**Effizienzsteigerung beim Implementieren von CSS Elementen für Appway AG Front-End Entwickler**

(Version 2.1 – 31.05.2014)

**1 Einleitung**

**TODO: Einleitung ist gleich dem Abstract!**

**1.1 Ausgangslage**

Das Unternehmen Appway AG entwickelt eine Webapplikation, um Geschäftsprozesse modellgetrieben zu entwerfen und zu entwickeln. Zu den Kernkompetenzen von Appway AG gehören die einfachen und selbsterklärenden Benutzeroberflächen. Benutzer können so mit vorgefertigten Komponenten Bildschirme für Prozessschritte ohne Programmiererfahrung bauen. Die Front-End Entwickler, welche für die Implementierung dieser Komponenten zuständig sind, schreiben viel CSS (Cascading Style Sheets), um das Aussehen im Browser steuern zu können. Der Aufbau und der Gebrauch von CSS führen zwangsweise zu Redundanzen im Quellcode und zum Überblicksverlust bestehender CSS-Klassen. Appway AG will mit ihrem Produkt alle gängigen Browser unterstützen, deshalb ist es notwendig die CSS-Klassen so zu implementieren, dass die Anweisungen von allen Browsern korrekt verstanden werden. Das Fehlen von Regelungen oder Prozessen führt dazu, dass Appway Entwickler viel Programmcode duplizieren müssen und somit fehleranfälliger werden.

**1.2 Ziel der Arbeit**

Das Ziel der Bachelorarbeit besteht in der Evaluation von geeigneten Techniken oder Tools für die Effizienzsteigerung von Appway Entwicklern bei der Implementation von CSS Programmcodes. Eine Anforderungsanalyse gemäss Requirement Engineering soll die Ausgangslage erfassen und die für Appway AG beste Lösung aufzeigen. Als weiteres Ziel soll eine Lösungsimplementation vorbereitet oder teilimplementiert werden, bei welchen die evaluierten Techniken oder Tools zum Einsatz kommen.

**1.3 Aufgabenstellung**

Im Rahmen dieser Bachelorarbeit werden vom Studenten folgende Aufgaben durchgeführt:

1. Ausgangslage erfassen.
2. Requirement Engineering für das gesamte Projekt.
3. Marktanalyse der verfügbaren Techniken und Tools zur CSS Verarbeitung.
4. Evaluation der zu verwendenden Technik oder Tool.
5. Konzept für Einführung der Technik oder Tool in den Arbeitsprozess von Appway AG.
6. Start der Umsetzung / Teilimplementation.
7. Evaluation von eingesparten Ressourcen.
8. Präsentation der Umsetzung oder der eventuellen Teilimplementation.

**1.4 Erwartete Resultate**

Im Rahmen dieser Bachelorarbeit werden folgende Resultate für die einzelnen (je Teilaufgabe) Teilaufgaben erwartet:

1. Dokumentation der Ausgangslage.
2. Anforderungsanalyse und Stakeholderanalyse für das Projekt.
3. Marktanalyseergebnisse auflisten und beschreiben.
4. Bewertungsanalyse und Dokumentation der Entscheidung.
5. Plan für die Einführung des Konzepts bei Appway AG.
6. Technik oder Tool wurde integriert und die Teilimplementation ist lauffähig.
7. Dokumentation der eingesparten Ressourcen.
8. Vorbereitete Demonstrationsumgebung.

**1.5 Termine**

|  |  |
| --- | --- |
| 21.12.2013 | Start Bachelorarbeit |
| 15.01.2014 | Kick-Off Meeting |
| 03.02.2014 | Statusmeeting mit Betreuer |
| 24.03.2014 | Statusmeeting mit Betreuer |
| 23.04.2014 | Design Review |
| 30.04.2014 | Statusmeeting mit Betreuer |
| 14.05.2014 | Statusmeeting mit Betreuer |
| 27.05.2014 | Statusmeeting mit Betreuer |
| 21.06.2014 | Abgabe Bachelorarbeit |
| ??? | Schlusspräsentation |

**2 Projektplanung**

Die Bachelorarbeit wurde am 21. Dezember 2013 offiziell freigegeben. Bis dahin wurde aber schon einige Zeit investiert um sich in die Thematik einzulesen. Im sechsmonatigen Zeitraum wurden alle Montage sowie gewisse Wochenenden für die Bachelorarbeit aufgewendet. Zusätzlich zu dieser Zeit konnte während zwei dedizierten Ferienwochen und für das Proof of Concept in der Arbeitszeit die Bachelorarbeit fertiggestellt werden. Trotz aufgestelltem Zeitplan kamen Zwei Seminararbeiten dazu, welche das Vorwärtskommen bremsten. Aber auch mit Zeitdruck des letzten Semesters ist die Bachelorarbeit erfolgreich fertiggestellt worden.

**2.1 Zeitplan**

**=> Im Google Kalender Einträge erfassen wann effektiv daran gearbeitet wurde.**

**=> EBS Kalender**

**Meilensteine**

**- Konzept & Aufgabenstellung**

**- Freigabe Bachelorarbeit**

**- Fehlerkatalog**

**- Lösungsentscheid**

**- Design Review**

**- Proof of Concept**

**- Abgabe Bachelorarbeit**

**- Präsentation**

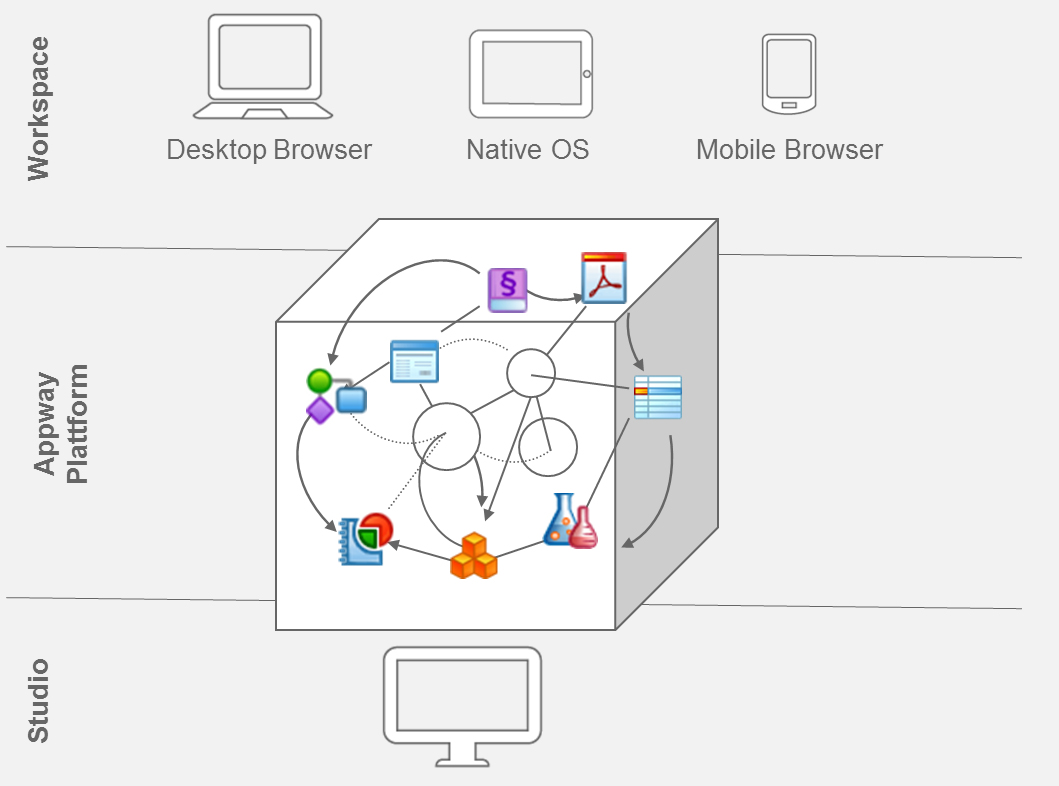
**2.2 Zeitplan SOLL / IST S**

|  |  |  |
| --- | --- | --- |
| **Projektphase** | **SOLL (in h)** | **IST (in h)** |
| **Vorbereitung** |  |  |
| Thema auswählen | 20 | 15 |
| Aufgabenstellung erfassen | 2 | 4 |
| In das Thema einlesen | 90 | 110 |
| **Anforderungsanalyse** |  |  |
| Fehlerkatalog erstellen | 10 | 15 |
| Anforderungen definieren | 5 | 2 |
| **Evaluation** |  |  |
| Recherche von Lösungsvarianten | 50 | 75 |
| Vergleich der Lösungsvarianten | 20 | 12 |
| **Umsetzung** |  |  |
| Proof of Concept | 40 | 70 |
| Akzeptanztests | 10 | 6 |
| **Bachelorarbeit** |  |  |
| Abstract | 2 | 2 |
| Plakat | 1 | 1 |
| Dokumentation | 60 | 80 |
| Präsentation | 5 | 4 |
| **Verschiedenes** |  |  |
| Termine | 15 | 15 |
| Pufferzeit | 30 |  |
| **Total** | **360** | **411** |

Die Werte in der obigen Tabelle weichen nur in wenigen Punkten mit mehr als 10 Stunden ab. Diese wurden aber meistens unterschätzt oder der dafür benötigte Zeitaufwand fiel grösser als geplant auf. Beim Einlesen in die Thematik sowie die Recherche der Lösungsvarianten wurde einiges mehr Zeit verwendet als angenommen. Dies ist darauf zurückzuführen, dass es viele Bücher und Webseiten mit guten Inhalten sowie über 100 Lösungsansätze für CSS im Internet zu finden sind. Die Integration von SASS ins Appway Produkt hat einige unvorhersehbare Probleme verursacht. Deshalb wurde für das Proof of Concept wurde fast die doppelte Zeit gebraucht. Die Dokumentation ist immer ein Punkt der sehr viel Zeit in Anspruch nimmt und ist trotz den 20 Stunden mehr Aufwand keine Überraschung.

