterations. Each iteration is called “Round”. After the testing environment has been prepared I will start by recording times on round one on all techniques. We will do 3 rounds for each technique and then calculate the average value. For every image size we start by loading the page and then refresh the page twice (see Table 5.2).

<table>
<thead>
<tr>
<th>Chrome - Picturefill</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image size</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>480*300px</td>
<td>2.37</td>
<td>1.81</td>
<td>1.83</td>
</tr>
<tr>
<td>480*300px</td>
<td>2.62</td>
<td>1.57</td>
<td>1.4</td>
</tr>
<tr>
<td>480*300px</td>
<td>2.78</td>
<td>1.77</td>
<td>1.72</td>
</tr>
<tr>
<td>AVG</td>
<td>2.59</td>
<td>1.72</td>
<td>1.65</td>
</tr>
<tr>
<td>960*600px</td>
<td>3.56</td>
<td>1.57</td>
<td>1.82</td>
</tr>
<tr>
<td>960*600px</td>
<td>3.33</td>
<td>1.83</td>
<td>1.41</td>
</tr>
<tr>
<td>960*600px</td>
<td>4.43</td>
<td>1.24</td>
<td>1.74</td>
</tr>
<tr>
<td>AVG</td>
<td>3.77</td>
<td>1.55</td>
<td>1.66</td>
</tr>
<tr>
<td>1920*1200px</td>
<td>7.13</td>
<td>1.65</td>
<td>2.45</td>
</tr>
<tr>
<td>1920*1200px</td>
<td>7.85</td>
<td>1.58</td>
<td>1.63</td>
</tr>
<tr>
<td>1920*1200px</td>
<td>8.24</td>
<td>1.80</td>
<td>1.26</td>
</tr>
<tr>
<td>AVG</td>
<td>7.74</td>
<td>1.68</td>
<td>1.78</td>
</tr>
</tbody>
</table>

Table 5.2 - Load time test with Picturefill on Chrome browser

This is the results from the speed test on the testing website with Picturefill applied on Chrome. Each image size is tested three times in three rounds. In each round the image is loaded 1st with all browser cache cleared and then the browser is refreshed two times for the 2nd and 3rd speed tests. Before the 1st load the browser cache is cleared. On the 2nd and 3rd refresh we will see the load time after the website has been cached on browser. The average is then calculated from each round and that is the data we will be focusing on.

The results from the pilot study will not be shown in this thesis, as they do not provide any further insight into the effect of responsive images than the final results. After the pilot study I needed to do some fundamental changes to the experiment that will be discussed in the next chapter.
6 Progression

The results from the pilot study will probably not be far from the results of the experiment but there are some details that will have to be changed. First of all we want to add detail to the study but also there will be changes in the terms we are conducting the study. The following is a list of necessary changes and then we will go into more detail for each change:

- More variation in network bandwidth
- Add website with 10 images
- Add more breakpoints (image sizes)
- Drop the requirements and focus on the results

More variation in network bandwidth
The pilot study tested the website and techniques on one bandwidth, 850 kbps, which mimics the connection speed of a 3G connection in good conditions. The final study will also do tests on faster connection that mimics average Wi-Fi connection (see Table 6.1). Changing the bandwidth will show us the effect a faster bandwidth will have on the responsive image techniques and tell us if they are more or less affective on slower networks. Applying responsive images can possibly have different effect on websites loading time for users depending on their bandwidth.

<table>
<thead>
<tr>
<th>Bandwidth (kbps)</th>
<th>Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>40960</td>
<td>Wi-Fi, Average case</td>
</tr>
<tr>
<td>850</td>
<td>3G, Good connectivity</td>
</tr>
</tbody>
</table>

Table 6.1 - Bandwidths tested

Add website with 10 images
In the experiment we will test two versions of the testing website, one version with 1 image and another with 10 images. Responsive images are probably more useful on websites with more images because the bigger the proportion images take of the whole size of the website the more effect the responsive images will have on the loading time.

