Responsive Web Design Workflow

Timo Laak

University of Tampere
School of Information Sciences
Interactive Technology
M.Sc. thesis
Supervisor: Timo Poranen
December 2013

University of Tampere
School of Information Sciences
Interactive Technology
Timo Laak: Responsive Web Design Workflow
M.Sc. thesis, 43 pages
December 2013

Responsive Web Design Workflow is a literature review about Responsive Web Design, a web standards based modern web design paradigm. The goals of this research were to define what responsive web design is, determine its importance in building modern websites and describe a workflow for responsive web design projects.

Responsive web design is a paradigm to create adaptive websites, which respond to the properties of the media that is used to render them. The three key elements of responsive web design are fluid layout, flexible media and media queries.

As the numbers of mobile device users are constantly increasing, responsive web design has become an important method to improve mobile device user experience and accessibility in browsing the web.

The workflow to build responsive websites consists of eight cumulative and iterative steps, which are discovery, planning, content design, sketching, prototyping, visual design, testing and discussion.

As each web design project is unique and has different content, audience and goals, it is difficult to create a perfect workflow model. The responsive web design workflow described in this thesis is a recommendation and a collection of best practices for building responsive websites.

Keywords: responsive web design, web design, workflow, progressive enhancement, mobile first, mobile web, adaptive design, content first

Preface

As a web developer and designer, I have often wondered how difficult it is to work on a web development project and try to keep in mind the best known practices, while keeping to the schedule, staying within budget and making both the client and the management satisfied in the outcome of the project.

The goal of this thesis is to collect many well-known modern web design guidelines and practices into a responsive web design project workflow model, or rather guidelines, which is a good fit for responsive web design projects and helps us developers and designers to build a better user experience for the web.

Helsinki, 10 December 2013
Timo Laak

Table of Contents

1 INTRODUCTION

The primary design principle of the World Wide Web, by Tim Berners-Lee, is universality. (Lloyd, 2012) The Web is open by nature and it should be accessible from any browser and hardware that can connect to the Internet. It should be accessible also to people with disabilities and work with any form of information. (Berners-Lee, 2010)

Early web pages were simple HTML files, which automatically adapted to fit the width of the browser when opened. This means that the web is responsive by default and it's the designers that have been breaking this paradigm by placing content in non-flexible fixed-width containers. (Hume, 2011) The very first web page in the world was mobile ready and responsive, long before smartphones, tablets and the concept of responsive web design. It was republished in 2003 to celebrate the 20th anniversary of the World Wide Web. (Epstein, 2013) The page consisted of structured plain text with hyperlinks, which makes it incredibly simple to view in different devices. (Hay, 2013, p. 51)

The most common web design layouts we see every day when browsing the web, be it on a desktop or on a mobile device, are based on typographic grids, popularized by graphic designers of the mid-1900s. A typographic grid is a rational system of rows and columns, which create modules where content can be placed. Although the typographic grids were originally designed for printed media, they have been the foundation of web designs and used even today as the basis of modern web designs. (Marcotte, 2011)

The flexibility of the web and inflexibility of typographic grids in fixed-width designs led to problems, when small screen devices started to gain popularity. Content could not be fitted on the screen anymore. The solution for this is responsive web design, which makes the typographic grid flexible and adaptive. Responsive web design is problematic in traditional waterfall projects, which sets demand for a different workflow. (Boulton, 2012)

It is necessary to understand the concept of responsive web design and why its importance has increased during the recent years to be able to conduct a successful responsive web design project. This literature review opens the different elements and provides background information about the paradigms and fac-

tors behind responsive web design. The concept of responsive web design and its three main elements are explained in Chapter 2. The cause for rising importance of responsive web design, the *evolution of mobile web* is discussed in Chapter 3 and the modern web design paradigms that make up the grounds for responsive web design workflow are discussed in Chapter 4. The *Responsive Web Design Workflow* showcased in Chapter 5 is based on eight different phases by Salminen (2013) and each of these phases is based on the tools and workflows of Hay (2013). I have included my own remarks on working in responsive web design projects in these phases and the surprising problems there sometimes may appear. I have also introduced a few typical responsive web design issues that can be detected and fixed with the help of prototyping in Chapter 5.7. Finally the summary in Chapter 6 lists many issues that may arise in responsive web design projects preventing the successful outcome.

2 WHAT IS RESPONSIVE WEB DESIGN?

Responsive Web Design, often abbreviated as RWD, is a web design and development paradigm, which uses web standards to create designs, which are flexible and adaptive to media that is used to render them. (Marcotte, 2010)

Roots of the term Responsive Web Design are in discipline called *Responsive Architecture* (Marcotte, 2010), introduced by Nicholas Negroponte in 1970s. A responsive architecture is the natural product of integrating computing into spaces and structures and thus embedding ubiquitous technology in buildings, which respond to data received via sensors and user input. (Sterk, 2005, pp. 225-227)

2.1 Why Websites Should Be Responsive?

User interfaces in the digital world are filled with analogies and metaphors to the physical world. There's a desktop on our computer screen. We store our digital files in folders and photos in albums. When we want to get rid of something, we move it to trash.

The World Wide Web, or simply just the web, is no different. The web is a network of websites. Each website consists of web pages. A web page is a metaphor to a physical printed page. Although they are both pages, they are of a very different nature. In the physical world, the page dimensions are always fixed. There are many paper size standards, but when a designer is creating a new design for printed media, they already know the size of the paper and the limitations can be taken into account accordingly.

The web is a totally different media. A web page can be viewed in several different browser applications on countless combinations of screen sizes and window sizes, screen resolutions and browser capabilities. These can be called contexts. There are different sized laptop and desktop computer screens, different sized screen resolutions, tablet screens, user defined custom styles, mobile devices with tiny screens and joystick navigation only, mobile devices with big touch screens, portable game consoles, fast modern browsers on modern computers, slow old browsers on old computers, huge billboard screens in shop-

ping malls, big television screens with a greater viewing distance, text based browsers, screen readers for visually impaired, web crawlers and indexing robots for search engines and, of course, a possibility to print the page on a sheet of paper, which can be any of the common paper size standards, in color or gray scale. The plethora of different browsers and devices makes it very difficult to design for the web. Grouping each context into different target groups is not practical, because the number of possible groups grows exponentially (Mohomed, Cai, Chavoshi, & de Lara, 2006, p. 44). Responsive design makes it possible to tailor the website for multiple different medias by using one content instead of building separate websites for each media.

The traditional practice has been to use a minimum target resolution and create fixed-width websites. This one-size-fits-all approach is problematic, because it ignores users with much larger and smaller screen sizes. It also creates a need to periodic re-designs, when new devices enter the market, changing the most common resolution. (Gardner, 2011) Besides just the most common resolution, typically also only the browsers with a significant market share have been targeted. These browsers have included mainly desktop browsers, e.g. Internet Explorer and Firefox, but as the numbers of mobile users are growing, this practice will alienate website visitors who are not using a supported browser with a supported resolution device. Some institutions have launched native mobile applications and built mobile websites to provide their content to all visitors, but the introduction of the iPad changed the game again by creating a new device group: tablets. (Joly, 2012) There are so many different devices and device categories, that it is impractical and even impossible to create separate websites for each of them. And the number of device categories keeps changing. There are laptops that support slate (tablet) mode, booklets and convertibles, which often are primarily mobile tablet devices, but have many laptop or even desktop computer characteristics and they may support also touch, pen and gesture-based input. There are also physically very large high-resolution displays, which go far beyond current desktop setups. (Nebeling & Norrie, 2013, p. 510).

Mobile browsing has been predicted to grow and outpace desktop browsing since the early 21th century (Schilit, Trevor, Hilbert, & Koh, 2001, p. 122) and the final shift to mobile dominance is expected to happen between 2013 and 2015 (Marcotte, 2010). As also tablet sales have been predicted to out ship PCs

and laptops in 2014 (Joly, 2013), the traditional target resolution approach can no longer be recommended. Web designs have to be adaptive and respond to different types of rendering media. Responsive web design has also changed the focus from being device centered to being context centered. A typical use case for looking up information such as addresses and phone numbers now happens in mobile context (Fox, 2012, pp. 119-122). Therefore a modern website design can be recommended to be built as responsive and adaptive to all sorts of rendering media and contexts.

2.2 Elements of Responsive Web Design

The three elements of responsive web design include a *fluid layout*, *flexible media* and *media queries*. Unlike fixed layout, which is specified in static units (pixels, points, inches), a fluid layout is specified in relative units, percentages (Nebeling & Norrie, 2013, p. 511). A fluid layout width is relative to the browser window (viewport) width and each child element is relative to the width of its parent element. Flexible media (images, embedded video or plugin content) can also be achieved by relative units, which specify the width of the media inside the parent container element. Third element is media queries, which can be used to serve different *Cascading Style Sheets* (CSS) in a different viewing context. (Gardner, 2011)

A fluid layout, also known as a liquid layout, is the opposite of fixed layout. As a fixed layout has a fixed width wrapping element, it will be of the same width for all visitors regardless of the device they are using to view the website, although the elements inside the wrapper can have a fluid width. A fluid layout has a fluid width wrapper and fluid width elements inside the wrapper with their width set as percentages so that it will adjust to the available screen resolution, or more precisely, viewport width. (Knight, 2009) This makes it perfect for creating designs for varying viewport sizes. A 90% wide fluid layout will always take 90% of the available viewport width and works in a similar way across all browser applications.

Flexible media is achieved simply by setting images to occupy no more than 100% of the width of the containing element with the *max-width: 100%* CSS property. Normally an image will take space according to its pixel dimensions,

but max-width property prevents images becoming larger than the specified max-width value (Bos, Çelik, Hickson, & Wium Lie, 2011). This also preserves the correct aspect ratio. The *max-width: 100%* rule can also be applied to other media elements, such as video and rich media. (Marcotte, 2011, p. 45).

Media queries were introduced as part of CSS3 standard by the World Wide Web Consortium (W3C) (Rivoal, Lie, Çelik, Glazman, & van Kesteren, 2012), the main international standards organization for the World Wide Web. Media queries are grouping directives for CSS, which are loaded when the condition matches the query. As an example, when browser window width matches the given width in the media query, browser uses the corresponding styles. The media queries make it possible to respond to different screen sizes and resolutions, and style the content accordingly. Media query directives are called *breakpoints* and they are often defined in pixel units. An 800-pixel max-width breakpoint will trigger, when the viewport width increases past 800 pixels by using the following media query:

```
@media screen and (max-width: 800px) {
    /* CSS rules for this query */
}
```

2.3 Adaptive Design

Responsive Web Design is adaptive by nature, but technically it is still very limited and adapts only to screen resolution. However, there is a growing need to adapt also to other conditions including network speed, touch interface, browser and operating system capabilities and viewing distance.

