Final Project Report

Michael Taylor, and Eamonn De'Leaster*

Waterford Institute of Technology,
Dept of Computing and Mathematics,
Cork Rd, Waterford City, Ireland
20064376@mail.wit.ie
http://www.wit.ie

Abstract. A complete UI framework designed for entry level users with a focus on easy to understand terminology, introducing templating, and partials with clear, concise, customisable themes.

Keywords: CSS, User Interface, framework, beginners, templates, partials, themes

* Supervisor

Table of Contents

Acknowledgements

I would like to thank ....

Introduction

User Interface (UI) and Cascading Stylesheets (CSS) frameworks are abundant and readily available and generally covered by permissible licence agreements [Arsenault, 2016]. This not only allows commercial use of the frameworks but ultimately encourages tie in to a particular framework. End users expect all websites to work across their devices, laptops, tablets and phones. This requirement of responsive web applications ensures that hand coding a solution is a daunting task for entry level developers.

Using a framework allows the user to speed up the initial mock-up process, they offer clarification on common CSS issues, have wide browser support. Frameworks are good for responsive design, offer clean, and tidy code. There are disadvantages to using frameworks such as an abundance of unused code left over, this is seen more so in large front end frameworks such as Twitter Bootstrap. There is a slower learning curve with using frameworks, and the user does not learn to do it themselves through problem solving.

Frameworks such as Twitter Bootstrap [et al, 2016b], and Zurb Foundation [et al, 2016a] offer complete solutions, with ready built code for forms, buttons, fluid layouts to popovers. Skeleton [Gamache, 2016] is an example of the other end of the spectrum. Skeleton is a boilerplate for responsive, and mobile first development. It is designed to be light, and is built with less than 400 lines of code. Unlike Bootstrap or Foundation, skeleton is designed to be the users starting point, not their full solution.

This project has being built to offer the end user a complete solution designed with an entry level user in mind. The framework is non opinionated unlike frameworks such as Bootstrap who are very opinionated about their design Guadia [2016]. This framework is be designed to be the starting point for a user, for them to add in their custom styling.

Instead of the user building large complex websites with repeating code, one of the project goals is to get the user familiar with concepts such templates, and partials. Partials break up the html code into smaller more manageable fragments that can be used across multiple html files. The framework has been built with this in mind, creating classes, and id's that can be reused instead of having a bloated package with a lot of unused code [K, 2016].

A major benefit of a project, is the ability to customise or use prebuilt themes also known an boilerplate code. Utilising boilerplate can allow the user to concentrate on the aesthetics more so than the structure of the project. Using boilerplate's does have its negatives, the boilerplate itself can be expensive costing hundreds if not thousands. Many shortcuts could be used in the development of boilerplate, not adhering to W3S standards. These boilerplate's can also offer little customisation [Weller, 2016].

Further attempts to incorporate Bootstrap into projects demonstrated the syntax as very unfriendly, and noticeably more difficult than hand coded CSS. Bootstrap syntax such as:

```
1 |   <div class="col-sm-4">
```

does not describe in any form that it would be displayed as a three column layout in the browser. Changing this to a one column layout by the syntax was found to be problematic and the frameworks were also found to be opinionated.

Additionally while working as a web developer mentor at Coderdojo, students were observed to have similar experiences. Many were reluctant to learn frameworks such as Bootstrap as they were too confusing.

Part I

User Manual

Introduction

The project is to build a full UI framework that is aimed towards an entry level user. The framework is be non-opinionated, light weight, easy to understand, theme-able, and most importantly easy to learn.

The framework has minimal styling, the author has only added styling where it is needed, and that styling is kept to basic colours. This was to ensure the framework does not carry the author's design opinions. With been non-opinionated the framework is bare so that the end user is encouraged to not rely upon the built in styling, but instead to use it as a starting point to build upon it. The framework is based on a responsive grid system of 12 grids, allowing the end user to build a seamless experience from desktops to mobile devices.

The hope is that the framework will be as small of possible as of the source code deadline, the framework weighs —— this allows projects using it load faster on mobile devices, and in countries where fast internet is either not available or expensive.

