

Bachelor's thesis
Information Technology
Internet Technology
2014

Krister Laakso

RESPONSIVE WEB DESIGN AND MAGENTO E-COMMERCE

– Creating a demo store



TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology

2014 | 52 pages

Patrick Granholm, Tomi Niemi

Krister Laakso

RESPONSIVE WEB DESIGN AND MAGENTO E-COMMERCE

This thesis is about responsive web design and Magento, the world's most popular e-commerce platform.

Today more and more people browse the web with mobile devices such as smartphones and tablets. Compared to a typical desktop screen, these devices limit the amount of pixels on the screen. Responsive web design is about having a one single HTML document that can be viewed on both mobile devices and on desktop screens without having to scroll the page horizontally. Hence, responsive web design eliminates the need for a separate mobile website.

As more and more people use their smart devices to browse the web, today many use their devices to make online purchases. In Finland, responsive web design combined with online stores has not been widely implemented.

In this thesis, a Magento-based online demo store was installed and configured with a responsive layout. The project was done for a small Finnish company specializing in web and mobile solutions, so that it could demonstrate the store to future clients.

The final piece of work can be found in <http://magentodemo.sofokus.com>

KEYWORDS:

Responsive, web design, Magento, e-commerce

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Information Technology

2014 | 52 sivua

Patrick Granholm, Tomi Niemi

Krister Laakso

RESPONSIIVINEN WEB-SUUNNITTELU JA MAGENTO VERKKOKAUPPA

Tämä opinnäytetyö kertoo responsiivisesta web-suunnittelusta sekä maailman suosituimmasta verkkokauppa-alustasta, Magentosta.

Älylaitteiden, kuten puhelimien ja tablettien käyttö on tänä päivänä hyvin yleistä ja ihmiset käyttävät internetiä päivittäin älylaitteillaan. Näiden laitteiden näytöt ovat pienempiä, niin fyysisesti kuin pikselimäärän suhteen, verrattuna perinteiseen pöytäkoneen näyttöön. Responsiivisen websuunnittelun tavoite on tehdä yksi sivusto, mitä on sujuvaa käyttää päätelaitteesta riippumatta, siten eliminoiden tarpeen erillisille mobiilisivuille.

Kun ihmiset käyttävät internetiä päivittäin, on myös tärkeää, että voi tehdä ostoksia verkossa mobiilisti. Responsiivinen web-suunnittelu yhdistettynä verkkokauppa-alustoihin on jotain, mitä Suomessa ei ole laajalti tehty.

Tässä opinnäytetyössä toteutetaan Magento demo-verkkokauppa responsiivisesti. Työ tehtiin suomalaiselle IT-ratkaisuja toimittavalle pienyritykselle. Demokaupan tarkoitus oli demonstroida Magento-pohjaisia verkkokauppoja ja niiden toimintaa tuleville asiakkailleen.

Lopullinen työ löytyy osoitteesta <http://magentodemo.sofokus.com>

ASIASANAT:

Responsiivinen, web-suunnittelu, Magento, verkkokauppa

CONTENTS

LIST OF ABBREVIATIONS (OR) SYMBOLS	5
1 INTRODUCTION	6
2 RESPONSIVE WEB DESIGN	7
2.1 What is responsive web design	7
2.2 Mobile first –principle	7
2.3 Pixels to percentages	8
2.4 Media queries	9
2.5 Flexible Images	12
2.6 Optimizing for a smaller screen	14
3 MAGENTO	20
3.1 What is Magento	20
3.2 Hierarchy	21
3.3 Products, catalogs and categories	22
3.4 Designs and themes	24
3.5 Checkout process	26
3.6 Extending Magento	30
4 DEMOSTORE - A PROJECT FOR SOFOKUS OY	32
4.1 Purpose of the project	32
4.2 Why Magento?	33
5 PROJECT IMPLEMENTATION	35
5.1 Planning	35
5.2 Installation	35
5.3 Configuring	38
5.4 Release	45
6 CONCLUSION	50
REFERENCES	51

LIST OF ABBREVIATIONS (OR) SYMBOLS

#	CSS ID selector. See CSS for more information.
Apache	HTTP server by Apache Software Foundation
CMS	Content Management System. A system for managing content and providing it in various formats
CSS	Cascading Style Sheets. A style sheet language for defining the visual image of HTTP documents
CSV	Comma-separated values. Files for storing tabular data.
FTP	File transfer protocol. A protocol used for transferring files over the Internet.
HTML	Hypertext Markup Language. The Main markup language for creating web pages.
HTTP	Hypertext Transfer Protocol. A protocol used in transferring HTTP documents over the Internet
JavaScript	A dynamically typed programming language. Used for creating web pages.
MySQL	Open-source relational database management system
PHP	Hypertext Preprocessor. A programming language for web development
PHTML	A file extension type. PHTML files contain both HTML and PHP code.
SSL	Secure Sockets Layer. A protocol for encrypting information over the Internet
XAMPP	A free and open source cross-platform web server solution stack package. Contains Apache Server, MySQL, PHP and Perl.
XML	Extensible Markup Language. Defines a set of rules for encoding documents

1 INTRODUCTION

In today's modern world people are becoming more and more dependent on the Internet. Many mobile devices exist, including smart phones and tablets, and they are at our disposal.

In Finland alone, smart phone penetration has risen to 45% of the population and 57% of smart phone users access the Internet every single day using their smart device (Google, 2013). Technology is evolving rapidly and mobile devices are here to stay. This growth obviously creates a vast demand for websites that are mobile responsive meaning that one single website should accommodate the needs of a mobile device screen and a desktop screen simultaneously.

In Finland 26% of smart phone users have made a purchase using on their phone (Google, 2013), which implies that more and more people are purchasing products and or services using their mobile device. This means that not only information-rich websites but also task-focused websites such as online stores should be responsive in a way that it is made easy for the consumers to browse the web.

None of the largest online stores in Finland are optimized for mobile users and that brings us to the purpose of this project. The purpose of the project was to combine responsive web design with the world's most popular open source e-commerce platform Magento (Magento Marketing, May 2013).

A responsive online store, from now on called Magento Demo Store, is created so that a company named *Sofokus Oy* could use it as a reference demonstrating what they do for their perspective clients.

Although there are many other e-commerce platforms available, Magento was chosen for three different reasons. First of all, it is open source (i.e., free). Secondly, it is the most popular in the world. Finally, during the implementation of this project there were people who work with Magento every day available to help.

2 RESPONSIVE WEB DESIGN

2.1 What is responsive web design

Responsive web design is web design with one key principle in mind: to optimize a website in a way that it can accommodate a screen with an arbitrary width. This screen could be a high definition desktop widescreen, a small mobile phone screen having a width of approximately 300 pixels or anything in between. A responsive website differs much from what people usually refer as a mobile site.

A mobile website is a separate website from the original website usually built after the original website is completed. This means that an administrator, who is responsible for maintaining the site, has two separate websites to maintain doubling the amount of work he or she has to do.

A responsive website has only one database behind it, meaning that when the administrator maintains the website, the changes take place immediately in both desktop view and mobile view. In other words designing websites to be responsive reduces the workload.

The key points of responsive web design are taking advantage of CSS3 and optimizing the content of a website for a small screen mobile device with a possibly slower Internet speed.

2.2 Mobile first –principle

Usually a mobile website or a mobile application is created after the original website designed for a wide desktop screen is completed (Wroblewski, 2011, p. 1). Either a separate mobile site is created or the current site is redesigned to adapt to a mobile. Regardless of which option is chosen, this could add up to an enormous amount of extra work to be done to accomplish this one goal. This is where the mobile first comes in.

The mobile first -principle refers to the idea that the designing process of the website should start with the idea that the website is first created for mobile and after that expanded to a desktop screen. This reverses the old way of a design process. Nevertheless, mobile first is a good and recommended practice because it can lead to a better overall user experience for a website (Wroblewski, 2011, p. 1).

As mentioned earlier in Finland alone 44% of smart phone users access the Internet every single day using their phone. On a global scale mobile devices account for 37% of the overall Internet traffic. These numbers are expected to grow higher every single day as the consumer PC sales are flat and unlikely to grow in the near future (Sterling, February 2013).

These figures support the fact that the mobile first -principle should be applied in today's web design due to the fact that mobile use is growing rapidly every year.

2.3 Pixels to percentages

One of the most important aspects of responsive web design is using percentages instead of regular pixels when defining width's in the CSS code. Suppose that a typical non-responsive website is designed to be 960px in width and, in case a browser window is greater than 960px in width, it is centered in the window. In terms of CSS, this could be represented as follows:

```
#page { width: 960px; margin: 0 auto; }
```

Now if the browser window is resized to a width of 800px, which is less than 960px, horizontal scrollbars appear. This means that in order to see all the content in this page one needs to scroll from left to right. The same effect takes place if viewed on a mobile phone. So, how can this be transformed into what is called responsive size?

Instead of having pixel values in our CSS, we need to express those widths in relative, proportional terms. Once this is accomplished, the resulting grid can

resize itself as the viewport changes, but without compromising the design's original proportions (Marcotte, 2011, p. 29).

This would transform into following:

```
#page { width: 100%; margin: 0 auto; max-width: 960px; }
```

Now that the width is 100%, the width of the page will be the width of a browser window regardless of the window being on a desktop or a mobile screen. The "max-width" property is entirely optional but recommended due to the fact that the browser window could be stretched all the way to 10000 pixels, which would be quite absurd.

As there are margins in a Word document, like this one that was written, margins are needed in a web page as well. Given that our page has a content container, #container, margins should be defined in percentages as well.

```
#container { width: 90%; margin: 20px 5%; }
```

The value 20px represents vertical margins rather than horizontal margins. This value was given because margins should exist not only on the left and right-hand side but also on the top and on the bottom.

2.4 Media queries

Even though transforming pixels to percentages is the key to responsive design it alone is not the answer. Suppose that our #page element had a content bar on the left hand side and a side bar on the right hand side defined as follows:

```
#contentArea { width: 60%; margin: 0 2%; float: left; }
```

```
#sideBar { width: 32%; margin: 0 2%; float: right; }
```

Given that in the desktop view the width is 960px, the widths of the bars would be 576 pixels and 307 pixels and approximately 77 pixels for margins.

What if the browser window were narrowed down to 320 pixels? This would result in the bars having widths of 192 pixels and 102 pixels. This hardly seems ideal. This is where media queries come in.

A media query consists of a media type and at least zero expressions that match conditions of particular media features. It is a logical expression that can be true or false (W3C, June 2012).

An example of a media query:

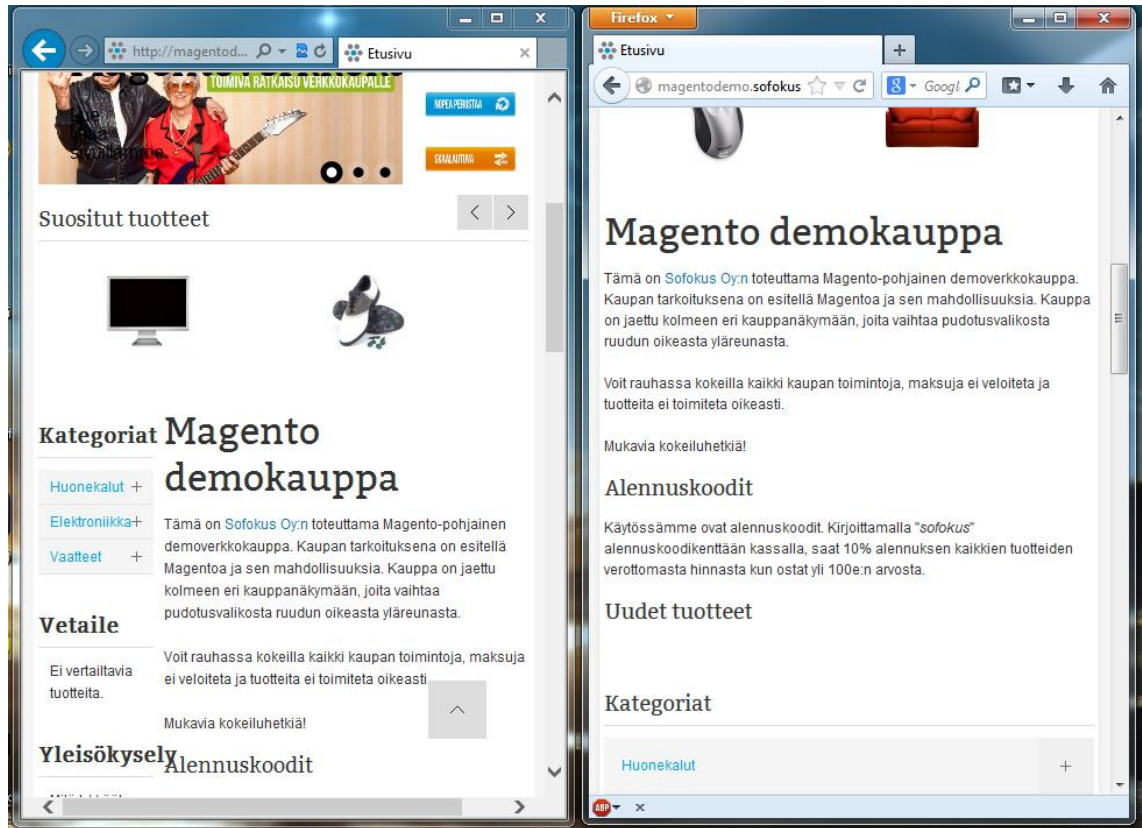
```
@media screen and (max-width: 480px) {  
  
#contentArea, #sideBar { width: 100%; }  
  
}
```

A media query consists of two individual parts: first, a media type “screen” and then the query itself. A media type screen is selected due to the reason that there is always a screen through which a web page is viewed. The query itself, in this example is “(max-width: 480px)”. This query can be split into two parts: a feature and a value, max-width and 480px respectively (Marcotte, 2011, p. 74).

