Bachelor's thesis

Degree programme: Information Technology

Bachelor of Engineering

2014

Larisa Deac

MODERN WEB DESIGN TECHNIQUES

– A Practical approach

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Degree programme: Information Technology

Completion of the thesis: 16.12.2014 | Total number of pages: 65

Instructor: Pasi Iivonen

Larisa Deac

MODERN WEB DESIGN TECHNIQUES

The aim of this thesis is to analyze the latest technologies, design techniques and theories in creating a website. Web design methods have been changing over the past decade with the arrival of web design technologies such as HTML5, CSS3 and JavaScript. This thesis points out the best approaches for creating a modern website and gives examples of projects where the author has applied this approach. The thesis mainly focuses on explaining practices such as responsive web design, CSS, HTML5, JavaScript, SCSS, Sass, jQuery and front-end frameworks.

KEYWORDS:

HTML5, CSS3, JavaScript, SCSS, Sass, jQuery, front-end, responsive, frameworks, modern design, Bootstrap, Sass, LESS, SCSS

CONTENTS

APPENDICES

PICTURES

TABLES

LIST OF ABBREVIATIONS (OR) SYMBOLS

CSS	Cascade Style Sheet
HTML	HTML, which stands for HyperText Markup Language, is the predominant markup language for web pages.
UX, UI	User experience, User interface/interaction
Sass	Syntactically Awesome Stylesheets
SCSS	Sassy CSS
IE	Internet Explorer
Div	division
HTTP	Hyper Text Transfer Protocol
SVG	Scalable Vector Graphics
API	Application Programming Interface
XHTML	Extensible HyperText Markup Language
Doctype	Document Type Declaration
DOM	Document Object Model

1 INTRODUCTION

1.1 Thesis topic and objective

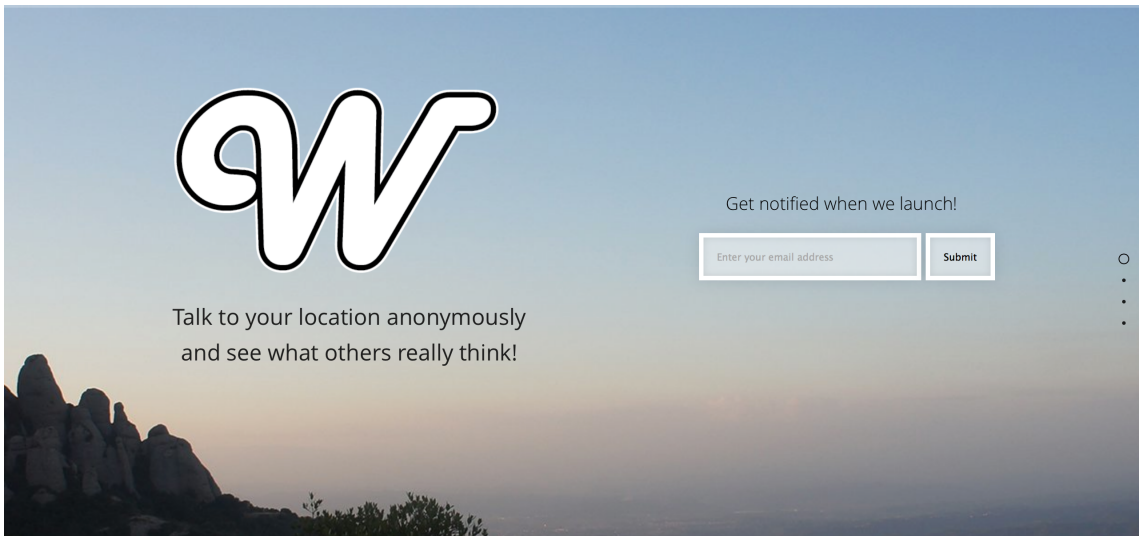
This thesis presents the principles and methods behind using modern web design technologies in the process of building a website. Over the study years the author has participated in projects where she had to apply these principles and be up to date with the latest trends. This experience has helped her assemble effective methods that she is now using in every new web project. Applying these principles has given the author the opportunity to provide a better user experience and interaction.

The purpose of this thesis is to give web designers an insight into the latest web technologies and to help them understand the benefits of responsive, mobile first design in their daily projects.

2 MODERN WEBSITE DESIGN

2.1 Principles of effective web design

Good web design fulfills the needs of the users and allows them to have an experience that will make them come back to the website. Referring to layout design, some effective strategies include: using clean typefaces that are easy to read online (for example, Sans Serif fonts like Arial and Verdana), using combinations of colors that enhance the user experience: vibrant colours for buttons, contrasting colors for the text and background, building a logical page structure, designing clickable buttons and following the “three-click rule”[1], using grid layouts. See Picture 1.



Picture 1. Use of colours and fonts[2]

When it comes to the development side of the website, there are a few key strategies to be taken into consideration. First of all, the load time of a page is very important to make the user stay on the page and read the content. Therefore, to reduce the page load time, a developer should optimize image

[1] Three click rule = users will be able to find the information they are looking for within three clicks
[2] Project Whispr - http://whisprapp.com

sizes (size and scale), concatenate files by appending one to the other, combine code into a central CSS or JavaScript file to reduce HTTP requests and minify HTML, CSS and JavaScript through compression. Then, a developer must remember that nowadays mobile phones and tablets are extremely used, so a website should be accessible from multiple devices with multiple screen sizes. In this sense, a responsive layout is the best approach. Responsive design can be achieved through CSS media queries or by using front-end frameworks that are built with a responsive grid system such as Bootstrap or Foundation. See Picture 2.

Picture 2. Mobile friendly principle

2.2 Best practices for modern web development

Among the most popular web standards are the technologies that work across multiple devices: HTML5, CSS3, SVG, Canvas, and so on. As Peter Gasston mentions in his book, “The Modern Web”, “HTML5 is basically an attempt to evolve the Web to meet the demands of the way we use it today, which has mutated dramatically from its earliest iteration as a simple network of linked documents”(Gasston, 2014, p13).

One of the best practices to do front-end development the right way is to use Modernizr. Modernizr is a JavaScript library that detects HTML5 and CSS3 features in the browser. According to http://modernizr.com/docs, “Modernizr is a small JavaScript library that detects the availability of native implementations for next-generation web technologies, i.e. features that stem from the HTML5 and CSS3 specifications” (Modernizr, 2014).

After including Modernizr JavaScript in the webpage, it checks the functionality of HTML5-CSS3 by default. The Modernizr library includes certain classes like video, no-video, audio, no-audio, etc. For example, if a video tag was used in the webpage and if the browser supports the video feature, then Modernizr will use the video class. If the browser does not support the video feature, then Modernizr will use a no-video class (see Table 1).

Table 1. Modernizr tag detection

```
<!DOCTYPE html>
<html class="no-js" lang="en">
<head>
  <meta charset="utf-8">
  <title>Hello Modernizr</title>
  <script src="modern.js"></script>
  <script src="yepnope.js"></script>
  <style>
      /* In your CSS: */
.no-video #music {
   display: none; /* Don't show Audio options */
}
.video #music button#play button#pause {
   width:100px;
}
  </style>
</head>
<body>

<!-- In your HTML: -->
<div id="video">
   <video>
      <source src="video.mp4" />
      <source src="video1.mp4" />
   </video>
   <button id="play">Play</button>
   <button id="pause">Pause</button>
</div>

</body>
</html>
```

Modernizr can be used in two essential ways: first through CSS and second, through conditional JavaScript (See Picture 3).

Picture 3. Modernizr js code

For example, when checking whether flexbox properties are available in the user's browser a custom version of Modernizr can be built with the build tool, making sure the flexbox option is checked. "The CSS3 Flexible Box, or flexbox, is a layout mode providing for the arrangement of elements on a page such that the elements behave predictably when the page layout must accommodate different screen sizes and different display devices" (Mozilla Developer

Network, 2014). After the flexbox option has been checked, a link must be included to the generated file in the head of the document. When the document has finished loading, a class of either flexbox or no-flexbox is added to the html element. That class could then be used to style the page depending on the level of flexbox support.