To stop code from been repeated unnecessary, code sections have been broken into partials allow for the ability for the content to be broken up into manageable pieces, removing receptive code such as headers, and footers. PHP is then used to combine the files into one when loaded in the browser.

The framework will contain similar features from both Bootstrap, and Foundation as seen in table **??** such as tables, lists, breadcrumbs, and pagination. Features such a tooltips, and right to left are outside the scope of the project. The hope for the project is to be complete framework but having a smaller file size than both foundation, and Bootstrap borrowing concepts from Skeleton in keeping the framework light, and nimble.

The proposed framework will be built using the three key points mentioned below:

- For Entry Level Users
- Themeable
- Introduce the concepts of Templates, and Partials

Partials

talk about partials here web components

Themes

Talk about themes here

Boilerplate

talk about portfolio, blog and commerce website here

Command Line Interface

talk about creating command line interface here

Part II

Development

Framework

The following section will detail the components that make up the Taylor'd UI framework. Each of the following components are what makes up the framework, and aid in the development of a website. All the components of the framework have been built in separate partial files, denoted by an underscore. The underscore also tells codekit, the compiler been used to not compile the files into their own css file. A separate file called taylord.scss was created, in this file the @include statement was used to pull all the separate partial files into one file and that file is then compiled to CSS.

0.1 Variables

At the start of framework development, a file called _foundation.scss was created, in this file all the variables for the framework are defined. In this file, the foundation colour variable is set and using the HSL function called lighten, a range of colours are created to be used in the framework. Other colour variables were added and then used to set the background colour, alert colours, and link colours.

```
1  $foundation-color: #000;
2  $foundation-color-2121: lighten($foundation-color,
3  13%);
4
5  h1, h2, h3, h4, h5, h6 {
6    color: $foundation-color-2121;
```

0.2 Font and Typography

In keeping with the framework being non opinionated, it was crucial to pick a font, and to have the typography nondescript so that the end user can change it to suit their needs. A range of fonts were looked at, the readability of the font in regards to large sections to single lines of text was looked at. Next, viewing the font on a range of devices ranging from mobile to desktop was looked at ensuing it was legible across these devices. The last aspect that was looked at was the range of the font, did it allow for non latin characters, accents and symbols.

The font that passed these tests, and was a joy to look at was Open Sans. The next step was to determine the font sizing and how to calculate it. When

decided on the best way to execute the font sizing, two methods were looked at, em and rem. Pixels was not looked at as early versions of Internet Explorer are not able to change the font size using browser functionality, a major usability issue. The em technique alters the base font size on the body element by using a percentage.

This adapts the font so that 1em is equal to 10 pixels, instead of the default 16 pixels. To change the font size to the equivalent of 14 pixels, the em needs to be 1.4em. The downside of using em to calculate the font sizing is that the font size compounds. This means that a list within a list isn't 14 pixels but rather 20 pixels. There is a work around where any child elements are declared to use 1em, but an entry level user would not know this.

With the advent of CSS3, rem which means root element was added, as previously mentioned, em sizing was relative to the font size of the parent whereas rem is relative to the root or html element. This means that a single font size can be defined for the html element, and all rem units will be a percentage of that base unit. Safari 5, Chrome, Firefox 3.6+, and even Internet Explorer 9 have support for rem units. Opera up to version 11.10, and early versions of Internet Explorer have yet to implement rem units. In order to display font on these browsers, a fallback pixel size is calculated using a mixin.

http://www.gunlaug.no/contents/wd$_a$*dditions*$_1$3.*html*

https://en.blog.wordpress.com/2012/10/09/open-sans-how-do-we-love-thee-let-us-count-the-ways/

0.3 Colour Scheme

When creating the colour scheme, it was important to keep design opinions to a minimum, to use colours that would get the frameworks point across. It was also key to use colours that the end user would recognise that they were for instructional purposes, and for them to change in their web development projects. The colour needed to look good when modified as well, this means if the hue, tone and vibrancy of the colour is modified, the resulting colour needed to look well.