This media query tackles our problem mentioned above. When the browser window is narrowed down, the content and the side bar become narrower pixel by pixel. At some point, these bars will be too narrow to fit content in such a way that a user would still find visually acceptable.

Inside the media query statement is the CSS code and the code will be applied when the conditions of that specific query are matched true.

Thus, when the width reaches a width of 480 pixels, these bars will receive new values for their widths. This results in the bars being re-positioned. The content will be on top of the sidebar both having width 100% of the width of the screen. An example can be seen in Picture 1.



Picture 1. On the left, media queries are not supported.

In this example, both the browsers have a width less than 480 pixels. The browser on the left side does not support media queries. However, the browser on the right does. On the right screen, the left side bar, which starts with “Kategoriat” (in English categories), has been assigned a width of 100% and has been placed after the content area.

There are no limits to the number how many @media queries one CSS file can have. In the example above, a width of 480px is considered a breakpoint because when the specific width is reached all the elements inside the media query will receive new values for their attributes. This is how a web page adapts to different widths.

A well designed web site will have many media queries at specific breakpoints. This allows more flexibility when adapting from desktop view to mobile view or even tablet view in between these two.

It could be easily assumed that the more media queries there are the better. This is only partly true. Consider that there would be a media query every 50 pixels. This would give us extreme flexibility and the website would look perfect in every given width. The problem at hand is that the more media queries there are, the more work there is to be done. Like in every situation everything should be in balance.

There are no common breakpoints because there are no common screen sizes. However, one approach would be to create breakpoints whenever the layout of a webpage breaks (van Gemert, 2013).

In the Demo Store, which is discussed in detail in Chapter 5, the following breakpoints were applied:

```
@media only screen and (min-width: 960px) { /* css code here */ }
```

```
@media only screen and (min-width: 768px) and (max-width: 959px) { /*  
css code here */ }
```

```
@media only screen and (min-width: 480px) and (max-width: 767px) { /*  
css code here */ }
```

```
@media only screen and (min-width: 320px) and (max-width: 479px) { /*  
css code here */ }
```

```
@media only screen and (max-width: 319px) { /* css code here */ }
```

Through testing these breakpoints resulted in a good, flexible design in both desktop and mobile view.

2.5 Flexible Images

By using percentages instead of pixels and implementing media queries we are on our way to a responsive way of designing websites. Nevertheless, one key aspect remains and that is flexible images. By default, if an image element is introduced into HTML code without an explicit width, what would be the width of

that image? Simple, it is the width of the image. So where is the problem? Assume that the following code is inserted into HTML:

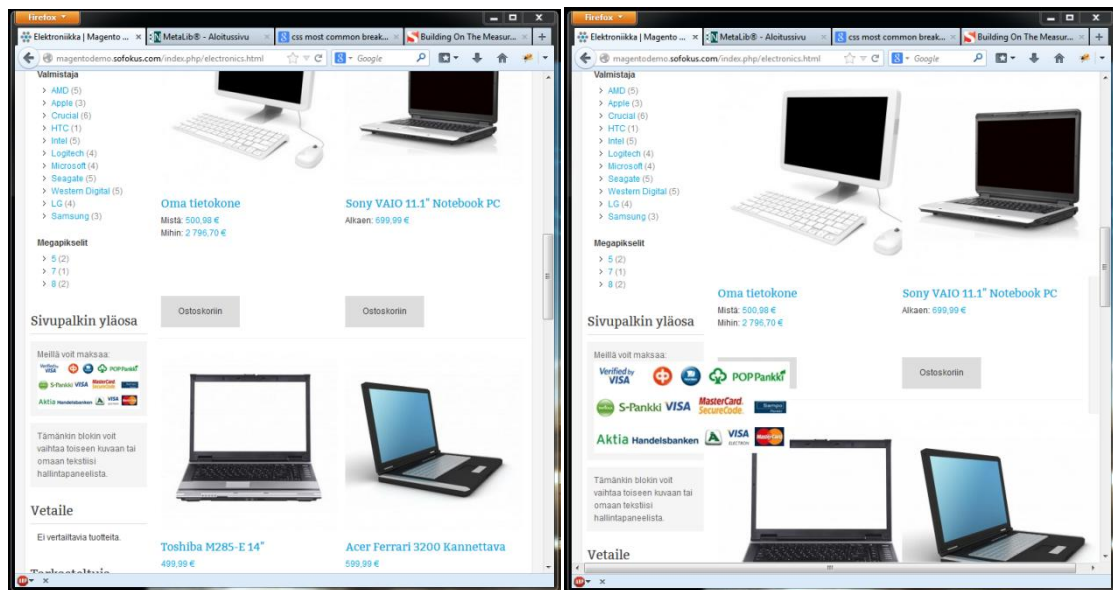
```

```

Given that no CSS code is applied to this img-element, the width of the image could be, for example, 600 pixels. If the width of the browser is 480 pixels and the image 600 pixels, this will break the layout. The image will overflow beyond the width of 480px. There is a simple way of avoiding this. In CSS, the following needs to be defined:

```
img { max-width: 100%; }
```

In all its simplicity this assures that the image will not be wider than its container. See Picture 2.



Picture 2. Flexible (left) and non-flexible image (right).

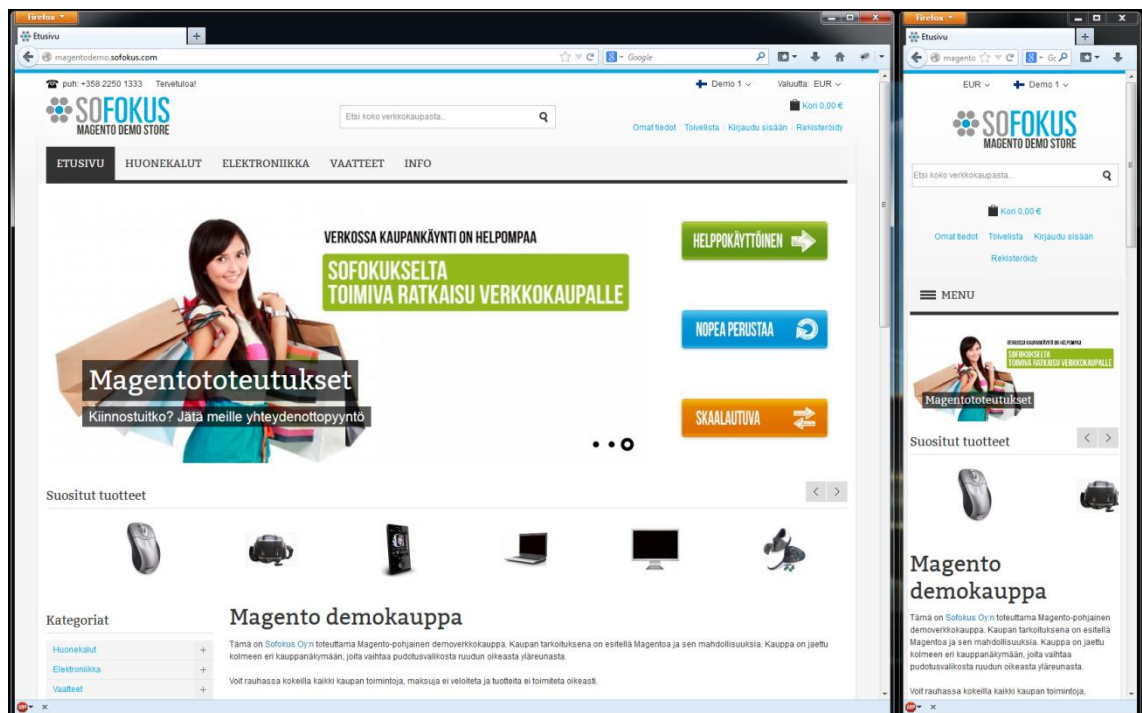
Notice that the image with logos of Finnish banks on the right does not look like something that the designer would want to do.

What makes this rule so simple is that newer web browsers have capabilities to render the image proportionally. In other words, the aspect ratio remains the same regardless of the current width of the resized image. "Max-width: 100%;"

is not limited only to img-elements in HTML but can be applied to video-, object- and embed-elements as well (Marcotte, 2011, p. 45-46).

2.6 Optimizing for a smaller screen

After scaling down the images a website feels responsive and scales to different widths. See Picture 3.



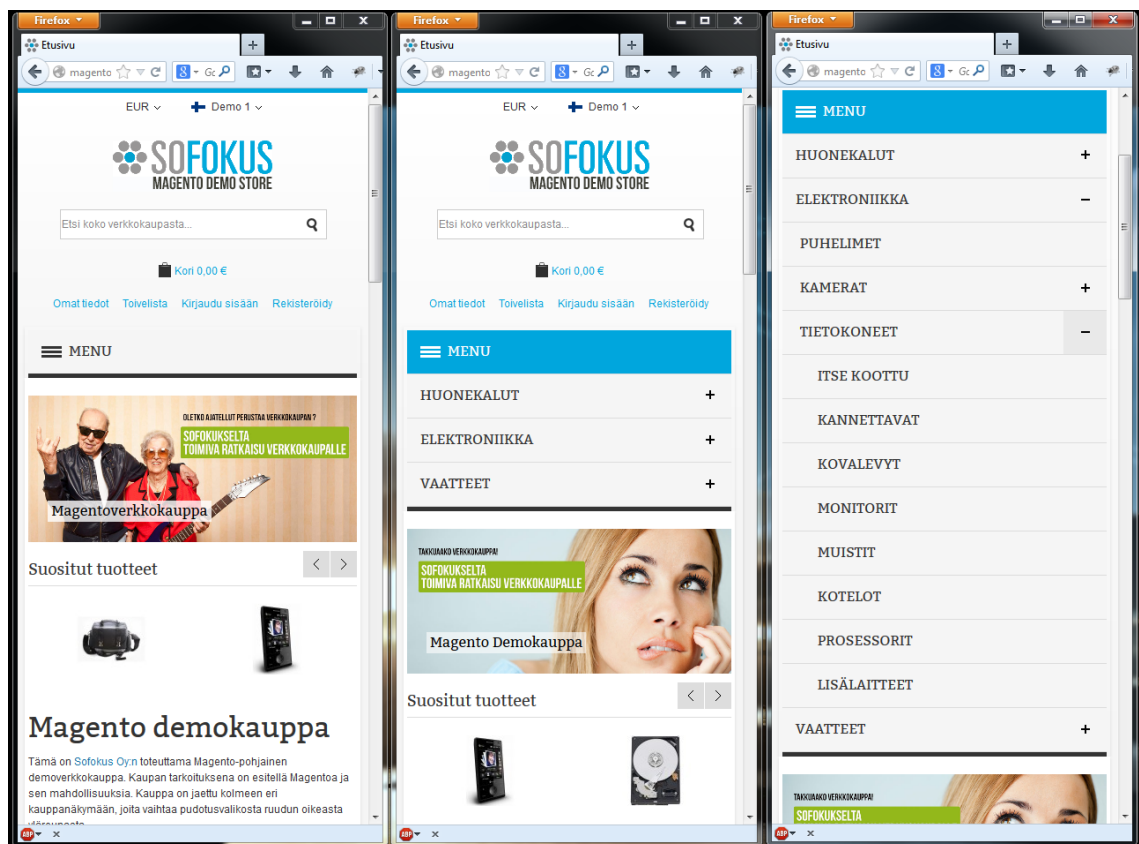
Picture 3. Desktop view and mobile view

After implementing percentages, media queries and flexible images it is time to start discussing the content. It is fairly obvious that when a user browses the website with his or her mobile device, there is far less pixels on the screen than on a 1920x1080 desktop screen.

This urges the designer to prioritize the content that is served to a user because the idea behind responsive web design is to serve one HTML document to a number of different browsers and devices. Mobile users will most likely want to have quicker access to different tasks than they would should they be browsing the site on their desktop computer (Marcotte, 2011, p. 107).

In Picture 3, two key elements have been put to the top, the logo and a search bar. Now when a user lands on this site, maybe through a Google search, the logo, *Sofokus Magento Demo Store*, is noticed immediately. By observing the logo, the user immediately knows what the site is about, a web store. After realizing it is an online store the search bar can be found just below the logo so it is easy to start looking for a product that might interest the user.

One of the most important aspects of mobile web sites is the navigation menu. By comparing the two views in Picture 3, the menu element is reduced into one Menu element in the mobile view. Furthermore, by observing Picture 4, the more categories and sub-categories are in the navigation menu, the greater the height of the element will be. By looking at the browser on the right, the menu fills the whole screen and there is no more room left for actual content. This greatly reduces the user experience and is not under any circumstances ideal.



