UNIVERSITY OF WATERLOO
Faculty of Engineering

An Analysis of Using Global Scoped Style Sheets and Atomic Scoped Style Sheets in the Context of Styling Component Based Applications

Yahoo!
Sunnyvale, CA

Aditya Sridhar
3B Software Engineering
Student ID 20539123
User ID a3sridha
September 8, 2016

Aditya Sridhar
201 Lester Street
Waterloo, ON N2L 3G1

September 10, 2016

Patrick Lam, Director
Software Engineering Program
University of Waterloo
200 University Ave. West
Waterloo, Ontario. N2L 3G1

Dear Patrick Lam:

This report, entitled "An Analysis of Using Global Scoped Style Sheets and Atomic Scoped Style Sheets in the Context of Styling Component Based Applications", was prepared during my 3B work term at Yahoo!, where I worked in the Mail Front-end department.

Description of your co-op position. This should be about three sentences

Description of report and how it related to co-op position.

Acknowledgements

I hereby confirm that I have received no help, other than what is mentioned above, in writing this report. I also confirm this report has not been previously submitted for academic credit at this or any other academic institution.

Yours,

_____

Aditya Sridhar, 20539123

Executive Summary

EXECUTIVE SUMMARY

Table of Contents

1 Introduction

1.1 Background Information

Yahoo! is an internet technology company and is recognized as one of the pioneers of the early-internet era. Yahoo! Mail is a web-based email service owned by Yahoo! with hundreds of millions of users worldwide. The next generation of Yahoo! Mail web client is currently being developed and is built using the ReactJS framework. ReactJS is a component-based JavaScript library used for building user interfaces. The idea behind a component-based framework is that a complex user interface should composed of multiple components, and each component should follow the Single Responsibility Principle (SRP). This implies implies that a particular component should be responsible for single part of the user interface.

Concern	**Resource**
Presentation	CSS
Structure	HTML
Behaviour	JavaScript

Table 1.1: Long table caption

Traditionally, the development of web pages was based on the principle of Separation of Concerns (SoC) which advocates breaking a problem into different concerns and using a resource to address a particular concern. In the context of web pages, the structure, presentation and behaviour layers were identified as separate concerns with Hyper Text Markup Language (HTML) defining the structure of a web page, Cascading Style Sheets (CSS) defining the content presentation styles and JavaScript (JS) defining how the web page behaves with user interaction. Table 1.1 summarizes the SoC principle in the case of web page development.

1.2 Introduction to the Issue

CSS is a collection of rules, each rule containing one or more selectors and a block of styles. These rules are written in style sheets which are then packaged in the main project. A selector is used to identify a particular HTML element in the web page content and the styles specified in the rule are applied to that particular HTML element. Figure 1.1 better elaborates how CSS rules work. The selector *div* targets the <*div*> HTML element and the styles mentioned in the corresponding style block are applied to the targeted HTML element.

```
1  <div>This is my web page</div>
2
3  div {
4      background-color: blue
5  }
```

Figure 1.1: Example of styling using CSS selectors

The scope of a CSS rule is a portion of the web page where that rule can be used to style an element on the web page. The rules contained in a global-scoped style sheet can be used to style any element on the web page. Hence the scope of the rules in a global style sheet is not limited to a particular portion of the web page. Atomic CSS on the other hand is a design principle that advocates restricting the scope of a CSS rule to a component of the web page.

This report compares CSS in global style sheets with Atomic CSS in the context of styling component-based applications. It first provides some additional information about the issue and lists a set of design constraints and the evaluation criteria. It identifies the benefits and pitfalls of each accepted solution. The report uses AHP to perform quantitative analysis of the alternatives against a set of evaluation criteria. The report then concludes by identifying the best solution and provides some recommendations.

2 Problem Specifications

2.1 Problem Description

CSS is an important component of the front-end toolkit. Since the internet is composed of billions of web pages, CSS is being used on a regular basis to govern the look and feel of web pages. One of the main concerns of using CSS is maintainability. CSS is easy to maintain when the web pages have less content, but as the size of the project increases, the maintainability of style sheets decreases. High maintainability implies that a developer can modify a particular rule without worrying about it adversely impacting other rules. Low maintainability hampers the scalability of a project from a design standpoint and significantly affects the developers in the team.

As mentioned earlier, a CSS selector targets a particular HTML element in the web page content and the styles linked to the selector is applied to element it targets. As the content of the web page grows, CSS selectors are more specific and descriptive to target a particular element in the web page. This is termed as selector specificity. If two selectors target the same element in the web page content, the selector with higher specificity tends to override the selector with lower specificity. The style sheet containing selectors with high specificity makes it harder for an existing style to be overriden by a new style. Hence low specificity leads to high maintainability.

```
1  <!-- HTML code -->
2  <div class="parent">
3      <div class="child1"></div>
4      <div class="child2"></div>
5  </div>
6
7  <!-- CSS code -->
8  .parent .child1 {
9      background-color: red
10 }
```

Figure 2.1: Example of styling using context-based selectors

At times, a developer might want to match a web page element that is a descendent of another web page element. For example in Figure 2.1 the <*div*> element whose class name is *child1* is a descendent of the <*div*> element whose class name is *parent*. The CSS selector *.parent .child1* is an example of a context-based selector. It is targeting the element with class

name as *child1* that is a descendent of the element with class name as *parent*. This means that the descendent element is being styled in the context of its parent, and that the rule is redundant outside of this context. This will ultimately lead to more rules being added to the style sheet if the requirement to style the same element outside of the context arises.

2.2 Design Constraints

A solution should adhere to the following design constraints, in order to be deemed as an accepted solution:

2.3 Design Criteria

The following evaluation criteria are used to compare the two accepted solutions:

1. **Minimize the usage of context-based styling**: The accepted solution should limit the usage of contextual styling as this demotes reusability and portability of styles. This ultimately makes style sheets hard to maintain.
2. **Low specificity of styles**: Low CSS specificity enables the developer to override a CSS rule when requirements change. This makes the style sheets more maintainable.
3. **Intuitive style changes**: The accepted solution should be intuitive and predictable enough for the developer to visualize any changes made to the style sheet in terms of web page presentation. This includes adding, removing and modifying rules.
4. **Style reusability** The accepted solution should promote reusability of styles. New style requirements being added to the specification should sparingly lead to new rules being added to the style sheet.
5. **Decouple markup and styles** The accepted solution should disallow coupling of web page content (markup) and styles. This enables the developer to make changes to the presentation without making changes to the markup.

2.4 Accepted Solutions

2.4.1 Atomic scoped

2.4.2 Global scoped

2.5 Evaluation Process

2.6 Alternative Not Considered

3 Conclusions

CONCLUSIONS

4 Recommendations

RECOMMENDATIONS

Acknowledgements

ACKNOWLEDGEMENTS