Chapter 1 – Introduction

Can a web site be made responsive without having the frontend do all the work?

# Background

Modern web sites contain a multitude of functionality and rich, interactive content. With this rich content and interactivity there comes a cost in the form performance issues. Mobile devices are quickly becoming peoples main channel for accessing content on the web, much of which is either not designed for- or is poorly implemented for use on such devices. Web applications are often highly optimized on the backend for the sake of scalability through database tuning, clustering, customized data caching and so on, which allows them to handle a large number of requests. Although this performance tuning helps the applications service a large number of users, the users themselves do not experience these optimizations in any tangible manner. Users only care about their own request, and if it is slow the quality of the user experience is severely diminished.

## Web performance

Steve Souders, author, creator of the web browser performance plugin YSlow and engineer at Google, suggests that because of this, we should focus on improving the response time on the frontend. The frontend, he says, stands for 80-90 percent of the response time. He suggests a list of best practices aimed at improving the performance of web pages through frontend optimization:

I set out to capture these best practices in a simple list that is easy to remember. The list has evolved to contain the following 14 prioritized rules:

1. Make fewer HTTP requests
2. Use a content delivery network
3. Add an Expires header
4. Gzip components
5. Put stylesheets at the top
6. Put scripts at the bottom
7. Avoid CSS expressions
8. Make JavaScript and CSS external
9. Reduce DNS lookups
10. Minify JavaScript
11. Avoid redirects
12. Remove duplicate scripts
13. Confi gure ETags
14. Make Ajax cacheable

- Steve Souders

With the increasing popularity of using mobile devices to web content, several design philosophies and best practices have come along to help web developers design their web pages to work for both mobiles and desktops alike. The one that has gained the most traction over the past couple of years is Ethan Marcotte’s book “Responsive Web Design”. Roughly speaking it aims to make web sites “respond” to the context in which they are being viewed. This is primarily achieved through something called “media queries” in CSS. Media Queries can be used to detect certain attributes of the device rendering the web page, e.g. screen width and height, which then can alter the layout of the web page to fit the result of the query. While this method is practical for developers in terms of giving them an easy way to make their web pages adapt to their environment, it leaves the whole job of making the web page responsive to the frontend. This is somewhat contradictory to Souders idea of optimizing the frontend of web applications, because that is where the largest chunk of the response time is spent in such a solution. Responsive Web Design (RWD) is reliant on heavy use of CSS (stylesheets) and also needs to use JavaScript to hide and alter content to fit each device. Both of these things go against Souders principles. Doing these kinds of alterations on the frontend also does not reduce the amount of HTTP requests, as these have their roots firmly set in the backend and the HTML document that is sent to the user.

## Making the backend smarter

Jon Arne Sæterås, Mobile Web evangelist, blogger and product director at Mobiletech suggested in his post “Next steps of Responsive Web Design” letting the device do all the hard work of being responsive is neither fair, right or smart.

It is not only the design of the web site and the layout of content that needs to be adapted of enhanced; the idea of being responsive, adaptive and enhancing, must be implemented in the whole value chain.

– Jon Arne Sæterås

In its current form, RWD sends the same markup, CSS, JS and images to all devices, regardless of their capabilities. What he means by this statement is that more parts of the Web hierarchy must be made to be responsive. Being able to move much of the work over to the backend can potentially reduce much of the current load on devices viewing Web pages designed using RWD. Involving the whole “value chain”, as he calls it, will result in Web pages consuming less time and resources on the frontend, leaving the responsibility of things like image scaling and markup processing to the backend.

In this thesis we present an implementation of a possible solution for making web pages responsive without having all of the work done on the frontend. This is done with the aim of improving web performance on both desktop and mobile devices. The focus has been on the mobile aspect as this is where the biggest gains can be found in the context of improving performance, and thus the user experience. The implementation is a user agent feature detection plugin for the Enonic CMS, which aims to accurately detect the features supported by each individual user agent, so that the HTML document served to the user can be tailored on the backend before ever reaching the requesting user agent. The idea is that by tailoring the HTML on the backend, the performance on the frontend can be improved, following the concepts of both Souders and Marcotte.

# Structure

The thesis is structured in a “bottom up” fashion to provide all necessary background knowledge before delving into the details of the implementation.

Chapter 2 presents the technologies and concepts that this thesis is based upon, as well as the related work in this field.

Chapter 3 presents the Enonic CMS, how it supports plugins and the details surrounding the implementation of our plugin.

Chapter 4 will detail how we did the performance testing of the plugin and also present the results of the tests.

Chapter 5 discusses the merits of the plugin, the results of the performance tests and attempts to look at the results in the context of related work in the same field.

Chapter 6 will tie it all together and summarize our key findings.