More advanced adaptive approach to create flexible media is to take into account the physical size and dimensions of the media. When a large image is set to occupy the full width of a small container to fit nicely inside a small viewport, its physical size still stays the same and creates a performance problem if the network connection is slow and if the processing power of the device is limited. Using techniques, where the screen width is used as a trigger to select a correctly scaled image, can solve this problem. Currently there are no standardized universal cross browser techniques to achieve this, but the proposed HTML *<picture>* element (Cáceres, Marquis, Weiss, & Bateman, 2013) will most

likely solve this problem in the near future. Other techniques include using *SVG* (Scalable Vector Graphics) images, JavaScript based *polyfills*, proxy servers or server side solutions (Alexander, 2013; Weyl, 2013).

A polyfill is a technique that replicates and mimics an API, *application programming interface*, not natively supported by the browser, thus providing a fallback mechanism for old browsers (Sharp, 2010). Polyfills are often used to bring user experience of modern browsers to older browsers, which do not support modern HTML5 or CSS3 features.

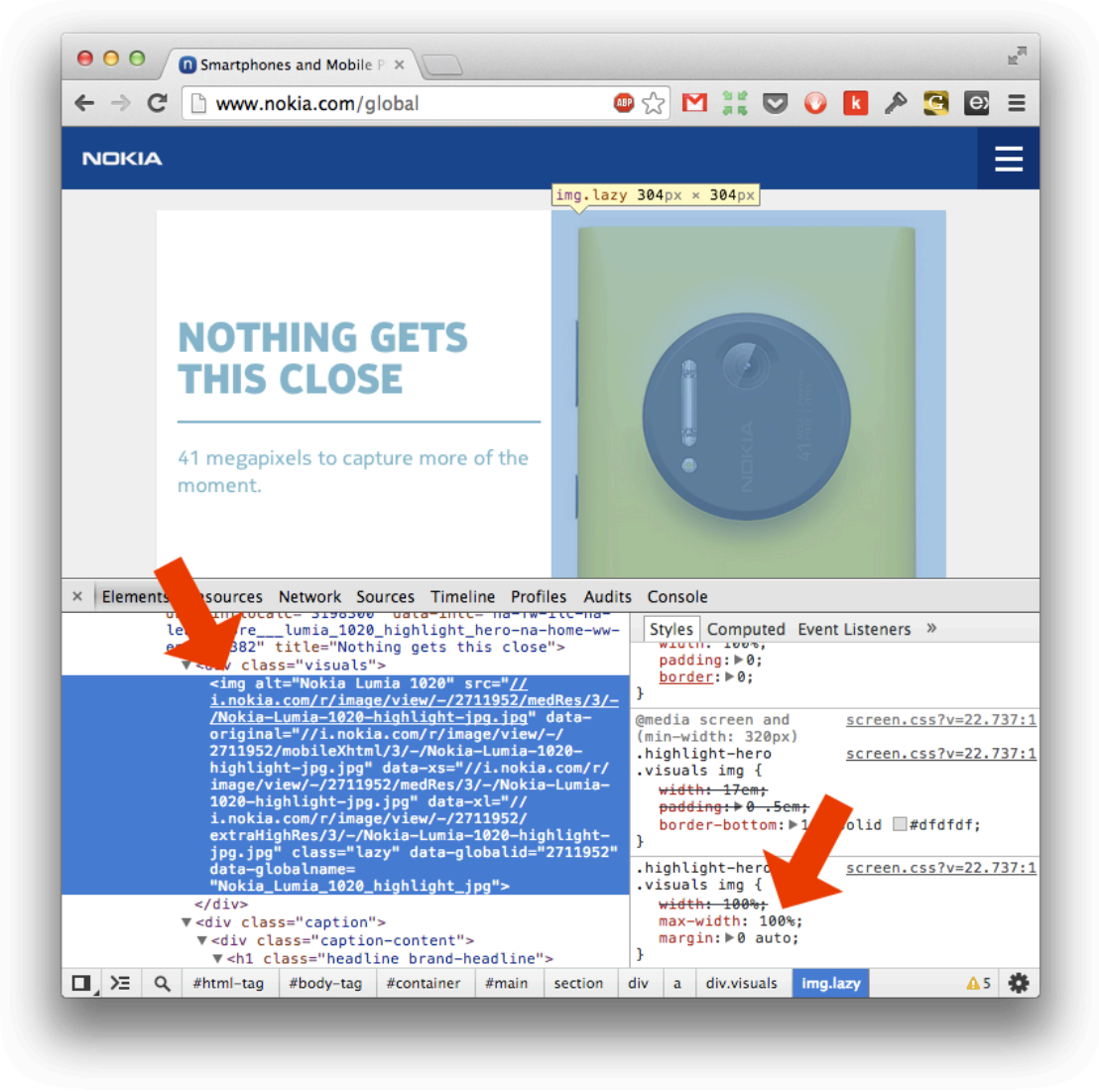

Figure 1. Nokia.com global website source code details in Google Chrome Developer Tools

The nokia.com global website (http://nokia.com/global) uses a technique, which downloads a default image for all browsers and sets alternative image

paths in data-attributes: *data-original, data-xs, data-xl.* The attribute names give a clue about the image size, but naturally the actual images defined in these attributes can be anything. In this case, the default image defined in the *src* attribute is the same as the image defined in *data-original* attribute, while the *data-xs* image is the smallest and the *data-xl* image the largest. The largest image will be loaded only when the viewport size exceeds 1024 pixels, thus making it available only when viewport resolution is big enough to show the image in full size. The HTML tag and its attributes can be seen highlighted on the *Elements* pane of the Google Chrome browser *Developer Tools* in Figure 1. Note the *max-width: 100%* property in the *Styles* pane on the right side of the developer tools.

Adaptive design itself is a very wide topic and can include not only adapting to rendering media, but also adapting to users and their abilities. An adaptive website could e.g. adapt its color palette and contrast settings to visually impaired users when it detects a certain text size setting or zoom level, but large parts of this kind of adaptation can be achieved by supporting CSS style sheets provided by the user.

Adaptive design does not only mean adapting to different browsers and viewports, but also formatting content in a way that enables sharing and distributing the content to any platform. There are not only different browsers on desktop or mobile devices, but also embedded browsers, *web views*, inside applications. Typical examples of this are Facebook and Twitter native mobile applications, which allow opening hyperlinks in web views inside the applications. The Twitter application also can show content embedded in the timeline and then there is the *Google Glass*, a futuristic head mounted display that can show web pages. This is beyond the responsive web design topic, but adaptive content is already present and implemented in modern applications and websites. (McGrane, 2012, p. 36)

3 MOBILE WEB

The term mobile web can be understood as a separate web, consisting of websites designed exclusively for mobile devices, or in this case, accessing the web from a mobile device. While there still are websites and services, which are designed solely for mobile devices and use cases that happen in a mobile context, the technology has grown mature enough to access almost any website in a mobile web browser. In the early days of mobile web, the situation was very different as it is today.

3.1 Accessing Desktop Websites from a Mobile Device

A separate mobile website or application often has very limited features compared to full-flavored desktop version. The cost of producing a mobile version has been a barrier to support more than the basic functionalities, which has often left enterprise software users without the ability to perform their work on a mobile device. Many different solutions to this problem have been introduced, but the most typical solution is a proxy server based application, which transforms and re-authors the desktop content for small screen devices. (Schilit et al., 2001, p. 122)

3.1.1 Early Solutions

Highlight system is a tool to re-author mobile web applications from existing websites developed by Nichols & Lau (2008). It is a Firefox web browser add-on, which enables users to create a working mobile version of a desktop website by selecting content and performing desired interactions. Its poor support for client-side JavaScript has probably made it obsolete, as numerous websites today are heavily dependent on JavaScript.

Another approach is automatic *server-side re-authoring*, which is based on device detection from HTTP request headers. Graphical elements can be omitted or compressed, content elements hidden and unhidden with JavaScript and tables converted into rows of text. (Artail & Raydan, 2005, pp. 368-373)

Digestor is a proxy, which splits a desktop web page into multiple smaller pages and creates navigational links between them. It is problematic on very small

screen devices, where the screen estate is limited and complicated structures are difficult to navigate. (Schilit et al., 2001, p. 122)

M-Links is a refactored interaction solution, a site navigation engine, for small screen devices. It divides web browsing into two categories, navigation and use. It provides a skeleton view of a website's content including the navigation links and content items. A link is marked with a folder icon and a content document is marked with a file icon. These items are presented to the user as a list, which is a typical navigation solution on a small screen device and can be used easily with four buttons. Navigation between items is similar to what can be found in the text-based Lynx browser. Opening a document triggers a list of available services, which are detected by the document's MIME type. These services can include an email, a fax or a language translation web service or a printer. M-Links was designed to work on "ancient" early 2000s smartphones, which could display only around ten 24-character lines of content. (Schilit et al., 2001, p. 123)

3.1.2 Modern Proxy Based Browsers

Opera Mini is a multi-platform lightweight mobile web browser that uses a proxy server to pre-process web pages and optimize their content for the small screen mobile device. It can automatically collapse long lists of links, which are often found in navigation elements and thus provide a much cleaner and less cluttered user interface for a small screen device. (Shrestha, 2007, pp. 187-194) Other similar browser applications are Nokia Xpress Browser, available for Asha and Lumia phones, and Amazon Silk Browser, pre-installed in Amazon's Kindle Fire tablet device. Amazon Silk uses a technique called cloud-acceleration, which runs on Amazons EC2 cloud computing platform and caches and optimizes content for the browser. (Schonfeld, 2011) Nokia Xpress Browser even decrypts encrypted and secure HTTPS data to optimize content, which has lead to privacy concerns. (Condliffe, 2013)

These techniques are mostly invented to enable mobile browsing on websites designed for desktop environment, which may not provide optimal user experience.