Table 2. Flexbox

```
.foo { display: block; }
.flexbox .foo { display: flex; }
```

The second use of Modernizr is for conditional JavaScript. Each test that is run creates a property for the Modernizr object, which has a true or false value for use with conditional functions.

Table 3. Conditional JavaScript

```
Modernizr.load({
  test: Modernizr.flexbox,
  nope: 'foo.js'
});
```

2.3 User interaction and user experience

User experience (UX) focuses on having a profound understanding of the type of users that a website is targeting, their needs, their abilities and their limitations. Peter Morville reveals the values of a good UX through his User Experience Honeycomb[3] (See Picture 4).

Picture 4. Peter Morville's UX Honeycomb (Morville, 2014)

His drawing reveals that in order to have a satisfying user experience, information must be:

- **Useful** – the content should be to the point and should fulfil the user's need
- **Usable** – the website must be practical, easily manageable
- **Desirable** – the brand logo, brand identity, images and other design elements should be used in a way that they raise emotion and appreciation
- **Findable** – the content must be accessible onsite and offsite
- **Accessible** – the website content must be accessible to users with disabilities (minimize the need for scrolling, use strong colour contrast between the text and the background, use alt-tags to label all the images and videos)
- **Credible** – users must trust the content (Morville, 2014)

2.4 Responsive web design approach

Over the past few years, there has been a significant burst in mobile growth (Gasston, 2014, p2). Nowadays, people are using Internet a lot more on their mobiles rather than on PCs. To address the need for people to have a satisfying experience while browsing the Internet on their smartphones or tablets, a designer must take into consideration the practice of responsive web design. Responsive web design refers to the practice of developing a website suitable to work on every device and every screen size, mobile or desktop (Frain 2014, p10).

3 MODERN TECHNOLOGIES

3.1 Introduction to modern web technologies

We are in the presence of a remarkable innovation on the Web. To be able to offer user satisfaction across all devices, newer technologies must be taken into consideration when building a website. These include: HTML5, CSS3, Sass, jQuery. At large, HTML5 “has become a shorthand term for a series of related and complementary technologies, including CSS3, SVG, JavaScript APIs, and more” (Gasston, 2014, p13). HTML5, CSS3, SVG, Canvas, and some device APIs are the most useful for constructing websites that work on multiple devices.

The following sections focus on explaining four of them: HTML5 (http://en.wikipedia.org/wiki/HTML), CSS3 (http://fi.wikipedia.org/wiki/Cascading_Style_Sheets), Sass (http://sass-lang.com) and jQuery (http://en.wikipedia.org/wiki/JQuery).

3.2 HTML5

HTML stands for HyperText Markup Language. HTML is the language that describes the structure and the semantic content of a web document. According to Wikipedia (http://en.wikipedia.org/wiki/HTML), HTML5 is the fifth revision of the HTML standard, thus it has new and improved features. Its purpose is to standardize the common hacks and design patterns used by designers over time, and to broaden as to meet the requirements of the modern Web.

The basic HTML template looks like this:

Table 4. HTML template

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
<title></title>
</head>
<body></body>
</html>
```

In HTML5, it is no longer necessary to mention the Doctype (strict HTML, transitional HTML, XHTML1.1). A few best practices should be considered when implementing HTML5. First of all, the developer is no longer required to use the type attribute when calling external resources. Using HTML4.01 or XHTML, a type had to be declared for each link, script, or style tag:

Table 5. Type declaration

```
<link href="foo.css" rel="stylesheet" type="text/css">
<script src="foo.js" type="text/javascript"></script>
```

When developing on the Web, CSS and JavaScript are actually default resource types used with these tags, so writing them out every time is redundant. In HTML5, the code looks like this:

Table 6. HTML5 code

```
<link href="foo.css" rel="stylesheet">
<script src="foo.js"></script>
```

Another best practice is to use an HTML5 cheat sheet for easy access to all the new various features. A cheat sheet is a list of all the new HTML5 tags and their definitions. A developer has to be aware of all the additions and changes to HTML5. These are happening quite fast. Therefore, one of the best tools to check out the HTML5 tags and properties is the cheat sheet created by InMotion Hosting (See Picture 5).

Picture 5. InMotion Hosting – HTML5 cheat sheet

New Elements in HTML5

In HTML5, there are new elements that are supposed to improve the structure of a page, and provide developers with more options for marking up areas of content.

For example, ... can now be replaced with <article>...</article>.

The W3C's[4] HTML5 specification has a list of ten new structural elements. As noted by David Storey in "The Modern Web" (Storey, 2014), these are:

article An independent part of a document or site, such as a forum post, blog entry, or user-submitted comment

aside An area of a page that is tangentially connected to the content around it, but which could be considered separate, like a sidebar in a magazine article

nav The navigation area of a document, an area that contains links to other documents or other areas of the same document

section A thematic grouping of content, such as a chapter of a book, a page in a tabbed dialog box, or the introduction on a website home page

The other three structural elements define areas within the sectioned content:

footer The footer of a document or of an area of a document, typically containing metadata about the section it is within, such as author details

header Possibly the header of a document, but could also be the header of an area of a document, generally containing heading (h1–h6) elements to mark up titles

hgroup Used to group a set of multiple-level heading elements, such as a subheading or a tagline (Storey, 2014)

HTML5 has a wide set of improved features that make it easier for the developer to build a website. One of the greatest advantages is that, used with jQuery, HTML5 is a very powerful tool to create games and applications. It can be used to create browser games or applications for smartphones. An example of a simple game made with HTML5 is the Snake Game.

Since making a game from scratch takes a very long time, there are HTML5 frameworks and libraries that contain a set of components used in the most

[4] W3C – World Wide Web Consortium

common games, and that can be used to develop a new game: LimeJS, Quintus, CraftyJS, MelonJS, Phaser, Pixie, BabylonsJS, and so on.

Despite the fact that HTML5 is not fully supported on all browsers, it is still a powerful and promising technology for creating browser-based applications. The aim of HTML5 is to create a common platform with more advanced features for the web. It is still an on-going process, but gradually all features and styles will be supported by all browsers.

3.3 CSS3

According to Wikipedia (http://en.wikipedia.org/wiki/Cascading_Style_Sheets), CSS is a style sheet language used for formatting the look of an HTML document. It is used to change the style of web pages, web applications and user interfaces.

Over the past few years, there have been many releases of CSS with improved functionality, and CSS3 is the most used. CSS3 introduces a whole new set of features. One of the most important ones is the use of media queries. Through media queries, a website can be made responsive or in other words, adapted to any kind of device size: desktop, tablet, smartphone, etc. Through media queries, we can now provide styles for each device width and each browser, based on their dimensions and capabilities. The way media queries are used will be covered later in the chapter “Responsive Design”.

CSS3 comes with animations and transitions that can be used to affect text, objects or images. It is no longer needed to use JavaScript when creating certain simple animations, because these can be achieved through CSS. For example, to change the background colour of an element, we can use easing (See Table 7):

Table 7. CSS3 transitions

```
div {
  transition: background-color 0.5s ease;
  background-color: red;
}
div:hover {
  background-color: green;
}
```

The example above changes the background colour of an element on hover.

Example: Transition

Besides animations, CSS3 introduces some new ***selectors***. A CSS selector is the part of a CSS rule set that identifies what part of the web page should be styled (SitePoint, 2014). By using these, we can choose Document Object Model (DOM) elements based on their attributes. “A DOM is a programming interface for HTML, XML and SVG documents. It provides a structured representation of the document (a tree) and it defines a way that the structure can be accessed from programs so that they can change the document structure, style and content” (Mozilla Developer Network, 2014). We no longer have to specify classes and IDs for every element.

The most advantageous attributes for selectors are[5]:

- [attr^=val] Â - matches a DOM element with the attribute attr and a value starting with val
- [attr$=val] Â - matches a DOM element with the attribute attr and a value ending with the suffix val
- [attr*=val] Â - matches a DOM element with the attribute attr and a value containing the substring val

[5] http://www.webreference.com/authoring/css3/index.html

Another feature of CSS3 is the possibility of using ***rounded corners***. Before, a developer had to set the width, height and position of these elements, but now CSS3 introduces the *border-radius* property. With the help of this property, we only have to set one style: for example "*border-radius: 25px;*". It does not require any JavaScript or need to include external images like before.

Other new CSS3 elements include: *border image, box shadow* and *text shadow* (See Table 8)

Table 8. Box shadow