Add more breaking points (image sizes)
In the pilot study the website started with an image in of 1920*1200px and then was scaled down to 960*600px and 480*300px. The break points are 1920px, 960px and 480px. The final experiment will test the techniques with 5 break points:

- 1920px
- 1440px
- 960px
- 720px
- 480px

This adds two more breaking points to the study, 1440 and 720. This means the images in the testing web site will also be scaled down to 1440*900px and 720*450px. By having more
break points in our test we cover more densely the range of screen sizes that are used for browsing the Internet.

**Drop the requirements and focus on the results**

In the pilot study I made requirements (in media queries) that the implementation of the responsive image techniques had to follow before the test (see Chapter 4.2). This approach assumes certain environment that limits the amount of computer systems this test will apply to. By doing this we are not focusing anymore on low- or high-resolution screens or small or big screens. By focusing on the results of the implementation we will make the test much more general in its approach and more researchers and developers can possible make use of the results. These changes will become clearer in the next chapter about the changes in implementation.

### 6.1 Changes in implementation

Some minor changes are done on the website and how the responsive image technique are implemented into the website. For Adaptive Images the cookie in the HTML header was changed:

```html
<script>
document.cookie = 'resolution=' + Math.max(screen.width, screen.height) + '; path=/';
</script>
```

This is the default code provided at adaptive-images.com. Before we used the alternative JavaScript that also sent information about device pixel density. The new break points are then added to the $resolutions array in adaptive-images.php:

```php
$resolutions = array(1440, 960, 720, 480);
```

For Picturefill the HTML markup is changed (see Table 6.2).

```html
<span data-picture data-alt="Calendula flower">
  <span data-src="img/480/0.jpg" data-media="(max-width: 480px)"></span>
  <span data-src="img/720/0.jpg" data-media="(min-width: 481px) and (max-width: 720px)"></span>
  <span data-src="img/960/0.jpg" data-media="(min-width: 721px) and (max-width: 960px)"></span>
  <span data-src="img/1440/0.jpg" data-media="(min-width: 961px) and (max-width: 1440px)"></span>
</span>
<noscript>
  <img src="img/mg/1920/0.jpg" alt="Calendula flower">
</noscript>
```

**Table 6.2 - HTML markup for Picturefill**

This markup will load different image sizes for different screen sizes defined with the media queries. `data-media="(min-width: 721px) and (max-width: 960px)"` will for example load `img/960/0.jpg` when the browser window is `721px – 960px` in width.
Both Adaptive Images and Picturefill have now been implemented without any special support for high pixel density screens. They will however return the same images, plus the extra two image sizes we added.

As mentioned before there will be two versions of the testing website with 1 and 10 images each. The images will be named from 0.jpg to 9.jpg. The images are scaled, compressed and saved in Photoshop. The images are then put in folders named after the width of the images. For example: Image 6.jpg that is 1440*900px will go to img/1440 folder. The folder structure for both websites is shown in image 6.1.

Figure 6.1 - The folder structure
7 Evaluation

In this test we are trying to figure out if responsive images decrease the loading time of websites that contain images. Two responsive image techniques are applied to testing website and then various tests are done. The results indicate that responsive images can reduce the loading time when website has much image data and the user is browsing on slow bandwidth. If the website does not have many images and/or the user is browsing on average Wi-Fi connection then the responsive image techniques become less useful, or even harmful.

7.1 Before testing

Before starting the experiment certain actions where performed to minimize the risk of our experiment being affected by things other than the variables we control. The most important thing is making sure we have control over cache both on browser and server side. Here is list of our procedure before testing:

1. Turn off all browser extensions
2. Turn off all unused programs
3. Clear browsing history
4. Turn on Chrome DevTools
5. Turn on Firebug

We turn off all extensions and programs because they could be using the bandwidth behind the scenes. To clear browser history we find the “Clear browsing data” dialogue and clear everything “from beginning of time”. Before we load the page we also have to turn on the development tools in the browsers (step 4 and 5) because otherwise the load time is not recorded and we do not get any results.