3.2 Mobile Web Revolution

On June 2007, Apple released the iPhone, which dramatically changed the web. Before that, browsing the web on a mobile device was often painfully slow and not very user friendly, because websites were mostly designed for the desktop environment and most phones had tiny screens, 12 button keyboards and joysticks. Text input was difficult and selecting links with the joystick was not the optimal way to navigate. Browsing a website designed for desktop required the user to scroll and position the window to find the desired content. (Chen, Ma, & Zhang, 2003, p. 225) The iPhone had a touch screen and a good mobile browser, *Safari*, which was able to view desktop websites on a 320x480 screen, which was relatively big at that time. The touch interface made it easy to scroll and zoom the content, interact directly with links and forms and browse the web in cafés and public transport. Between 2006 and 2009 AT&T's mobile data traffic increased nearly 5000%. During that time AT&T was the exclusive carrier of the iPhone in USA. (Wroblewski, 2011, pp. 10-11)

It has been estimated that by 2013 the number of mobile browsers in the Internet will outrun desktop computers. Global smartphone shipments were predicted to surpass PC shipments in 2012, but it actually happened two years earlier in the last quarter of 2010. (Wroblewski, 2011, pp. 7-8) It means that more and more people are browsing websites and using web applications on a mobile device: a phone, tablet or a hybrid of a tablet and a laptop computer. Pew Internet Research Center's report states that 74 percent of teens aged 12 to 17 are browsing the web on cell phones, tablets and other mobile devices, while the remaining 26 percent are using only their smartphones. (Joly, 2013)

As the share of mobile browsers started to rise, website owners had to pay more attention to mobile users and tailor their websites to be more mobile friendly. One common solution for this problem was, and still often is, offering a separate mobile website to mobile users, usually with limited functionality, different content and simpler layout. This solution has its drawbacks because it requires maintaining two separate websites instead of one. Detecting mobile devices by a user agent string or touch capabilities is prone to errors and some devices have screens as large as desktop and laptop computers, although they can also

be considered mobile devices. These devices include tablets and smartphones with very large screens.

3.3 Multi-Screening Context

The rise of mobile computing has not only made small screen support more important, it also affects how the web is being used. Web browsing does not happen only on desktop devices, mobile devices or tablet devices. It can happen on all of them simultaneously or the browsing can be started on one device and continued on another. This phenomenon is called multi-screening and it can be divided into two different modes: sequential screening, where user moves between devices and simultaneous screening, where multiple devices are used at the same time. ("The New Multi-screen World," 2012)

Context plays an important role in multi-screening. A mobile device enables web browsing in any location where a network connection is available. Browsing may happen at home, sitting on a sofa while watching TV, or it may happen in public transport, while commuting to work. Sometimes there is a need to use a specific service or app in several different contexts and devices, which requires not only multiple responsive user interfaces, but also means to synchronize the data so that it's always up-to-date.

As the context switches to mobile and multi-screening, a responsive website or web application has to support the new use cases, unless the majority of users are still using a desktop browser, which may be typical to intranets and relying on tightly controlled environments. Web analytics can be used to collect data about users and their primary means to access the website. If the majority of users are browsing a website on a mobile device, a mobile content strategy is recommended to create a smartphone friendly website. Google recommends responsive web design in their Zero Moment of Truth marketing research, but a mobile strategy may also include a separate mobile application in specific cases. (Leibtag, 2012)

Instead of relying in proxy servers and adaptive web browsers, a recommended solution for designing and building mobile websites is to create everything for mobile in the first place. This paradigm is called *mobile first* and it will be discussed in Chapter 4.1.

4 MODERN WEB DESIGN PARADIGMS

4.1 Mobile First

To create a great user experience in mobile context, it is recommended to design everything mobile first. This method also prepares the website content for the growth of mobile web and forces focus on essential content and functionality. (Wroblewski, 2011, p. 5) It is easier to design for a small screen and add more content dynamically when viewport size increases, than designing for a large screen and removing content when viewport size decreases. When there is space to fill, it is often filled with banners, promotions, calls to action and other non-essential content. A tiny mobile phone screen does not allow for this kind of content, because there simply isn't any room for it. Designing for a small screen requires knowledge of user behaviour on the site. Designers must understand what matters most and focus on the most important features for visitors, users, customers and business. The same principle could also be utilized in desktop website design, but when there is room, the different stakeholders often demand their content to be present on the website and the end result is chaotic and filled with interface debris and each piece of content is fighting for the user's attention. (Wroblewski, 2011, pp. 19-22)

Less content to view also means less content to load. Mobile networks are not as fast and reliable as wired connections and that makes the amount of data to be transferred very important. Data transfer may also be expensive, depending on the data plan. More data means slower page loads and higher cost. Designing for mobile is also designing for better performance, which will benefit desktop and tablet users too. (Wroblewski, 2011, pp. 22-23) Browsing the web on a large screen device doesn't always mean that the network connection is fast. The screen size cannot be taken as an indicator of a fast network connection or a fast processor. A modern powerful desktop computer may still use an often slow and lagging 3G network and many old computers with slow processors and limited memory still have large screens. A notable 74% majority will leave a website that does not load in five seconds or less. (Brown, 2013)

There are several techniques to improve website performance, and they can be divided into three main categories: minimize HTTP server requests, minimize

the data to be loaded and optimize the page rendering speed. Some of these techniques can belong into more than one category.

An HTTP server request is a request sent from the client (browser) to the (web) server. The client requests a document from the server, and the server serves a response. Each request requires an HTTP connection, which must be handled. More requests will mean more waiting time, while the connection is opened, request handled and response received and processed. Slow network connection, big latency and a sluggish server altogether produces a web page that loads very slowly, if at all, and each HTTP request will make the page load time longer and longer. It is a recommended practice to minimize HTTP server requests by grouping assets and libraries into bigger chunks of data. Images can be combined into *sprites*, which can contain dozens of small images in a single file and only a small part of the larger sprite sheet is displayed at once. Sprites are typically used for small image files, e.g. icons and buttons. CSS and JavaScript files can also be concatenated and bundled into single files, which may include entire libraries, plugins and site-specific code and styles. Caching should also be used to prevent any extra HTTP requests when new pages are loaded.

CSS and JavaScript bundling, as well as image compression and optimization, can be used to minimize the data to be transferred. A bundled CSS or JavaScript file is not only concatenated but also minified. Minifying and obfuscating will create smaller files by removing extra whitespace and re-authoring the code by shortening variable names and creating logic for repeated structures. Images should be compressed with optimal settings to preserve desired quality while reducing the file size. Some image formats like PNG do not support compression, but removing any extra information, which is not required to show the actual image, can be used to optimize them. This includes the transparent alpha channel and unused colours. Images could also be as small as possible by default and replaced dynamically if a larger version is needed. This may increase HTTP server requests, when viewing a single image can require loading of two versions of the same image: small low-quality and low-resolution version for tiny screens (the default image) and large high-quality and high-resolution version for large screens (an alternative image). Web servers should also compress

the data they send to the browser, if the browser application supports compression.

Modern CSS3 properties should be used instead of images to create gradient backgrounds, rounded corners and box shadows, but they should not be overused to avoid slowing down the page rendering.

Finally, of course not including anything that is not required is a good practice when optimizing a website. Large libraries and frameworks, such as heavy JavaScript UI frameworks and CSS grid frameworks, should be avoided and abandoned, unless they are absolutely needed.

Loading speed is important also on desktop computers and even small delays may disrupt the conversion path, which is very crucial to business websites and has a direct effect on revenue. (Wroblewski, 2011, pp. 23-24)

The mobile first paradigm also creates better support for accessibility. Accessibility is part of the original idea in Tim Berners-Lee's World Wide Web as a discipline to provide access to the web for everyone regardless of the browser they are using or disability they may have. The website content should be accessible in legacy browsers, screen readers and other assistive devices, gaming consoles as well as modern smartphones and desktop browsers. (Berners-Lee, 2010) Accessibility will therefore benefit also those with technological limitations. (Hay, 2013, p. 52) When websites are built to be responsive, it should be relatively easy to also take the accessibility into consideration. *Accessibility First* is a similar paradigm to Mobile First and Progressive Enhancement. Supporting basic accessibility is easier when websites are built for the lowest common denominator, separating content from presentation. Good accessibility does not only assist the browsing experience of visually impaired, but it also enables new ways to consume website content in the form of reading enhancement add-ons, e.g. *Reader* functionality in Apple Safari browser, *Instapaper* (http://www.instapaper.com/) and *Readability* (http://www.readability.com/) services aimed to provide a better reading experience. They are mostly used to get clutter-free access to content or to save content for reading it later (Keith, 2011; Zeldman, 2013). But the content should really be designed for good readability and accessibility so that the need to use these services never arises. Accessibility ensures the website content can be indexed by search engine robots,

because it relies on meaningful metadata and semantic markup. There are also legal regulations for building accessible websites. *Section 508* of the *Rehabilitation Act* in the United States requires that content provided by federal agencies has to be accessible to people with disabilities. (Rosmaita, 2006, p. 271) A Similar recommendation by JUHTA also exists in Finland. (*JHS 129 Julkishallinnon verkkopalvelun suunnittelun ja toteuttamisen periaatteet*, 2005)

4.2 Graceful Degradation and Progressive Enhancement

Graceful degradation and *progressive enhancement* are related to each other, but while graceful degradation targets the most advanced browsers first and removes features if the browser does not support them, progressive enhancement will take the lowest common denominator as the starting point for the design and adds features when the browsers support them. (Nebeling & Norrie, 2013, p. 511) Progressive enhancement means good experience for everyone, because the websites built with progressive enhancement in mind will most likely work on every device that supports HTML. (Zeldman, 2013)

Graceful degradation is about fault tolerance: the system degrades gracefully when there is an error or when the browser does not support a certain feature. A problem is, that building support for old browsers is not a top priority and the starting point for the design is the powerful and modern desktop browser. (Gustafson, 2008)

Graceful degradation is often a trade-off between desired outcome seen in visual layouts and the difficulty of the technical implementation. If the layout uses rounded corners on content boxes and buttons, they cannot be implemented in browsers not supporting the *border-radius* CSS property without using polyfills or static background images, which are more cumbersome techniques.