Picture 4. Responsive menu

To solve this issue, a responsive menu should be applied. The menu element simply reduces the menu into one small element and by clicking (or tapping) it, it opens. This way, if a user wants to read what is on the front page he or she might not be interested in exploring the menu. Thus unnecessary scrolling is avoided.

Another important technique is to implement the CSS Display property. This simply hides a given element in the HTML document without removing it. An example is below:

```
#nonDesktopElement { display: none; }
```

Looking back at picture 3 (desktop view), next to the slider element there are three colored boxes on stacked on top of each other. On the right side they are nowhere to be seen. This is due to the reason that “display: none” was applied. The first reason was that if it had been there, there would not have been enough room to accommodate the slider element properly. The second reason was that the hidden element was not considered that important to show to the user. This is because in the mobile view the space is limited. However, the hidden images were still loaded when the page was loaded but this is not necessarily a problem due to the reason that the images were small in size.

Loading times are crucial when browsing the web with a mobile device. Although Finland may have decent mobile network coverage, other countries might not. Another aspect to consider is that some mobile operators or Internet service providers might have a type service which limits the data usage for the user. For example, a user might pay a fixed amount of money per month to use mobile data, but only up to two gigabytes a month. After two gigabytes have been used the service provider might the user extra.

This also emphasizes the importance of hiding content, which is not considered mandatory, but is recommended for good user experience. By applying the display none rule, it is possible to prevent content from loading at all which saves the user some data and loads the page faster.

This rule could be applied in a case where in a desktop view there is a large background image and a smaller optimized image in the mobile view.

In HTML:

```
<div id="desktopParent"><div id="desktopImage"></div></div>
```

```
<div id="mobileParent"><div id="mobileImage"></div></div>
```

And in CSS:

```
#desktopImage { background: url('large.jpg'); }
```

```
#mobileImage { background: url('small.jpg'); }
```

```
#desktopParent { display: block; }
```

```
#mobileParent { display: none; }
```

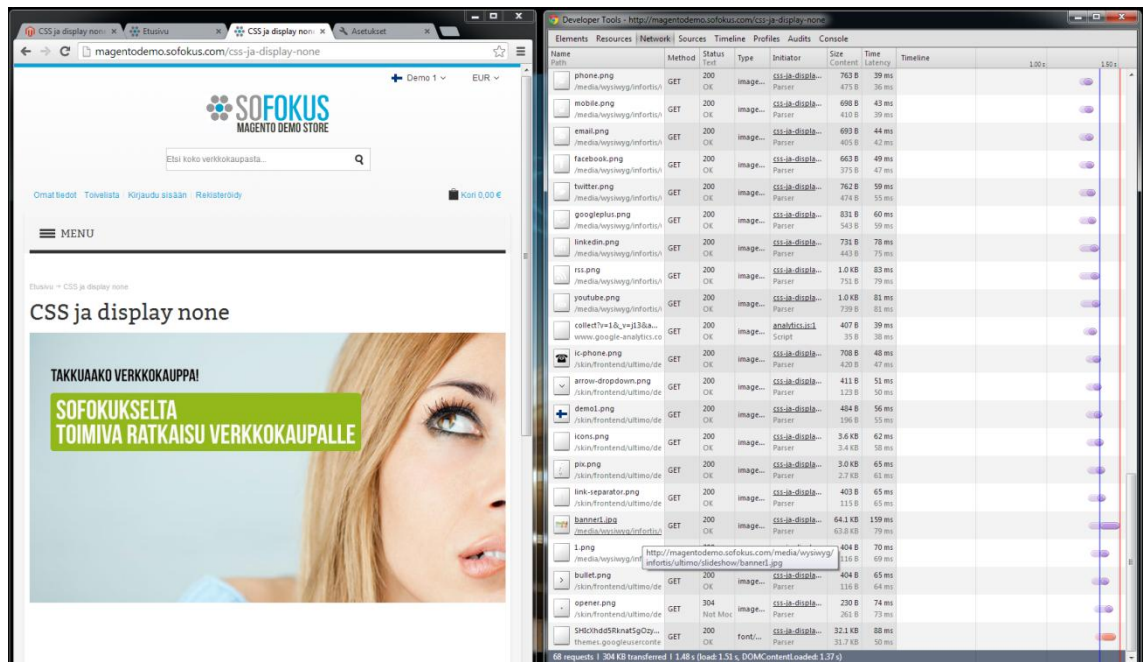
```
@media all and (max-width: 768px) {
```

```
#desktopParent { display: none; }
```

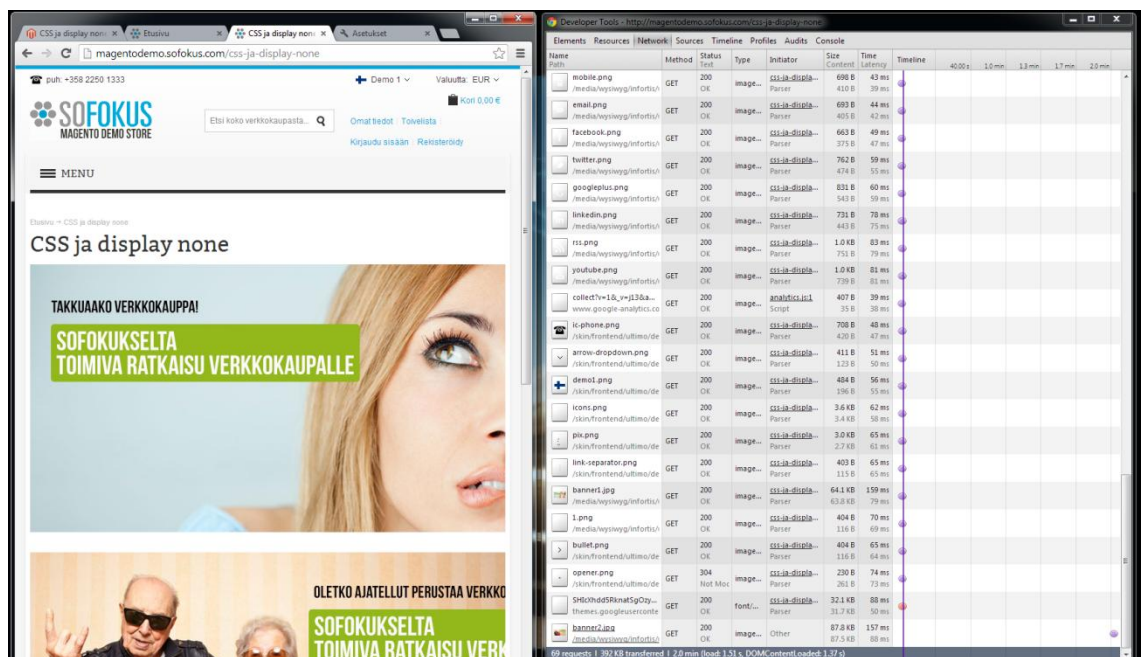
```
#mobileParent { display: block; } }
```

When the width is smaller than 768 pixels, the larger element is hidden and the smaller element is displayed. By wrapping the image in a parent element and hiding it, the child element is not requested, i.e., the image will not be loaded.

This feature was tested to confirm that this was true. A simple page was created applying the same principle mentioned above and two screenshots were taken.



Picture 5. One image visible, the other hidden. The browser width is 767 pixels.



Picture 6. Both images are visible. The browser width is 768 pixels.

On right side are Google Chrome developer tools which were used to monitor traffic from the server when loading a page. By comparing Pictures 5 and 6, it can be observed that the second image was not loaded until the browser width reached a breakpoint of 768 pixels. As a result, by applying the principle above

it is clear that by hiding such elements and implementing CSS display none the loading times are reduced.

3 MAGENTO

3.1 What is Magento

Magento, in short, is an e-commerce web application or in simpler terms a web browser based online store. It was originally launched in 2008 and its latest stable release version as of September 2013 is version 1.8.

Magento is an open source product, which means that the source code is open for anyone to explore and modify much like Linux operating systems. Magento's source code is written in PHP and for storing data it uses a MySQL database. Magento's PHP code runs within an Apache web server (Williams, 2012, p 14).

In terms of e-commerce platforms, Magento is the most popular e-commerce platform in the world (Magento Marketing, 2013) and by 2012 Magento Inc. itself claimed that users had downloaded the software package more than 2.5 million times and that it was used by over 80 000 companies worldwide (Mastering Magento, 2012).

There are three different versions of Magento available today. The first and the most important one is the Community Edition, which is completely free. Second and third are the Go Edition and Enterprise editions respectively. However, these versions cost \$15 per month and \$15 550 per year respectively. In this thesis the Community version was used due to the fact that it is free of charge.

The Community Edition, albeit free, is a versatile and highly scalable framework. It is certainly feature rich, from products and categories to search engine optimization and sales reporting to name a few (Magento Inc. 2012). Considering the large number of features supported by Magento this thesis focuses on the basic features in order to gain an understanding of the most important aspects of an e-commerce platform.

3.2 Hierarchy

One of the reasons Magento is flexible and scalable is the hierarchy. Magento calls this Global-Website-Store, or GWS, methodology. See picture 7.



Picture 7. Global-Website-Store methodology (Source: http://www.magentocommerce.com/images/uploads/multiple_websites_diagram.gif)

Global refers to a Magento installation. Global data is shared between all the websites and stores. An example of a common shared value is a product price. (Williams, 2012, p. 16)

A website refers to a website itself. One Magento installation can include one or more websites, each with their own domain name. The website is the root of a Magento store. A website may consist of a number of stores where each store represents a different set of products for example (Williams, 2012, p. 17).

A store, in a hierarchical structure, is used to associate different product catalogs to different stores within a website (Williams, 2012, p. 17). Suppose there is an entrepreneur who office supplies and printers. Although the supplies and the printers could be sold in a single catalog, one might consider dividing these products into two separate catalogs and thus two separate stores, both within a

single website. Regardless of the decision, Magento supports this feature making it flexible.

In addition to Picture 7 above, each individual store may include one or more store views. These views can be implemented for different language versions. These views can be identical but the other one could be e.g. in Finnish and the other one in Swedish. These store views are discussed further in Section 3.4 Designs and themes.

3.3 Products, catalogs and categories

In Magento a catalog represents a set of products. A catalog may include a variety of different products. These products can be divided into smaller groups, categories, and within these categories are the products themselves.

Every single store within a Magento installation can be assigned a catalog but a single product may be included into one or more catalogs (Williams, 2012, p. 58).

A category represents a sub section in a catalog. This can be something as simple as "laptops" in a catalog called "computers". However, Magento also supports special categories for special display options, such as "New products" or "Featured products". It is very common for a retail store to have something like this on their front page (Williams, 2012, p. 58-59).

The concept of a product is simple and self-explanatory. A product could be a simple piece of equipment or a repair service. This is where flexibility of product types in Magento comes in. Potential customers first decide the style of a product they find interesting and after that the possible variations. If someone is interested in acquiring a new shirt, they will go shopping for a shirt. It is after they enter the store when they start thinking about the color or the size of the shirt. This is why products in a store need to be presented in a logical and convenient way. (Williams, 2012, p.66)

Magento groups these products into six different types: Simple, grouped, configurable, virtual, bundle and downloadable (Williams, 2012, p 62-66).

A simple product is an individual product. An example could be an issue of a magazine. There might be a number of different issues of a certain magazine and most likely a number of magazines by a different publisher but in general, this product is sold individually.

Grouped products are products including two or more simple products displayed on a single product page. A shopper may choose only the products he or she wishes to buy. The grouping is usually applied to a group of products in a same category (Williams, 2012, p 62).

Configurable products are products with different variations. A t-shirt could be sold in three different colors and three different sizes. This adds to a total of nine different t-shirts each with their own stock-keeping-unit identifier. A configurable product combines these products into one single product with options to select for a customer, for example, a green shirt with in size M. This is a more convenient way of shopping for a customer (Williams, 2012, p 63).

A virtual product is a product that requires neither shipping nor inventory. This could be as simple as a repair service or a warranty (Williams, 2012, p 64).

A bundle product is a very customizable type of product. A bundle product, as its name suggests, bundles a list of products and gives the customer the option to create a product they desire by choosing the combination they like. A typical example is a "Build your own PC". Customers may select the components he or she will to get the combination that is best suited for him or her (Williams, 2012, p 64-65).

Downloadable products are distributed electronically. These are typically an eBook or a music file. Magento also keeps track of how many times a specific file has been downloaded (Williams, 2012, p 65).