**3 Appway**

Appway als Applikation ist die erste 100% Modell getriebene Geschäftsprozessmanagement (GPM) Plattform auf dem heutigen Markt. Die Applikation schliesst die Lücke zwischen Softwareentwicklung und Geschäftsprozessmanagement, indem jedes visuelle Modell direkt in eine Prozessapplikation umgewandelt wird. Appway ist eine Webapplikation, mit welcher bestehende Systeme integriert und komplexe Prozesse oder Aufgaben ausgeführt werden können. Geschäftsprozesse erstellen oder ändern ist aber nur ein kleiner Teil der Appway Plattform. Die Webapplikation bietet automatische Tests und Auslieferungen, sowie Vereinfachung beim gesamten Softwarelebenszyklus (von der Anforderungsanalyse bis zur Bereitstellung).

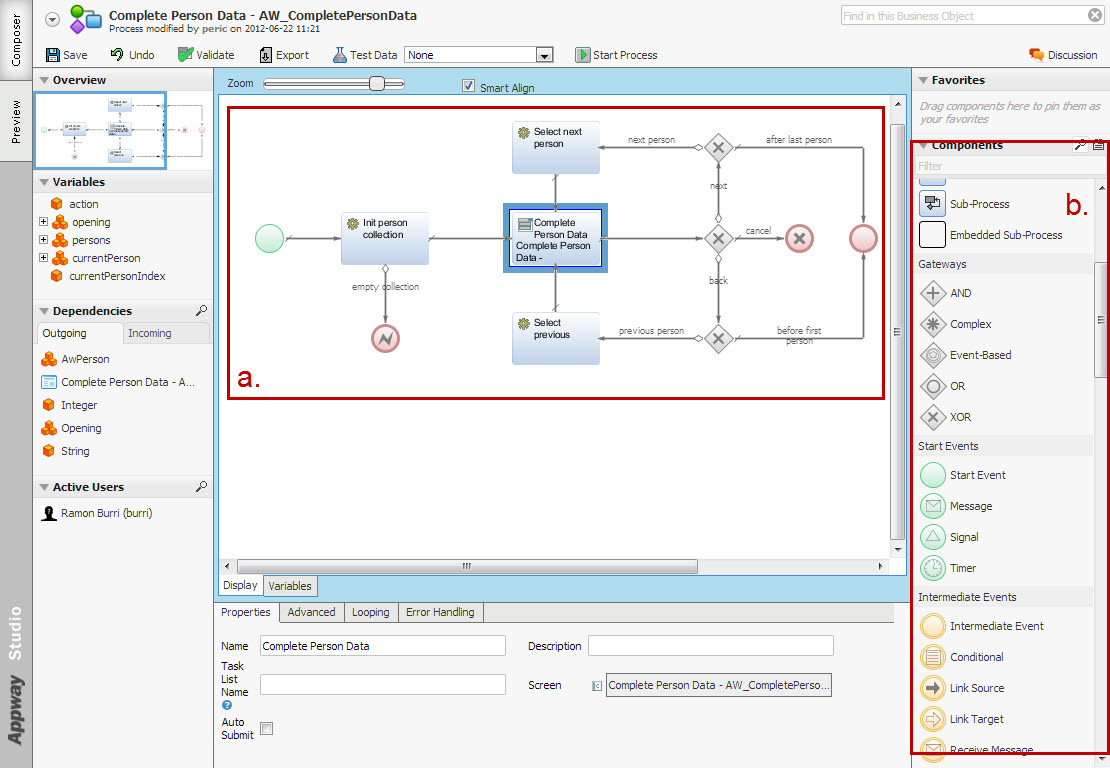


**Abbildung : Das Appway Modell**

**3.1 Appway als GPM Tool**

Geschäftsprozessmanagement beinhaltet die Automatisierung, Koordination und Steuerung aller Geschäftsprozesse innerhalb eines Betriebs. Technische, wie auch organisatorische Fragestellungen werden mit der Hilfe von Personen und Systemen adressiert. Appway bietet mit dem Prozesseditor die Möglichkeit bestehende oder auch neue Prozesse modellbasiert zu erstellen und zu testen. Der Editor wurde an die Geschäftsprozessmodell Notation (BPMN) angelehnt und ist so für die Personen der Geschäftsseite einfacher zugänglich, da die Modellsprache weltweit verständlich ist.

Ein Geschäftsprozess besteht aus vielen kleinen Aufgaben oder Teilaufgaben, welche in Appway durch Symbole visuell dargestellt werden. Symbole verkörpern Aufgaben, wie der Start oder das Ende eines Prozesses, Bedingungen, Teilungsknoten und einigen andere komplexere Aufgaben. Die Erstellung solcher Prozesse benötigt kein Wissen eines Softwareentwicklers und kann von der Geschäftsseite her einfach und wie in einem Diagrammtool manipuliert werden.



**Abbildung: Appway Prozesseditor**

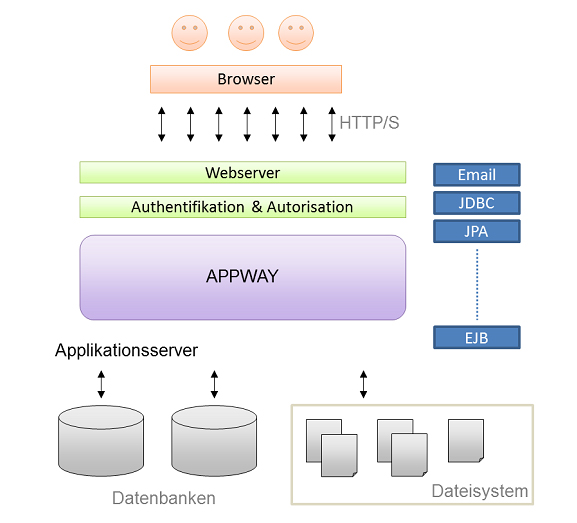
a. Inhaltsbereich des Prozesseditors

b. GPM Elemente die verwendet werden können

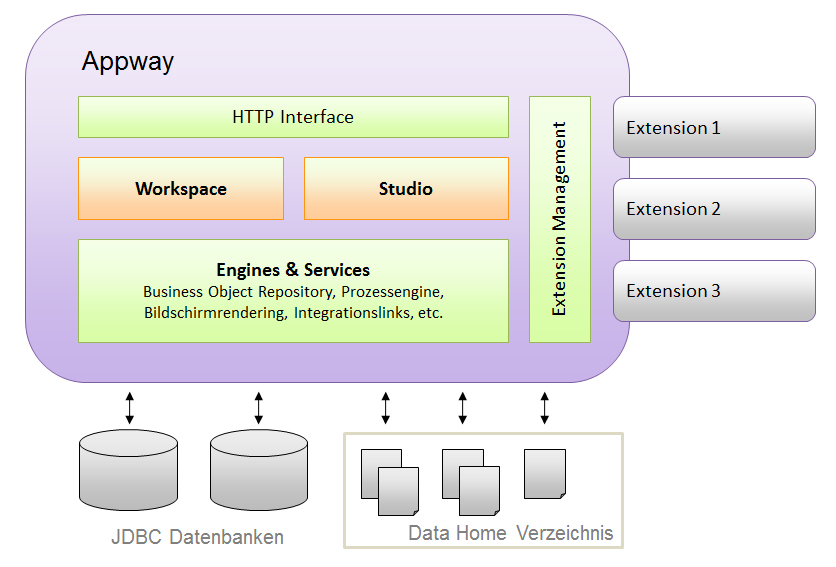
Durch das Einfügen von Screens, ein einzelner Bildschirm einer Webanwendung, werden ein oder mehrere Geschäftsprozesse in eine zusammenhängende Webapplikation umgewandelt. Der Editor ist zudem in der Lage automatisch gewisse Logiken zu prüfen, um nicht terminierende Prozesse oder fehlende Abhängigkeiten zu erkennen und anzuzeigen.

**3.2 Systemabgrenzung**

Appway ist eine Webapplikation welche auf einem Java unterstützten Webserver läuft. Anwender greifen über einen Browser auf den Webserver zu und werden zur Applikation weitergeleitet. Appway bietet Schnittstellen zu Webservices, Datenbanken oder Extensions an. Die Daten welche Appway produziert werden standartmässig im Dateisystem gespeichert.



**Abbildung : Sytsemabgrenzung der Appway Applikation**

****

**Abbildung : Aufbau der Appway Applikation**

**3.3 Appway Philosophie**

Die Grundidee von Appway ist die Lücke zwischen Geschäftsseite und Entwicklungsseite zu schliessen. Dies funktioniert nur mit einer gemeinsamen Basis, oder auch Sprache, die von beiden Seiten gleich interpretiert werden kann. Das Ziel wird durch den Modellgedanken erreicht, damit kein Programmcode nötig ist, um ein Geschäftsprozessproblem zu lösen. Trotzdem soll das Modell Fehler erkennen, komplexe Aufgaben lösen und intelligent mit weiteren Systemen interagieren.

Für die Bachelorarbeit sollen folgende Punkte, in Bezug auf die Philosophie von Appway, berücksichtigt werden:

**Kompatibilität**

Entwicklung sollte immer mögliche Migrationen von anderen Appwayversionen berücksichtigen. Auch ältere Browser sollen stets so gut wie möglich unterstützt werden.

**Design**

Module oder Komponenten in Appway sollten selbst erklärend und einfach zu bedienen sein, ohne den Aspekt eines guten Designs ausser Acht zu lassen. Die mit Appway erstellten Lösungen haben als Basis ein solides Appway Design, welches von den Front-End Entwicklern und Designer von Appway jeweils freigegeben wird. Lösungen sollten aber jederzeit durch CSS angepasst werden können.

**DRY (Don't repeat yourself)**

Bei der Entwicklung auf Bestehende Objekte und Module von Appway aufbauen.