Progressive enhancement, or *inclusive web design*, as it was first named (Champeon, 2003; Gustafson, 2008), is based on the idea of separating content, presentation and behaviour. It embraces accessibility, semantics, forward-compatibility, search engine optimization and usability. A progressively enhanced website is designed in a content first method by starting with a simple HTML skeleton for the content, without any CSS styles or functionality provided by JavaScript. This skeleton can be called the *content layer*. It is available to

any browser and device regardless of their CSS and JavaScript support. On top that layer is the *presentation layer*, which contains the visual look and feel built with CSS styles and images. Browsers will ignore CSS rules and properties that they do not understand, so it is fairly easy to write styles that are supported in all browsers and modern styles that are ignored by old browsers but rendered correctly by modern browsers. The topmost layer is the *behaviour layer*, adding interaction to the mix with JavaScript. Behavioural functionalities may include drop down navigations, showing and hiding of content or updating subsection content dynamically. (Wells & Draganova, 2007, pp. 56-57) Progressive enhancement and graceful degradation are visualized in Figure 2, showing also different kinds of content blocks and their positioning on the screen on different layers.

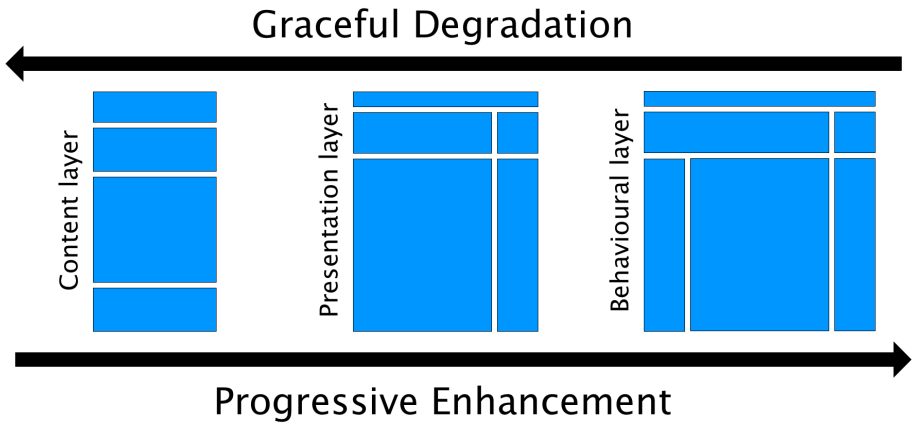

Figure 2. Graceful Degradation and Progressive Enhancement comparison

4.3 Content First

Content is a design problem, but content also precedes design. Design without the content is not design, it is just decoration. (Zeldman, 2013)

It is quite logical that users are accessing websites because of the published content they contain. The content can be anything: text, image galleries, social networks, video and music streaming services, games and applications. Although the division between a website and a web application may not be always clear, there is still content being delivered to visitors. Users are coming to the website to get the content, when they want and how they want. Therefore the designer's job is to serve the customer: the user. Some would call it user experience, but in

the end it is just about keeping customers happy through good service. Responsive design is also designing for small screens, which means that content should come first, because there is no room for extra clutter. The right content should be served to the right user at the right time, otherwise they might not bother to read the content at all and if the purpose of the website is to sell something, it may not be very profitable in the end. Mobile web, small screens and responsive design are creating a new interaction design landscape, where users and content come first. (Zeldman, 2013)

Designing for every screen size in existence is extremely difficult, so the traditional *canvas in* approach does not work anymore. There are no pages or recommended resolutions to design for. Hence the design process must change its focus from canvas in to *content out*. Design for pieces of content, not for an imaginary page. (Boulton, 2011a)

5 RESPONSIVE WEB DESIGN WORKFLOW

Design, also responsive design, is problem solving within a set of constraints and if the constraints are unknown, the designer must find out what they are. It is the designer's job to ask questions to gain insight about the goals and talk to the client. (Monteiro, 2012, pp. 10-12) In responsive design, the constraint is no longer a single browser on a single resolution desktop computer; the constraint is the content itself.

Many websites are designed and built from a *user interface in* or *canvas in* starting point instead of *content out*. Responsive web design is based on structured content, which allows websites to respond to the rendering media. In short, responsive web design begins with the data, focuses on the most basic important content and advances to sketching, prototyping, visual design and finally into a responsive website. (Hay, 2013, pp. 52-54)

5.1 Abandon The Waterfall

Web projects have been more or less reminiscent of the classic waterfall project model, although agile or any suitable methods can be used during the development phase, and visual design workflow is often iterative by nature: sketch, visualize, review and back to the drawing board. Many design projects start with a planning phase, which produces a concept, wireframes and graphical layouts. (Salminen, 2013) The development phase does not begin, until the layouts are finished and approved. Sometimes there is also a specification phase, which happens simultaneously with graphic design. When the developers and web designers finally get the specification and layouts, they can start working. At this point, it is often difficult to fix any flaws in the layout and the specification.

According to Boulton (2012), the traditional waterfall workflow has consisted of three phases:

1) Planning and design of artefacts including site maps, wireframes, user flows, user journeys, scenarios, all of which are to be reviewed and approved by the client

2) Photoshop composition (a layout image) producing based on the final artefacts created during the first phase, again reviewed and approved by the client
3) HTML templates and website implementation, reviewed and approved by the client before publishing

The waterfall model is not optimal for web design projects and responsive web design projects are even more challenging than the traditional ones. Each step of the waterfall process may be executed by a different team, perhaps even by a different company. Without clear understanding of responsive web design requirements, the end result can be anything between an expensive catastrophe for the client or a partially working website, which matches only the needs of desktop users. When the client has approved and paid for the concept and layouts made by an agency, there is little opportunity to change anything without strong arguments and if the client does not really understand what responsive web design is all about, it may be a helpless situation. If there are several vendors working on a single project, they should co-operate and try to fulfil the client's goals by aiming for the perfect end-user experience on any device and platform. In the optimal situation, a single vendor can deliver the project by using a talented cross-competence team, but the team still needs a method to follow.

The *Responsive Web Design Workflow* model is a multidisciplinary, cumulative and iterative model based on Boulton's (2012), Hay's (2013) and Salminen's (2013) workflow models and my own hands-on experience in building responsive websites for organisations and private businesses. It allows clients to see and experience the evolution of the design from the very beginning into the final release. (Hay, 2013, p. 8) It is divided in eight phases starting from discovery and ending in feedback and discussion phase.

5.2 Discovery

The first step in a responsive web design project, or any project, is to do a background research and discover information about visitors (website users), the website (business) owner and possible competitors and project goals.

Most of the information about visitors is gathered from web analytics and statistical data. Web analytics is an important tool to gain insight of your website users' behaviour. (Leibtag, 2012) Analytics can tell which parts of the website are rarely accessed and where the conversion path is disrupted. Analytics can also tell about the demographics of the visitors: which browser versions and operating systems are they using, which screen resolutions are the most popular ones, which country the visitors come from and so on. Analytics reveals the technical limitations of visitors' browsers and devices, which is helpful information when designing a new site. If a notable portion of visitors is still using outdated or ancient browsers, it is recommended to continue supporting these browsers. Analytics can also spot potential usability problems, which could be crucial factors affecting the website owner's business by disrupting the conversion path.

To evaluate mobile browser usage, analytics data that can be gathered includes percentage of visits with mobile browsers, pages or sections of content being accessed with mobile browsers, search queries from mobile browsers and exit pages, which are the last visited pages before a user leaves the site. (McGrane, 2012, pp. 55-56)

Sometimes there is no statistical information available at all. Reasons for this are obvious, if there is no existing website from where the analytics could be collected from, but sometimes the website owner has not installed any tracking system on their website and all the useful information is lost. In cases like these, the only way to gain insight is to collect information from the website owner. It is advisable to ask a lot of questions to understand the website owner's business and the goals of the project. Examples of typical questions by Salminen (2013) are: "*Why would people come to your site?*", "*What is the main goal you are trying to achieve?*" and "*Who are your main competitors?*".

Without the discovery phase and proper knowledge it is nearly impossible to understand what the website owner wants or needs. Unfortunately, it is not easy to complete the discovery phase and gain all the required insight. Companies and organizations often request quotations from vendors and the scope of work must be offered during a short time frame and it must be based on limited amount of information. There may be not enough information about technical requirements, e.g. integrations to different systems and interfaces, or there may

not even be a chance to meet the business owner to ask questions. Even the users of the website could be unknown as there may be no access to statistical information.

It is recommended to collect every piece of information that will be helpful in building the actual website and base the quotation on these facts, or choose a more radical approach and offer a different solution, which will be better suitable for responsive web projects. It is likely that the requesting party will reject any quotations that do not fit their processes or match the original request, so it is always risky for a vendor's business, as lost tenders do not produce any revenue. When the tender is won and the project starts, it is still possible and advisable to return to the discovery phase, although the project schedule and scope of work could have already been fixed. Insight gathered in this phase can still prevent unpleasant surprises in later phases and it may reveal problems in the client's business goals as well as technical obstacles that must be solved before release.

5.3 Planning

The planning phase is based on the information gathered during the discovery phase. It includes writing user stories and designing the information architecture, which is the foundation of the website structure. The planning phase can also specify different content elements, placed in order of importance. It is common to do sketching and very rough prototyping (e.g. drawings by hand in a sketchbook) during early stages of the project, but real content will make it easier to create more detailed and accurate prototypes. (Salminen, 2013) Planning can include also creating the content inventories, which are complete lists of existing content, but this process is often very time consuming and causes too much work. It is recommended to list instead only the things that absolutely need to be on the website, regardless of their existence during the planning phase, and keep that list very simple. (Hay, 2013, p. 16) At simplest form, a content inventory is a short list of items, which will work as a rough guideline for later work. The table of contents in this thesis is also a content inventory.

The planning phase may also include the creation of content reference wireframes, which are not as detailed as classic wireframes, also called sche-

matics. A content reference wireframe should not look like a finished page, but rather a minimal model of the placement of different content sections on the website. (Hay, 2013, pp. 24-27) An example of a content reference wireframe can be seen in Figure 3.