```
.shadow {
  -moz-box-shadow:    3px 3px 5px 6px #ccc;
  -webkit-box-shadow: 3px 3px 5px 6px #ccc;
  box-shadow:         3px 3px 5px 6px #ccc;
}
```

Observing the release of all these new features, we can conclude that CSS is evolving continuously. As Peter Gasston mentions in his book "The book of CSS3", "[...] from its humble beginnings as a way to provide simple decoration to text documents, CSS is moving toward a future where it becomes almost a language in itself, capable of adapting to the many devices that we will use to access the web in the future." (Gasston, 2014 p 255)

3.4 Sass

Sass (Syntactically Awesome Stylesheets) is a stylesheet language that first appeared in 2007. Its latest release was in 2014. It is an extension to CSS that makes developing easier and it is used to enhance CSS. Sass is open-source and coded in Ruby[6].

Sass uses variable and functions, which output to plain CSS and are then read by the browser. The use of variables and functions makes coding faster and easier to maintain.

[6] Ruby is a general purpose programming language

Sass uses two syntaxes: Sass and SCSS (Sassy CSS). The Sass syntax is indented and removes the need for semi-colons and braces. In addition, it does not complain about missing semi-colons. SCSS is the new main syntax and is a superset of CSS3's syntax. SCSS files use the extension "*.scss*". The basic difference between the two is the user interaction: Sass has a loose syntax, while SCSS resembles more CSS (See Table 9).

Table 9. Sass, SCSS & CSS

SASS

```
$color: red
$color2: lime

a
  color: $color
  &:hover
    color: $color2
```

SCSS

```
$color: #f00;
$color2: #0f0;

a {
    color: $color;
    &:hover {
        color: $color2;
    }
}
```

CSS

```
a {
    color: red;
}
    a:hover {
        color: lime;
    }
```

SASS vs. LESS

Sass and Less are both CSS Preprocessors. In other words, they are intended to make CSS more dynamic, better structured and efficient. To compare, we will take a few factors into consideration: installation, extensions, languages, nesting, mixins and selector inheritance, operations, and documentation.

Installation

Sass and LESS are built on different platform: Sass runs on Ruby[7] while LESS is a JavaScript library.

Sass: On Windows, Ruby needs to be installed so that Sass can work while on Mac Ruby is pre-installed. In addition, Sass must be installed through commands on the Terminal or Command Prompt.

[7] Ruby is a dynamic, open source programming language

LESS: Since LESS is developed on JavaScript, it only requires linking JavaScript library to the HTML document

Extensions

Sass: Sass uses the Compass[8] extension, which has a number of Mixins[9] to write CSS3 syntax using less time. Compass has many features like Helpers, Layout, Typography, Grid Layout and Sprite Images.

LESS: LESS also has several extensions but they are not gathered in one place like in Compass. They are built by different developers, so it might be harder for beginner developers to start to use them. LESS extensions include: LESS Elements, LESS Mixins, 960.gs, Semantic.gs, Even.less, Twitter Bootstrap.

Languages

Both Sass and LESS use variables: Sass defines them with a **$** sign while LESS defines them with an **@** sign. There is no real difference between them.

Nesting

Sass allows nesting of individual properties, while LESS does not (See Table 10).

[8] Compass is an open-source, CSS framework - http://compass-style.org
[9] Mixins are directives that group CSS declarations meant to be reused throughout a website - http://sass-lang.com/guide

Table 10. Nesting in Sass

```
nav {
    margin: 50px auto 0;
    width: 788px;
    height: 45px;
    ul {
        padding: 0;
        margin: 0;
    }
border: {
    style: solid;
    left: {
      width: 4px;
      color: #333333;
    }
    rightright: {
      width: 2px;
      color: #000000;
    }
 }
}
```

Mixins and Selector Inheritance

Sass uses the @mixin directive, while LESS defines it with class selectors. Mixins, in LESS and Sass, is used to include properties from one ruleset to another. The difference between the two is that Sass will extend or group

selectors that have the same properties and values using the @extend directive (See Table 11).

Table 11. @extend directive

```
.circle {
    border: 1px solid #ccc;
    border-radius: 50px;
    overflow: hidden;
}
.avatar {
    @extend .circle;
}
```

Operations

Both Sass and LESS can perform math operations. In Sass, mixing units like "%" and "px" in the same operation will result in a syntax error, while LESS can make operations with incompatible units. In this case, Sass is doing it more accurately because different units are not equivalent and should return an error.

Error Notifications

Sass: Sass is slightly vague about where the errors are (for example, it will notify there is an error on line 7, while it is actually on line 6).

LESS: LESS is more accurate because it indicates where the error is exactly, and it will point to the file in question.

Documentation

Sass: The documentation is very understandable and complex. Still, the way it is displayed is not very appealing to the eye (plain text on white background).

LESS: LESS documentation is clearer and gives lots of examples. There is a better color palette and the layout of the documentation is more elegant.

In conclusion, we cannot conclude whether one preprocessor is better than the other. It depends on each developer's needs, knowledge and personal preference.

3.5 jQuery

jQuery has become a standard for working on the web. jQuery is a JavaScript framework that simplifies the way scripts are written. It focuses on abstracting common functions and providing a unified experience across all browsers.

jQuery is a lightweight, cross-platform JavaScript library. It was designed to be reusable and to reduce complex code. jQuery is one of the most widely used JavaScript libraries available, with support from some of the most heavily trafficked websites in the world. According to jquery.com, Google, Amazon, Microsoft, Twitter and many others use it.

Here is a simple example:

Table 12. jQuery

```
$(document).ready(function () {
  $('h1').addClass('foo');
});
```

The first line is required to use jQuery. In the second line, the first part is a CSS selector that selects a node for the action to be applied to, and the second part is the method that states what action will be applied: a class of *foo* will be added to all *h1* elements.

Another advantage in using jQuery is the possibility to chain several methods in one sequence, to create long statements (See Table 13). For example, we can add a click event listener to all elements in the same class (*foo*), then run an anonymous function when that event is fired and assign the event object to a variable (*ev*); in that object, find all elements with a class of *bar* that are children of the element that the event was fired on and change their background color to *#f00*.

Table 13. Chaining