Instead of researching colour theory and choosing the best colours, it was decided to use Google's Material Design colour palette, and choose from their wide range of colours. The palette gave a variety of colours along with modifications of the colour such as different hues and saturations. The colours selected to use

in the framework were kept to primary colours to keep it as non opinionated as possible.

https://material.io/guidelines/style/color.htmlcolor-color-tool

0.4 Tables

Tables in HTML should only be used for rendering data that naturally belongs in a grid based system. This is data where the data characterised is similar across a number of objects. Tables should not be used for the layout of content in a website, divs should be used for this. The key to designing the tables was to demonstrate to the end user that the tables were for data, and not for layout.

Five table variations were created for the framework, ranging from default table to striped tables, and included is table modifiers. Table modifiers take the colours that are used to dictate success or warning, and add them to a row in the table. The tables were also designed to be responsive, to achieve this the padding of table is calculated by dividing the $baseline height by a set value.

```
1    .table.table-condensed > thead > tr > th,
2    .table.table-condensed > tbody > tr > td {
3      padding-top: $baseline-size / 2.4; // 5px
4      padding-bottom: $baseline-size / 2.4;  // 5px
5      padding-left: $baseline-size / 1.5;  // 8px
6      padding-right: $baseline-size / 1.5; // 8px
7    }
```

0.5 Buttons and Button Groups

Buttons are an integral part of a framework. The styling, and the functionality of the buttons are key in the end users goal of using a website. If a button does not look like a button or if the styling of a button is overdone, the user can get confused, and not know how to proceed. With this in mind, the development of the buttons continued throughout the development of the framework. Originally, the buttons had round corners but this was removed in trying to keep the design non opinionated.

Project Plan

add stuff about what was built each week - similar to the engineering thing

Future Work

The original plan of the framework was to create a documentation website showcasing the framework, and detailing everything about the framework with code snippets, and examples of usage. When the framework was been developed, a kitchen sink website was created first to ensure that the code was working as it should. Due to time constraints, the finished website hasn't been developed. For the continuation of the project, this website will be built using the framework, and have documentation on each of the core functionality of the framework.

In continuing on developing the framework, the following features would be added to the framework to make it more robust, and eliminate the need to write extra CSS; simple built in navigation so that the user does not have to create their own. Badges to show unread content or to be used to display notifications. Breadcrumbs to show the user where they are are on a website. Tabs for tabbed navigation. Off canvas sidebar that slides in and out of a page, ideal for mobile design.

In addition to CSS, jQuery features to be added that would help the framework work more seamlessly such as making links active in the navigation bar, ensuring that the footer section always stays at the bottom of the page, regardless of page height.

Conclusions

Bibliography

Cody Arsenault. *Top 10 Front-End Frameworks of 2016*, 2016. Available at: https://www.keycdn.com/blog/front-end-frameworks/ [Accessed: 03/12/2016].

Alexi Sellier et al. *Getting Started*, 2016a. Available at: http://lesscss.org [Accessed: 26/11/2016].

Hampton Catlin et al. *CSS with superpowers*, 2016b. Available at: http://SASS-lang.com [Accessed: 26/11/2016].

Dave Gamache. *A dead simple, responsive boilerplate.* , 2016. Available at: http://getskeleton.com [Accessed: 26/11/2016].

Kemie Guadia. *CSS Frameworks- comparing Bootstrap alternatives*, 2016. Available at: http://www.monolinea.com/css-frameworks-comparison/ [Accessed: 08/12/2016].

Karol K. *The Bootstrap Framework Controversy ... Should You Use It or Not?*, 2016. Available at: http://www.htmlcenter.com/blog/the-bootstrap-framework-controversy-should-you-use-it-or-not/ [Accessed: 08/12/2016].

Nathan B. Weller. *Custom vs Pre-made WordPress Themes A Look at the Pros and Cons*, 2016. Available at: http://wplift.com/custom-vs-pre-made-themes [Accessed: 08/12/2016].