3.4 Designs and themes

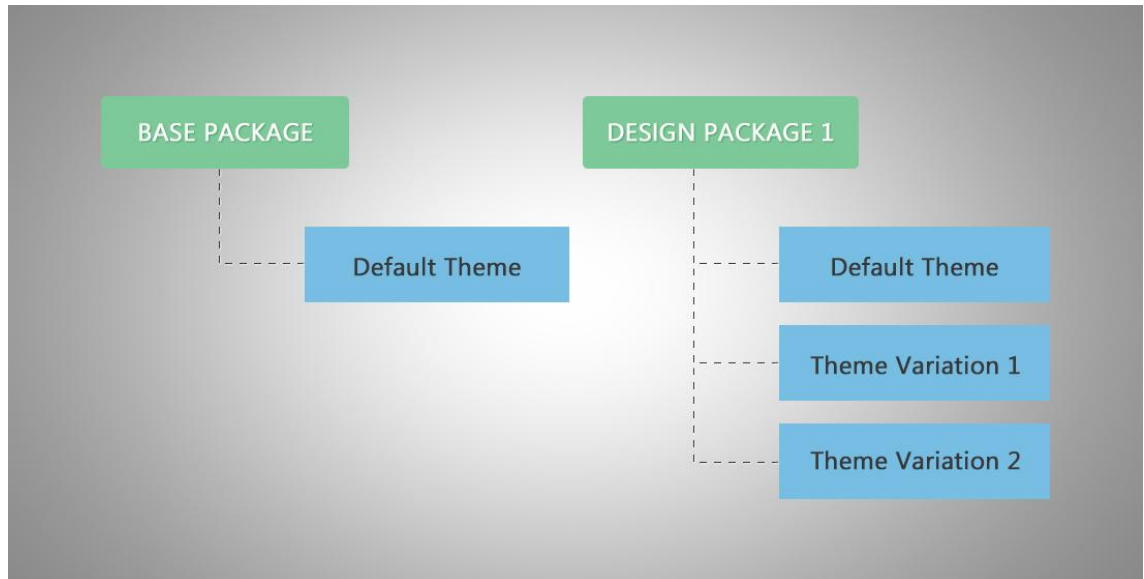
Having a business and selling products is not only about the products or the inventory. The store owner has to consider the interior design of the store. This is because if there was a jewelry store selling high end products, it would neither make much sense to have the products on display sitting on an old dusty table nor have the walls full of cracks. The same analogy applies to an online store. The graphics of a website have to have credibility telling the customer that it is a vendor one can trust. Respectively, the content has to be easily accessible for a natural browsing experience for it takes only a few seconds for the visiting person to decide whether to stay on the website or move one. (Williams, 20012, p. 83)

As mentioned in Section 3.2, a store in the GWS hierarchy may contain one or more store views. This allows customization at many levels within a website. (Williams, 20012, p. 84)

These store views may be customized with something that is referred to as a design package. When installing Magento, it comes with two design packages:

- A base package: a special type of package containing all the core files.
- A default package: a package containing the files for a default store.

A design package must contain at least one theme (Williams, 20012, p. 84).



Picture 8. Design package hierarchy (Source: <http://blog.belvg.com/create-custom-themes-magento-front-end-developer-certification.html>)

The design packages contain the following files:

- Layout files: XML files to define areas of the pages and how the HTML code of a page is structured.
- Template files: PHTML files which contain HTML and PHP code to define the visual structure of the pages.
- Locale files: Files that are used in language translations.
- CSS files: CSS code to give the elements a certain outlook.
- Image files: The image files used on the pages
- JavaScript files: JavaScript files for example for a slider element on the front page. (Williams, 20012, p. 84)

The package structure can be grouped into two different groups:

- Theme structure: This group contains layout files, template files and locale files.
- Skin structure: This group contains CSS files, Images and JavaScript files. (Williams, 20012, p. 85)

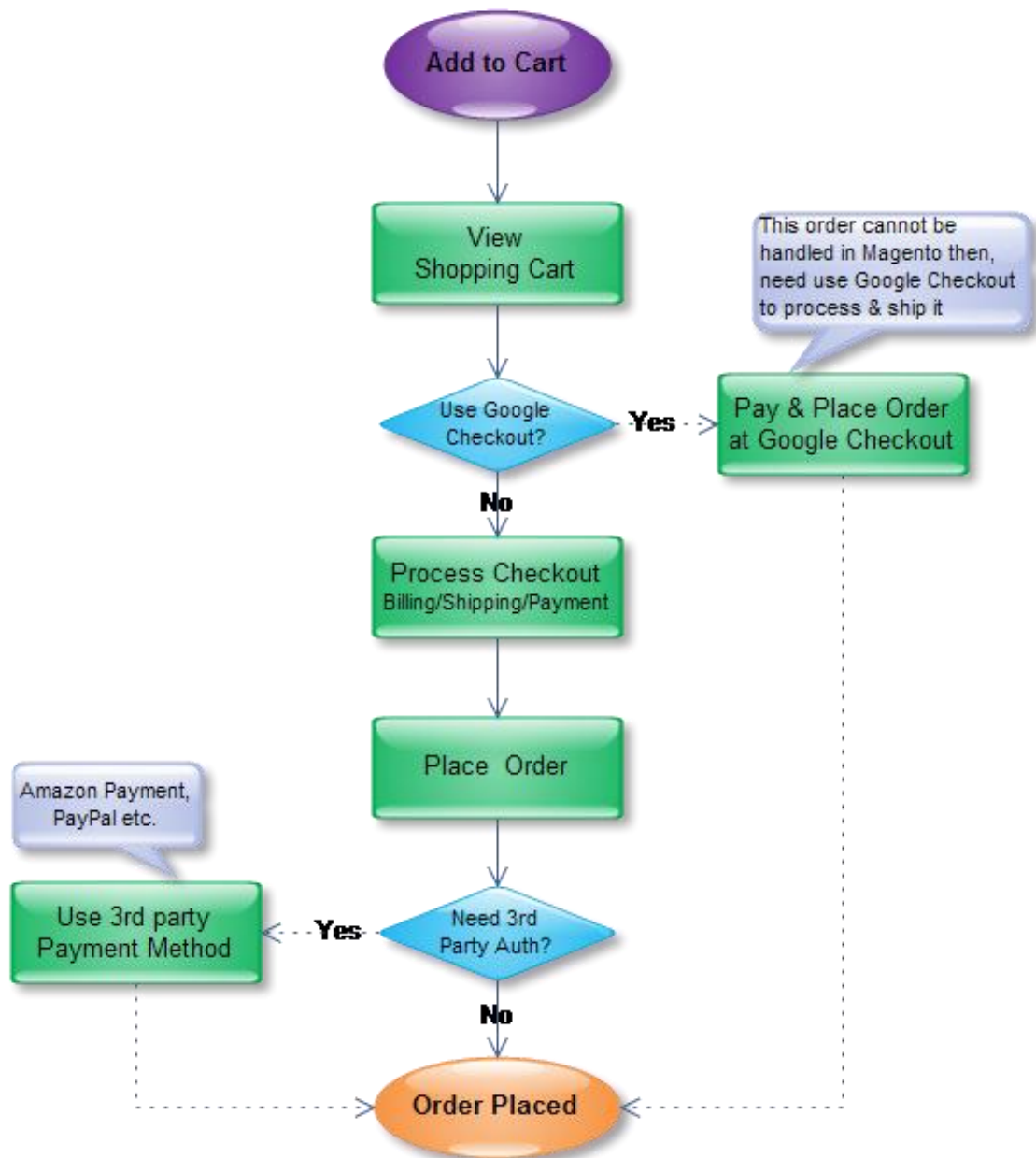
If a theme is missing one file in a package (reasons could be many), this is not considered a problem since Magento has implemented a theme fallback model. In other words, when Magento builds a page it first looks into the theme used in a given store view. If there is a file missing it looks back into default theme within that same package. If there still is no file to be found it finally looks back into the base package's default theme and finds a match. This is why the files in the base package should never be modified in any way due to the fact it is the key to Magento's fallback model. (Williams, 2012, p. 86)

These themes could be used to give the store a different look during Christmas time, for seasonal sales or perhaps use the same look for two different store views but have one view in English and the other one in Finnish.

3.5 Checkout process

From a customer's point of view, purchasing online is a simple process. First, a product is inserted into the shopping cart. After that, the customer proceeds to checkout. During the checkout, the customer supplied the merchant with required information, such as contact information, shipping information and payment information. Finally, an order is made and ready to be processed by the merchant.

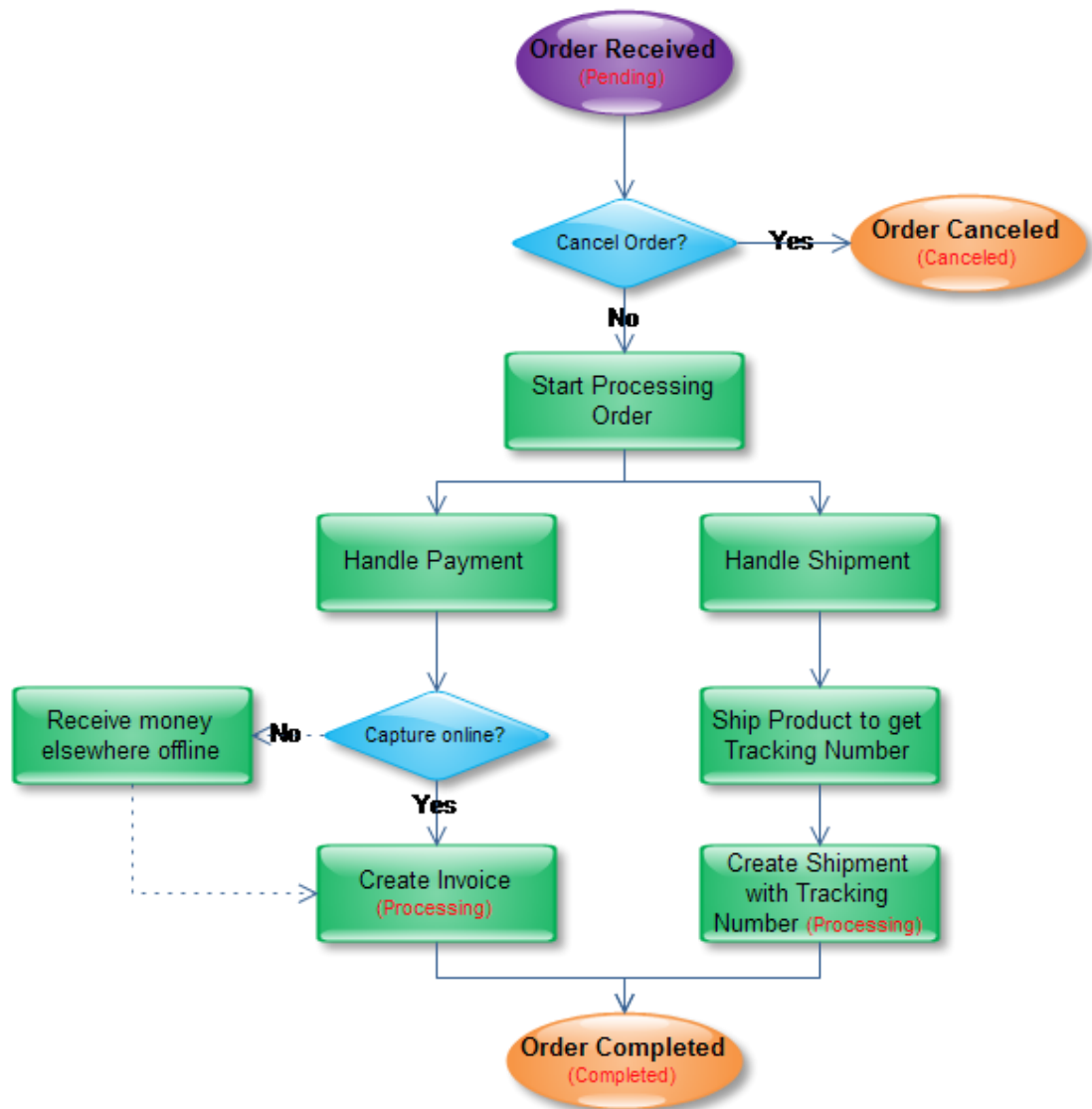
Magento® Purchase Flow



Picture 9. Magento purchase flow chart. (Source: www.magentocommerce.com/wiki/_media/general/magento_order_flow.png)

From the merchant's point of view, there many aspects to consider. Not only having an inventory of products is sufficient but when a purchase is made, there are payment methods, shipping methods and taxes to consider.

Magento® Order Processing Flow



Picture 10. Order processing flow chart (Source: www.magentocommerce.com/wiki/_media/general/magento_order_flow.png)

Selling, whether online or not, merchants are always subject to paying taxes to the government and tax rules differ from country to country. In Magento tax rules are applied. A tax rule consists of three pieces of information: customer tax class, product tax class and tax rate & zone. Based on the rule, Magento calculates the tax for a product. (Williams, 2012, p. 130)

After the tax rules have been applied to the products in the inventory, it is important to consider the possible shipping methods. Magento offers three different shipping methods: flat rate shipping, free shipping and table rate shipping. The first two are quite self-explanatory as flat refers to a fixed amount of money for the shipping fees and free is simply free.

The third, table rates, offer more flexibility when shipping different products to various shipping addresses.

	A	B	C	D	E	F
1	Country	Region/State	ZIP/Postal Code	Order Subtotal (and above)	Shipping Price	
2	USA	*	*	0	15	
3	USA	*	*	50	10	
4	USA	*	*	100	5	
5	USA	AK	*	0	20	
6	USA	AK	*	50	15	
7	USA	AK	*	100	10	
8	USA	HI	*	0	20	
9	USA	HI	*	50	15	
10	USA	HI	*	100	10	
11						

Picture 11, example of table rates (Source: http://www.magentocommerce.com/images/uploads/admin_shipping_tablecsv_hiak.jpg)

Referring to Picture 11, the example shows an example of price-based shipping fee calculations. The asterisk symbol refers to the keyword “any”. In other words, in the example regions are divided into three groups: all states except Alaska and Hawaii, Alaska and Hawaii. Next, the column order subtotal defines the shipping rates based on the subtotal of the shopping cart.