**3.4 Appway Plattform**

**3.4.1 Verwendete Technologien**

Die Appway Plattform ist eine Webapplikation, welche mit mit der Programmiersprache Java geschrieben wurde. Die Entscheidung Java zu verwenden führt bis auf die Gründung des Unternehmens zurück und wird damit belegt, dass Java die meist verbreitete Sprache unter den Entwicklern war. Der Softwarekern ist somit in Java geschrieben und die Schnittstelle zum Web wird über JSP (JavaServer Pages) gemacht.

Die im Web ausgelieferten und im Browser dargestellten Seiten bestehen in erster Linie aus HTML Code. Dynamische Elemente werden meist mit Hilfe von JavaScript behandelt. Für das Aussehen werden Cascading Style Sheets (CSS) verwendet.

**3.4.2 Verfügbares Toolset**

Für die Entwicklung der Screen-Komponenten kann auf eine Sammlung bereits integrierter Bibliotheken, Modulen und Tools zugegriffen werden. Hier ist eine kurze Aufzählung mit den für die Bachelorarbeit relevanten Komponenten:

**Intellij**

Integrierte Entwicklungsumgebung um Javaprogrammcode, HTML, JavaScript und CSS zu schreiben.

**Apache Tomcat**

Open Source Webserver, um Java Applikationen über JSP Seiten auszuführen.

**FogBugz**

Ein webbasiertes Projektmanagement Tool für Fehlertracking, Diskussionsforums und Wikis.

**4 Grundlagen**

Bevor es zu den Anforderungen geht wird kurz beschrieben was CSS genau ist, wozu es gebraucht wird sowie die gängigsten Anwendungsvarianten. Zurzeit während diese Bachelorarbeit verfasst wird ist CSS Level 3 (CSS3) immer noch in der Entwicklung. Die Kerneigenschaften von CSS3 sind im W3C-Standart schon aufgeführt jedoch unterstützen die Modernen Browser bereits schon mehr Funktionalität.

**4.1 Cascading Style Sheets**

Cascading Style Sheets oder kurz auch CSS ist eine deklarative Sprache für Stilvorlagen von Strukturierten Dokumenten wie HTML oder XML. Die Grundidee von CSS war die Trennung von Inhalt und Darstellung. Während HTML oder XML die Struktur eines Dokumentes beschreibt können mit CSS Darstellungen wie Farbe, Layout oder Typographie für Elemente definiert werden. Im Verlauf der Zeit wurde die Sprache immer mehr durch zusätzliche Elemente wie Animationen oder Unterscheidungen vom Ausgabemedium hinzugefügt. In dieser Arbeit gehe ich nur auf die Aspekte ein welche für die Darstellung von HTML notwendig sind.

CSS nutzt sogenannte Selektoren damit auf Elemente in einem Dokument gesucht oder zugegriffen werden kann. Die Identifikation eines Elementes ist deshalb ein wichtiger Faktor beim schrieben von CSS Code. Elemente können durch diverse Merkmale Identifiziert werden:

* Elementnamen (Beispiel: „ul“ für ein unsortiertes Listenelement)
* HTML ID eines Elementes
* Position eines Elementes im Dokument (Beispiel: Alle Unterelemente des „li“ Elementes)
* Attributnamen und Attributewerte (Beispiel: Alle „li“ Elemente mit dem Attribut „class“ und dem dazugehörigen Wert „navigation“.

Mit diesen Selektoren können Stilvorlagen mit CSS-Regeln direkt auf einzelne Elemente appliziert werden.

CSS-Regeln können dabei zentral direkt beim Element, im Kopfbereich einer HTML-Seite oder in einem separaten Stylesheet definiert werden. Die Letztere ist die bevorzugte Variante, da sie ermöglicht Stylesheets für weitere Dokumente wiederzuverwenden und der gesamte CSS Code an einer Stelle ist. Ein aufblähen eines Stylesheets, durch zu viele Definitionen, wird durch ein Vererbungsmodell für Auszeichnungsattribute vermindert. Dieses Vererbungsmodell („cascading“) ermöglicht das verschachteln von Selektoren.

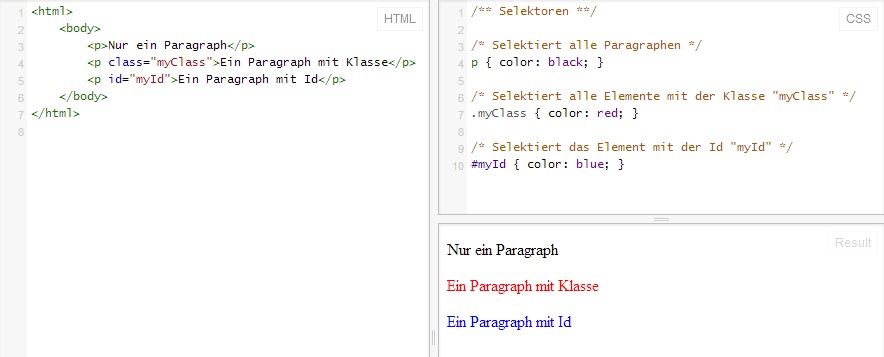
**4.2 Syntax**

Der CSS Syntax besteht aus verschiedenen Stilen. Ein einzelner Stil besteht immer aus einem Selektor und einer Regel. Es besteht die Möglichkeit für mehrere Selektoren die gleiche Regel zu applizieren oder umgekehrt mehrere Regeln auf einen Selektor. In den CSS Beispielen in dieser Bachelorarbeit tauchen Zeilen auf die mit **/\*** beginnen und **\*/** enden. Solche Zeilen sind Kommentare und werden beim lesen vom CSS ignoriert.

**4.2.1 Selektoren**

Mit Hilfe von CSS Selektoren wird bestimmt auf welche HTML Elemente Stile appliziert werden sollen.

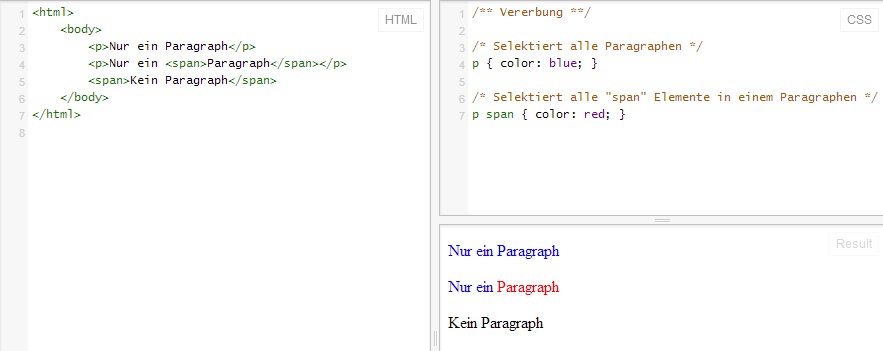
Es können nach HTML Elemente, Klassen oder Ids selektiert werden.

****

**BILD SELEKTOREN**

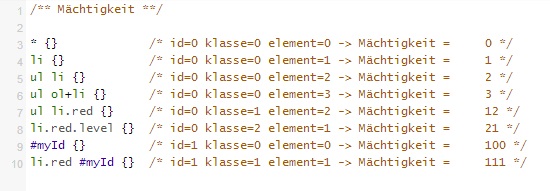
**http://superuser.com/questions/419814/create-a-microsoft-word-quick-style-for-a-code-snippet-box**

Selektoren funktionieren nach kaskadierendem Prinzip. Das heisst, die Regeln werden von Oben nach Unter gelesen und können sich gegenseitig überschreiben. Der Aufbau der HTML Seite, wo das CSS wirksam gemacht wird, spielt hierbei eine wichtige Rolle. Kinderelemente erben die Stile von ihren Elternelementen. Deshalb können die Selektoren selbst auch verschachtelt sein, um in der hierarchischen Baumstruktur die richtigen Elemente auszuwählen.



**BILD VERERBUNG**

Es gibt unzählige Selektortypen, welche bei der Programmierung mit CSS zur Hilfe kommen. Damit bei den vielen unterschiedlichen Selektoren die Übersicht bewahrt werden kann, bietet sich an zuerst alle Allgemeinen Stile zu definieren und immer spezifischer zu werden. Dies verhindert ungewollten überschreiben von Stielen. Aber auch trotz kaskadierungsregeln haben die einzelnen Selektortypen verschiedene Mächtigkeiten. Es wird dabei unterschieden, ob es sich um einen Id-, Klassen- oder Elementselektor handelt. Je spezifischer die Art und Anzahl Vorkommnisse eines Selektores, desto höher der Mächtigkeitswert.



**BILD MÄCHTIGKEIT**

**4.2.2 CSS Regeln**

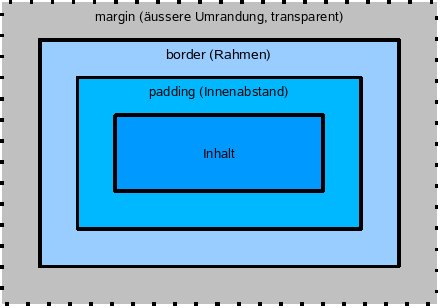
Eine CSS Regel besteht immer aus einem Eigenschaftsnamen und deren Wert. Diese Eigenschaften wirken auf die durch Selektoren bestimmten Elemente. Der Name und der Wert werden immer durch einen Doppelpunkt getrennt und die Eigenschaft durch ein Semikolon abgeschlossen. Es gibt unzählige Eigenschaften, die Aussehen, Form und Verhalten beschreiben.

**SELEKTOR { EIGENSCHAFTSNAME : WERT }**

**BILD REGELN**

**4.2.3 Box-Modell**

Das Box-Modell definiert die Berechnung der Breite und Höhe von HTML Elementen. Gemäss den Spezifikationen des W3-Konsortiums errechnet sich die Gesamtbreite eines Elementes aus einer Addition von Inhaltsbreite, Innenabstand, Rahmenstärke (border) und dem Aussenabstand. Für die Höhe ist die Berechnung analog zur Breite.