7.2 The experiment

The website load time was measured as described in the pilot study (see Chapter 5.2). Every number presented here is the average load time of three recordings. Every responsive image technique and image size is loaded once with browser cache cleared and then the browser is refreshed two times, denoted with 1st, 2nd and 3rd. To begin with the whole study results will be presented in two tables, 7.1 and 7.2, where each table represents different bandwidth. In the Analysis chapter we will dive into the numbers in more detail and present the data in different graphs and tables to analyse and understand the results better.
## Table 7.1 - Results on 40960 kbps bandwidth

<table>
<thead>
<tr>
<th>1 img website</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>Image size</th>
<th>10 img website</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>Image size</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chrome</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic</td>
<td>1.11</td>
<td>0.75</td>
<td>0.88</td>
<td>480</td>
<td>Basic</td>
<td>2.03</td>
<td>1.20</td>
<td>1.00</td>
<td>480</td>
</tr>
<tr>
<td>Adaptive Images</td>
<td>1.51</td>
<td>1.36</td>
<td>1.25</td>
<td>480</td>
<td>Adaptive Images</td>
<td>4.04</td>
<td>4.18</td>
<td>3.83</td>
<td>480</td>
</tr>
<tr>
<td>Picturefill</td>
<td>1.22</td>
<td>0.94</td>
<td>0.94</td>
<td>480</td>
<td>Picturefill</td>
<td>2.35</td>
<td>1.35</td>
<td>1.48</td>
<td>480</td>
</tr>
<tr>
<td>Basic</td>
<td>1.34</td>
<td>1.09</td>
<td>0.73</td>
<td>720</td>
<td>Basic</td>
<td>2.15</td>
<td>1.23</td>
<td>1.16</td>
<td>720</td>
</tr>
<tr>
<td>Adaptive Images</td>
<td>1.36</td>
<td>1.41</td>
<td>1.33</td>
<td>720</td>
<td>Adaptive Images</td>
<td>4.04</td>
<td>4.13</td>
<td>5.07</td>
<td>720</td>
</tr>
<tr>
<td>Picturefill</td>
<td>1.29</td>
<td>0.93</td>
<td>0.91</td>
<td>720</td>
<td>Picturefill</td>
<td>2.28</td>
<td>1.32</td>
<td>1.38</td>
<td>720</td>
</tr>
<tr>
<td>Basic</td>
<td>1.25</td>
<td>0.74</td>
<td>0.78</td>
<td>960</td>
<td>Basic</td>
<td>2.25</td>
<td>1.10</td>
<td>1.18</td>
<td>960</td>
</tr>
<tr>
<td>Adaptive Images</td>
<td>1.51</td>
<td>1.34</td>
<td>1.31</td>
<td>960</td>
<td>Adaptive Images</td>
<td>4.20</td>
<td>3.95</td>
<td>4.29</td>
<td>960</td>
</tr>
<tr>
<td>Picturefill</td>
<td>1.41</td>
<td>0.92</td>
<td>0.92</td>
<td>960</td>
<td>Picturefill</td>
<td>2.72</td>
<td>1.37</td>
<td>1.76</td>
<td>960</td>
</tr>
<tr>
<td>Basic</td>
<td>1.31</td>
<td>0.73</td>
<td>0.84</td>
<td>1440</td>
<td>Basic</td>
<td>2.58</td>
<td>1.03</td>
<td>1.00</td>
<td>1440</td>
</tr>
<tr>
<td>Adaptive Images</td>
<td>1.92</td>
<td>1.86</td>
<td>1.83</td>
<td>1440</td>
<td>Adaptive Images</td>
<td>4.29</td>
<td>4.48</td>
<td>4.21</td>
<td>1440</td>
</tr>
<tr>
<td>Picturefill</td>
<td>1.58</td>
<td>0.97</td>
<td>0.91</td>
<td>1440</td>
<td>Picturefill</td>
<td>3.14</td>
<td>1.35</td>
<td>1.36</td>
<td>1440</td>
</tr>
<tr>
<td>Basic</td>
<td>1.69</td>
<td>0.75</td>
<td>0.73</td>
<td>1920</td>
<td>Basic</td>
<td>2.68</td>
<td>1.08</td>
<td>1.23</td>
<td>1920</td>
</tr>
<tr>
<td>Adaptive Images</td>
<td>1.91</td>
<td>1.80</td>
<td>2.05</td>
<td>1920</td>
<td>Adaptive Images</td>
<td>4.71</td>
<td>4.26</td>
<td>5.03</td>
<td>1920</td>
</tr>
<tr>
<td>Picturefill</td>