```
$('.foo').on('click', function (ev) {
  $(ev.currentTarget).find('.bar').css('background-color','#f00');
});
```

An extension of jQuery that provides features suitable for mobile devices is jQuery Mobile. Considering that a website must work very well on any platform nowadays, this extension is very useful. According to Peter Gasston in his book, “The Modern Web”, “jQuery Mobile is actually an extension to jQuery that provides cross-platform widgets and styles, as well as new events and methods that take advantage of the new capabilities provided by mobile devices. It requires the jQuery library to run” (Gasston, 2014, p 100).

jQuery provides developers with a great amount of plugins that can be used in different web projects. Here is a list: Alertify.js, File Upload, jQuery Knob, Pickadate.js, Sticky, Super Scroll Obama, etc.

For responsive design, we recommend the following: Response.js, Responsly, Responsive Menu, Blueberry, PhotoSwipe, FlexSlider, Seamless Responsive Photogrid.

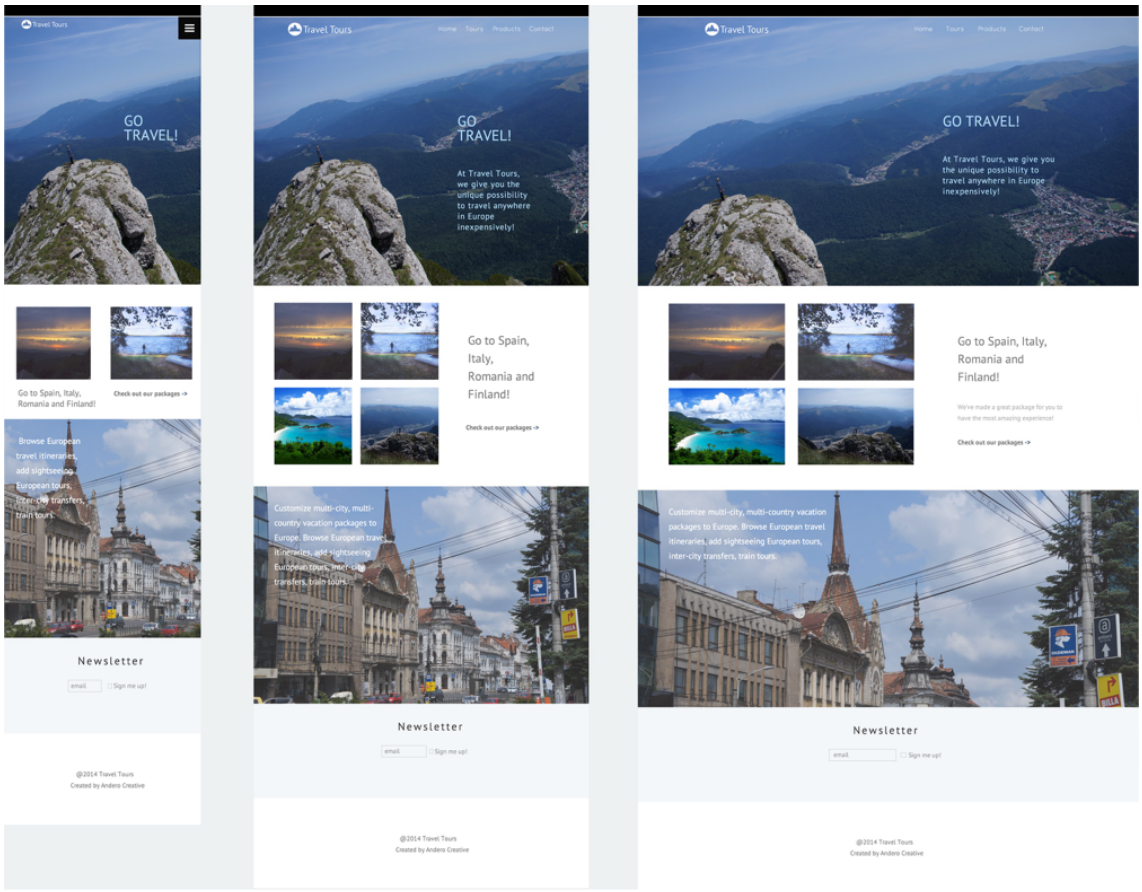
4 RESPONSIVE WEB DESIGN

4.1 Introduction to responsive web design

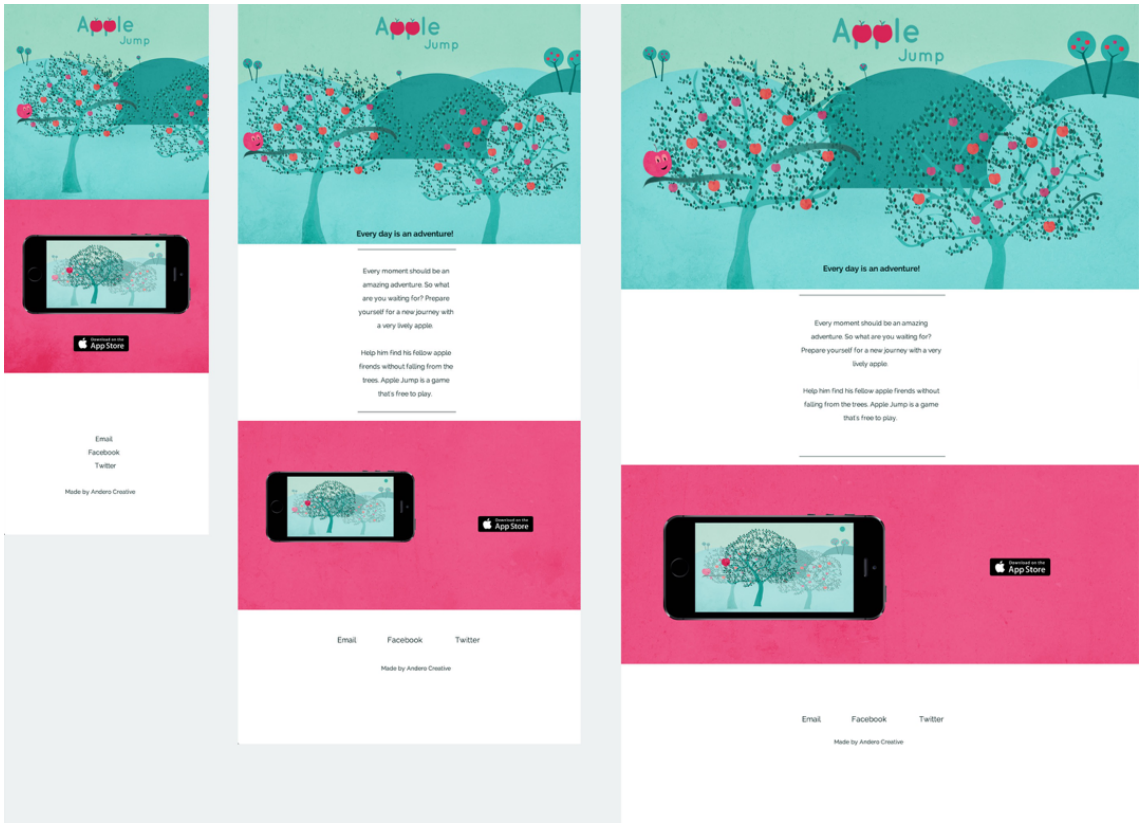
Until 2011-2012, websites could be built at a fixed width, such as 960 pixels. All users were expected to receive more or less a consistent experience. But now, there are tablets, netbooks and smartphones. People are browsing the Web on their mobile devices more often than they used to. It is therefore very important for the end user to have a satisfying experience on the small screen. The solution to the ever-expanding browser and device landscape is a responsive web design, built with HTML5 and CSS3, that allows a website to adjust to multiple devices and screens.

Ethan Marcotte (2014) coined the term “responsive web design”. In his List Apart article, he consolidated three existing techniques (flexible grid layout, flexible images, and media and media queries). This thesis focuses on explaining flexible layouts and media queries.

Examples of responsive websites that the author has designed for mobile, tablet and desktop:



Picture 6. Mockup of Travel Tours website



Picture 7. Mockup of Apple Game website

4.2 Flexible Layouts

Flexible designs adapt beautifully to devices that have portrait and landscape modes (See Picture 8). There are many CSS grid systems that can be used to create a responsive design. Responsive grids are made in CSS, using several methods: negative margins, using *box-sizing*: *border-box*, using *table display*, using *flexbox*.

Negative margins

This method uses negative margins to create grid blocks with a fixed margin in between each block. Negative margins in CSS are values that allow the overlapping of objects in a document (Mozilla Developer Network, 2014). Example: Negative Margins

Using box-sizing:border-box

Using this feature, we can create a flexible grid using fixed “margins”. Example: Box-sizing

Using Table Display

This method uses the table functionality, so the visible elements are block-level by default. At some point in this method, grid rows become tables and columns become table cells. Example: Table Display

Flexbox

According to MDN[10], “The CSS3 Flexible Box, or flexbox, is a layout mode providing for the arrangement of elements on a page such that the elements behave predictably when the page layout must accommodate different screen sizes and different display devices.” Example: Flexbox

Each method has its own benefits and each is easily customizable and adaptable to the developer’s needs.

Picture 8. Flexible layout

[10] MDN = Mozilla Developer Network

4.3 Media Queries

Responsive design can also be achieved through CSS media queries. Media queries are filters that can be applied to CSS styles to act for specific device width, type, height, orientation and resolution.

Table 14. Using @media