**BILD BOXMODELL**

<http://upload.wikimedia.org/wikibooks/de/8/85/Boxmodel.jpg>

HTML Elemente können entweder Blockelemente sein mit eigener Höhe und Breite oder eine Inlineelement, welches dann die Masse durch den Inhalt übernimmt. In CSS wird die Darstellung immer durch ein solches Box-Modell berechnet. Die Aussen- und Innenabstände sowie die Rahmenstärke können bei einem Element auf allen Vier Seiten vorhanden sein. Ein Blockelement mit zum Beispiel einer Inhaltsbreite von 30px, einer Rahmenstärke von 5px und eine, Aussenabstand von 15px besitzt eine Gesamtbreite von 70px. Es wird ein Mal die Inhaltsbreite genommen, dazu zwei Mal die Rahmenstärke (links und rechts) sowie zwei Mal den Aussenabstand addiert.

**4.2.4 CSS Hacks**

CSS Hacks bestehen aus einem Set an Stilen, um Unterschiede bei der Darstellung in verschiedenen Browsern auszugleichen. Der Begriff Hack bezeichnet dabei nichtstandardisierte Stile, mit welchen Interpretationsschwächen von Browsern ausgenutzt werden, um möglichst in jedem Browser das gleiche Ergebnis angezeigt zu bekommen. Dabei kann der Hack fehlerhaft angegebene Selektoren in Kombination mit zusätzlichen Zeichen angereichert werden oder enthält Anweisungen, die bestimmte Browser nicht kennen. Ein Beispiel ist der „Star-HTML-Hack“. Das **\***-Zeichen dient als Universalselektor und ist vor dem **html** Selektor sinnlos.

**BEISPIEL CSS-HACK**

p { **background-color**: **blue**; }

\* html p { **background-color**: red; }

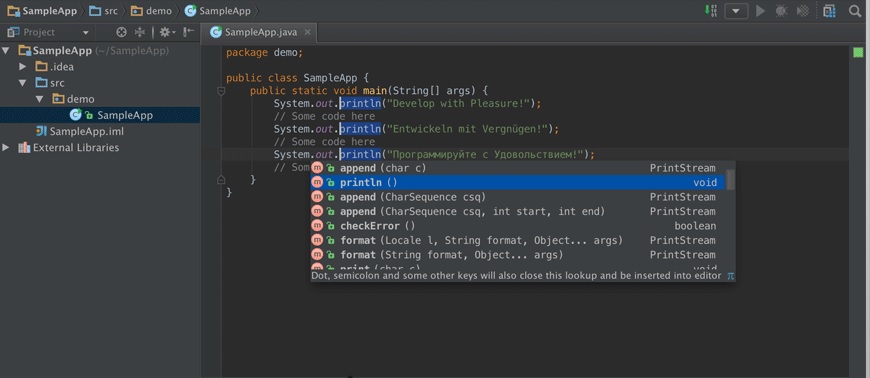
Die Browser interpretieren die p-Elemente und stellen diese mit einem blauen Hintergrund dar. Der Internet Explorer 7 interpretiert aber auch die zweite Zeile und färbt den Hintergrund wieder rot, obwohl html kein Eltern-Element besitzt, auf das \* zutreffen könnte.

Solche Hacks werden immer wieder gebraucht, damit eine einheitliche Browsererscheinung erreicht werden kann. Das Anwenden zu vieler Hacks kann zum Übersichtsverlust führen und erschweren bestehende Stile anzupassen.

**5 IST Analyse**

**5.3.1 Appway AG Implementation**

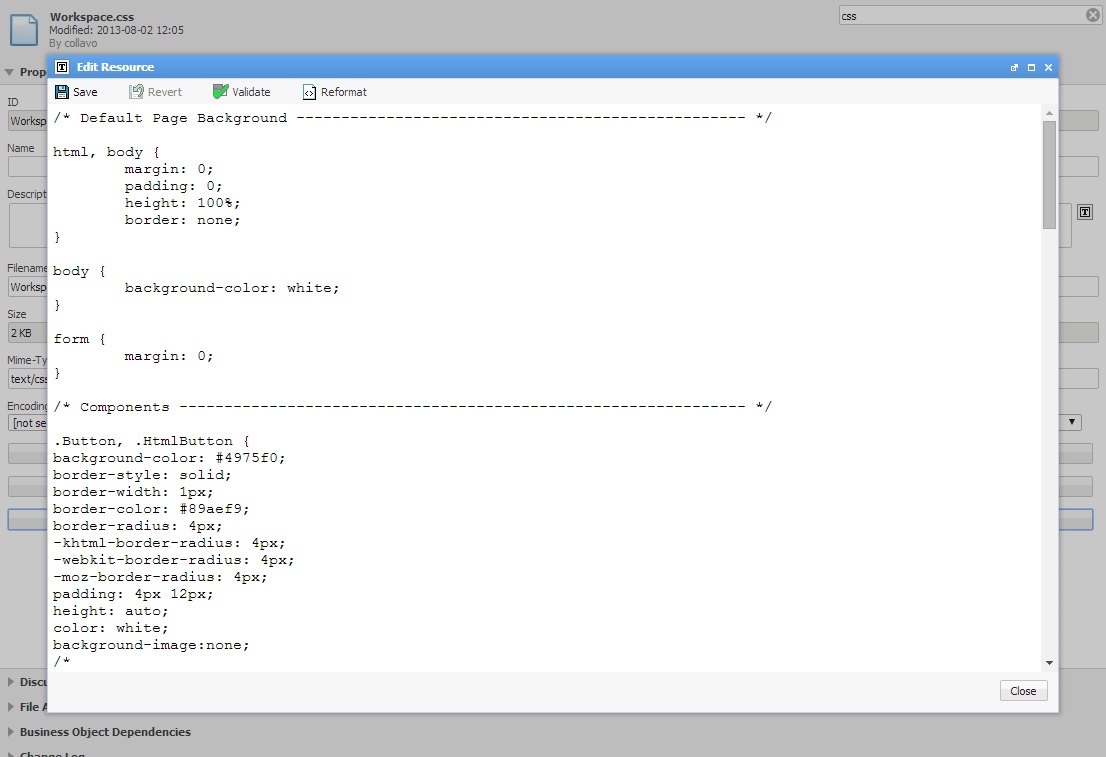
Die Arbeitsweise von Appway Entwicklern und Studiobenutzern unterscheidet sich nur darin, dass eine andere Arbeitsumgebung vorhanden ist um CSS zu schreiben. Die Entwickler arbeiten meist in einer integrierten Entwicklungsumgebung (IDE) wie IntelliJ. Dies hat Vorteile wie Codehervorhebung, automatische Fehlererkennung oder Codeergänzung.



**BILD IDE IntelliJ**

http://www.jetbrains.com/idea/

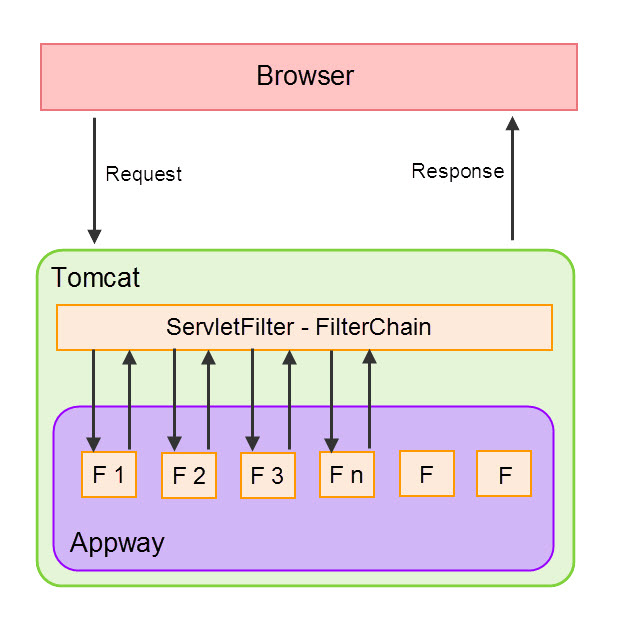
Die Studiobenutzer hingegen sind auf das Appway Studio begrenzt und haben eine deutlich weniger komfortablere Arbeitsumgebung, da der CSS Editor im Appway Studio ein simpler Texteditor ist. Es ist aber auch möglich, die Entwicklung und Lagerung von CSS Dateien extern zu machen und in Appway Studio darauf zeigen lassen.



**BILD STUDIO CSS EDITOR**

Das Appway Studio bietet bereits Mechanismen an, welche auch in komplexeren Präprozessoren vorkommen. Es können Farben als Businessobjekte im Stuio erfasst werden und danach im CSS Code referenziert werden. Ebenfalls kann auf eine Anzahl von globalen Platzhaltern, wie der Kontextpfad für Verlinkungen, zugegriffen werden. Diese sind aber alle fest im Kern von Appway definiert und können nur über eine Konfiguration geändert werden.

Der Webserver liest mit Hilfe des Servlet Filters beim Seitenaufruf die Antwort des Servers aus und ersetzt in JavaScript und CSS Dateien die Platzhalter. Im Quellcode einer Webseite sieht man so die ersetzten Werte und der Browser kann diese Information auch verarbeiten.



**BILD: Diagramm des Ablaufes von CODE replacement**



**BILD: BEISPIEL VORHER**

Neben der Codeersetzung arbeiten die Appway Entwickler mit einem Styleguide. Dieser Styleguide wird als Richtlinie im Umgang mit CSS Code geführt und wird vom Entwicklerteam stetig weiterentwickelt. Das Einhalten der Richtlinien des Styleguide wird nicht überwacht und ist daher an die Motivation des Entwicklers gebunden. Ein Styleguide bringt in der Regel nur Erfolg, wenn er auch eingehalten wird. Der Styleguide von Appway AG an dessen auch der Student dieser Bachelorarbeit mitgewirkt hat findet sich im Anhang.