<td>1.78</td>
<td>1.01</td>
<td>1.05</td>
<td>1920</td>
<td>Picturefill</td>
<td>3.24</td>
<td>1.34</td>
<td>1.30</td>
<td>1920</td>
</tr>
<tr>
<td><strong>Firefox</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic</td>
<td>1.27</td>
<td>0.74</td>
<td>0.76</td>
<td>480</td>
<td>Basic</td>
<td>2.06</td>
<td>1.02</td>
<td>0.99</td>
<td>480</td>
</tr>
<tr>
<td>Adaptive Images</td>
<td><strong>2.13</strong></td>
<td>1.41</td>
<td>1.64</td>
<td>480</td>
<td>Adaptive Images</td>
<td><strong>4.47</strong></td>
<td>3.99</td>
<td>3.73</td>
<td>480</td>
</tr>
<tr>
<td>Picturefill</td>
<td>1.34</td>
<td>0.87</td>
<td>0.88</td>
<td>480</td>
<td>Picturefill</td>
<td>2.25</td>
<td>1.38</td>
<td>1.38</td>
<td>480</td>
</tr>
<tr>
<td>Basic</td>
<td>1.32</td>
<td>0.83</td>
<td>0.71</td>
<td>720</td>
<td>Basic</td>
<td><strong>2.22</strong></td>
<td>1.09</td>
<td>1.18</td>
<td>720</td>
</tr>
<tr>
<td>Adaptive Images</td>
<td><strong>2.06</strong></td>
<td>1.53</td>
<td>1.44</td>
<td>720</td>
<td>Adaptive Images</td>
<td><strong>4.93</strong></td>
<td>3.95</td>
<td>3.96</td>
<td>720</td>
</tr>
<tr>
<td>Picturefill</td>
<td>1.38</td>
<td>0.88</td>
<td>0.88</td>
<td>720</td>
<td>Picturefill</td>
<td>2.46</td>
<td>1.34</td>
<td>1.41</td>
<td>720</td>
</tr>
<tr>
<td>Basic</td>
<td>1.40</td>
<td>0.77</td>
<td>0.70</td>
<td>960</td>
<td>Basic</td>
<td>2.40</td>
<td>0.98</td>
<td>1.03</td>
<td>960</td>
</tr>
<tr>
<td>Adaptive Images</td>
<td><strong>2.09</strong></td>
<td>1.72</td>
<td>1.77</td>
<td>960</td>
<td>Adaptive Images</td>
<td><strong>5.11</strong></td>
<td>4.66</td>
<td>4.33</td>
<td>960</td>
</tr>
<tr>
<td>Picturefill</td>
<td>1.71</td>
<td>0.88</td>
<td>0.97</td>
<td>960</td>
<td>Picturefill</td>
<td>3.02</td>
<td>1.46</td>
<td>1.39</td>
<td>960</td>
</tr>
<tr>
<td>Basic</td>
<td>1.55</td>
<td>0.71</td>
<td>0.67</td>
<td>1440</td>
<td>Basic</td>
<td>2.87</td>
<td>1.08</td>
<td>1.08</td>
<td>1440</td>
</tr>
<tr>
<td>Adaptive Images</td>
<td><strong>2.13</strong></td>
<td>1.75</td>
<td>1.83</td>
<td>1440</td>
<td>Adaptive Images</td>
<td><strong>5.25</strong></td>
<td>4.55</td>
<td>4.96</td>
<td>1440</td>
</tr>
<tr>
<td>Picturefill</td>
<td>1.57</td>
<td>0.89</td>
<td>0.87</td>
<td>1440</td>
<td>Picturefill</td>
<td>3.40</td>
<td>1.44</td>
<td>1.48</td>
<td>1440</td>
</tr>
<tr>
<td>Basic</td>
<td>1.60</td>
<td>0.67</td>
<td>0.69</td>
<td>1920</td>
<td>Basic</td>
<td>3.06</td>
<td>1.31</td>
<td>1.02</td>
<td>1920</td>
</tr>
<tr>
<td>Adaptive Images</td>
<td>2.17</td>
<td>2.31</td>
<td>2.17</td>
<td>1920</td>
<td>Adaptive Images</td>
<td>5.06</td>
<td>5.43</td>
<td>5.07</td>
<td>1920</td>
</tr>
<tr>
<td>Picturefill</td>
<td>1.98</td>
<td>0.90</td>
<td>0.89</td>
<td>1920</td>
<td>Picturefill</td>
<td>2.96</td>
<td>1.41</td>
<td>1.43</td>
<td>1920</td>
</tr>
</tbody>
</table>
These tables show the average load time of the websites, with and without responsive images. Table 7.1 shows loading times on bandwidth 40960 kbps that mimics average Wi-Fi
connection. Table 7.2 shows loading times on 850 kbps bandwidth. Both tables are split in to four. The left side is the loading times for 1 image website and the right is for 10 image website. The upper part (blue) of the tables is for loading times in Chrome and the lower (orange) is for Firefox.