```
@media (query) {
  /* CSS Rules used when query matches */
}
```

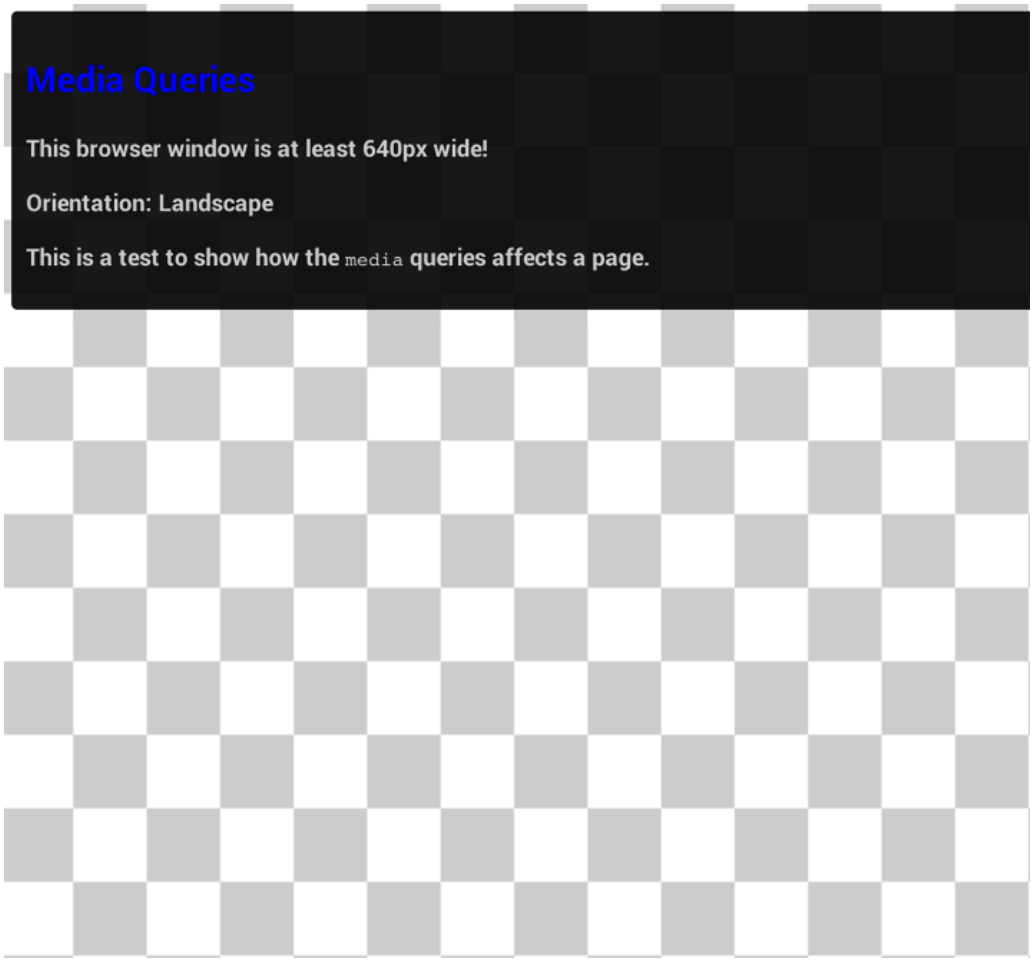
For responsive web design, the most used ones are: *min-width*, *max-width*, *min-height* and *max-height*.

According to Pete LePage (2014) on Google Developers, these are the results achieved by using the four most popular attributes (See Table 15):

Table 15. Media Queries

Attribute	**Result**
min-width	Rules applied for any browser width over the value defined in the query.
max-width	Rules applied for any browser width below the value defined in the query.
min-height	Rules applied for any browser height over the value defined in the query.
max-height	Rules applied for any browser height below the value defined in the query.
orientation=portrait	Rules applied for any browser where the height is greater than or equal to the width.
orientation=landscape	Rules for any browser where the width is greater than the height.

In the following example, the website can be resized to different widths and heights: Media Queries (See Picture 9)



Picture 9. Media Queries

- If the browser width is between 0px and 640px, styles from max-640px.css will be applied.
- If the browser width is between 500px and 600px, styles within the @media will be applied.
- If the browser is 640px or wider, min-640px.css will be applied.
- If the browser width is greater than the height, landscape.css will be applied.

- If the browser height is greater than the width, portrait.css will be applied.[11]

  Media queries are very useful because they allow the developer to take a step further into the process of responsive web design.

[11] *Device width*

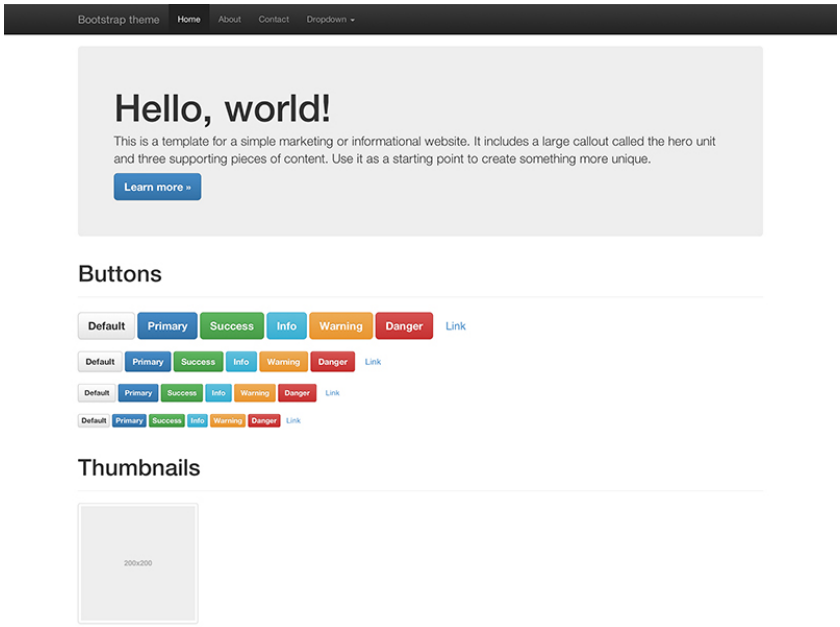
5 FRONT-END FRAMEWORKS

Front-end frameworks are platforms on which ready software solutions like web interfaces are built. They consist of ready components that can be modified or adjusted to current needs (Merix Studio, 2014). In the following section, we will present the Bootstrap framework.

5.1 Bootstrap (Twitter Bootstrap)

In order to create a responsive website, web designers also have the possibility to use front-end frameworks. A CSS framework (or front-end framework) is a library of files. According to the book “The Modern Web”, a framework consists of a set of predefined CSS rules that can be used for fast development; they cover features like typography, forms and layout patterns (Gasston, 2014, 18). The most famous CSS frameworks include: Bootstrap by Twitter, Foundation, Blueprint.css.

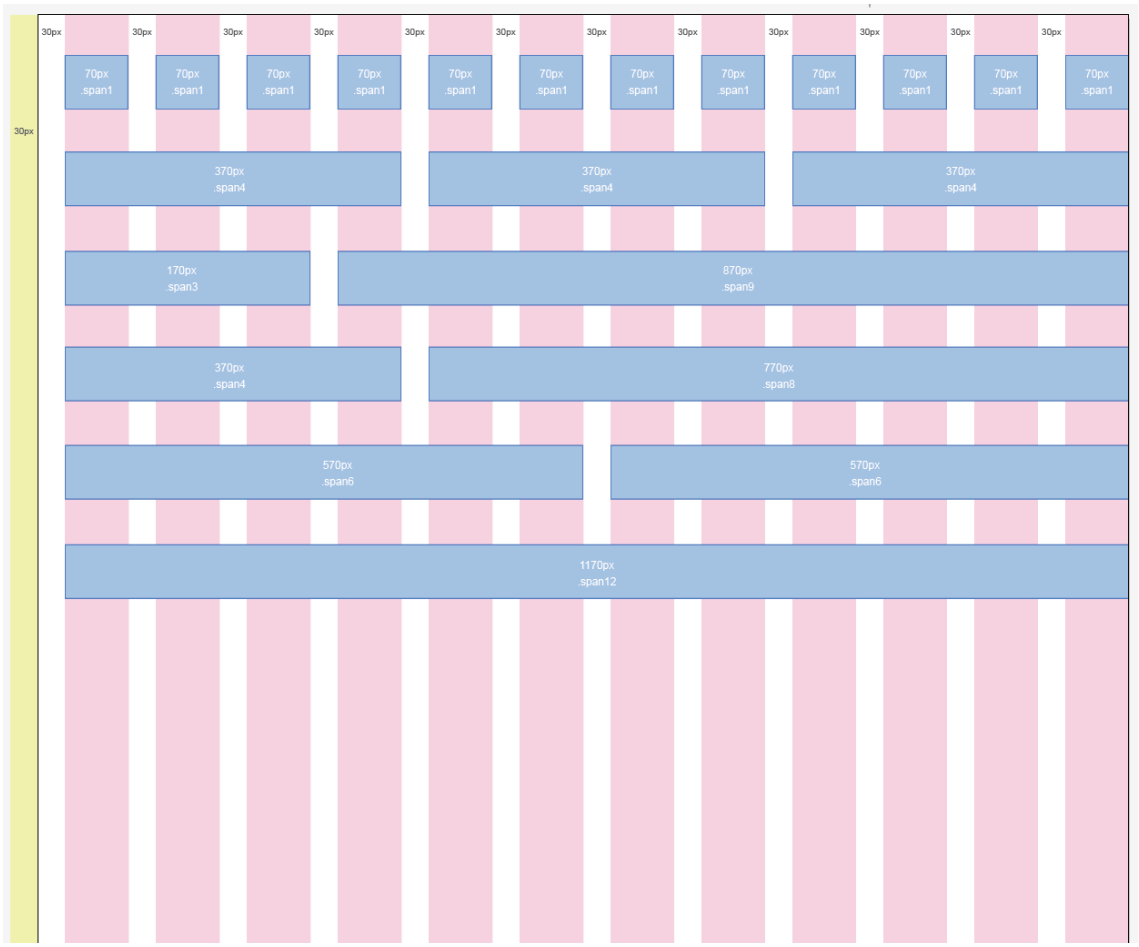
Bootstrap is a responsive front-end framework that has a collection of tools using HTML, CSS and JavaScript. It uses the grid system, which provides a flexible layout, as explained in the section “Flexible Layouts”. It is open source, which means that developers can add their own improvements to it and make their code public to the community. A great advantage in using Bootstrap is that it provides ready-made templates (see Picture 10) that can be further developed into more complex websites. For example, the *Dots* website was created with Bootstrap.