**6 Anforderungsanalyse**

**6.1 Stakeholder**

Mit dieser Bachelorarbeit kommen bei der Durchführung einige Stakeholder in kontakt. Ein Stakeholder hat jeweils einen Bezug sowie spezifische Erwartungen an die Bachelorarbeit.

|  |  |  |
| --- | --- | --- |
| **Stakeholder** | **Beschreibung** | **Erwartungen** |
| Ramón Burri (Student) | Mitarbeiter von Appway AG und Teil des RnD Teams. Verfasser der Bachelorarbeit. | Ablieferung einer guten Bachelorarbeit sowie die Effizienzsteigerung des eigenen Arbeitsprozesses. |
| Entwicklerteam (Appway AG) | Mitarbeiter von Appway AG vom Research and Development (RnD) Team. Das Team besteht aus 13 Personen die in Berührung mit CSS kommen und von der Effizienzsteigerung betroffen sind. Im Verlauf dieser Arbeit werden Sie unter dem Begriff **Appway Entwickler** referenziert. | Gut einsetzbare Lösung für Effizienzsteigerung. |
| CTO, Oliver Brupbacher (Appway AG) | Direkter Vorgesetzter des Studenten und Repräsentant vom Unternehmen Appway AG. | Hat Interesse eine Effizienzsteigerung im Umgang mit CSS herbeizuführen. |
| Studio Anwender | Entwickler die im Appway Studio Lösungen bauen. Schreiben selber CSS Code um Lösungen für die Kunden zu designen. Im Verlauf dieser Arbeit werden Sie unter dem Begriff **Studio Anwender** referenziert. | Eine professionelle Lösung. |
| Beat Seelinger | Betreuer der Bachelorarbeit und Fachperson im Bereich Webengineering. | - |
| ZHAW | Auftragsteller und Abnehmer der Bachelorarbeit. | Will eine gemäss Reglement durchgeführt Bachelorarbeit haben. |
| Experte | Von der ZHAW gestellter Experte um als externe Fachperson die Bachelorarbeit zu bewerten. | - |

**6.2 Anforderungen**

**6.2.1 Daten Akquirieren**

Die Liste der Anforderungen spiegelt eine Fehlerliste wieder, welche aus verschiedenen Quellen zusammengetragen wird. Damit diese Liste möglichst vollständig wird sind folgende Quellen verwendet worden:

**Fogbugz**

Appway Entwickler nutzen das Ticketsystem Fogbugz um Fehler, Ideen und auch Wünsche festzuhalten. Für die Bachelorarbeit wurde eine Suche nach relevanten Begriffen wie CSS, Stylesheet, Style und Farbe auf dem Ticketsystem abgesetzt und die Resultate nach Einträgen durchforstet, welche typische Fehler im Umgang mit CSS aufweisen.

**Feldbeobachtung**

Im Unternehmen Appway AG arbeiten mehr als 20 Berater, die selbständig für Kunden im Appway Studio entwickeln und somit auch in den Kontakt mit CSS kommen. In einem Feldversuch wurde im Verlauf einer Woche mehrmals verschiedenen Beratern beim anwenden vom CSS über die Schulter geschaut. Die Anwesenheit des Studenten wurde aber meist als störend empfunden weshalb nicht viel aufschlussreiches Material mit dieser Methode erschlossen werden konnte.

**Internetrecherche**

Mit einer Suche nach Tipps und Tricks im Umgang mit CSS sind unzählige Webseiten erschienen. Viele davon behandeln nur gewisse Aspekte oder sind viel zu generalisiert für eine brauchbare Auflistung. Die wenigen Webseiten mit guten Inhalten wurden studiert und die Fehlerliste mit diesen Punkten ergänzt.

**Teamsitzung**

Bei Appway AG wird wöchentlich eine Teamsitzung durchgeführt. Diese Gelegenheit wurde genutzt um von Mitentwicklern zu erfahren, welche Probleme ihnen beim programmieren mit CSS begegneten und wie diese von ihnen gelöst wurden. Diese Methode ist im Gegensatz zum Ticketsystem oder der Internetrecherche die Aufschlussreichste. Dies ist darauf zurückzuführen, dass eine direkte Frage-Antwort Situation entsteht und bei lückenhaften Antworten nochmals nachgefragt werden kann.

**6.2.2 Fehlerliste**

Folgend ist die Liste mit allen Fehlern die im Kontext von Appway aufgetaucht sind oder auftauchen können. Die Fehler sind jeweils mit einer einmaligen Identifikationsnummer, einem Titel und Beschreibung versehen. Im Ausblick auf eine spätere Priorisierung werden zusätzlich die Häufigkeit des Auftretens und die Grösse des Schadens bei Nichtbehebung angegeben.

|  |  |  |  |
| --- | --- | --- | --- |
| **Häufigkeit** | | | |
| **0** | **1** | **2** | **3** |
| Bisher noch nicht aufgetreten | 1-2 Mal im Jahr | 1 Mal im Monat | Mehrmals pro Woche |

|  |  |  |  |
| --- | --- | --- | --- |
| **Kritikalität** |  |  |  |
| **0** | **1** | **2** | **3** |
| Nicht auffallend und kann ignoriert werden | Schönheitsfehler aber beeinträchtigt die Benutzung nicht | Kann zu Beeinträchtigung beim Nutzen des Produktes führen | Produktionsfähigkeit ist beeinträchtigt |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: ERR\_001** | **Wiederholung von Farbendefinitionen** | | |
| Beschreibung: | In grossen Stylesheets werden an vielen Orten Farben für Schriften, Hintergründe oder andere Elemente definiert. Falls keine klare Designrichtlinie oder Farbschema vorhanden ist kommt es schnell zu Abweichungen von Farben. Ebenfalls müssen bei Änderungen immer alle Vorkommnisse gesucht und angepasst werden. Dies führt dazu, dass die Designrichtlinien nicht eingehalten werden und das Produkt nicht stimmig erscheint | | |
| Häufigkeit: | 3 | Kritikalität: | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: ERR\_002** | **Browserspezifische CSS Eigenschaften** | | |
| Beschreibung: | CSS Eigenschaften die spezielle Ausprägungen für verschiedene Browser haben werden oft nicht komplett oder in falscher Reihenfolge geschrieben. Dies führt dazu, dass gewisse Browser die Eigenschaft nicht oder falsch interpretieren.  Wenn ein Verlauf dargestellt werden soll via CSS, dann haben die verschiedenen Browser jeweils eine unterschiedliche Syntax um diesen darzustellen. Ein Browser nimmt immer die erste Anweisung, welche er lesen kann. Deswegen ist die Reihenfolge der Regeln im Stil wichtig.  C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\gradient_example.jpg  **BILD GRADIENT EXAMPLE** | | |
| Häufigkeit: | 2 | Kritikalität: | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: ERR\_003** | **Welcher Stil wird hier appliziert** | | |
| Beschreibung: | CSS Stile fliessen von verschiedenen Orten in das HTML hinein und sind meist für den Entwickler nur schwer nachzuvollziehen. Aus diesem Grund werden immer wieder Inline Stilvorlagen geschrieben mit dem **!important** Zusatz. Dies führt dazu, dass diese Eigenschaft nicht mehr geändert werden kann und verhindert das Designen von HTML Elementen. | | |
| Häufigkeit: | 2 | Kritikalität: | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: ERR\_004** | **Redundanz im CSS Code** | | |
| Beschreibung: | CSS Programmierung verleitet oft zu Code Duplikation. Deswegen ist an vielen Stellen Redundanz vorhanden.  In grösseren Stylesheets kommt es immer wieder vor, dass durch die fehlende Übersichtlichkeit die gleichen Regelstile wiederholt angewendet werden.  C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\example_code_duplication.jpg  **BILD CODE DUPLICATION** | | |
| Häufigkeit: | 3 | Kritikalität: | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID: ERR\_005** | **Debuggen von Stylesheets erschwert** | | | |
| Beschreibung: | Fehlersuche in Stylesheets gestaltet sich schwer. Meistens fehlen die Kenntnisse über vorhandene Tools. Zudem verleitet der Appway Screenbau durch dessen Architektur den Entwickler dazu viele Stile direkt auf einzelnen Komponenten zu setzten. | | | |
| Häufigkeit: | 3 | Kritikalität: | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: ERR\_006** | **CSS Selektorenhirarchie verwirrend** | | |
| Beschreibung: | CSS Selektoren werden falsch angewendet und führt dazu, dass Stile auf falsche Elemente appliziert werden. Die Verschachtelungsmöglichkeit der Selektoren sind für die Studio Anwender nicht nachvollziehbar. | | |
| Häufigkeit: | 2 | Kritikalität: | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: ERR\_007** | **ID Attribute** | | |
| Beschreibung: | Verwendung von ID Attributen im HTML für CSS ist nicht immer ratsam. ID Selektoren überwiegen normale Klassen Selektoren und können somit zu Problemen im Aufbau der Stile führen. | | |
| Häufigkeit: | 2 | Kritikalität: | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: ERR\_008** | **Veraltete HTML Elemente übersteuern CSS** | | |
| Beschreibung: | In HTML 4 konnte Text mit dem <center> Element zentriert werden. In HTML 5 sind <center> und ähnliche Elemente als „Depricated“ markiert und führen durch dessen Einsatz dazu, dass das Styling auf mehrere Ebenen verteilt wird. | | |
| Häufigkeit: | 2 | Kritikalität: | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: ERR\_009** | **Browser Standartstile beachten** | | |
| Beschreibung: | Jeder Browser bringt einen gewissen Satz an Standartstilen mit was zu unterschiedlichen Resultaten führen kann. | | |
| Häufigkeit: | 3 | Kritikalität: | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: ERR\_010** | **Shorthand CSS Darstellung fördern** | | |
| Beschreibung: | Es gibt einige Eigenschaften die im CSS Code durch eine kompakte Schreibweise ersetzt werden können. Das weglassen solcher Shorthand Schreibweisen kann zu unübersichtlichen und langen Stildefinitionen führen.  C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\example_shorthand.jpg  **BILD SHORTHAND EXAMPLE** | | |
| Häufigkeit: | 2 | Kritikalität: | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: ERR\_011** | **Relative Grösseneinheiten besser strukturieren** | | |
| Beschreibung: | Im CSS Code können diverse Einheiten verwendet werden um Grössen auszudrücken. Einige davon sind relativ zu bereits definierten Grössen. Bei komplexeren Strukturen führt dies leicht zu Unübersichtlichkeit.  C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\example_units.jpg  **BILD UNIT EXAMPLE** | | |
| Häufigkeit: | 2 | Kritikalität: | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: ERR\_012** | **Farbenwertdarstellung im Browser** | | |
| Beschreibung: | Farben können in CSS auch durch Namen ausgedrückt werden (red, blue, yellow…). Verschiedene Browser können diese Farbwerte unterschiedlich interpretieren und darstellen. | | |
| Häufigkeit: | 1 | Kritikalität: | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: ERR\_013** | **Fallback von neuen CSS Eigenschaften** | | |
| Beschreibung: | Gewisse Eigenschaften können auf älteren Browsern nicht korrekt dargestellt werden. Ohne einen Fallback zu implementieren werden ältere Browser nicht mehr unterstützt.  C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\example_fallback.jpg  **BILD FALLBACK EXAMPLE** | | |
| Häufigkeit: | 1 | Kritikalität: | 2 |