The numbers marked with red are results of strange behaviour when Firefox loads the website with Adaptive Images for the first time. What happens is that the browser will start to load the images before the cookie is read on the server. When there is no cookie available, adaptive-images.php will check if the user is browsing on a mobile. If true, the smallest image in the $resolutions array (discussed in 4.4 and 6.1) is loaded, otherwise the biggest image is loaded. In this case the biggest image is loaded because we are testing on laptop so the results always shows loading time when loading the biggest version of the images. When the website is refreshed the browser will however load the right image size. This is what is called cookie race condition on adaptive-images.com. We are not sure why Firefox does this. This problem did not occur in Safari or Opera. More discussion on browsers is in chapter 7.3.

Just for testing purposes all image sizes where tested on the website without responsive images. In reality the website would only have one image size that would not change. To change the image size we had to change the markup of the website so right image would be loaded each time. By doing this we can see the impact of responsive images on all image sizes.

### 7.3 Analysis

In this chapter the results from the study will be analysed. The data from tables 7.1 and 7.2 are displayed in a different way to make it easier for the reader to understand the results and their implications (see Figure 7.1 and 7.2).
Figure 7.1 shows an important table and graph in this experiment. What graph shows is the loading time on the website, loading the biggest image (1920*1200px) compared against the website with responsive image techniques, loading the different image sizes we have defined. The “Difference” column shows the calculation (with responsive images) / (no responsive image technology). 1 means that there is no time difference. If the number is <1, the website loads faster with responsive images. If >1, then the website is loading slower with the responsive images implemented.

When the websites are loaded on Wi-Fi connection the loading time in most cases increases with responsive images technology applied. With Picturefill this is not so bad, and even better in two cases. Adaptive images on the other hand have a negative impact on the loading time, increasing the loading time from 51 – 76%.

The responsive images techniques obviously do better on slower network connection. In all cases the loading time decreases when smaller image than the original is loaded. This is expected with the responsive image technologies applied as the same image is loaded, plus the extra code it needed to implement them.
Figure 7.2 - 1 image vs. 10 images – 850 kbps - Chrome

Figure 7.2 shows the difference between having 1 image and 10 images on the website on a 3G bandwidth. When there is only one image on the website responsive image technologies have less effect. 10 images on 850 kbps connection load much faster with the responsive image technologies applied.
Here we are looking at comparison between Chrome (blue) and Firefox (orange) loading the 10 image website versions at 850 kbps. The first load time is always longest, which is normal, as the content has been cashed in the second and third refresh. Basic (no responsive image website) and Picturefill are handled well by the browsers in regards to caching data. Adaptive images on the other hand will not cache the images so the user will have to reload them each time they visit the page. As discussed before, Adaptive Images load the biggest image first every time on Firefox (denoted with ! in the graph). Apart from handling Adaptive Images the browsers are fairly similar, Chrome being slightly faster.
7.4 Result summary

There is no absolute winner in this comparison. Each implementation of the website had its pros and cons. The website with no responsive images is of course easiest to implement and there is no need for markup change. Using Adaptive Images and Picturefill seems to only be feasible for users who are browsing on low bandwidth and on websites with more images. If the bandwidth is sufficient the responsive image technologies will do little to Decrease the load time and even increase the loading time as we saw Adaptive Images do in figure 7.1.