Other methods include weight-based shipping and price-based shipping. Using these methods, the table looks the same except for the fact that the column “Order subtotal” is changed to either “Order weight” or “Item quantity”. (Williams, 2012, p. 129)

Finally, after taxes and shipping methods are in order, it is time to accept the payment from the customer. Magento supports two types of payment systems: on-site and off-site (Williams, 2012, p. 121).

On-site payments refer to a payment system where the transaction takes place on the same site as the online store itself. The advantages are that the customer is kept on the same site as opposed to redirecting to an outside payment system such as PayPal thus eliminating the need for the buyer to register to an outside payment system. The disadvantages are that if the merchant is new, the customer might not trust the merchant enough to actually submit credit card information to the store owner. Moreover, on-site payments may make the merchant subject to PCI compliance, which in turn introduces concerns for security issues. Common examples of on-site payment systems include MoneyBookers and PayPal Pro. (Williams, 2012, p. 122-123)

Off-site payments take place on another website. In some cases, a customer might consider this a more secure way of submitting credit card information in order to make the purchase. After the customer has paid the off-site payment service, the payment service verifies that the merchant is legitimate and then pays the merchant. The advantages are more protection to customers against suspicious merchants and that there are no PCI compliance requirements. The disadvantages are that the customer is taken off the site thus giving the merchant limited access to buyer information. In addition, the customer may be required to register in a third-party payment system. Common off-site payment systems include PayPal Express and Google Checkout. (Williams, 2012, p. 121-122)

3.6 Extending Magento

Although Magento is a powerful and feature rich e-commerce framework sometimes a need to extend Magento's functionality might occur.

As Magento is a flexible and scalable framework, it is possible to install extensions on top of Magento. There are thousands of extension modules available

online, which include themes, utilities, site management tools and integrations to name a few. All of these extensions are found on Magento's website, <http://www.magentocommerce.com/magento-connect/>. (Williams, 2012, p.193)

There are free extensions available as well as paid extensions. Free extensions are easy to install using Magento's built-in Magento Connect Manager but paid extensions require manual installation. Once a suitable, and free, extension is found all the user needs to do is copy an extension key given and paste it into Magento Connect Manager. After that, Magento installs the extension automatically. (Williams, 2012, p.198-200)

The screenshot displays the Magento Connect Manager interface. At the top, there is a navigation bar with 'Extensions' and 'Settings' tabs, and links for 'Return to Admin' and 'Log Out'. The 'Settings' section includes a checkbox for 'Put store on the maintenance mode while installing/upgrading/backup creation' (checked), and a 'Create Backup' option with a dropdown menu set to 'Database'. Below this is the 'Install New Extensions' section, which has a search box for 'Magento Connect' and a text input for 'Paste extension key to install:' with an 'Install' button. The 'Direct package file upload' section has a 'Browse...' button and an 'Upload' button. The 'Manage Existing Extensions' section shows the channel 'Magento Community Edition' and a 'Check for Upgrades' button. A table lists installed packages with columns for Package Name, Installed version, Actions, and Summary.

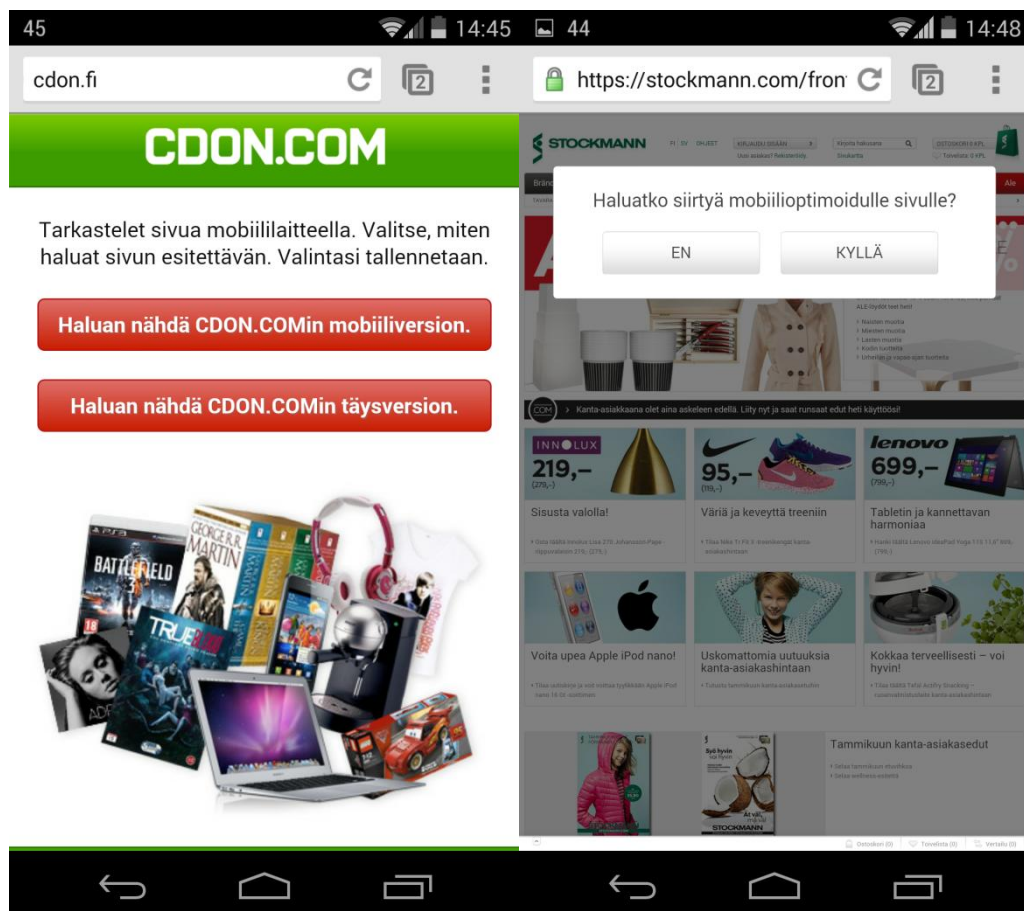
Package Name	Installed	Actions	Summary
Interface_Adminhtml_Default	1.7.0.1 (stable)		Default interface for Adminhtml
Interface_Frontend_Base_Default	1.7.0.1 (stable)		This is a Magento themes base
Interface_Frontend_Default	1.7.0.0 (stable)		Default interface for Frontend
Interface_Install_Default	1.7.0.0 (stable)		Default interface for Install
Lib_Google_Checkout	1.5.0.0 (stable)		Google Checkout Library
Lib_Js_Calendar	1.51.1.1 (stable)		Javascript Calendar for Magento
Lib_Js_Ext	1.7.0.0 (stable)		Extjs Javascript Libraries for Magento
Lib_Js_Mage	1.7.0.1 (stable)		Javascript Libraries for Magento
Lib_Js_Prototype	1.7.0.0.3 (stable)		Prototype and Scriptaculous Javascript Libraries for Magento
Lib_Js_TinyMCE	3.4.7.0 (stable)		TinyMCE Javascript Libraries for Magento
Lib_LinLibertineFont	2.8.14.1 (stable)		Libertine Open Fonts Project fonts for PDF print-outs

Picture 12. Magento Connect Manager

4 DEMOSTORE - A PROJECT FOR SOFOKUS OY

4.1 Purpose of the project

Online stores in Finland, or in the whole world, are nothing new and they have been around for a while now. By observing some of the largest online stores in Finland such as CDON, Gigantti, NetAnttila, Stockmann and Verkkokauppa on a mobile device, it is clear that none of these websites are built to be responsive. They either have no mobile optimized version of the site or upon entering the site, a prompt is displayed to the user to confirm whether to stay on the site or move to a mobile optimized one.



Pictures 13. Upon entering the site the user must select between the original site and a mobile optimized one, as opposed to one responsive site.



Pictures 14. The sites are not mobile optimized.

Therefore, the purpose of this project was to create an online store which was built responsive, adapting to the screen regardless of the width of the screen. This was also an opportunity to study Magento, the world's most popular ecommerce platform (Magento Marketing, May 2013). Hence, responsive web design and Magento were ideal topics to combine in order to write a thesis.

The goal was to create a responsive online demo store for Sofokus Oy, a small Finnish IT company, to use for marketing. Ideally, this store would demonstrate what Sofokus does – specialize in web and mobile solutions.

4.2 Why Magento?

Magento ecommerce platform was selected for the following reasons:

- It is the world's most popular ecommerce platform.
- It is feature rich.
- Its Community Edition is free and open source.
- *Sofokus Oy* specializes in Magento-based solutions and embraces open source technologies, thus giving the writer of the thesis much needed support in setting up the website.
- Magento has a very steep learning curve (Boag, 2011). Although not all the features of Magento were used during this process, it is always a wise decision to start from something difficult and later adapt to something easier than vice versa.

5 PROJECT IMPLEMENTATION

5.1 Planning

Once the idea of the project was clear, like every project, everything starts from planning. From the beginning it was clear that the Magento Demo Store was supposed to be a store where no real life money transactions were ever to be made. Nevertheless, the store still needed to look and feel real as if it were a legitimate online store.

- During the first meeting the following requirements were set:
- One Magento installation with one database
- Three different responsive design themes
- Localization for both Finnish and English
- A few products
- Appropriate extensions such as Checkout Finland. Checkout Finland is a payment service of Finnish banks such as Nordea and OP.
- Indexing and caching for better performance

With the following goals set, it was time to start setting up the store.

5.2 Installation

As Magento requires PHP and MySQL in order to operate, these services were needed to be installed. Luckily, a package called XAMPP is available free of charge and the package can be downloaded from their website <http://www.apachefriends.org/en/xampp.html>. XAMPP includes Apache Server (HTTP server), PHP (programming language compiler) and MySQL (database).

After the installation of XAMPP, two tasks had to be performed before installing Magento itself. First of all, a modification to the Apache server configuration was made. The document root folder and the server name had to be changed.

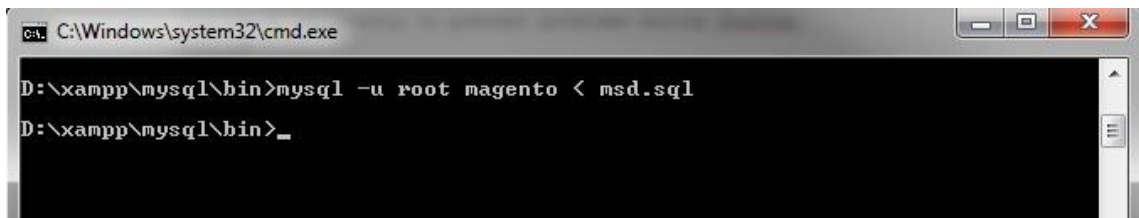
```

207 #
208 # ServerName gives the name and port that the server uses to identify itself.
209 # This can often be determined automatically, but we recommend you specify
210 # it explicitly to prevent problems during startup.
211 #
212 # If your host doesn't have a registered DNS name, enter its IP address here.
213 #
214 ServerName localhost:80
215
232
233 #
234 # DocumentRoot: The directory out of which you will serve your
235 # documents. By default, all requests are taken from this directory, but
236 # symbolic links and aliases may be used to point to other locations.
237 #
238 DocumentRoot "D:/xampp/htdocs/Magento"
239 <Directory "D:/xampp/htdocs/Magento">

```

Pictures 15a and 15b. Apache configuration changes

Secondly, Magento provides a download link for sample data for testing purposes. In other words, the sample data populates the database with sample products. Given that the store was a demo store, it was considered easier to use sample data rather than manually input products to the database. The data was inserted to the database with the following MySQL command:



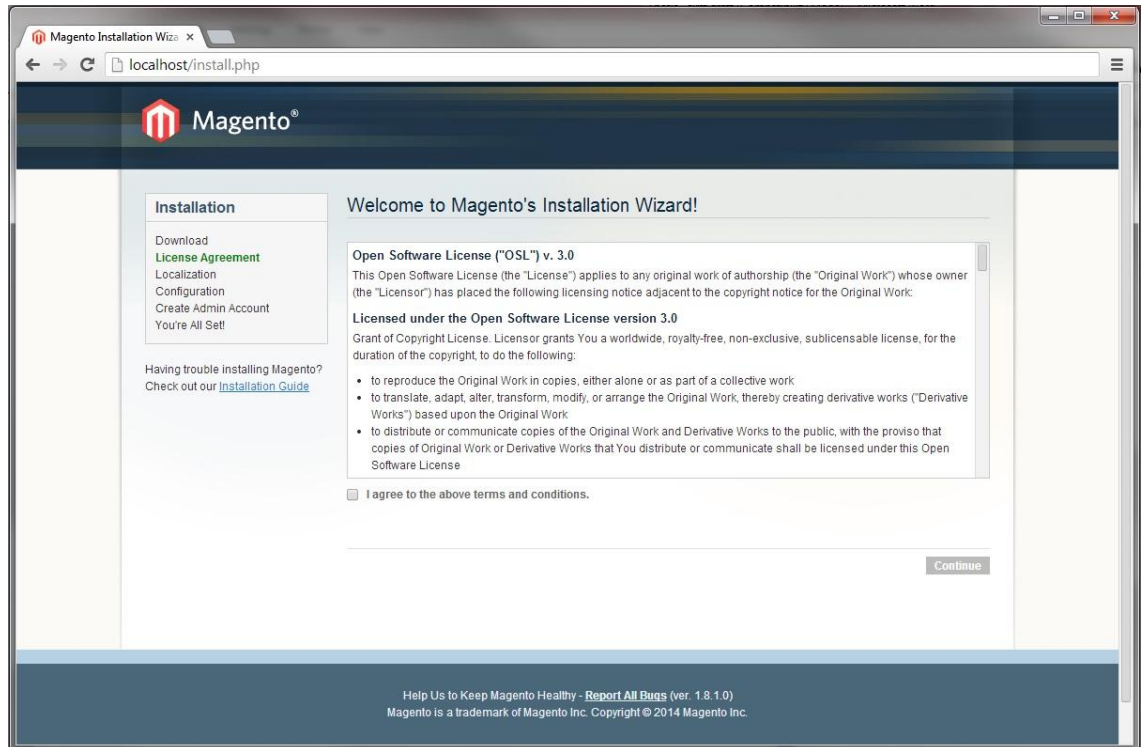
```

C:\Windows\system32\cmd.exe
D:\xampp\mysql\bin>mysql -u root magento < msd.sql
D:\xampp\mysql\bin>_

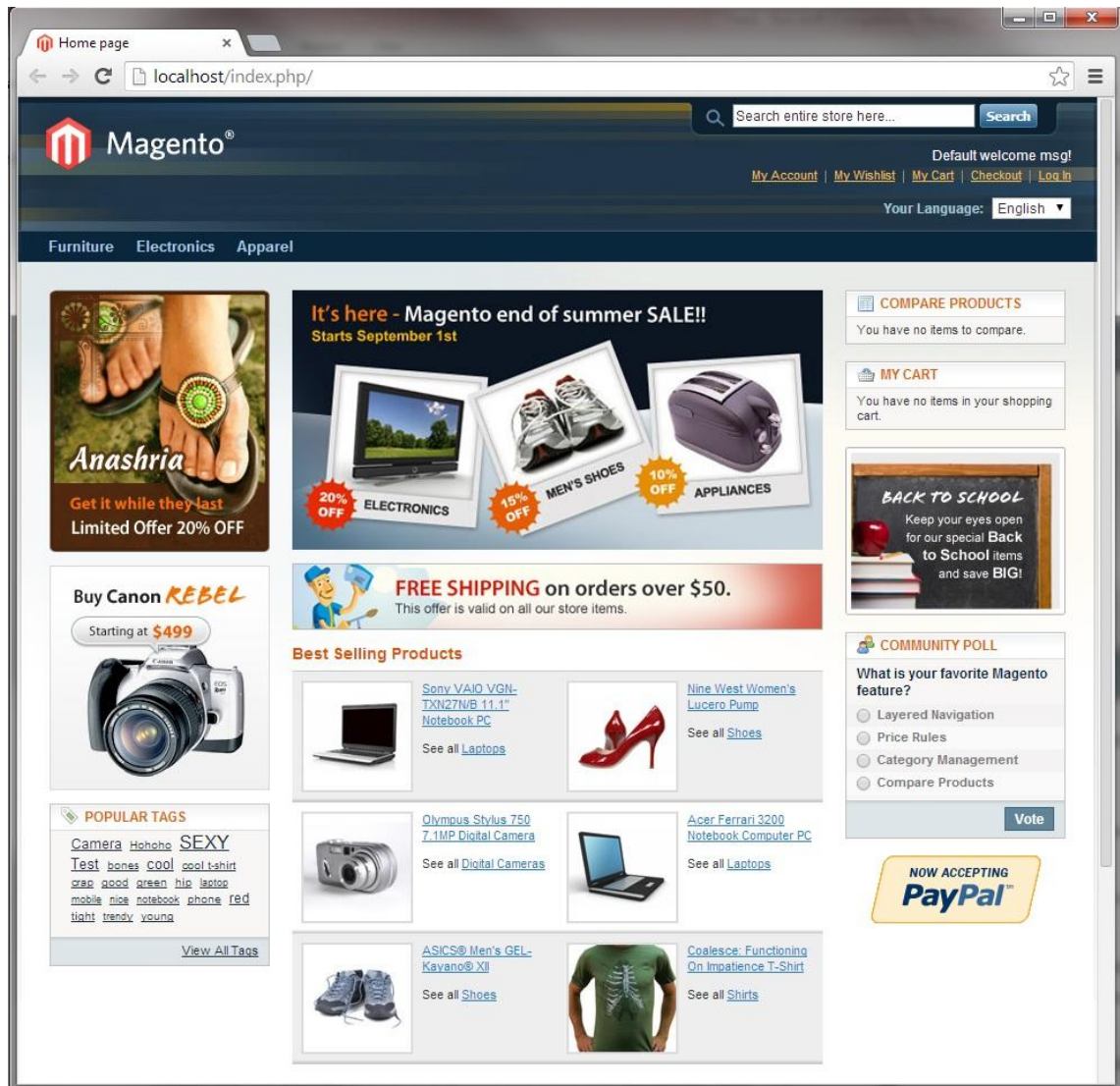
```

Picture 16. The command used to populate the database with sample data. The “msd.sql” refers to Magento sample data SQL file.

After these initial steps, it was time to install Magento itself by navigating to <http://localhost/install.php>. The installation was relatively simple as basic parameters were configured such as the language, time zone, currency, database settings and URL settings.



Picture 17. Magento Installation wizard

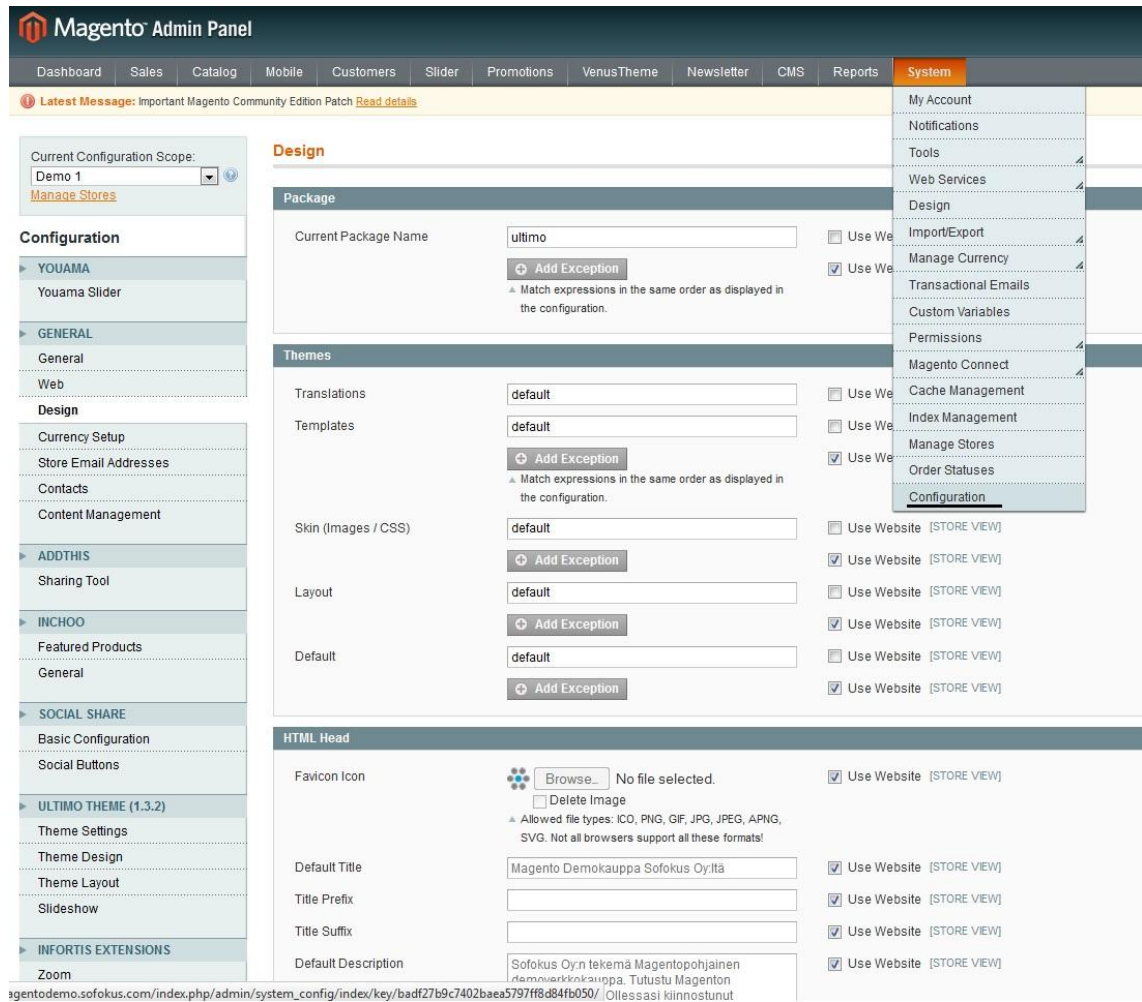


Picture 18. The front page after the installation

5.3 Configuring

The first step of configuring was to change the default theme (see Picture 18) to a theme that was responsive. For this demo store, two different themes were selected: one called “Ultimo” and another one called “Ves Hitech”. Most of the screenshots in this thesis taken from the demo store are from the theme “Ultimo”.

Theme installation was a simple and straightforward procedure. After downloading the source files, they were merged with the Magento installation folder and then by navigating to Admin backend -> System -> Configuration -> Design.



Picture 19. Configuring themes in Magento Admin Panel.

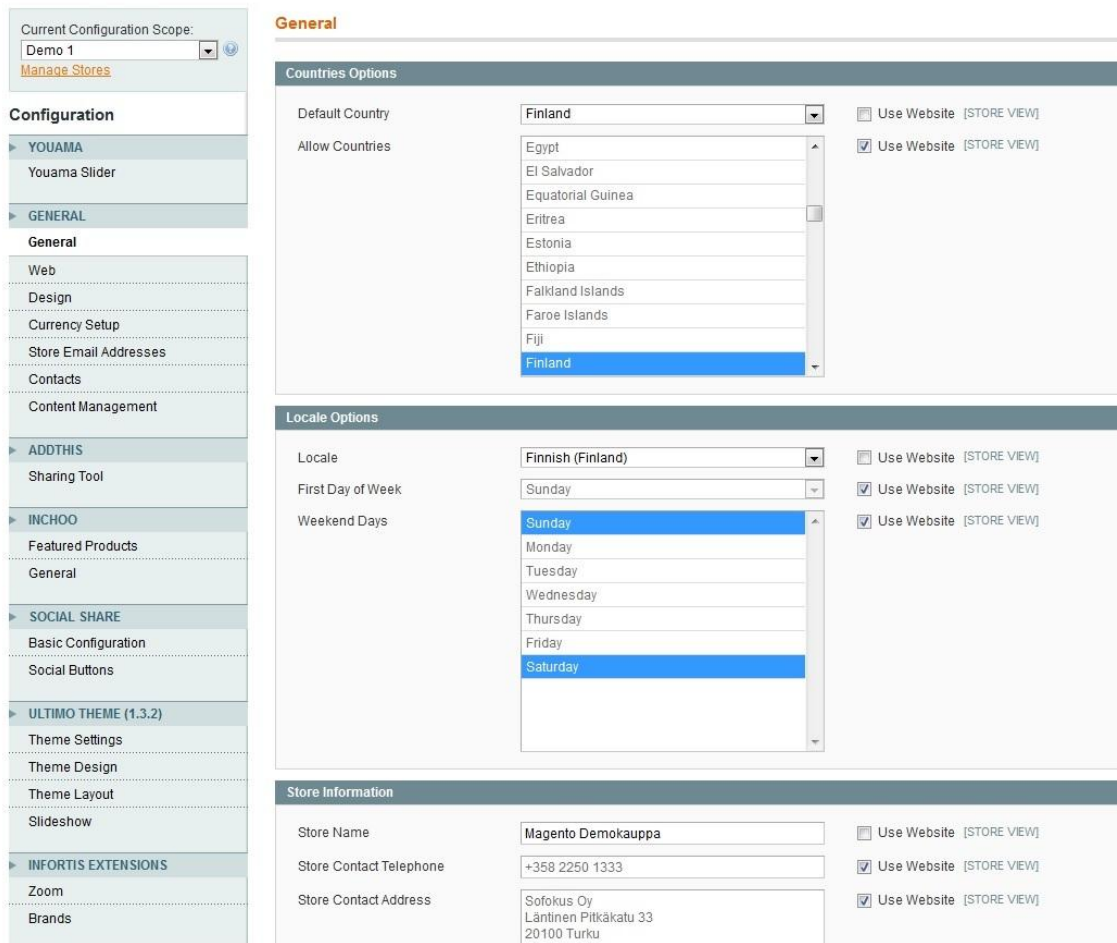
Changing the “Current package name” from “default” to “Ultimo” Magento looks for a package called Ultimo and its source files. Recall that if files are missing in package “Ultimo”, Magento will use its theme fallback model and use files from the default base package thus preventing any errors.