Picture 10. Bootstrap theme

Bootstrap Grid

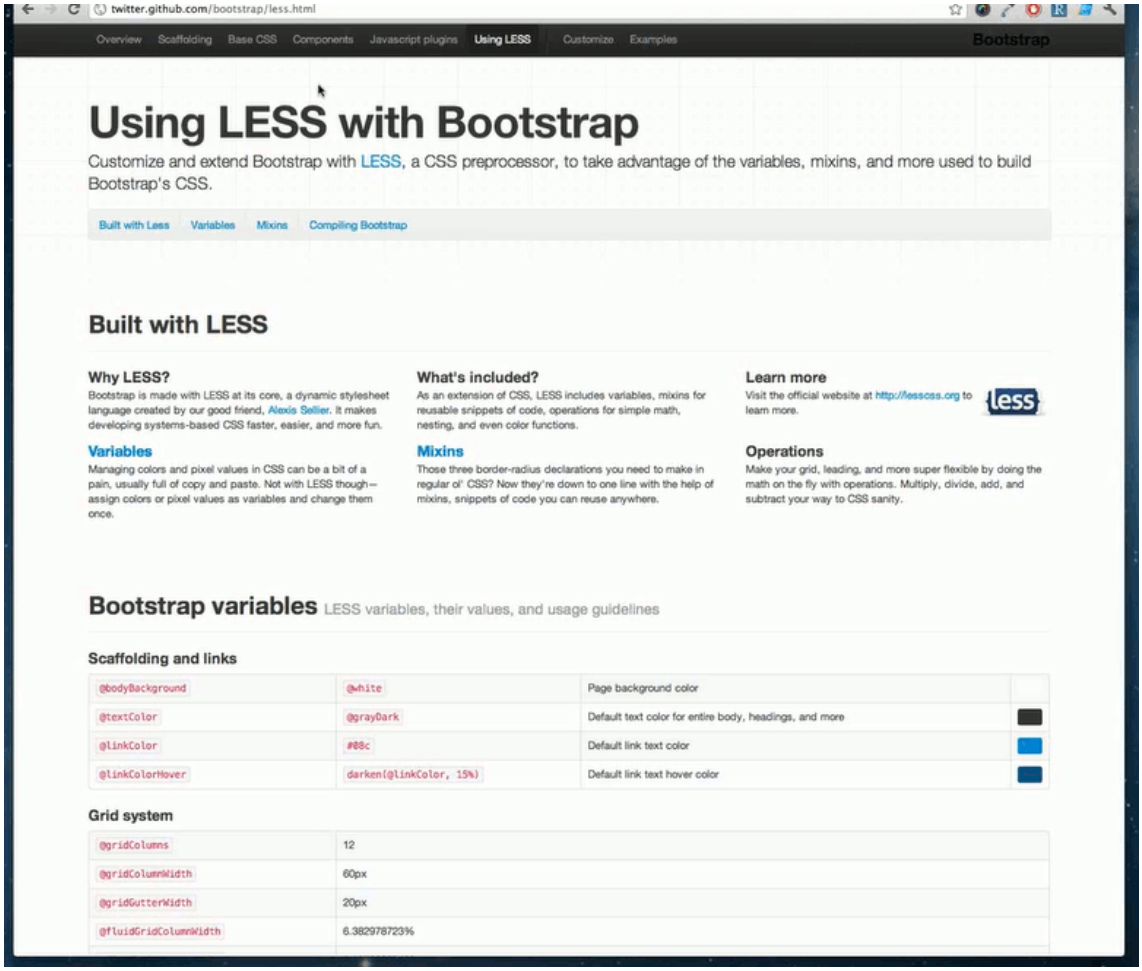
The platform's grid is not mandatory to use, but it makes the process much easier. The platform provides a 16-column grid, which is 940px wide. A single column holds 40px with and additional 20px as the gutter (See Picture 11).



Picture 11. Bootstrap Grid

LESS is more

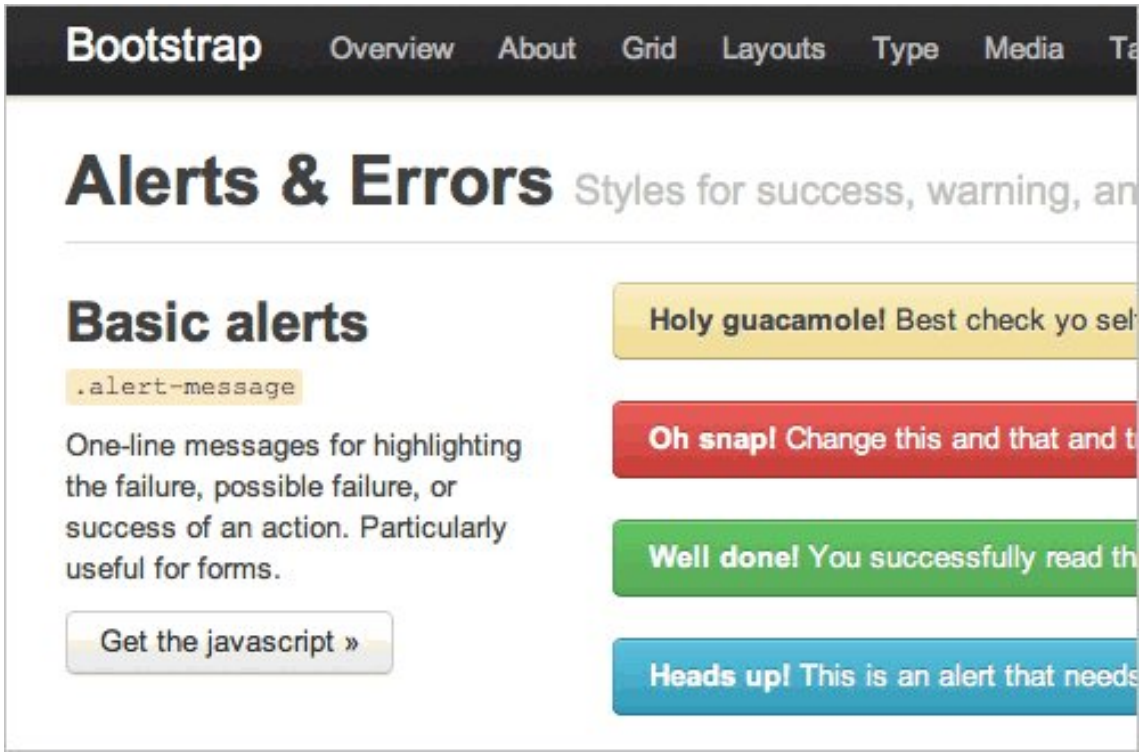
Bootstrap uses LESS mixins and CSS that can be edited to customize the default grid (See Picture 12). Mixins are functions used to embed all the properties of a class into another class by simply including the class name as one of its properties (SitePoint, 2014).



Picture 12. LESS

JavaScript

Bootstrap uses JavaScript libraries that come with complex elements that can be later modified by the developer: windows alerts, tooltips, Button, Typehead, Popover (See Picture 13).



Picture 13. JavaScript in Bootstrap

Cross-browser compatibility

The results from Bootstrap are uniform across all web browsers, which makes it much easier for the developer to focus on other issues rather than fixing pieces of code.

Documentation

The documentation on Bootstrap is very complex and there is a wide community of developers who are using it (See Picture 14).

Picture 14. Bootstrap Documentation by Mark Sandstrom

Bootstrap is a useful tool that can provide great results and productivity. Along with Bootstrap, there are other frameworks that can be used to achieve the same kind of results: HTML5 Boilerplate, Zurb Foundation, jQuery UI, etc.

6 CASE STUDY (ANDERO CREATIVE WEBSITE)

In order to present the modern web design techniques explained in the previous chapters, we chose to analyse the website Andero Creative that was created by the author.

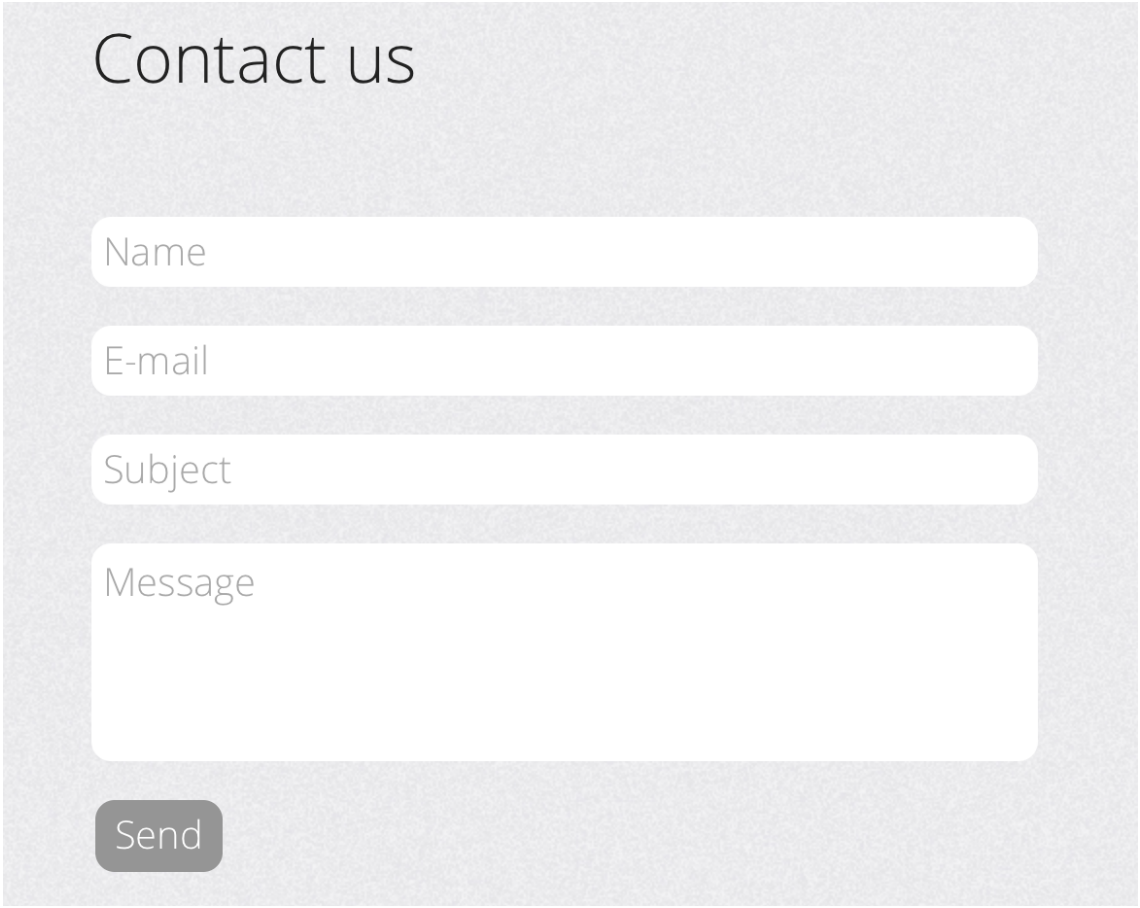
Design

The website has a modern, sleek design with smooth transitions (See Picture 15). The content is spacious and allows the user to find what he/she needs easily. Each section of the website can be accessed by scrolling down or by clicking one of the buttons in the menu.

Picture 15. Andero Creative

HTML5

The website uses HTML5 form tags such as “form”, “required”, “placeholder”, etc. (For contact form, see picture 16)



Picture 16. Contact Form

CSS3

There are several CSS3 elements in the website: “transform”, “opacity”, “hover”, “transition”, “border-radius”, etc. See Appendix 1. See Picture 17.

Picture 17. CSS3 styles

Responsive

The website has a responsive design, achieved through CSS media queries. The media queries are set for 600px, 320px and 480px. See Appendix 2.

jQuery

The website uses jQuery plugins for transitions and flipping images, such as: flipshow.js, smooth-scroll.js. In addition, the Modernizr library has been linked in order to detect HTML and CSS3. See Appendix 3.

7 CONCLUSION