Adaptive Images had the big advantage of not having to do any changes in the markup. It is fairly easy to implement but it is limited, as you do not get total control of every image if you want to scale them differently or have art direction. You have to define one rule to fit all images. A big disadvantage is that the images would not be cached in users browsers so images had to be loaded every time.

With the Picturefill technique there was a sense of better control over the changes we wanted to make because you can do art direction on individual images on the website. It is convenient to be able to use media queries as when working with CSS. Art direction was no problem and you have total control over your images. The plugin is lightweight and the images are cached on users browser. The big disadvantage is the additional markup. For only one image you have to add quite some markup to your web page and with many images or lots of legacy code, managing your images can become hard and tedious job.

The results show us that using these responsive image techniques are not perfect. Finding the right solution for a website will depend on the content of the website and the bandwidth the users have available.
8 Conclusion

In this thesis two well known and widely used responsive image techniques have been tested and compared with website with no responsive images implemented. The results are not as promising as was postulated in the hypothesis. The load time of the website decreased when the responsive image techniques where implemented, but only if the website had many images and was loaded on slow bandwidth. There are a lot of things that have to be considered when looking for the best solution for responsive images. Adaptive Images and Picturefill use very different approach to make images responsive.

In their paper Performance comparison of Dynamic Web Technologies, Titchkosky, Arlitt, and Williamson compare how fast server technologies such as PHP and JAVA, render dynamic content. The results in their paper indicated that JAVA was fastest at this certain task. If I

It is hard to find the right solution for responsive images. In this thesis we have only tested two solutions that we choose because of how adopted and accessible and how they approached the problem. There are other solutions that would be interesting to test. The difficult part is to test them in a fair way because the techniques are so different from each other. Some use third party servers to detect and scale images while other plugins focus on providing images for high-resolution screens, like HiSRC.

So what solution would we choose for our next web project? Well, that depends. If the site does not have too much of images it is probably best to skip any additional code to your project and just keep it simple without responsive image. If you, however, want to create website with many images you can certainly benefit from implementing responsive images. We would probably choose Adaptive Images just because you can add and remove the it without any additional changes to your website and it is backward compatible. Neither solution feels like a long-term solution for responsive images. When a better solution comes along, removing Adaptive Images would cause fewer problems.

In this experiment I was able to get accurate results based on repeatable experiment. The experiment built on similar ground as the experiment in the paper Performance comparison of Dynamic Web Technologies by Titchkosky, Arlitt, and Williamson. In the paper they compare how fast server technologies such as PHP, Pearl and JAVA render static and dynamic content on webpage. They used benchmark test similar to what I did in this experiment where they got accurate quantitative data to compare the technologies.

8.1 Discussion

8.1.1 Social and ethical issues

As responsive image techniques are built today they will not create any social or ethical issues because they are built on top of current technologies. They can use cookies or user agent string, but the cookies or user agent strings can have social or ethical issues concerning privacy, but not responsive images. There are ideas of making more information available about users to make responsive images more efficient (discussed in more detail in chapter 8.4). This information could be for example users bandwidth that could be made available in the user agent string. With this information available it would be easy to detect slow bandwidth and load smaller image sizes depending on that. Making more information
available about users can however create some privacy concerns as more personal information makes it easier for others to spy on users.

Other issue that could arise from using responsive images is that even though users are browsing on a small screen or slow bandwidth they could possibly want to download the whole image, like when viewing map for example.

8.1.2 Critical perspective
When recording the loading times in the study the loading times usually had some fluctuation. Even the average of three load times could change. One reason this could be is that the testing website was kept on a public server and any difference in traffic to the server could potentially have some effect on the test. Solution to this could have been to set up our own private server that would be less affected by traffic on the network. This is just a theory though. There where some fluctuations but I am not sure why exactly.