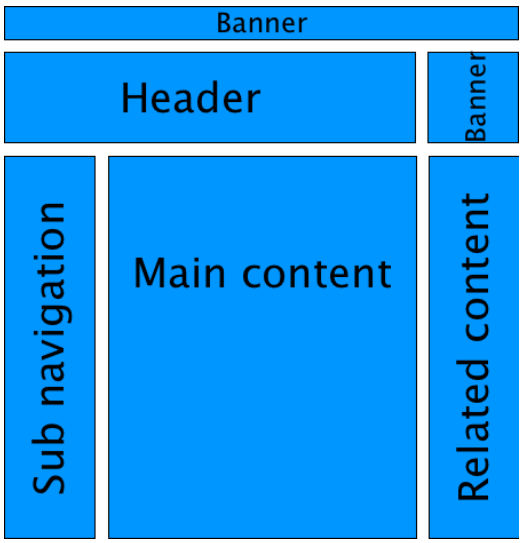

Figure 3. A content reference wireframe

5.4 Content Design

Content or text design phase includes the design of the textual content. In many web projects it is typical, that the actual site design is based on existing content from the old website, or in the worst case, dummy content, which can be anything from random characters typed by just banging the keyboard or more commonly, using pseudo Latin phrases, the *Lorem Ipsum* (http://en.wikipedia.org/wiki/Lorem_ipsum). The final content is often designed and created by the client and published after the web design and implementation project has already been completed. It is often forgotten that the content is the most important thing on the website and everything should be designed to support the delivery of content. (Salminen, 2013)

Content can be designed with any text editor or word processor, but it should not contain too specific formatting. Many organisations are using Microsoft Word or other similar *WYSIWYG* (What You See Is What You Get) word processor for content design, but it is recommended to use a *plain text markup language* instead. The idea behind plain text markup languages like *Markdown* is that they allow writing text in a human readable way, without tags and com-

plex formatting commands and offers tools to convert the text to clean and tidy HTML, which can be used in the prototyping phase. This HTML is an excellent foundation, the lowest common denominator for progressively enhanced mobile first websites, as it will most likely work in any device, even in text browsers. (Hay, 2013, pp. 55-69) Plain text markup language also separates content from form. Many content authors may want to control the styles of their content and content management system (CMS) editors are often WYSIWYG rich text formatting tools, which cannot create reusable content. (McGrane, 2012, pp. 45-47) It is good to remember that content is not just text; it can consist of tables, images, videos and other rich media, but working from the content out method can expose issues and problems, e.g. missing important content or navigational problems.

Responsive content is content that can reformat itself automatically for different devices and screens, showing dynamically more or less content depending on device capabilities and the network bandwidth. Responsive content requires metadata and other background information as well as a smart application that can automatically display the right content in the right context. (McGrane, 2012, p. 45)

5.5 Sketch Design

Sketching is a method to quickly create rough low-detailed drawings of design ideas. Many designers do sketching all the time, so it does not have to be a separate step done only during a certain phase of the project. Sketches can be made with any tools from pencil and paper to computer programs, but usually pencil and paper are the easiest and also the most efficient tools. The cost of pencil and paper are very low, using them does not require any learning effort, in contrast to what is often the case with software. Sketch drawings can be easily presented to other team members and they can also be scanned into an electronic format for online sharing. Sketches are often used as paper prototypes in usability testing. The fundamental idea behind sketching is to quickly test and find the best design concepts and abandon those that are not good enough to be developed any further. Dropping out failed concepts at an early stage will save time and resources and reduce the risk of bringing bad designs into production. The more sketching is done, the better solutions it may help finding. Sketching is

about thinking and problem solving. It does not have to produce any deliverables to be shown to anyone and the tools used can be anything. (Hay, 2013, p. 108)

Hay (2013, pp. 109-115) introduces a sketching process, which includes four different steps:

1) Start with small low-detail thumbnail sketches
2) Select the best ideas for further exploring
3) Create more detailed and larger rough sketches of the selected ideas
4) Create realistic compositions of the best sketches

Thumbnail sketches should be small, low-detailed, quick and dirty drawings about anything that comes to the designer's mind. They should be so simple that several sketches should be possible to create in a short time. Quantity is more important than quality, because thumbnail sketches are based on the assumption that resulting design ideas will increase when the number of generated ideas increase and the designer will learn during sketching which ideas could work and which will not. (Hay, 2013, pp. 109-110) An example of a thumbnail sketch can be seen in Figure 4.

After selecting a few best ideas from the thumbnail sketches, it is time to explore them further and create more detailed rough sketches. It is not necessary to draw any text, but the sketches can be annotated and commented. The sketches will communicate the essential information of the design idea to the person or stakeholder, who is viewing them. Rough sketches can be drawn on paper or on real device, e.g. a tablet. Using a real device for drawing will let the designer to explore the size of different elements on the screen. An example of a rough sketch can be seen in Figure 5.

The final step is to create realistic compositions of the best sketches. For a responsive web design project, these compositions should be HTML mock-ups that can be tested in a browser. The rough sketch can be used as a background image for the mock-up to get a starting point for laying out elements on the HTML page. (Hay, 2013, p. 114)

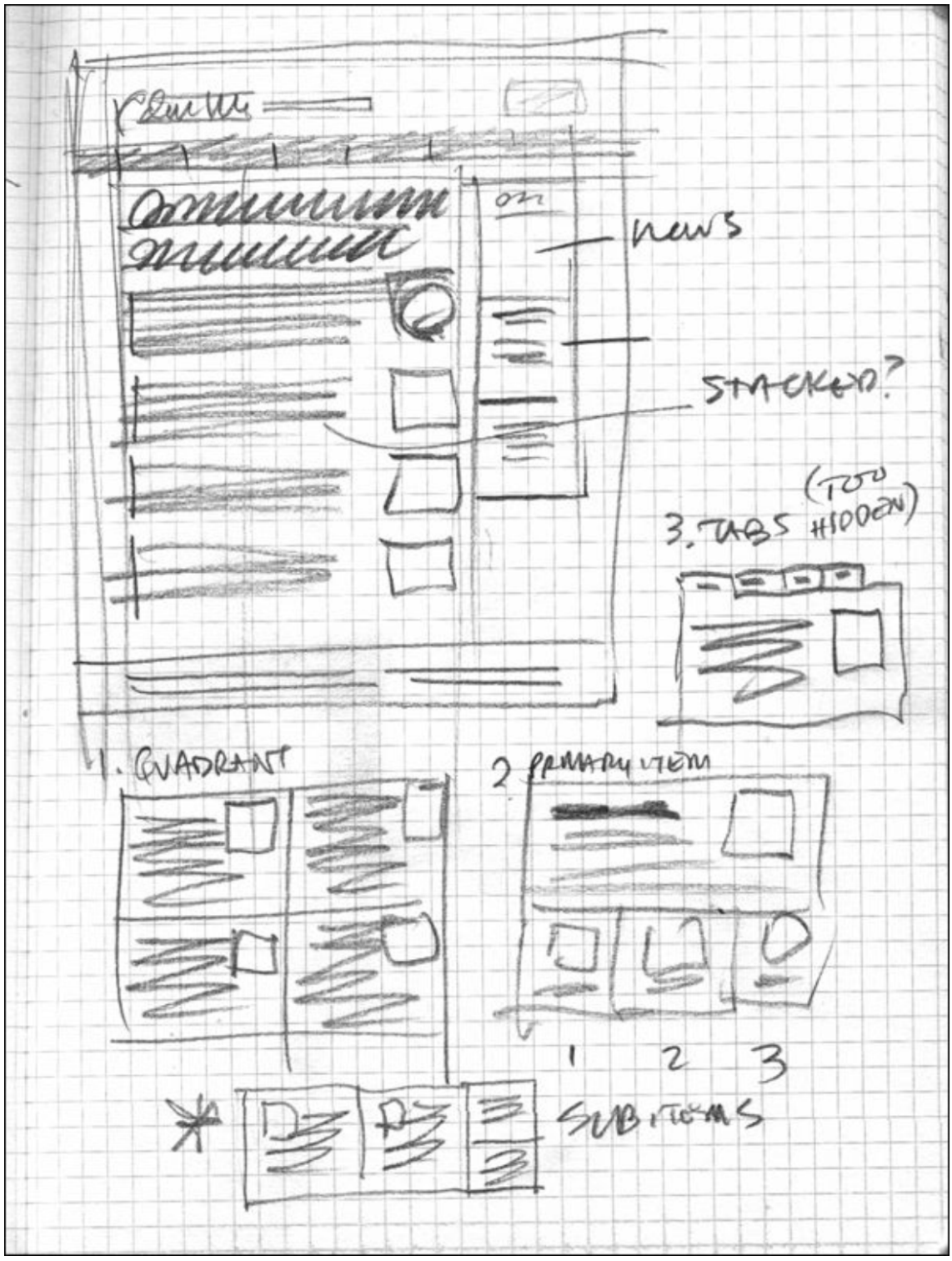

Figure 4. A thumbnail sketch (Hay, 2013, p. 110)

The HTML mock-up can be used to design the breakpoints, i.e. CSS media query directives, for the website. Breakpoints can be divided in two categories: major and minor. A major breakpoint will trigger a major change in the design. An example of a major change is a layout change from two columns to four columns. A minor breakpoint is a small change, e.g. moving a form label above the

form field from left of it, while rest of the page remains unchanged. (Hay, 2013, pp. 115-117) Breakpoints should not be the main constraints in the design, because a breakpoint can be considered similar to a specific screen size. They should not lead the design, because otherwise they easily form a target to follow and content will lose the main focus. Instead of sticking to pre-defined breakpoints, which are often based on popular device screen sizes, designers should use breakpoints to lay out content in a different way, when the content cannot be otherwise presented in accessible and readable manner. Let the content define the breakpoints.

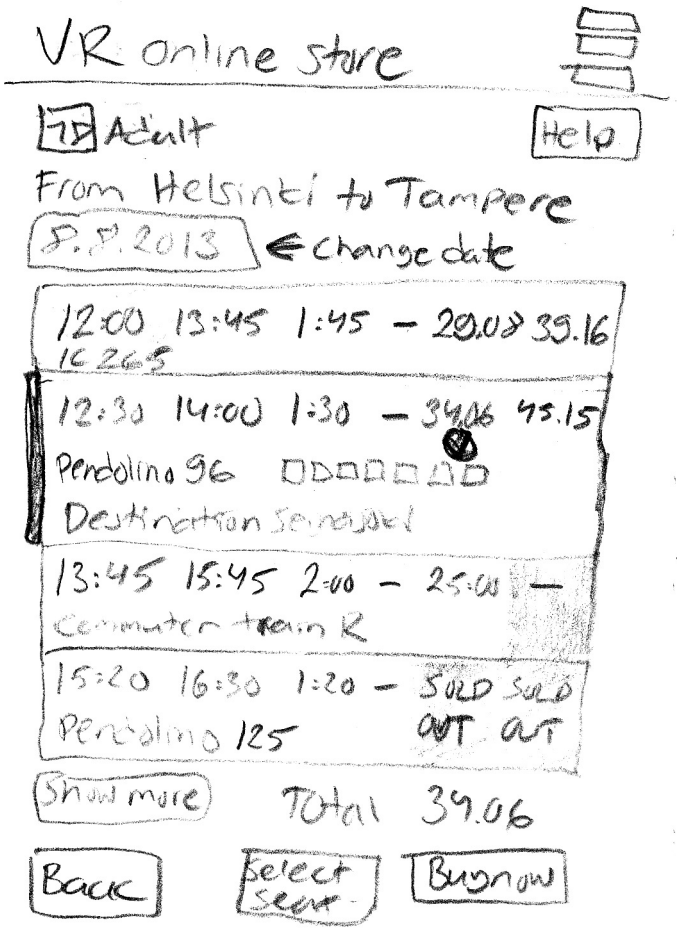

Figure 5. A detailed rough sketch

5.6 Prototyping

HTML mock-ups created in sketching phase are one step closer to prototyping. A prototype is an implementation of the design sketch, which is created only for testing and review purposes. Prototypes are often meant to be quick and dirty implementations of the sketch design, so they may or may not contain in-

teractive elements or dynamic content. Prototypes can vary from rough drawings on paper to fully interactive implementations that work in browser.