**6.2.3 Priorisierung**

Im Rahmen dieser Bachelorarbeit können nicht alle Punkte der Fehlerliste berücksichtigt und als Anforderung erfasst werden. Welche Punkte relevant sind wird anhand von folgender Priorisierungsmatrix entschieden. Listeneinträge im roten Bereich sind sehr kritisch und bekommen eine höhere Priorität im Gegensatz zu den Punkten im grünen Bereich.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Häufigkeit** | | | | |
| **Kritikalität** |  | **0** | **1** | **2** | **3** |
| **3** |  |  |  |  |
| **2** |  | **ERR\_013** | **ERR\_002**  **ERR\_006** | **ERR\_005** |
| **1** | ERR\_003  ERR\_007  ERR\_008 | ERR\_012 |  | **ERR\_001**  **ERR\_009** |
| **0** |  |  | ERR\_010  ERR\_011 | ERR\_004 |

Es wird versucht alle Punkte im orangen und gelben Bereich durch die Effizienzsteigerung zu lösen. Die auftretenden Fehler, welche jetzt im grünen Bereich sind werden nicht explizit als Anforderung erfasst können aber durch die Effizienzsteigerung ebenfalls gelöst werden.

**6.3 Anforderungskatalog**

Aus den ausgewählten Fehlern werden folgende Anforderungen definiert:

**TODO: Anforderungen in Tabellarischer Form niederschreiben!**

**ANF\_001:**

Der Lösungsansatz soll dem Entwickler die Möglichkeit bieten Werte global und einmalig zur Wiederverwertung definieren zu können.

Abhängigkeit: ERR\_001

Prio: 3

**ANF\_002:**

Der Lösungsansatz soll dem Entwickler das abdecken von browserspezifischen Eigenschaften automatisch abnehmen.

Abhängigkeit: ERR\_002

Prio: 2

**ANF\_003:**

Der Lösungsansatz soll dem Entwickler Hilfestellung beim Debuggen von CSS Code liefern.

Abhängigkeit: ERR\_005

Prio: 2

**ANF\_004:**

Der Lösungsansatz soll dem Entwickler helfen CSS Selektoren übersichtlich zu sortieren.

Abhängigkeit: ERR\_006

Prio: 3

**ANF\_005:**

Der Lösungsansatz soll die in den Browsern vorhandenen Standarteinstellungen egalisieren.

Abhängigkeit: ERR\_009

Prio: 2

**ANF\_006:**

Der Lösungsansatz soll für nicht vom Browser unterstütze CSS Eigenschaften einen Fallback zur Verfügung stellen.

Abhängigkeit: ERR\_013

Prio: 3

**ANF\_007:**

Der Lösungsansatz muss mit älteren Browsern, ab Internet Explorer 7, kompatibel sein.

Abhängigkeit: Appway Philosophie

Prio: 1

**ANF\_008:**

Der Lösungsansatz muss das Appway Front-End Design stützen und fördern.

Abhängigkeit: Appway Philosophie

Prio: 1

**7 Lösungsansätze**

CSS Code in einem separaten Stylesheet löst bereits viele Probleme. Es gibt jedoch einige Techniken und Tools, welche dafür gemacht sind, die Arbeit mit CSS effizienter zu gestalten. Dabei zielen diese auf unterschiedliche Elemente der Effizienzsteigerung. In den folgenden Kapiteln werden die Techniken und Tools vorgestellt.

**7.1 Techniken**

**7.1.1 Objektorientiertes CSS**

Objektorientiertes CSS oder auch OOCSS ist keine weitere Programmiersprache sondern vielmehr ein Paradigmenwechsel im Umgang mit CSS. Aus diesem Grund handelt es sich nicht um das gleiche Objektorientierte wie von JAVA bekannt ist, sondern vielmehr um ein paar Muster und bewährte Praktiken im Umgang mit CSS, welche wiederverwendet werden können. Dabei ist das Ziel den Code nach Modulen, Inhalt, Zuständen und weiteren Spezifischen Eigenschaften und Anwendungsfällen zu gliedern. In der Theorie sollen so Webapplikation einfacher skalieren, da der CSS Code flexibler auf Änderungen reagieren kann. Für die folgenden Beispiele und OOCSS Regeln haltet sich dieses Dokument an die Vorgaben von Jonathan Snook's Buch „SMACSS - Scalable and Modular Architecture for CSS“.

Beim erstellen einer Webseite wird beim OOCCS die Inhalt von der Struktur getrennt. Dies geschieht durch ein Basislayout aus verschiedenen Containern. In modernen Architekturen werden dafür meistens Gridsysteme verwendet. Diese helfen die Webseite in einheitliche Container zu teilen. Die Trennung dient auch dazu, dass jeglicher Inhalt in die Container gefüllt werden kann und dieser sich automatisch dem äusseren Layout anpasst. Eine Linkliste zum Beispiel soll auf der Übersichtseite dominanter und grösser angezeigt werden als in einem Artikel. Der Inhalt ist in beiden Fällen genau gleich, doch das aussehen wird durch den Container bestimmt. Dies fördert die Wiederverwendbarkeit solcher Inhaltsblöcke. Das Anwenden der objektorientierten Paradigmen hat auch Effekte auf das Stylesheet. Zum Einen werden diese viel kleiner und zum Anderen kann das Design schneller und einfacher gewechselt werden dank der strikten Trennung von Inhalt und Struktur.

In diesen Beispielausschnitten eines CSS wird ersichtlich welche Ausprägungen OOCSS hat.



BEISPIEL OOCSS!!!!

Das gebrauchen von Klassenselektoren erhöht die Wiederverwendbarkeit der Stile. Zu sehen ist auch dass weder Breiten- oder Höhenangaben für die Inhaltsstile verwendet werden, da diese zur Struktur gehören und nicht zum Inhalt. Die Klasse „!!!!!!!!!!!“ kann so an allen Orten zum Einsatz kommen an dem „!!!!!!!!!!!!!“ werden soll. Stiele sollten gemäss OOCSS auch auf überliegende Objekte oder Komponente angewendet werden und nicht direkt auf die Kinderelemente. Wird dies eingehalten, können später einzelne Elemente einfacher umgestaltet werden. Am nächsten Beispiel wird ersichtlich wie verschiedene Klassen für unterschiedliche Stile und Ausprägungen für einen Button verwendet werden.



BEISPIEL OOCCSS MANY CLASSES

Zusammengefasst wird durch ein Objektorientiertes CSS die Webseite schneller, wartbarer und das Stylesheet übersichtlicher. Jedoch bringen diese Vorteile nichts wenn nicht alle zusammen strukturiert und gemäss Vorlagen arbeiten.

**7.1.2 Styleguide 2.0**

Für Projekte oder Teams ist es durchaus üblich, dass Regeln für den Umgang mit Designfragen in einem Styleguide festgehalten werden. Appway AG arbeitet auch mit einem Styleguide, um den Code zu vereinheitlichen. Im Allgemeinen gibt es keine Vorschriften was in einem Styleguide stehen muss. Jedes Entwicklerteam hat andere Präferenzen und Anforderungen an ihren CSS Code. Aus diesem Grund gleicht auch selten ein Styleguide dem Anderen.

Das führen und nutzen solcher Richtlinien ist in grösseren Teams mehr anzutreffen, damit die dessen Koordination verbessert wird. Aber auch Einzelpersonen haben einen Styleguide. Dieser wird aber selten schriftlich niedergeschrieben sondern implizit verwendet. Dennoch arbeitet ein Entwickler nach gewissen Richtlinien und Erfolgsmethoden, die sich in der Vergangenheit bewährt haben.

Es können unterschiedliche Dinge in einem Styleguide geregelt werden. Zum Beispiel wie die Syntax aussehen soll, Farben definiert werden oder was vermieden werden soll.



BEISPIEL STYLEGUIDE

Ein Sytleguide ist also eine Sammlung aus Regeln, Richtlinien und Erfolgsmethoden, welche helfen den CSS Code übersichtlicher zu gestalten, Konsistenz zu wahren oder Fehler im vornherein zu vermeiden. Wie auch beim OCCSS gilt, dass jeder Entwickler verantwortlich ist sich an den Styleguide zu halten, ansonsten gehen die Stile auseinander und der Nutzen geht verloren.