Nowadays the Web is in a continuous change and innovation, a matter that inspires developers to be more aware of the trends and to keep up to date with the latest improvements. Despite the fact that the Web had advanced enormously, there are still challenges that need to be overcome: development for Internet Explorer could be made easier unless the browser stops to exist, modern technologies like HTML5 and CSS3 need to be improved in order to function perfectly across all browsers. Even without using front-end frameworks, browsers need to support modern features extensively.

The purpose of this thesis was to present and analyse the web technologies that are used at the moment for developing modern websites. The thesis compared technologies and demonstrated a few examples that use these modern principles of development. In conclusion, currently, responsive web designs built with HTML5 and CSS3 represent a great development option for most websites.

As mobile device usage continues to extend, the techniques analysed in this thesis provide the most certain and future proof way of building websites that will function on any device and on any viewport.

REFERENCES

Bootstrap. http://getbootstrap.com. Consulted 11.12.2014

Bradley, S. 2014. CSS Animations and Transitions for the Modern Web. US: Adobe Press.

Carver, M. 2014. The Responsive Web. New York: Manning Publications.

Comparison between Front-End Frameworks. http://usablica.github.io/front-end-frameworks/compare.html. Consulted 1.12.2014

CSS3. http://en.wikipedia.org/wiki/Cascading_Style_Sheets. Consulted 12.09.2014

Duckett, J. 2014. JavaScript & jQuery, New Jersey: Wiley.

Felke, Morris T. 2012. Web Development and Design Foundations with HTML5. 6th edition. Boston: Addison-Wesley.

Frain, B. 2012. Responsive Web Design with HTML5 and CSS3. Birmingham: Packt Publishing.

Gasston, P. 2013. The Modern Web. San Francisco: No Starch Press.

HTML5. http://en.wikipedia.org/wiki/HTML5. Consulted 4.09.2014

jQuery. http://jquery.com. Consulted 11.11.2014

Levin, M. 2014. Designing Multi-Device Experiences: An Ecosystem Approach to user Experiences across Devices. Sebastopol: O'Reilly Media.

Marcotte, E. 2014. A List Apart. http://alistapart.com/article/responsive-web-design. Consulted 1.12.2014

Media Queries. https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/use-media-queries. Consulted 16.11.2014

Modernizr documentation. http://modernizr.com/docs. Consulted 11.10.2014

Podjarny, G. 2014. Responsive & Fast: Implementing High-Performance Responsive Design. Sebastopol: O'Reilly Media.

Principles of Website Design. http://goo.gl/3QQvv2. Consulted 1.12.2014

Robinson, A. E. 2014. The Principles of Modern Web Design. Seattle: CreateSpace.

Sass. http://sass-lang.com. Consulted 14.11.2014

Sass vs. LESS. http://css-tricks.com/sass-vs-less/. Consulted 12.11.2014

User Experience Design. http://semanticstudios.com/user_experience_design/. Consulted 3.09.2014

Appendix 1 – CSS3 Elements in the website Andero Creative