Paper prototypes are often rough sketches, which are fast to create and test, but they only represent small portion of the whole website and are not interactive. Sometimes paper prototypes are just low-detailed wireframes or even higher-detailed visual layouts, but nevertheless, they still lack the interactivity, which is quite important in software design. This does not mean that they are useless, rather the opposite, but as they are static designs, they cannot be used to test the responsiveness.

Interactive prototypes can be created with software designed for prototyping or by using front-end frameworks, static site generators and programming languages. Popular software tools to create interactive prototypes are Axure, OmniGraffle, Visio and Adobe Flash. Bootstrap framework is a widespread front-end framework, originally developed by Twitter and often used as a basis for user interfaces. An interactive prototype can be anything from a collection of wireframes as hyperlinked PDF documents or a fully functional visual browser compatible application with animations and transitions. The more detailed it is, the more expensive and time consuming it is to create.

A big problem with many prototyping tools is that they cannot produce responsive prototypes. To get a realistic responsive prototype, the recommended solution is to create an HTML prototype. Although HTML prototypes are often introduced as rapid prototypes, they may not be significantly faster to create compared to static mock-ups. A static site generator is an application, which can generate a static website from templates, supporting content formatted by a plain text markup language and providing templating languages to write the HTML structure instead of writing raw HTML manually. (Hay, 2013, pp. 139-140)

Responsive prototypes should be created and tested in the real environment, which is the browser. The main goal of an HTML prototype is to find problems and prevent the *big reveal,* which will give unrealistic and false expectations to the client. (Boulton, 2012) A big reveal happens, when client sees the Photoshop composition of the layout and expects to get a website looking exactly the same, pixel by pixel. An HTML and CSS prototype working in a browser is the only

way to give realistic expectations of the final outcome and test breakpoints and responsiveness in different sized viewports and devices. If the same content were to be tested with static prototypes, it would be very difficult to spot potential problems. Cross-browser compatibility should not be the main focus in prototyping, because the prototype is a replacement for the Photoshop layout. (Hay, 2013, p. 138) Typical responsive design problems are *overflowing content*, *unrecognizable images* and *complex navigation structures*.

5.7 Problems Revealed by Prototyping

Content overflow happens, when long words do not fit in the horizontal space of the viewport or in the container element. This is often a problem when content is presented in a language containing lots of compound words, e.g. Swedish, Finnish or German. Overflow can be visible (automatic) or hidden, depending on the CSS *overflow* property value, which is *visible* by default in most browsers. Visible overflow means that long words will spread outside of their containers, thus creating a broken layout, which is so common that it is visualized on a popular coffee mug theme seen in Figure 6, or they may even spread outside of the viewport, bringing up the horizontal scrollbar, forcing the user to scroll the page horizontally to see the overflowing content. If the overflow property is set to hidden, overflowing content will disappear, the layout will not break and there will be no scrollbars, but then the content is only partially visible. A vertical overflow may happen, when background images or fixed height containers are used as wrappers. As the viewport width decreases, content will usually need more vertical space and it will easily overflow from a fixed height container, unless the hidden value is concealing the overflowing content. The same issue can happen, when the user increases the text size. These problems can be spotted when real content is tested in a real browser. Forcing word-breaks, which will break an overflowing word into several lines, can prevent horizontal overflow, but it may bring an unprofessional touch to the overall impression. A better solution is to use hyphenation, but it is not yet well supported across browsers. Hyphenation cannot be recommended for all long text strings, e.g. web URLs or email addresses, because the hyphen is not part of the string and may lead to error situations. Therefore, it is crucial to find these issues and solve them in an appropriate manner, case by case. When the content does not fit into its container, either the container is too small, or the font is too

large. Making the container bigger or decreasing the font size can often prevent the overflow issue.

Figure 6. Content overflow problem visualized in popular coffee mug theme (Waugh, 2009)

Unrecognizable images contain details that are not readable when the visible or physical image size shrinks as a result of the max-width: 100% CSS property setting or a minimum sized default image. There may be text inside the image or other visual elements, which cannot be read or recognized in a tiny image. Compression algorithms can also create artefacts that will have an influence in the overall quality and clearness of the image. A high compression setting shouldn't be used for images that contain text, e.g. buttons, badges and calls to action, because the compression will make the text blurry. A very high-resolution display may also make images blurry, because their size is scaled up. Every image should be reviewed and thoroughly tested in different sized view-ports, low-resolution and high-resolution displays to ensure a polished look and feel.

Navigation structures are problematic on large websites that may have hun-dreds or even thousands of pages, several sections and dozens of categories and subcategories. They create a complex navigational maze that cannot be easily presented in a small viewport. Some websites have solved this issue by creating collapsible navigation wrappers, which only show a single level at once. Toggle buttons are used to open and close nodes, which contain sub level navigations.

These structures can be several levels deep and they often do not fit into the viewport, hence scrolling is required. A complex navigation is often a result of poorly designed information architecture. It will reveal issues in the actual content, how it is structured and categorized. A large screen can forgive much of these issues by providing more space to show navigation items, but testing on a small screen will reveal potential problems.

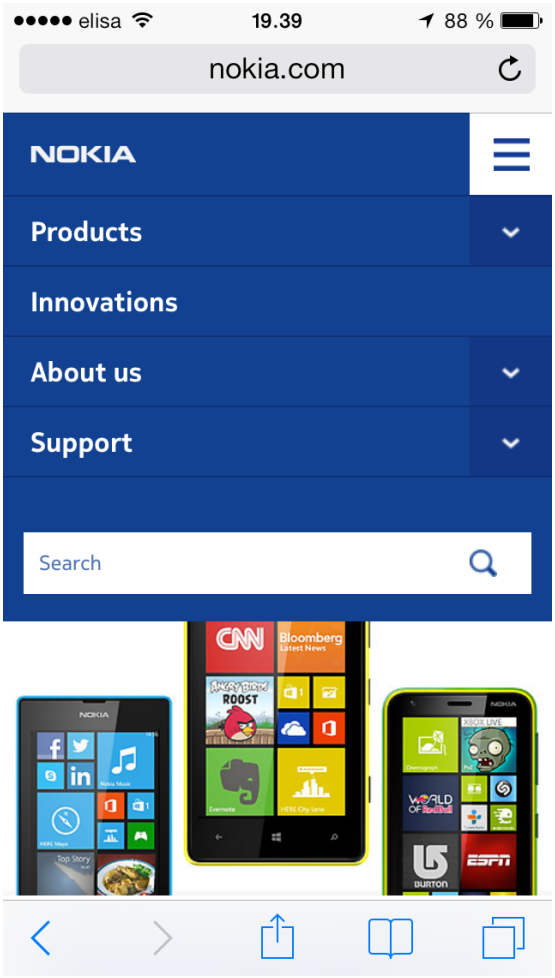

Figure 7. Nokia.com global website navigation open in Safari browser on iPhone 5

The nokia.com global website main navigation contains only four sections on the root level as shown in Figure 7, which will still occupies half of the iPhone 5 screen height, which is 1136 actual pixels and 568 CSS pixels. Actual pixels are the physical pixels that will provide the maximum resolution for the device, while CSS pixels are the pixels used programmatically. The iPhone 5 has a high-resolution display, which means that a pixel used in CSS styles, e.g. margin, padding or image width and height, consists of four actual pixels. The Nokia

global website uses a typical collapsible navigation, which is hidden by default and every subnode must be opened by clicking or tapping the toggle button shown on the far right next to each navigation item.

When each node is open, the whole navigation does not fit on the display. Combined screen captures of the whole navigation tree are shown in Figure 8. Dashed red line marks the merging point of the two screen captures. The browser toolbar is not visible on this screen capture. It can be very difficult to get a good grasp of the website structure, if the levels are deeply nested. The nokia.com example is very simple and contains only four main level items plus the link to the home page, 12 sub level items and the additional search field. It is recommended to include access to search on every page, preferably at the top of the page. When the navigation is not clear enough to provide clues about the page contents, it can be really frustrating to browse through multiple pages on a slow network connection. Search is helpful in cases where navigation does not lead to the content the user is looking for.