**7.2 Tools**

**7.2.1 CSS-Präprozessor**

Ein CSS-Präprocessor interpretiert Code (in einer eigenen Syntax) und generiert daraus gültigen CSS-Code. Auf dem Markt gibt es mittlerweile unzählige Präprozessoren, welche auf unterschiedlichen Technologien basieren. Zu den gängigsten gehören SASS, LESS, Stylus und Turbine. SASS basiert auf der Programmiersprache Ruby, LESS und Stylus sind in JavaScript geschrieben und Turbine läuft mit PHP. Alle dieser Präprozessoren haben einen ähnlichen Syntax, welcher am normalen CSS Syntax angelehnt ist, jedoch handelt es sich eher um einen programmatischer Entwicklungsprozess.

Ein Präprozessor bringt wesentliche Erleichterungen für die Entwickler wie den Einsatz von Variablen, Dynamische Grössenberechnung, Funktionen und Verschachtlungen. All diese Elemente helfen bei grosssen Projekten die Fehler zu reduzieren. Aber zuvor muss der Präprozessor installiert und die neue Syntax erlernt werden.

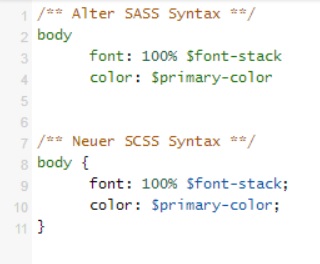




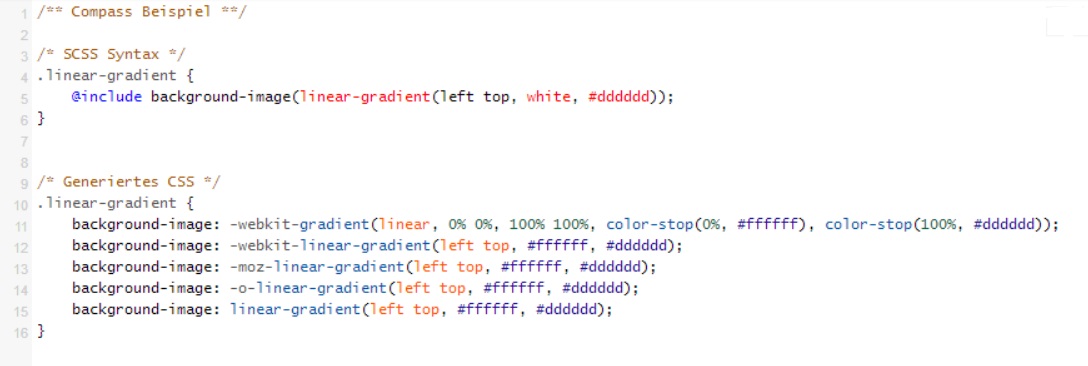
BEISPIEL SCSS zu CSS

Für die Bachelorarbeit wird SASS (Syntactically Awesome Style Sheets) als Repräsentant für die Präprozessor verwendet. Einerseits bietet SASS den grössten Funktionsumfang und andererseits wird SASS, dank der riesigen Benutzergemeinde, stetig weiterentwickelt. Mit SASS kann man zusätzlich Compass verwendet. Compass ist ein Authoring Framework, welches viele vorgefertigte Funktionen liefert, um CSS3-Eigenschaften oder Sprites (Computergrafik) einfach zu nutzen. Der Nutzen von SASS wird durch dieses ergänzende Framework nochmals erheblich gesteigert.

SASS wurde über die Zeit weiterentwickelt woraus SCSS, ein neuer Syntax, entstanden ist. Der Vorteil von SCSS ist, dass Punktation und geschweifte Klammern verwendet werden, was die Lesbarkeit erhöht und die Fehleranfälligkeit mindert.



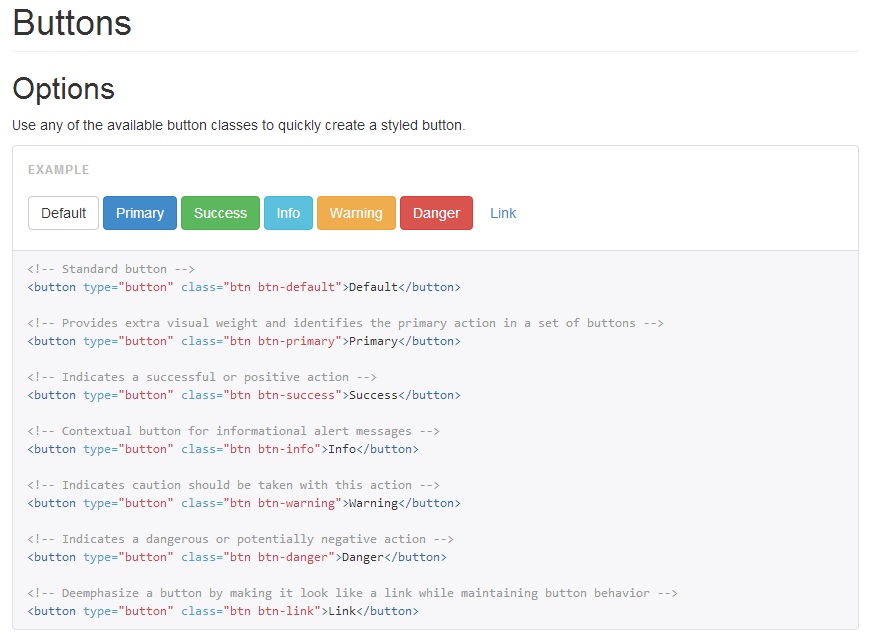
BEISPIEL SASS VS SCSS



BEISPIEL COMPASS

**7.2.2 Framework**

Um den Umgang mit CSS zu vereinfachen gibt es auch die Möglichkeit ein CSS Framework zu verwenden. Hierbei liefern die Frameworks ausgereifte CSS Strukturen und Vorlagen, welche mit simpler Klassenzuweisung im HTML genutzt werden können. Mit dieser deklarativen Vorgehensweise können komplexere Layoutstrukturen in kürze aufgebaut werden. Während sich zum Beispiel das Foundation Framework vor allem auf responsive, automatisch sich dem Platz anpassende, Gridsysteme spezialisiert, bietet Pure CSS ein Set von HTML Grundelementen an. Der weitaus bekannteste Repräsentant aus der Gruppe Framework ist Bootstrap von Twitter. Bootstrap bietet ein Umfangreiches Set an allen gängigen HTML Elementen, ein eigenes Gridsystem sowie interaktive Elemente mit eigener JavaScript Bibliothek.

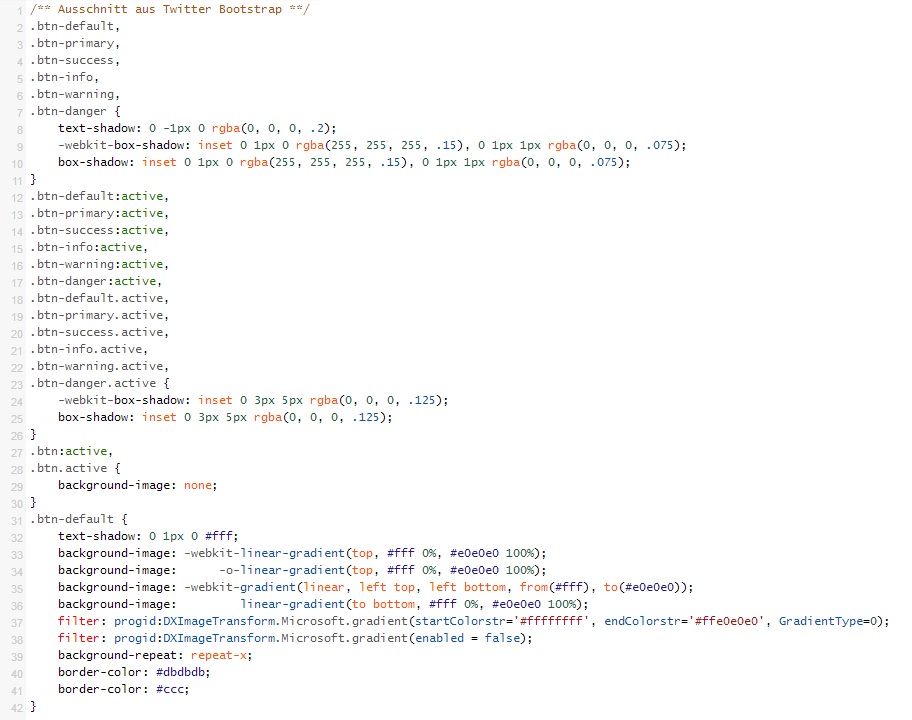
****

**BEISPIEL DEFAULT BOOTSTRAP BUTTONS**

http://getbootstrap.com/css/#buttons

Im Oberen Beispiel ist dargestellt wie bei Bootstrap verschiedene Buttons, anhand von einfachen Klassenzuweisungen, designt werden. Dies zeigt aber nur ein kleines Spektrum der Möglichkeiten dieses Frameworks. Dank dieses deklarativen Designprozesses können Webseiten in Windeseile erstellt werden.

Ein detaillierterer Einblick in den Quellcode von Bootstrap zeigt ein für seine Grösse gut strukturiertes CSS. Die Entwickler des Frameworks folgen vielen OOCSS Prinzipien und bestätigen somit wie Wertvoll gut strukturiertes CSS Code ist. Bootstrap nutzt aber nicht nur OOCSS für die Entwicklung sondern bietet die Unterstützung eines Präprozessors wie LESS oder SASS.