The test could also have been more detailed. Adding more bandwidths, browsers or images to our test could possibly have added more detail to the study. The problem was that it takes quite some time to make the test and with every new variable added the time it takes to make the test grows exponentially. For example, adding one extra bandwidth speed to the test would increase the testing time by around 50% as the number of iterations we would have to do would increase by 1.5. All loading times recorded where done manually and the results where documented by hand into an Excel document. A testing tool that would automate some or all of the process could allow for a much easier, faster and more detailed study.

8.1.3 Benefits to society
The main purpose of responsive images is to make browsing faster and possibly cheaper by reducing webpage size. With responsive images it is possible to make the website adapt to users device, much like responsive web design, hopefully making the browsing experience more enjoyable. In places where the Internet access is more limited responsive images can be of big help for the users by reducing the size of websites. It is also common that users have to pay for their download and responsive images can certainly help reduce the cost of browsing.

8.2 Future Work
This thesis has given us much better insight into strengths and weaknesses of responsive images. There is a lot to be gained in terms of loading time for users on slow Internet connection. By implementing responsive images to website the loading time can be reduced dramatically. The issue is still being discussed and people are still trying to find more permanent and future-proof solution. There are more solutions available that could be interesting to test, for instance HiSRC, Sencha and Foresight.js (Schmitt, C. 2013, Sencha.com, 2014, Bradley, A. 2012). The thesis did not test all available responsive image techniques and trying out more approaches to this problem could be interesting.

High-resolution (retina) displays are becoming increasingly more popular. Many high-resolution displays have double the pixel density and devices with up to 3 times the pixel density are becoming available for users. It would be interesting to study responsive image techniques that focus on providing higher resolution images for retina displays.
There is also other factor that is hard to deal with but is also an important factor when deciding what image to provide to user; the bandwidth. As browsers are built now there is no good way to determine the user bandwidth. One solution used by HiSRC is to send a file to the user and measure how long time it takes. This costs extra HTTP request and has negative effect on loading time. In our test the techniques where only really effective when the website was tested on less bandwidth. By knowing users bandwidth the website could be implemented in a way that responsive images would only be used if the user is browsing on low bandwidth.

One possible solution to this problem and others is that the browsers could send more information about individual user environment, maybe in the user agent string. This could be information about users bandwidth, screen size and pixel density. With this information it would be possible to approach this problem from a whole different angle. The available responsive image techniques are interesting but if the browsers would make more information available about users it would be really interesting to try and research different approaches of how to use this data to implement truly responsive website. By adding this information to the user agent string there would be no need to create and send cookie before deciding which image size is appropriate. This information would be available as soon as the server gets the HTTP request and appropriate image could be served right away.

To make this a reality we would possibly have to see change in the HTTP protocol or changes in the W3C standards. An interesting study would be to change open source browser such as Firefox or Chromium to provide this information in the user agent string. The effect of this implementation could then be studied and analysed.

As discussed in chapter 8.1 responsive images can create a problem where users want to download the full size image even though they are browsing on small screen or on slow bandwidth. One solution that could be interesting to test is to allow choosing image quality, in browser’s settings for example. The options could be for example: Auto, Low, Medium, and High. Auto would provide images depending on users screen size, pixel density or bandwidth while the other options would provide the image quality you want regardless of your browsing environment. This would allow users to reduce download on expensive Internet connection even though they have fast connection and big display. Users would also be able to download high-resolution images even though they have slow bandwidth.

The possibilities are many and we will certainly see this field continue to evolve fast over the next years. Responsive images are still unsolved puzzle in web development and it will be interesting to watch where researchers will lead us in coming years.
References


case studies approach”. The American Biology Teacher 59(1).