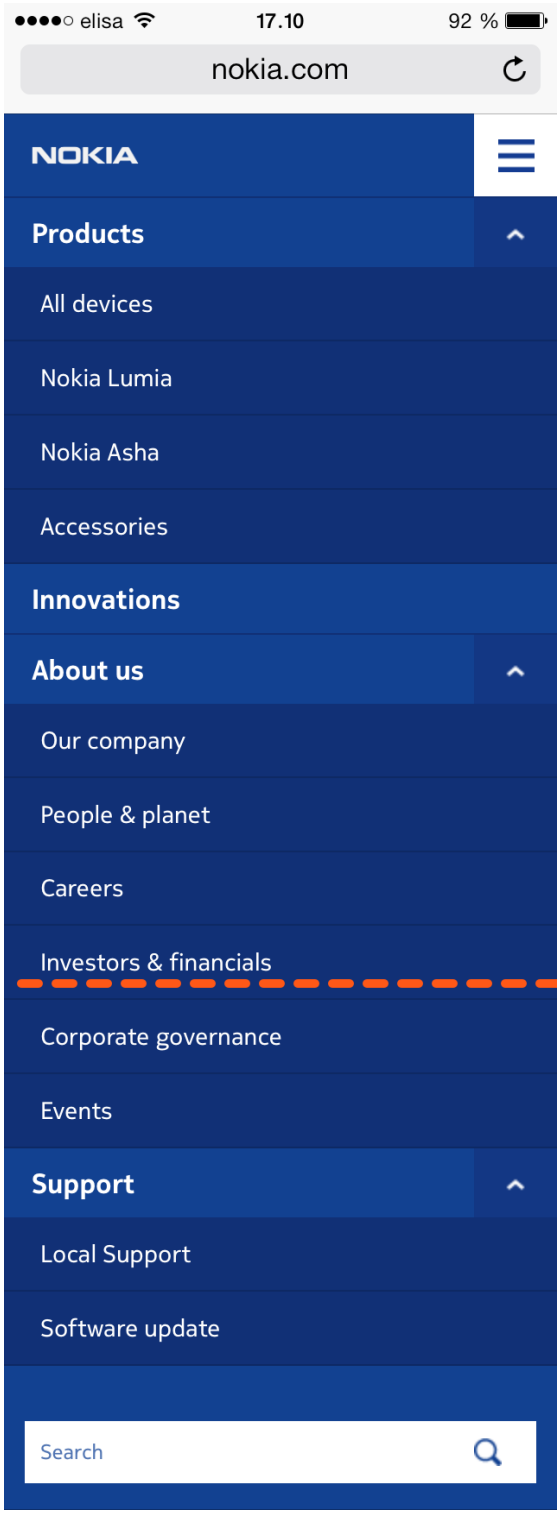

Figure 8. Nokia.com global website: all navigation levels open

In landscape mode, the navigation eats even more space, because it does not use the available space on screen very efficiently. In Figure 9 can be seen that the navigation items uses up only about one fourth of the available horizontal space and most of the page is empty, while the vertical size of the navigation is uncomfortably long in landscape mode. Prototyping with real content can easi-

ly reveal similar issues and allows experimenting and finding a decent solution through iteration and usability testing.

Figure 9. Nokia.com global website navigation in landscape mode

Finnish tabloid newspaper Ilta-Sanomat uses a two-column navigation to save space. It is not a fully responsive website, but a separate mobile website and is used here only as a reference of a different approach to solving the space required by navigation problem. In Figure 10 is shown a space saving two-column navigation in landscape mode. Browser toolbars are hidden automatically in iOS 7 when user scrolls, but as can be seen in Figure 9, they can eat up lots of precious space.

5.8 Visual Design

Responsive Web Design has created new requirements for the visual design phase. It is no longer practical to create static compositions (comps) and fully designed layouts in Adobe Photoshop or other image editing and vector drawing tools (e.g. Adobe Illustrator, Adobe Fireworks, Corel Draw, Pixlr, Acorn, Sketch, GIMP, Pixelmator, Paint Shop Pro). Photoshop is practically the industry standard editing software for layout and composition design. The problem with static compositions is, that their behaviour and transformation on different sized displays and between breakpoints cannot be estimated or tested. To be able to design for different types of media, dozens, even hundreds of composi-

tions must be created and reviewed and it is a time consuming and expensive process. Static compositions also cannot easily present other differences between browsers, operating systems and devices, e.g. font rendering, colour settings, and not to mention the potential user settings, such as custom CSS style sheets and disabling of image rendering. And what happens if something has to be changed? Editing hundreds of images because of a relatively minor issue as a link colour can take several days to complete.

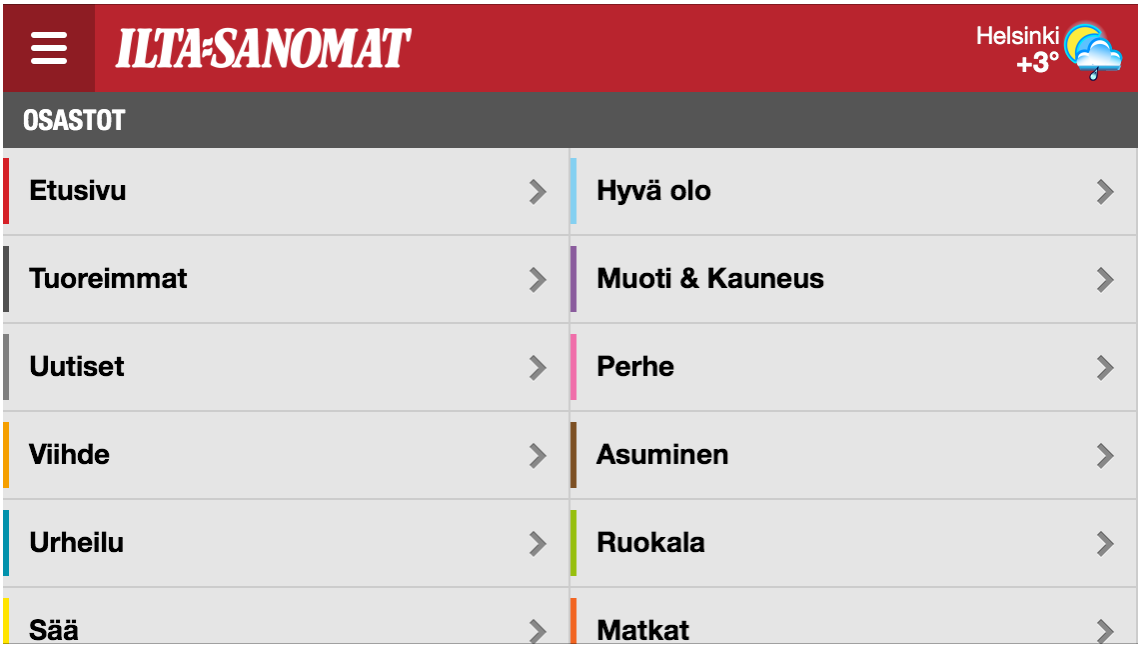

Figure 10. Iltasanomat.fi navigation is divided in two columns

One of the biggest problems in Photoshop driven design is, that it creates a strong first impression and expectations for the actual outcome of the end-user product, when clients review the compositions for the first time. It is often erroneously assumed to be a WYSIWYG tool, producing images of websites. It has its own font rendering and anti-aliasing settings, which do not present the actual outcome of font rendering engines and technologies in operating systems, such as ClearType, GDI, DirectWrite and Quartz. The fonts rendered by the browser can also be of different type with different kerning and ligature settings, so there may be significant differences in font appearance between static composition and text rendered in the browser. (Ahrens, 2012) Photoshop is also not an interaction design tool, as presenting different states and transitions of elements is not very practical.

In short, static compositions shouldn't be used as blueprints for responsive design, because they are not responsive and do not adapt to the conditions of the rendering media.

This problem is very real and a major issue especially in cases where the visual design is created by a third party agency. The client may have already seen the layouts, approved them and expects to get a website which looks exactly like the layouts suggest.

However, the visual design phase is still needed to create graphics and maybe also a proposal of the actual design for the website, but static compositions should be considered only as suggestive representations of the actual outcome of the website, not pixel perfect blueprints. The actual responsive web design should happen in the prototyping phase in the browser itself.

5.9 Testing & Feedback

Testing in Responsive Design Workflow means testing the working prototype in real devices and browsers, not just unit testing, acceptance testing, regression testing or usability testing. It can include all of them, but the main focus in this model is to test on real devices. It is impossible to do comprehensive testing in every device ever released, because there are too many of them. In larger companies, there may be several projects running simultaneously and there can be shortage of test devices, or there may be difficulties accessing the development environments from the device if they are in different networks, usually for security reasons. The purpose of testing in real devices is to find the quirks of different browsers, browse the website by using the input methods the device offers and be able to get the same user experience as the end user owning a similar device will get. Testing in real devices reveals not only device specific browser quirks, but also performance problems, such as if the website is too heavy to load and process in the browser.

Feedback and discussion are recommended during the whole process. There should be no big reveal, where the client does not see the website they will get until it is finished. Big reveals seldom cause cheers, praises or satisfied customers. Big reveal usually leads into numerous bug reports, change requests and arguments about the cost of fixing the website to be what the client expected it

to be. It is recommended to regularly collect feedback and discuss with the client during every phase of the project. It will help to avoid the contrast between expectations and the final outcome, because the client will know what they are getting.

5.10 Iteration

After sketching, prototyping, visual design, testing and feedback, it is time to learn which parts still need work, which sections are complete and what could be improved. This means starting the process over from sketching phase and repeating the cycle until the website is ready for launch. It is typical in responsive web design, that new problems appear every time something is changed and fixed. The differences between the rendering engines in different browsers are so big, that it is sometimes difficult to predict how a small change will affect all the devices without testing.

When the website is published, it still often requires work and active development. Not just bug fixes or changes to technical implementation, but updating the content, learning from web analytics data and responding to the eternally evolving technical progress, such as new devices and interaction methods coming to the market.

6 DISCUSSION

Responsive Web Design as described by Marcotte (2010) is a simple technique, which alone does not really give web designers and developers tools to create fully adaptive websites or applications for different kinds of rendering media and viewing contexts, but it helps to create a solid user experience for information rich websites and has changed history as the "*designed for 1024x768 resolution and Internet Explorer*" websites have slowly been replaced by modern responsive ones and thus provided access to the web to almost any device, just as Tim Berners-Lee designed the web to be.

Device fragmentation has made it almost impossible to test if a website really works in more than a small number of devices and while the modern browsers are developing in rapid cycles, their predecessors still have lots of users. Internet Explorer 6 was released in 2001 and although it is not commonly supported in western web industry any more, it is still widely used in China up to this date. ("Internet Explorer 6 usage around the world," 2013) Many organizations may also have outdated default browsers. Often these organizations demand that their website has to provide identical user experience, look and feel for the old browser they are using, although they would be the only ones accessing the site using that specific browser. In the worst-case scenario, there is a single stakeholder using their favorite browser from the last decade. Although a website does not have to look exactly the same in every browser, clients often assume that they will get an identical website across the thousands of different devices.

As a designer cannot really know which kind of rendering media will be used to view the website, it is recommended to build a decent user experience for the lowest common denominator. Targeting a small screen feature phone on a slow network connection encourages abandoning anything not providing extra value. Many websites have been plagued by flashing banners, ads, calls to action, related content lists and other secondary level content, which eat up the precious space, distract user focus and slow down page loading. Responsive design has enforced stakeholders to decide, which content is essential and which is not. Large companies have the resources to develop native applications for different platforms, but for smaller ones it is often more practical to create a re-

sponsive website or a web app. While it is nearly impossible to test the website in every device ever made, a good and simple, clean layout will probably work well enough in an old feature phone to fulfill the *accessible from any device and software* principle.

