POTTERY: A CSS FRAMEWORK FOR CLAY

B. TECH PROJECT

CS498 PROJECT REPORT

SUBMITTED BY:

PRANJAL GARG(10509)

VIBHAV AGARWAL(10795)

PROJECT GUIDE:

Prof. PIYUSH P. KURUR

ACKNOWLEDGEMENT

We would like to thank our project guide, Dr Kurur for his help throughout the project. It would not have been possible to complete this without his able guidance.

BACKGROUND

The World Wide Web is a system of interlinked hypertext documents accessed via the Internet. Using a software called web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them via hyperlinks. Since its invention in 1989 at CERN, the web has seen enormous transformations over the years- not least in how web pages appear.

The primary language used for creating web pages is the Hypertext Markup Language (abbreviated as HTML). It is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>), within the web page content. It allows images, objects, scripts etc. to be embedded and enables the development of structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items.

However, with the evolution of the web, content on web pages has increased dramatically. Therefore, keeping content and the design layout of the page on the same HTML document leads to code that is extremely cluttered and tedious. This lead to the development of a style sheet language called Cascading Style Sheets (CSS) which is used for describing the presentation semantics (the look and formatting) of a document written in a markup language like HTML. It can include the layout, fonts and much more.

Now, while CSS has proved to be extremely helpful in the development of richer web pages, it is not without its drawbacks. One of its major drawbacks is that any bugs in the code only show up at run-time in the browser. There is no way to check the correctness at compile time. To eliminate this, there has been increasing development of pre-processors for CSS. The foremost of these are LESS and SASS. These pre-processors not only allow more efficient debugging, the style-sheets that they generate are also much cleaner. Additionally since they allow incorporation of mixins and variables, the process of writing code also becomes less tedious.

Like SASS and LESS Clay is a CSS pre-processor. What distinguishes it from the other two languages though is the fact that it has been implemented as an Embedded Domain Specific Language (EDSL) in Haskell. Because of this, it is much more powerful than the other two and also much more re-usable. Clay, however, is in its infancy. It lacks an extensive framework like Compass (for SASS) which would make streamline the process further still.

The primary aim of the project is to add an extensive, functioning library to Clay on the lines of Compass.

MOTIVATION FOR THE PROJECT

In today's world, the importance of design cannot be overstated. The user interface often decides the fate of products/ websites. In such circumstances, the fact that CSS is a dumb language (owing to the reasons mentioned previously) can prove to be a deterrent for many people. Thus it is necessary to have an efficient pre-processor and a large, functional library for it. It is this gap that Pottery aims to fill.

Although some might argue that with libraries like Compass, Foundation, Bourbon etc. already present for pre-processors like SASS and LESS Pottery is redundant, the fact that Clay is an EDSL of Haskell endows it with several features that SASS and LESS lack. Firstly, being based on the Functional Programming paradigm, Haskell enhances productivity over Imperative languages. The biggest advantage, though, is that since Haskell is already supported by a large user community, it has a huge library of functions that can be accessed when using Clay. Thus, a well-developed model for Pottery would, in fact, render SASS and LESS redundant.

METHODOLOGY

We started the project by playing around with the Compass libraries in SASS to get first-hand knowledge of some of the nuances in web design that we aimed to implement and also to gain an insight into how the library needed to be designed. This proved a great learning step for the rest of the project.

After gaining a good understanding of Compass and SASS, we then moved on to HASKELL in order to understand the fundamentals of the language better. In particular, we went through the concepts of Monad theory which later proved fundamental during the project.

Following this, we went through the documentation for Clay and after gaining proficiency in Clay, we set about implementing all that we had learnt.

We started by implementing the module on Vertical Rhythm. This (as has been explained later) is one of the fundamental concepts in web design. After initially managing to implement the functionality in an imperative manner, we moved to a purely functional design using state monads.

After successfully completing the first module, we moved on to implement more modules including- golden ratio etc. (A detailed list has been supplied in the next section.)

TYPOGRAPHICAL DESIGN CONCEPTS IMPLEMENTED

VERTICAL RHYTHM

CONCEPT

In design, vertical rhythm is the structure that guides a reader's eye through the content. Good vertical rhythm makes a layout more balanced and beautiful and its content more readable. There are 3 primary factors governing rhythm in a document- the font size, line height and margin or padding. All of these factors must calculated with care in order that the rhythm is maintained. The basic unit of vertical space is line height. Establishing a suitable line height that can be applied to all text on the page is the key to a solid dependable vertical rhythm, which will engage and guide the reader down the page.