Next, the theme needed to be customized by changing the colors of the site and adding sliders to website. Luckily, the “Ultimo” package provided a menu for simple configuration under System -> Configuration -> Ultimo Theme

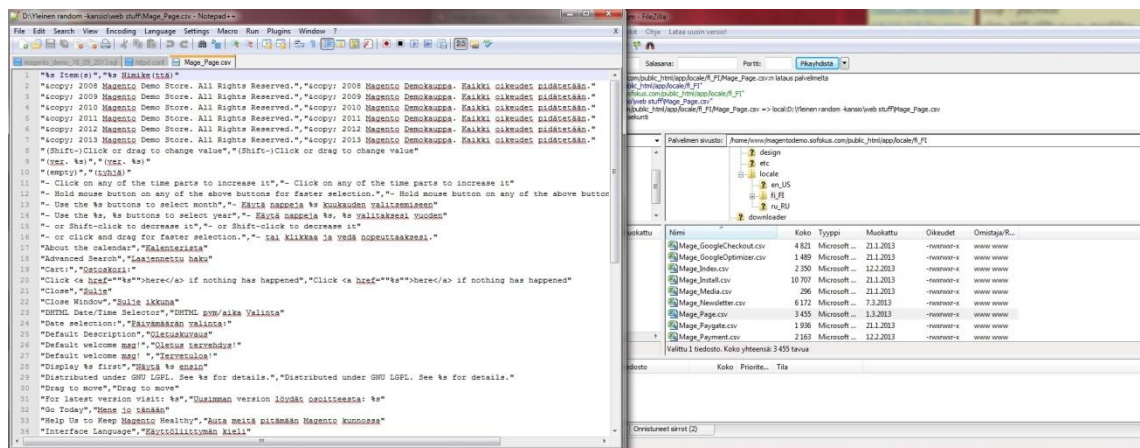
The screenshot displays the Magento Admin Panel interface. At the top, there is a navigation bar with various menu items: Dashboard, Sales, Catalog, Mobile, Customers, Slider, Promotions, VenusTheme, Newsletter, CMS, Reports, and Sys. Below this is a 'Latest Message' banner. The left sidebar contains a 'Configuration' menu with categories like YOUAMA, GENERAL, ADDTHIS, INCHOO, SOCIAL SHARE, and ULTIMO THEME (1.3.2). The 'ULTIMO THEME (1.3.2)' section is expanded, showing 'Theme Settings' as the active option. The main content area is titled 'Theme Settings' and is divided into two sections: 'Main Menu' and 'Disable Sidebar Blocks (On Home Page)'. The 'Main Menu' section lists various menu items like Category View, Product Page, and Footer, each with an 'Enable' dropdown set to 'Enable'. The 'Disable Sidebar Blocks' section lists various sidebar blocks with 'Disable' dropdowns, most set to 'No'.

Picture 20. Configuring the Ultimo theme.

Now that the theme was basically set up, it was time to set the locale to Finnish and add static content, i.e., pages (in Finnish). The locale settings are under System -> Configuration -> General. When a locale is set to Finnish, Magento looks into a folder Magento root/app/locale/fi_FI where the translation files are located. These files are in CSV, comma separated value, format.



Picture 21. Setting the locale to Finnish.



Picture 22. Translation files on the server. CSV file on the left hand side.

Obviously, changing a locale does not mean that all static content will be translated as well. Therefore, a few pages were needed to be made. Adding static

Categories

[Add Subcategory](#)

Choose Store View: Demo 1

[Collapse All](#) | [Expand All](#)

- Root Catalog (0)
- Huonekalut (7)
- Elektroniikka (42)
 - Puhelimet (6)
 - Kamerat (8)
 - Tarvikkeet (3)**
 - Digikamerat (5)
 - Tietokoneet (28)
- Vaatteet (66)
- Household Items (0)
- Muut (0)

Tarvikkeet (ID: 25)

[Reset](#) [Delete Category](#) [Save Category](#)

General Information | Display Settings | Custom Design | Category Products

General Information

Name * [STORE VIEW] Use Default Value

Description [STORE VIEW] Use Default Value

Image No file selected. [STORE VIEW] Use Default Value

Thumbnail Image No file selected. [STORE VIEW] Use Default Value

Page Title [STORE VIEW] Use Default Value

Meta Keywords [STORE VIEW] Use Default Value

Manage Products

Choose Store View: Demo 1

[Add Product](#)

Page 1 of 7 pages | View 20 per page | Total 121 records found | [Notify Low Stock RSS](#) [Reset Filter](#) [Search](#)

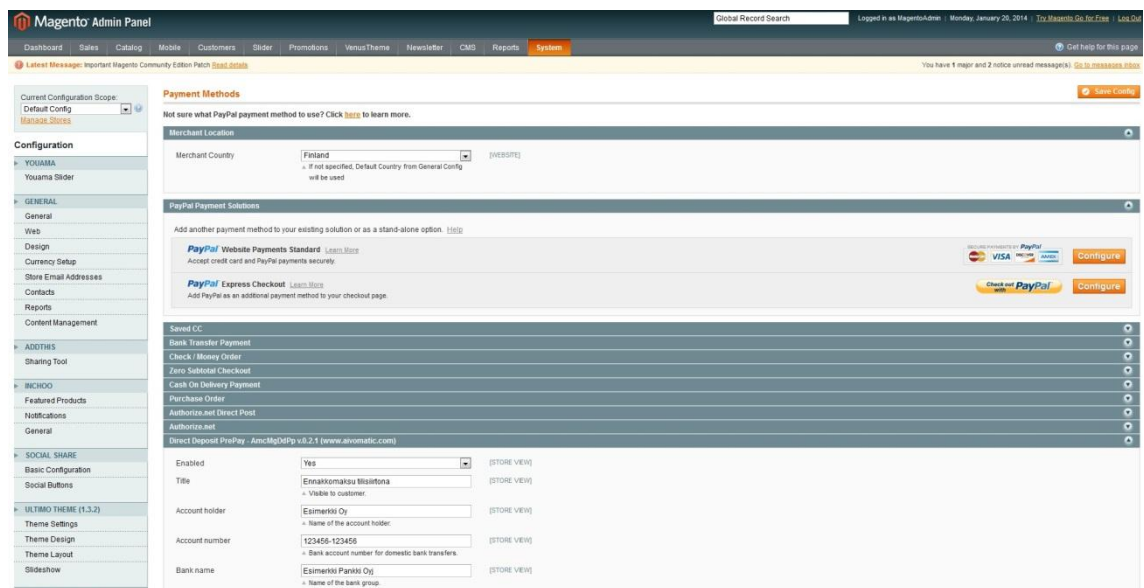
Select All | Unselect All | Select Visible | Unselect Visible | 0 items selected

ID	Name	Name in Demo 1	Type	Attrib. Set Name	SKU	Price	Qty	Visibility	Status	Websit
168	Sohva - sininen	Sohva - sininen	Simple Product	Default	couch-blue	€599.00	100	Catalog, Search	Enabled	Main Websit
166	HTC Touch Diamond	HTC Touch Diamond	Simple Product	Cell Phones	HTC Touch Diamond	€599.00	849	Catalog, Search	Enabled	Main Websit
165	Oma tietokone	Oma tietokone	Bundle Product	Computer	mycomputer		0	Catalog, Search	Enabled	Main Websit
164	Pelitietokone	Pelitietokone	Bundle Product	Default	computer_fixed	€4,999.95	0	Catalog, Search	Enabled	Main Websit
163	Pöytäkone	Pöytäkone	Bundle Product	Default	computer		0	Catalog, Search	Enabled	Main Websit
162	Microsoft Wireless Optical Mouse 5000	Microsoft Wireless Optical Mouse 5000	Simple Product	Default	micronmouse5000	€59.99	760	Catalog, Search	Enabled	Main Websit
161	Logitech diNovo Edge Keyboard	Logitech diNovo Edge Keyboard	Simple Product	Default	logidinovo	€239.99	209	Catalog, Search	Enabled	Main Websit
160	Logitech Cordless Optical Trackman	Logitech Cordless Optical Trackman	Simple Product	Default	logitechcord	€79.99	617	Catalog, Search	Enabled	Main Websit
159	Microsoft Natural Ergonomic Keyboard 4000	Microsoft Natural Ergonomic Keyboard 4000	Simple Product	Default	microsoftnatural	€99.99	275	Catalog, Search	Enabled	Main Websit
158	Sony VAIO 11.1" Notebook	Sony VAIO 11.1" Notebook PC	Bundle Product	Computer	VGN-TXN27N/BW		0	Catalog, Search	Enabled	Main Websit

Picture 25. Managing products and catalogs

Still, due to the imported sample data, the currency and tax rules did not correspond to a Finnish online store. Therefore, the currency was changed from US Dollar to Euro and the tax rates were changed to VAT 24%.

At this point, the store was already looking neat but one key aspect still remained, which was the checkout. There is an extension available for Magento called Checkout Finland. This enables the customer to pay directly to the merchant through Finnish online bank payment method. Considering that no real life transactions were to be made on this site because it was a demo store, luckily, there was a demo version of Checkout Finland available for testing purposes. This method was enabled along with credit card payment option and an advance payment option (“Ennakkomaksu tilisiirtona”). The payment methods are configured under System -> Configuration -> Payment methods. Note that the credit card payment option was enabled for credibility reasons and did not actually work properly due to the fact that this was a demo store.










The screenshot shows the Magento Admin Panel interface for configuring payment methods. The main content area is titled "Payment Methods" and includes a "Merchant Location" section where "Finland" is selected. Below this, there are sections for "PayPal Payment Solutions" (Website Payments Standard and Express Checkout) and a "Select CC" section with various payment methods. The "Direct Deposit PääPay - AmcMgDPp v8.2.1" method is expanded, showing fields for Enabled (Yes), Title (Ennakkomaksu tilisiirtona), Account holder (Esimerkit Oy), Account number (123456-123456), and Bank name (Esimerkit Pankki Oy).

Pictures 26. Managing payment methods

KASSA

Täytä tiedot varmistaaksesi ostosi!

Rekisteröitynyt? Kirjaudu sisään.

NIMI & OSOITE	TOIMITUSTAPA	TARKISTA TILAUKSESI																					
<p>Jos alla olevat kentät ovat tyhjiä, päivitä osoitteesi</p> <p>Etunimi * <input type="text"/> Sukunimi * <input type="text"/></p> <p>Sähköpostiosoite * <input type="text"/></p> <p>Osoite * <input type="text"/></p> <p>Kaupunki * <input type="text"/></p> <p>Postinumero * <input type="text"/> Puhelinnumero * <input type="text"/></p> <p>Maa <input type="text" value="Suomi"/></p> <p><input type="checkbox"/> Luo lili myöhempää käyttöä varten</p> <p><input checked="" type="checkbox"/> Lähetä tähän osoitteeseen</p>	<p>Täytä tietoihin osoitteesi</p> <p>MAKSUTAPA</p> <p><input type="radio"/> Ennakkomaksu tilisiirtona</p> <p><input type="radio"/> Credit Card / Visa, Mastercard, AMEX, JCB, Diners</p> <p><input checked="" type="radio"/> Checkout Finland</p> <p>     </p> <p>    </p> <p>ALENNUSKOODIT</p> <p>Syötä Alennuskupongin koodi jos sinulla on sellainen.</p> <input type="text"/> <p>Syötä Kuponki</p>	<table border="1"> <thead> <tr> <th>TUOTENIMI</th> <th>LUKUMÄÄRÄ</th> <th>YHTEENSÄ</th> </tr> </thead> <tbody> <tr> <td>24" Widescreen Flat-Panel LCD Monitor</td> <td>1</td> <td>399,99 €</td> </tr> <tr> <td colspan="2">Välisumma</td> <td>399,99 €</td> </tr> <tr> <td colspan="2">Lähetys- ja käsittelykulut (Kiinteät toimituskulut)</td> <td>5,00 €</td> </tr> <tr> <td colspan="2">Yhteensä (veroton)</td> <td>327,57 €</td> </tr> <tr> <td colspan="2">Verot</td> <td>77,42 €</td> </tr> <tr> <td colspan="2">YHTEENSÄ (VEROINEEN)</td> <td>404,99 €</td> </tr> </tbody> </table> <p>HYVÄKSY TILAUS</p>	TUOTENIMI	LUKUMÄÄRÄ	YHTEENSÄ	24" Widescreen Flat-Panel LCD Monitor	1	399,99 €	Välisumma		399,99 €	Lähetys- ja käsittelykulut (Kiinteät toimituskulut)		5,00 €	Yhteensä (veroton)		327,57 €	Verot		77,42 €	YHTEENSÄ (VEROINEEN)		404,99 €
TUOTENIMI	LUKUMÄÄRÄ	YHTEENSÄ																					
24" Widescreen Flat-Panel LCD Monitor	1	399,99 €																					
Välisumma		399,99 €																					
Lähetys- ja käsittelykulut (Kiinteät toimituskulut)		5,00 €																					
Yhteensä (veroton)		327,57 €																					
Verot		77,42 €																					
YHTEENSÄ (VEROINEEN)		404,99 €																					

Picture 27. Checkout page with Checkout Finland enabled.