However, the cold fact is, that although building a perfect mobile first, accessibility first, content first, progressively enhanced responsive and adaptive website is probably be the dream of every agency, designer and client, there are limitations to what can be done in a given timeframe and budget. Many clients are not willing to pay for a responsive website if they do not understand or see the reasons why they should do so. If there is no *return on investment* (ROI), there will be no responsive website.

There can also be advertising banners and integrated 3rd party content, which may not be responsive at all and out of the designer's control. Compared to building a classic static desktop website, a responsive website will require more work, more planning, more skills and expertise and more resources, thus it will be more expensive for the client and the outcome may still not be 100% responsive. For example advertising banners are standardized elements, but their size is fixed and their placement on a responsive website may change. Advertising brings revenue to many websites so it is crucial to find solutions how banners can be included also on responsive websites. (Boulton, 2011b)

Although there are lots of good guidelines and paradigms to follow, it will not be easy, because each project will be unique with different goals, users and content, budget and schedule, client and designers. Responsive design workflow will however provide good tools to reach for the optimal solution, which will benefit the users by making the content easily accessible and bring revenue to commercial website owners via increased sales and decreased abandon rate. It's the designer's job to convince the client to engage in a responsive web design project. The designer and the client must work in co-operation, on the same side of the table. The designer must therefore provide information to the client about the project progress, discuss and ask for feedback regularly. A good responsive website will not come into being without all parties working together for a common goal.

And the most important thing to remember: it's all about the content.

REFERENCES

Ahrens, T. (2012, April 24). A Closer Look At Font Rendering. *Smashing Magazine*. Retrieved December 4, 2013, from http://www.smashingmagazine.com/2012/04/24/a-closer-look-at-font-rendering/

Alexander, S. (2013, July 8). Choosing a responsive image solution. *Smashing Magazine*. Retrieved October 8, 2013, from http://mobile.smashingmagazine.com/2013/07/08/choosing-a-responsive-image-solution/

Artail, H. A., & Raydan, M. (2005). Device-aware desktop web page transformation for rendering on handhelds. *Personal and Ubiquitous Computing*, *9*(6), 368–380.

Berners-Lee, T. (2010, November 22). Long Live the Web: A Call for Continued Open Standards and Neutrality. *Scientific American*. Retrieved May 13, 2013, from http://www.scientificamerican.com/article.cfm?id=long-live-the-web

Bos, B., Çelik, T., Hickson, I., & Wium Lie, H. (2011). *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification* (No. 20110607). W3C. Retrieved from http://www.w3.org/TR/2011/REC-CSS2-20110607/

Boulton, M. (2011a, March 24). A Richer Canvas. *The Personal Disquiet of Mark Bolton*. Retrieved December 8, 2013, from http://www.markboulton.co.uk/journal/a-richer-canvas

Boulton, M. (2011b, November 15). Responsive Advertising. *The Personal Disquiet of Mark Bolton*. Retrieved October 13, 2013, from http://www.markboulton.co.uk/journal/responsive-advertising

Boulton, M. (2012, February 24). Responsive Summit Workflow. *The Personal Disquiet of Mark Bolton*. Retrieved December 4, 2013, from http://www.markboulton.co.uk/journal/responsive-summit-workflow

Brown, M. (2013, June 7). 8 Reasons Why to Invest in Responsive Web Design. Retrieved October 13, 2013, from http://www.zaginteractive.com/about/blog/blog-post/the-z-drive/2013/06/07/8-reasons-why-to-invest-in-responsive-web-design

Cáceres, M., Marquis, M., Weiss, Y., & Bateman, A. (2013). *The picture element* (No. 20130226). W3C.

Champeon, S. (2003, March 11). Inclusive Web Design for the Future. *hesketh.com*. Retrieved December 7, 2013, from

http://www.hesketh.com/thought-leadership/our-publications/inclusive-web-design-future

Chen, Y., Ma, W.-Y., & Zhang, H.-J. (2003). Detecting web page structure for adaptive viewing on small form factor devices (pp. 225–233). Presented at the WWW '03: Proceedings of the 12th international conference on World Wide Web, New York, New York, USA: ACM.

Condliffe, J. (2013). Nokia's Xpress Browser Decrypts Your HTTPS Data. *Gizmodo*. Retrieved from http://gizmodo.com/5975095/nokias-xpress-browser-decrypts-your-https-data

Epstein, Z. (2013, April 30). World's first web page brought back from the dead for 20th anniversary. *BGR*. Retrieved December 6, 2013, from http://bgr.com/2013/04/30/first-website-anniversary-cern/

Fox, R. (2012). Being responsive. *OCLC Systems & Services*, *28*(3), 119–125.

Gardner, B. S. (2011). Responsive Web Design: Enriching the User Experience. *Connectivity and the User Experience*, 13.

Gustafson, A. (2008). Understanding Progressive Enhancement. *A List Apart*, (269). Retrieved from http://alistapart.com/article/understandingprogressiveenhancement

Hay, S. (2013). *Responsive Design Workflow*. New Riders.

Hume, A. (2011, July 7). Responsive by Default. *andyhume.net*. Retrieved May 13, 2013, from http://blog.andyhume.net/responsive-by-default/

Internet Explorer 6 usage around the world. (2013). Internet Explorer 6 usage around the world. *Modern IE*. Retrieved December 8, 2013, from http://www.modern.ie/en-us/ie6countdown

JHS 129 Julkishallinnon verkkopalvelun suunnittelun ja toteuttamisen periaatteet. (2005). *JHS 129 Julkishallinnon verkkopalvelun suunnittelun ja toteuttamisen periaatteet*. Retrieved from http://docs.jhs-suositukset.fi/jhs-suositukset/JHS129/JHS129.html

Joly, K. (2012). One Design to Rule Them All. *University Business*, *15*(2), 49–50. Retrieved from http://www.universitybusiness.com/toc/6387

Joly, K. (2013). To Go or Not to Go … Responsive. *University Business*, (May 2013), 58. Retrieved from http://www.universitybusiness.com/internettech

Keith, J. (2011, April 27). Content First. *Adactio*. Retrieved December 8, 2013, from http://adactio.com/journal/4523/

Knight, K. (2009, June 2). Fixed vs. Fluid vs. Elastic Layout. *Smashing Magazine*. Retrieved April 10, 2013, from http://coding.smashingmagazine.com/2009/06/02/fixed-vs-fluid-vs-elastic-layout-whats-the-right-one-for-you/

Leibtag, A. (2012, October 23). Is a Mobile-First Mentality Right for Your Organization? *EContent*. Retrieved October 21, 2013, from http://www.econtentmag.com/Articles/Column/Content-Ahas!/Is-a-Mobile-First-Mentality-Right-for-Your-Organization-85363.htm

Lloyd, P. R. (2012, September 25). The Web Aesthetic. *A List Apart*. Retrieved April 13, 2013, from http://alistapart.com/article/the-web-aesthetic

Marcotte, E. (2010). Responsive Web Design. *A List Apart*, (306). Retrieved from http://alistapart.com/article/responsive-web-design/

Marcotte, E. (2011). *Responsive web design*. A Book Apart.

McGrane, K. (2012). *Content Strategy for Mobile*. A Book Apart.

Mohomed, I., Cai, J. C., Chavoshi, S., & de Lara, E. (2006). *Context-aware interactive content adaptation*. *the 4th international conference* (pp. 42–55). New York, New York, USA: ACM. doi:10.1145/1134680.1134686

Monteiro, M. (2012). *Design is a Job*. A Book Apart.

Nebeling, M., & Norrie, M. C. (2013). Responsive design and development: methods, technologies and current issues (Vol. 7977, pp. 510–513). Presented at the ICWE'13: Proceedings of the 13th international conference on Web Engineering, Berlin, Heidelberg: Springer-Verlag.

Nichols, J., & Lau, T. (2008). Mobilization by demonstration: using traces to reauthor existing web sites (pp. 149–158). Presented at the IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces, New York, New York, USA: ACM Request Permissions.

Rivoal, F., Lie, H. W., Çelik, T., Glazman, D., & van Kesteren, A. (2012). *Media Queries*. W3C.

Rosmaita, B. J. (2006). Accessibility first!: a new approach to web design, *38*(1), 270–274.

Salminen, V. (2013, May 7). Responsive Workflow. Retrieved December 4, 2013, from http://viljamis.com/blog/2012/responsive-workflow/

Schilit, B. N., Trevor, J., Hilbert, D. M., & Koh, T. K. (2001). m-links: An infrastructure for very small internet devices (pp. 122–131). Presented at the MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking, New York, New York, USA: ACM Request Permissions.

Schonfeld, E. (2011). The Kindle Fire Will Have A Whole New "Cloud Accelerated" Mobile Browser Called Amazon Silk | TechCrunch. *TechCrunch*. Retrieved from http://techcrunch.com/2011/09/28/amazon-silk/

Sharp, R. (2010, August 10). What is a polyfill? *Remy Sharp's b:log*. Retrieved October 8, 2013, from http://remysharp.com/2010/10/08/what-is-a-polyfill/

Shrestha, S. (2007). Mobile web browsing (p. 187). Presented at the the 4th international conference on mobile technology, applications, and systems and the 1st international symposium, New York, New York, USA: ACM Press. doi:10.1145/1378063.1378094

Sterk, T. D. (2005). Building upon Negroponte: a hybridized model of control suitable for responsive architecture. *Automation in construction*, *14*(2), 225–232.

The New Multi-screen World. (2012). The New Multi-screen World.

Waugh, J. (2009). *Truly, CSS is awesome*. *Be the signal*. Retrieved from http://bethesignal.org/blog/2009/06/13/truly-css-is-awesome/

Wells, J., & Draganova, C. (2007). Progressive enhancement in the real world (pp. 55–56). Presented at the HT '07: Proceedings of the eighteenth conference on Hypertext and hypermedia, New York, New York, USA: ACM Request Permissions.

Weyl, E. (2013, June 2). Clown Car Technique. *Smashing Magazine*. Retrieved October 8, 2013, from http://coding.smashingmagazine.com/2013/06/02/clown-car-technique-solving-for-adaptive-images-in-responsive-web-design/

Wroblewski, L. (2011). *Mobile first*. New York: A Book Apart.

Zeldman, J. (2013). *Content First by Jeffrey Zeldman*. Jeffrey Zeldman. Retrieved from http://vimeo.com/70977623