Appendix

Testing website

index.html

```html
<!DOCTYPE html>
<html>
<head>
  <!-- Basic Page Needs -->
  <meta charset="utf-8">
  <title>Responsive Images</title>
  <!-- Mobile Specific Meta -->
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
  <!-- Scripts -->
  <!-- CSS -->
</head>
<body>
<div class="container">
  <div>
    <h1>Responsive Images | BASIC</h1>
  </div>
  <nav>
    <ul>
      <li><a href="../basic">Basic</a></li>
      <li><a href="../adaptive_images">Adaptive Images</a></li>
      <li><a href="../picturefill">Picturefill</a></li>
    </ul>
    <hr/>
  </nav>
  <div>
    <img src="img/1920/0.jpg" alt="Calendula flower">
  </div>
</div>
</body>
</html>
```

style.css

```css
html, body, div, span, h1, h2, h3, h4, h5, p, a, img, ul, li, nav, menu {
  padding: 0; margin: 0; border: 0;
  font-size: 100%; font: inherit; vertical-align: baseline;
}

body { color: #333; } 

h1 { font-size: 38px; margin: 14px 0 10px; } 

h2 { font-size: 24px; margin-top: 20px; }

hr { border: solid #ddd; border-width: 1px 0 0; clear: both; margin: 4px 0 10px; }

a, a:visited { color: #555; }

a:hover, a:focus { color: #000; }

ul { list-style: none outside; }

li { margin: 4px 14px 2px 0; float: left; }

img { width: 100%; height: auto; }

.container { position: relative; width: 100%; margin: 0 auto; padding: 0; }
```
Adaptive Images

index.html

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Responsive Images</title>
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
  <script src="adaptive-images.php"></script>
  <link rel="stylesheet" href="style.css">
</head>
<body>

<div class="container">

<h1>Responsive Images | ADIMG</h1>

<nav>
  <ul>
    <li><a href="../basic">Basic</a></li>
    <li><a href="../adaptive_images">Adaptive Images</a></li>
    <li><a href="../picturefill">Picturefill</a></li>
  </ul>
</nav>

<hr />

<!-- JPG image -->

</div>
</div>

</body>
</html>

adaptive-images.php

<?php
$resolutions = array(1440, 960, 720, 480);
$cache_path = "stuff/ri/1/adaptive_images/ai-cache";
$jpg_quality = 92;
$sharpen = TRUE;
$watch_cache = TRUE;
$browser_cache = 60*60*24*7;

$document_root = $_SERVER['DOCUMENT_ROOT'];
$requested_uri = parse_url(urldecode($_SERVER['REQUEST_URI']), PHP_URL_PATH);
$requested_file = basename($requested_uri);
$source_file = $document_root.$requested_file.$requested_uri;
$resolution = FALSE;

// Rest of file not included - Download on www.adaptive-images.com

.htaccess

<IfModule mod_rewrite.c>
Options +FollowSymLinks
RewriteEngine On
RewriteCond %{REQUEST_URI} !assets
RewriteCond %{REQUEST_URI} !ai-cache
</IfModule>
RewriteRule \.(?:jpe?g|gif|png)$ adaptive-images.php
</IfModule>

Picturefill

index.html

<!DOCTYPE html>
<html>
<head>
  <!-- Basic Page Needs -->
  <meta charset="utf-8">
  <title>Responsive Images</title>
  <!-- Mobile Specific Meta -->
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
  <!-- Scripts -->
  <script src="picturefill.js"></script>
  <!-- CSS -->
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <div>
      <h1>Responsive Images | PICFL</h1>
    </div>
    <nav>
      <ul>
        <li><a href="../basic">Basic</a></li>
        <li><a href="../adaptive_images">Adaptive Images</a></li>
        <li><a href="../picturefill">Picturefill</a></li>
      </ul>
    </nav>
    <hr />
    <!-- JPG image -->
    <div>
      <span data-picture data-alt="Calendula flower">
        <span data-src="img/480/0.jpg" data-media="(max-width: 480px)"/>
        <span data-src="img/720/0.jpg" data-media="(min-width: 481px) and (max-width: 720px)"/>
        <span data-src="img/960/0.jpg" data-media="(min-width: 721px) and (max-width: 960px)"/>
        <span data-src="img/1440/0.jpg" data-media="(min-width: 961px) and (max-width: 1440px)"/>
        <span data-src="img/1920/0.jpg" data-media="(min-width: 1441px)"/>
      </span>
      <noscript>
        <img src="img/mg/1920/0.jpg.jpg" alt="Calendula flower">
      </noscript>
    </div>
  </div>
</body>
</html>