5.4 Release

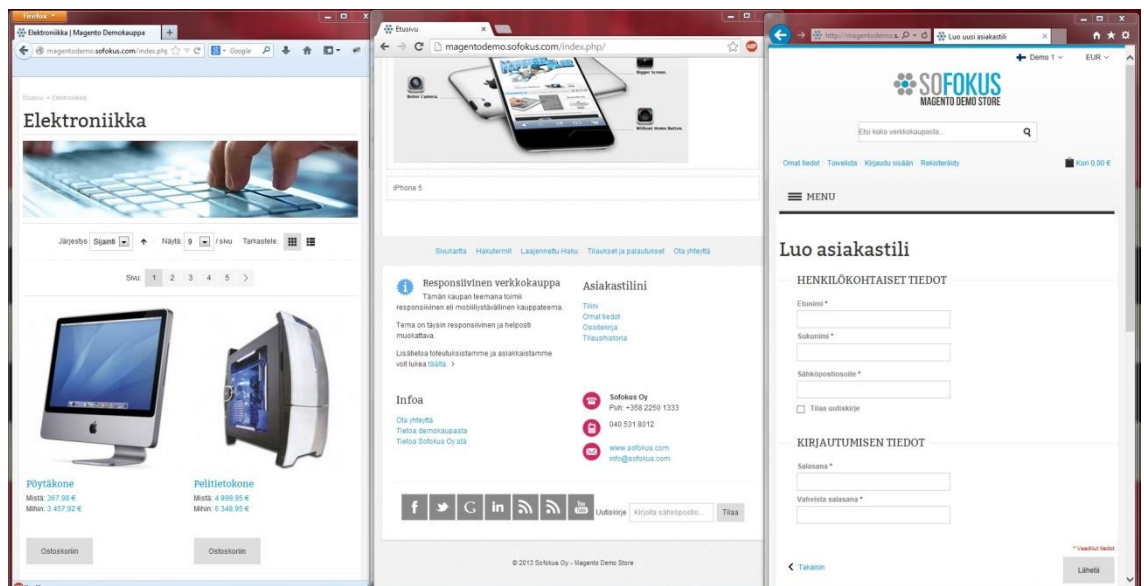
The most important step before releasing the demo store was testing. This testing included general testing and cross-browser testing. General testing included browsing through the site to assure that all the pages function accordingly. Cross browser testing, on the other hand, was CSS-related testing. Different web browsers, such as Mozilla Firefox, Google Chrome and Microsoft Internet Explorer interpret CSS code differently so it was crucial to assure that the pages looked the same regardless of the browser used. This step was time-consuming because small bugs were found throughout the site and needed to be fixed. In addition, some CSS code alterations had to be made.

```

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
httpd.conf styles.css
121 /* Form Elements
122 ~~~~~
123 */
124 input, select, textarea, button { font:12px/15px Arial, Helvetica, sans-serif; vertical-align:middle; }
125 input.input-text, select, textarea { background:#fff; border:1px solid #ccc; }
126 input.input-text, textarea { padding:7px 2px; }
127 select { padding:6px 6px 6px 2px; }
128 select option { padding-right:10px; }
129 select.multiselect option { border-bottom:1px solid #e5e5e5; padding:2px 5px; }
130 select.multiselect option:last-child { border-bottom:0; }
131 textarea { overflow:auto; }
132 input.radio { margin-right:3px; }
133 input.checkbox { margin-right:3px; }
134 input.qty { width:2.5em !important; }
135
136 input.input-text:hover, select:hover, textarea:hover { border-color:#999; }
137 input.input-text:focus, select:focus, textarea:focus {
138   border-color:#999;
139   outline:none;
140   -moz-box-shadow:inset 0 0 4px rgba(0, 0, 0, 0.2);
141   -webkit-box-shadow:inset 0 0 4px rgba(0, 0, 0, 0.2);
142   box-shadow:inset 0 0 4px rgba(0, 0, 0, 0.2);
143 }
144 input.input-text:disabled,
145 select:disabled,
146 textarea:disabled {}
147
148
149 /* Buttons
150 ~~~~~
151 */
152 button.button::-moz-focus-inner { padding:0; border:0; } /* FF Fix */
153 button.button {
154   -webkit-border-fit:lines; /* <- Safari & Google Chrome Fix */
155   overflow:visible; width:auto; border:0; padding:0; margin:0; background:transparent; cursor:pointer;
156 }
157 button.button span {
158   float:left;
159   display:block;
160   padding:0;
161   font-size:12px;
162   text-align:center;
163   white-space:nowrap;

```

Picture 28. Styles.css file located in (Magento root)/frontend/ultimo/default/css.



Picture 29. Cross browser testing: Firefox, Chrome and Internet Explorer. This picture shows that the site looks good regardless of the browser or the page.

Magento also features methods to improve performance such as cache management and file merging. Cache management pre-renders XML and HTML files, to name a few, so whenever a page is requested by a client, i.e., the user Magento delivers the HTML document faster. File merging includes merging CSS and JavaScript files into one single file thus limiting the HTTP requests that are made by the user's browser. These performance improvements might not be huge but it is worth considering that the faster the page loading time, the smoother the experience for the user. Therefore, these features were turned on before the release.

Cache Storage Management Flush Magento Cache Flush Cache Storage

Select All | Unselect All | Select Visible | Unselect Visible | 0 items selected Actions Refresh Submit

Cache Type	Description	Associated Tags	Status
<input type="checkbox"/> Configuration	System(config.xml, local.xml) and modules configuration files(config.xml).	CONFIG	ENABLED
<input type="checkbox"/> Layouts	Layout building instructions.	LAYOUT_GENERAL_CACHE_TAG	ENABLED
<input type="checkbox"/> Blocks HTML output	Page blocks HTML.	BLOCK_HTML	ENABLED
<input type="checkbox"/> Translations	Translation files.	TRANSLATE	ENABLED
<input type="checkbox"/> Collections Data	Collection data files.	COLLECTION_DATA	ENABLED
<input type="checkbox"/> EAV types and attributes	Entity types declaration cache.	EAV	ENABLED
<input type="checkbox"/> Web Services Configuration	Web Services definition files (api.xml).	CONFIG_API	ENABLED
<input type="checkbox"/> Web Services Configuration	Web Services definition files (api2.xml).	CONFIG_API2	ENABLED

Additional Cache Management

Flush Catalog Images Cache Pregenerated product images files.

Flush JavaScript/CSS Cache Themes JavaScript and CSS files combined to one file.

Current Configuration Scope: Default Config Manage Stores

Configuration

- YOUAMA
 - Youama Slider
- GENERAL
 - General
 - Web
 - Design
 - Currency Setup
 - Store Email Addresses
 - Contacts
 - Reports
 - Content Management
- ADDDTHIS
 - Sharing Tool
- INCHOO
 - Featured Products
 - Notifications
 - General
- SOCIAL SHARE
 - Basic Configuration
 - Social Buttons
- ULTIMO THEME (1.3.2)
 - Theme Settings

Developer Save Config

Developer Client Restrictions

- Debug
- Template Settings
- Translate Inline
- Log Settings

Enabled: Yes [STORE VIEW]

System Log File Name: system.log [STORE VIEW]
▲ Logging from Mage::log(). File is located in {{base_dir}}/var/log

Exceptions Log File Name: exception.log [STORE VIEW]
▲ Logging from Mage::logException(). File is located in {{base_dir}}/var/log

JavaScript Settings

Merge JavaScript Files: Yes [STORE VIEW]

CSS Settings

Merge CSS Files: Yes [STORE VIEW]

Pictures 30. Cache management and file merging settings.

Finally, all the files were uploaded to a server and a new database was set up. The store URL was configured to be <http://magentodemo.sofokus.com> and the required DNS configurations were made in order for the URL to correspond with the IP-address. Typically, an online store requires an SSL certificate to encrypt the traffic (HTTPS) over the public Internet but given that this was a demo store, the certificate was not considered mandatory.

Current Configuration Scope:
Default Config ▼
[Manage Stores](#)

Configuration

- ▶ YOUAMA
 - Youama Slider
- ▶ GENERAL
 - General
- Web**
- Design
- Currency Setup
- Store Email Addresses
- Contacts
- Reports
- Content Management
- ▶ ADDTHIS
 - Sharing Tool
- ▶ INCHOO
 - Featured Products
 - Notifications
 - General
- ▶ SOCIAL SHARE
 - Basic Configuration
 - Social Buttons
- ▶ ULTIMO THEME (1.3.2)
 - Theme Settings
 - Theme Design
 - Theme Layout
 - Slideshow

Web Save Config

Url Options ▲

Add Store Code to Urls	<input type="text" value="No"/>	[GLOBAL]
	<small>▲ Warning! When using Store Code in URLs, in some cases system may not work properly if URLs without Store Codes are specified in the third party services (e.g. PayPal etc.).</small>	
Auto-redirect to Base URL	<input type="text" value="Yes (302 Found)"/>	[GLOBAL]
	<small>▲ I.e. redirect from http://example.com/store/ to http://www.example.com/store/</small>	

Search Engines Optimization ▼

Unsecure ▲

Base URL	<input type="text" value="http://magentodemo.sofokus.com/"/>	[STORE VIEW]
Base Link URL	<input type="text" value="{{unsecure_base_url}}"/>	[STORE VIEW]
Base Skin URL	<input type="text" value="{{unsecure_base_url}}skin/"/>	[STORE VIEW]
Base Media URL	<input type="text" value="{{unsecure_base_url}}media/"/>	[STORE VIEW]
Base JavaScript URL	<input type="text" value="{{unsecure_base_url}}js/"/>	[STORE VIEW]
	<small>▲ Warning! When using CDN, in some cases JavaScript may not run properly if CDN is not in your subdomain</small>	

Secure ▲

Base URL	<input type="text" value="https://magentodemo.sofokus.com/"/>	[STORE VIEW]
	<small>▲ Make sure that base URL ends with '/' (slash), e.g. http://yourdomain/magento/</small>	
Base Link URL	<input type="text" value="{{secure_base_url}}"/>	[STORE VIEW]
	<small>▲ Make sure that base URL ends with '/' (slash), e.g. http://yourdomain/magento/</small>	
Base Skin URL	<input type="text" value="{{secure_base_url}}skin/"/>	[STORE VIEW]
Base Media URL	<input type="text" value="{{secure_base_url}}media/"/>	[STORE VIEW]
Base JavaScript URL	<input type="text" value="{{secure_base_url}}js/"/>	[STORE VIEW]
	<small>▲ Warning! When using CDN, in some cases JavaScript may not run properly if CDN is not in your subdomain</small>	

Picture 31. URL settings, secure and unsecure.

TURKU UNIVERSITY OF APPLIED SCIENCES THESIS | Krister Laakso

6 CONCLUSION

Finally, the project has been completed. The idea was not only to study Magento but to use that knowledge to set up a live store. What gave this project an extra challenge was applying a responsive layout to this store. Considering these facts, the goal was achieved with concentration, hard work, and technical assistance from Sofokus personnel. Even though this store was a demo store, it still looked and functioned like a legitimate one. This project was a very good opportunity to extend skills in web designing and web developing.

As Magento is an extremely feature rich framework and has a steep learning curve, the store could be extended in the future by studying features that were not in the scope of this project. Examples could be for example creating new plugins for the store, customizing the back end admin panel or even creating a new theme package.

Though it is not considered an enormous task to install and set up a Magento store, this project had some significance. This was through applying to responsive layout because this is something that has not been done in Finland. As mentioned earlier, many Finnish online stores have to separate sites: a desktop site and a mobile site. The key is this project was to have one single website and to find a balance between a desktop site and a mobile site.

Personally, I think that this project was not only something I liked but also very challenging. The first challenge was the complexity of Magento. After a while things started to make sense and I felt that I could actually accomplish something in Magento. The second challenging aspect was translating the store in Finnish. As Finnish and English are two very different languages, it is not always easy to find perfect translations. This is because if an online store is poorly translated into Finnish, this takes away the credibility of the store for Finnish customers.

REFERENCES

Boag, P. (2011). *Magento Review: Why Magento is worth it... for some*. Available at: <http://boagworld.com/reviews/why-magento-is-worth-it-for-some> [accessed 27.1.2014]

Google Inc. (2013). *Our Mobile Planet: Finland. Understanding the Mobile Consumer*. Available at: <http://services.google.com/fh/files/misc/omp-2013-fi-en.pdf> [accessed 18.2.2014]

Magento Inc. (2012). *Magento Community Edition User Guide*. Available at: www.magentocommerce.com/resources/user-guide-download [accessed 27.1.2014]

Magento Inc. (2012). *Magento Features List*. Available at: <http://www.magentocommerce.com/images/uploads/magento-feature-list.pdf> [accessed 27.1.2014]

Magento Marketing (2013). *Magento Is The Leading eCommerce Platform For Alexa's Top 1M Sites*. Available at: <http://www.magentocommerce.com/blog/comments/magento-is-the-leading-e-commerce-platform-for-alexas-top-1m-sites/> [accessed 27.1.2014]

Sterling, G. (2013). *Report: Nearly 40 Percent Of Internet Time Now On Mobile Devices*. Available at: <http://marketingland.com/report-nearly-40-percent-of-internet-time-now-on-mobile-devices-34639> [accessed 27.1.2014]

van Gemert, V. (2013). *Logical Breakpoints For Your Responsive Design*. Available at: <http://www.smashingmagazine.com/2013/03/01/logical-breakpoints-responsive-design/> [accessed 27.1.2014]

W3C® (2012). *Media Queries*. Available at: <http://www.w3.org/TR/css3-mediaqueries/> [accessed 27.1.2014]

Marcotte, E. (2011). *Responsive Web Design*. New York, New York: A Book Apart.

Williams, B. (2012). *Mastering Magento*. Birmingham, UK: Packt Publishing Ltd.

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