```
.ch-item {
	width: 100%;
	height: 100%;
	border-radius: 50%;
	position: relative;
	cursor: default;
	-webkit-perspective: 900px;
	-moz-perspective: 900px;
	-o-perspective: 900px;
	-ms-perspective: 900px;
	perspective: 900px;
}
.ch-info{
	position: absolute;
	width: 100%;
	height: 100%;
	-webkit-transform-style: preserve-3d;
	-moz-transform-style: preserve-3d;
	-o-transform-style: preserve-3d;
```

```
        -ms-transform-style: preserve-3d;
        transform-style: preserve-3d;
}
.ch-info > div {
        display: block;
        position: absolute;
        width: 100%;
        height: 100%;
        border-radius: 50%;
        background-position: center center;
        -webkit-transition: all 0.4s linear;
        -moz-transition: all 0.4s linear;
        -o-transition: all 0.4s linear;
        -ms-transition: all 0.4s linear;
        transition: all 0.4s linear;
        -webkit-transform-origin: 50% 0%;
        -moz-transform-origin: 50% 0%;
        -o-transform-origin: 50% 0%;
        -ms-transform-origin: 50% 0%;
        transform-origin: 50% 0%;
}
```

Appendix 2 – Media Queries in Andero Creative project

```
@media screen and (max-width: 600px) {
        body {
                min-width:510px;
        }

        #right {
                width:100%;
                padding-left:0;
                padding-right:0;
        }

   #contact_info{
      width: 69%;
   }

   #design_services{
      width: 100%;
      padding-left: 0;
      padding-right: 0;
```

```
}
#info {
    width:100%;
    padding-left:0;
    padding-right:0;
}

.menu ul{
    display: none;
}
.menu select {
    display: inline-block; color: #000;
}

      #web_design h2 {
            padding-left:0;
      }

      form {
            width:100%;
            padding-left:0;
            padding-right:0;
```

```
            padding-top:30px;
        }

        #parent1 {
            width:85%;
            padding-left:25px;
            padding-right:25px;
        }

    #parent2 {
        width:85%;
        padding-left:25px;
        padding-right:25px;
    }

    #parent3 {
        width:85%;
        padding-left:25px;
        padding-right:25px;
    }

    #parent4 {
```

```
    width:85%;
    padding-left:25px;
    padding-right:25px;
}

#parent5 {
    width:85%;
    padding-left:25px;
    padding-right:25px;
}

#parent6 {
    width:85%;
    padding-left:25px;
    padding-right:25px;
}

#contact_form_message_sent{
    padding-top: 270px;
}

      #logo {
```

```
        height:300px;
        width:500px;
        position:absolute;
        top:50%;
        left:50%;
        margin-left:-250px;
        margin-top:-150px;
        overflow:hidden;
}

#logo img {
        width:264px;
        margin-left:100px;
        padding-bottom:50px;
}

#software {
}

header a {
        font-size:40px;
}
```

```
        .link {
                width:50px;
                height:50px;
        }

#background{
    background: none;
}

header{
    background: #f9f9f9 url("../images/fields_2k.jpg");
    width: 100%;
    height: 100%;
    position: relative;
    min-height: 300px;
    overflow: hidden;
}

#photography{
    background: #f9f9f9 url("../images/fields_2k.jpg");
}
```

```
    .menu{
        display: none;
    }
}

/* Smartphones (portrait and landscape) ---------- */
@media only screen
and (min-device-width : 320px)
and (max-device-width : 480px) {
    body {
        min-width:320px;
    }

    #logo {
        height:300px;
        width:320px;
        position:absolute;
        top:50%;
        left:50%;
        margin-left:-160px;
        margin-top:-70px;
```

```
    overflow:hidden;
}

#logo img {
    width:164px;
    margin-left:78px;
    padding-bottom:0;
}

/* Web design portfolio thumbnails */
.innerContent a img {
    max-width: 95%;
    max-height: 9999px;
    padding-right: 0;
}

#parent2 h2{
    line-height: 35px;
    padding-bottom: 0;
}

#parent5 h2{
```

```
    line-height: 35px;
    padding-bottom: 0;
}

#people img{
    width: 260px;
}

#contact_form{
    width: 100%;
}

/*#contact_info{
    padding-right: 80px;
}*/

#media_buttons{
    display: inline;
    padding-right: 100px;
}

.fc-slideshow img{
```

```
    width: 260px;
}
#photo_detail{
    font-size: 25px;
}

#photo_detail a{
    font-size: 27px;
}

.lari{
    margin-top: 2px;
}

#right {
    width:100%;
    padding-left:0;
    padding-right:0;
}

#contact_info{
    width: 100%;
```

```
    padding-right: 0;
}

.lari{
    margin-top: 3px;
}

#background {
    background: none;

}

header{
    background: #f9f9f9 url("../images/fields_2k.jpg");
    width: 100%;
    height: 100%;
    position: relative;
    min-height: 300px;
    overflow: hidden;
}

#photography{
```

```
    background: #f9f9f9 url("../images/fields_2k.jpg");
}

.menu{
    display: none;
}

#design_services{
    width: 100%;
    padding-left: 0;
    padding-right: 0;
}
#info {
    width:100%;
    padding-left:0;
    padding-right:0;
}

.menu ul{
    display: none;
}
.menu select {
```

```
    display: inline-block; color: #000;
}

#web_design h2 {
    padding-left:0;
}

form {
    width:100%;
    padding-left:0;
    padding-right:0;
    padding-top:30px;
}

#parent1 {
    width:85%;
    padding-left:25px;
    padding-right:25px;
}

#parent2 {
    width:85%;
```

```
    padding-left:25px;
    padding-right:25px;
}

#parent3 {
    width:85%;
    padding-left:25px;
    padding-right:25px;
}

#parent4 {
    width:85%;
    padding-left:25px;
    padding-right:25px;
}

#parent5 {
    width:85%;
    padding-left:25px;
    padding-right:25px;
}
```

```
    #parent6 {
        width:85%;
        padding-left:25px;
        padding-right:25px;
    }

    #contact_form_message_sent{
        padding-top: 270px;
    }

    #software {
    }

    header a {
        font-size:40px;
    }

    .link {
        width:50px;
        height:50px;
    }
}
```

Appendix 3 - Including Modernizr library for the project Andero Creative

```
<script type="text/javascript" src="js/modernizr.custom.79639.js"></script>
```