OUR IMPLEMENTATION

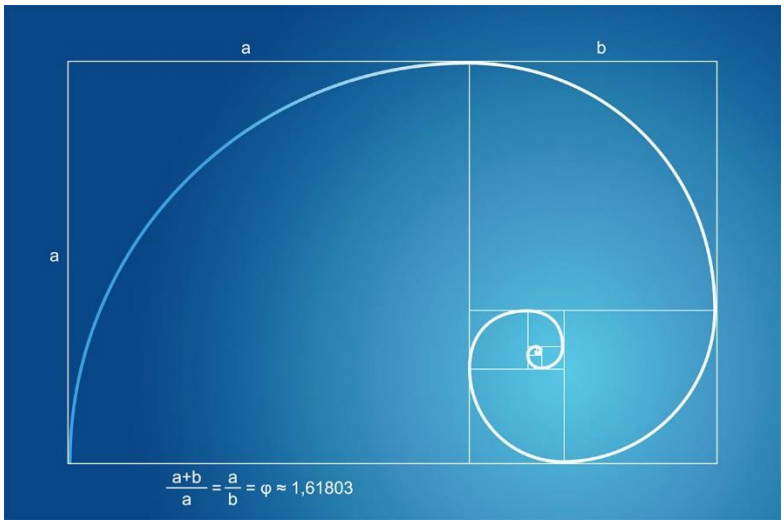
We have tried to mirror Compass's design for the module. Salient features:

- Configurable Variables: Base Font Size, Base Line Height, Rhythm Border Style, Relative Font Sizing, Round To Nearest Half Line, Font Unit.
- Mixins Included:
    - establishBaseline : Establishes the baseline for the given CSSState
    - Baseline : Returns baseline CSS
    - Rhythm : Calculate rhythm units
    - toFontSize
    - fontSize : returns Css to adjust Fontsize of a element
    - linesToFontSize : Calculate the minimum multiple of rhythm units needed to contain the font-size.
    - Leader : Apply leading whitespace. The property can be margin or padding. By default property is margin.
    - paddingLeader : Apply leading whitespace as padding.
    - paddingTrailer : Apply trailing whitespace as padding.
    - marginLeader : Apply leading whitespace as margin.

- Trailer : Apply trailing whitespace. The property can be margin or padding. By default property is margin.
- marginTrailer : Apply trailing whitespace as margin.
- propertyRhythm : Shorthand function to apply whitespace for top and bottom margins and padding. Takes the number of lines for each property with default value 0.
- applySideRhythmBorder : Apply a border & whitespace to any side without destroying the vertical rhythm. The whitespace must be greater than the width of the border.
- rhythmBorders : Apply borders and whitespace equally to all sides.
- leadingBorder : Apply a leading border.
- trailingBorder : Apply a trailing border.
- horizontalBorder : Apply both leading border and trailing border
- hBorder : Alias for horizontal-border.

GOLDEN RATIO

CONCEPT



When designing the visual aspect to a website or creating images in general the dilemma of choosing proportions inevitably comes up. The Golden Ratio or 'ϕ' which equals 1.618 in decimal terms helps to solve this. The Golden Ratio and the Golden Rectangle is used in many forms of art and design. In the Renaissance period, many artists proportioned their artworks according to this ratio and rectangle. In ancient Greece, architects used this rectangle in the design of the buildings; the Parthenon is a good example of this. Even in modern architecture, the golden rectangle has a strong presence. Even in modern web design, the Golden ratio has proven its worth for aesthetics.

OUR IMPLEMENTATION

This module has not been implemented in Compass. Hence, we have implemented it ourselves. The mixins included are:

- grTitleSize : Returns CSS of title size based on content width.
- grHeadlineSize : Returns CSS of headline size based on content width.
- grSubHeadlineSize : Returns CSS of subheadline size based on content width.
- grFontSize : Returns CSS of Font size based on content width.
- grSecondaryText : Returns CSS of Secondary Text based on content width.
- grBaseLineheight : Returns CSS of default line-height based on content width.
- grCustomLineheight : Returns CSS of Golden-ratio line-height based on a customvalue specified by the user.

REFERENCES

1. http://fvisser.nl/clay/
2. http://compass-style.org/
3. www.haskell.org/haskellwiki/
4. http://www.w3schools.com/css/
5. http://sass-lang.com/
6. http://www.wikipedia.org/
7. http://www.pearsonified.com
8. http://webdesign.tutsplus.com/articles/design-theory/improving-layout-with-vertical-rhythm/
9. http://24ways.org/2006/compose-to-a-vertical-rhythm/
10. http://webdesign.tutsplus.com/articles/design-theory/mathematics-and-web-design-a-close-relationship