**BILD BOOTSTRAP CODE EXAMPLE**

<https://github.com/twbs/bootstrap/blob/master/dist/css/bootstrap-theme.css>

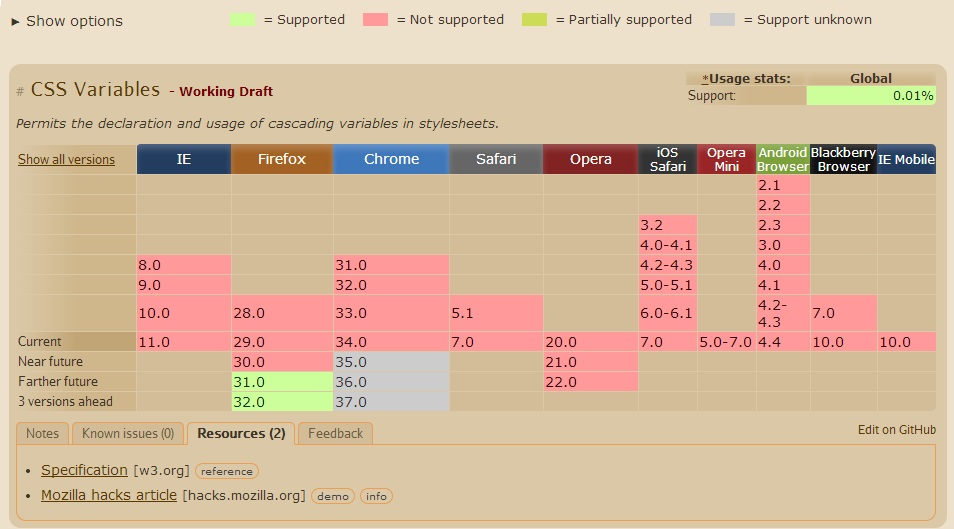
Die grosse Benutzergemeinde wie auch die stetige Weiterentwicklung sind Grund für den grossen Erfolg im Internet. Dieses Framewokr ist kostenlos und steht unter der MIT-Lizenz (Massachusetts Institute of Technology), was für die Wiederverwendung ideal wäre in Bezug auf Appway AG. Aber auch ein solch durchdachtes Framework kommt mit ein paar negativen Punkten. Das so gut miteinander abgestimmte Grunddesign von Bootstrap ist durch die riesige Bibliothek nur schwer anzupassen. Aus diesem Grund wird eine Webseite, die Twitter Bootsrtap verwendet, auch auf der Stelle als solche erkannt. Dies muss nicht negativ sein, verhindert aber einen individuellen Designcharakter auszustrahlen.

**7.2.3 CSS 4.0**

Der CSS Sprachsyntax ist in den letzten Jahren stark gewachsen und wird auch in Zukunft immer wichtiger werden um Webseiten bequem und Browserübergreifend zu erstellen. Das World Wide Web Consortium kurz W3C ist ein Gremium zur Standardisierung aller das Web betreffenden Technologien. CSS gehört als Teil des Webs ebenfalls in Ihren Zustellungsbereich. CSS3 (Cascading Style Sheets – Level 3) wird seit dem Jahr 2000 von W3C entwickelt. Viele der Funktionen von CSS3 sind in heutigen Browsern als Standard verbaut und können verwendet werden. Seit 2012 arbeitet W3C am Nachfolger CSS Level 4.

Die neue CSS Version soll unter anderem Variablen, neue Selektoren, inline Scripts und Mobile Geräte besser unterstützen. Der Aspekt Variablen nutzen zu können, was bisher nur durch den Einsatz von Präprozessoren erreicht wird, wird einen Umschwung in der Webgemeinde herbeiführen.

Das Problem mit Standards im Allgemeinen ist, dass diese sehr lange Vorlaufzeiten haben während individuelle Tools sich viel schneller auf neue Funktionen einstellen können. Variablen werden zurzeit noch in keinem Browser unterstützt und auch nur Mozilla verweist darauf diese Später verwende zu wollen.



**BILD CAN I USE VARIABLES**

Das warten bis neue Technologien genügend verbreitet sind, um diese Produktiv zu nutzen, dauert lange. Und wenn die neuen Funktionen unterstützt werden ist die Rückwärtskompatibilität für ältere Browser noch nicht sicher gestellt.

**8 Vergleich der Lösungen**

Bevor die verschiedenen Lösungsansätze miteinander verglichen werden können benötigen wir alle Kriterien und Anforderungen, welche der Lösungsansatz erfüllen soll und muss. Die Anforderungen mit einer Priorität von 1 sind als zwingend anzusehen und gelten nachfolgend als KO-Kriterien. Alle weiteren Anforderungen, welche durch den Fehlerkatalog ermittelt wurden, gelten als SOLL-Kriterien. Der Lösungsansatz soll möglichst viele dieser Kriterien abdecken oder erfüllen können.

**8.1 KO-Kriterien**

Für die Lösungsauswahl gibt es zwei KO-Kriterien. Zum einen haben wir die Anforderung ANF\_007 mit welcher die Kompatibilität sichergestellt wird. Und zum anderen die ANF\_008, welche das Design des Appway Produktes schützt.

|  |  |  |
| --- | --- | --- |
|  | **KO-Kriterium** | |
| **Kompatibilität (ANF\_007)** | **DESIGN (ANF\_008)** |
| OOCSS | Check | Check |
| Styleguide 2.0 | Check | Check |
| CSS 4.0 | X | Check |
| SASS | Check | Check |
| Bootstrap | Check | X |

Für CSS 4.0 ist noch offen, ab wann die Funktionen Browserübergeifend unterstützt werden. Zudem ist sicher, dass ältere Browser die neuen Funktionen nicht implementieren und somit die Kompatibilität nicht gewährleisten werden kann. Deshalb fliegt CSS 4.0 als Lösungsvariante aus der obigen Aufstellung raus.

Die Designanforderung kann Bootstrap trotz seiner Komplexität nicht erfüllen. Anpassungen in diesem Framework sind zu Umständlich und führen dazu, dass Updates von der Bootstrap Bibliothek zukünftig verwehrt bleiben. Das Appway Studio enthält über 70 verschiedene Komponenten, um das Layout sowie das Aussehen einer Webseite zu gestalten. Die mit Bootstrap mitgelieferten CSS Stile und Elemente könnten gerade einmal 26 dieser Komponenten ersetzten. Dies bedeutet, dass mehr als die Hälfte aller vorhandenen Komponenten nicht von der Bootstrap Bibliothek profitieren können.

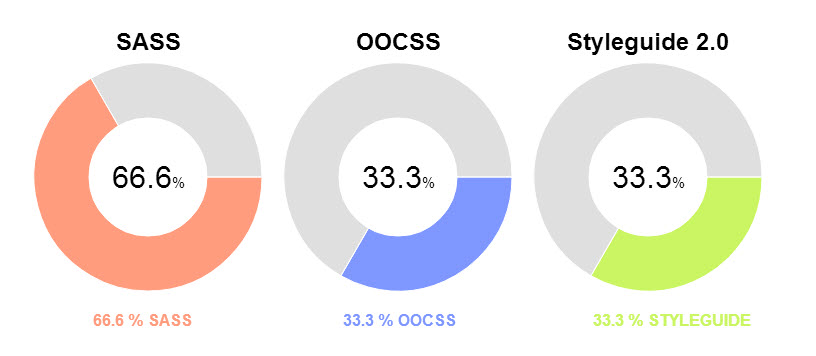
Für die kompatiblen Komponenten wie zum Beispiel ein HTML Button würde die Anpassung vom Bootstrap zum Appway Stil und das Testen circa 1 Arbeitstag dauern. Dieser Aufwand bei jedem Bootstrap Update und für alle Komponenten durchzuführen wäre kostentechnisch nicht tragbar.

Ein Vorteil von Bootstrap ist sein integriertes Grid, welches dazu verwendet wird die einzelnen Elemente auf dem Bildschirm auszurichten. Dieser Zusatz, denn Bootstrap für so viele Weblösungen so wertvoll macht, divergiert aber mit der Appway Architektur. Die einzelnen Screen Komponenten von Appway sind nicht zwingend auf fixe Rastergrössen ausgelegt und deswegen lässt es keine solchen Gridsysteme zu. Somit wird Bootstrap für den Lösungsentscheid nicht mehr in Betracht gezogen.

Somit werden für den Lösungsvergleich nur noch die Übrigen Lösungsansätze angeschaut und bewertet.

**8.2 SOLL-Anforderungen**

Für die SOLL-Anforderungen aus dem Fehlerkatalog wurden die Fehler entgegengestellt und aufgelistet welche Lösungsvariante wie viele der Anforderungen abdecken kann.



**DIAGRAMM FEHLERABDECKUNG 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Anforderungen** | | | | | |
| **001** | **002** | **003** | **004** | **005** | **006** |
| **SASS** | X | X | X |  |  | X |
| **OOCSS** |  |  | X | X |  |  |
| **Styleguide 2.0** |  |  |  | X | X |  |

**ANF\_001 – Wiederholung von Farbendefinitionen**

Die Wiederverwendung von statischen Werten kann im Moment nur durch den Einsatz eines Tools wie der SASS Präprozessor erreicht werden.

**ANF\_002 – Browserspezifische CSS Eigenschaften**

Ein Styleguide oder ein Modularer CSS Aufbau können nur bedingt dafür sorgen, dass dieser Fehler nicht auftritt. Während SASS diesen Fehler, durch den Einsatz von Mixins, endgültig eliminieren kann.

**ANF\_003 – Debuggen von Stylesheets erschwert**

Debuggen von CSS fällt generell schwer aus. Die modernen Browser bieten alle mittlerweile gute Entwicklungstools an, um Herkunft und auch Ausprägungen von CSS Stilen nachzuvollziehen. Je besser das Stylesheet gegliedert und strukturiert ist, desto besser fällt die Fehlersuche mit diesen Tools aus. Aus diesem Grund erfüllt SASS und OOCSS mit jeweils fixen Strukturvorgaben diese Anforderung. Für SASS gibt es zusätzlich noch ein Chrome Entwicklungstool, was mit Hilfe von „CSS Source Maps“ \* das präzise debuggen von SASS Dateien erlaubt.

**ANF\_004 – CSS Selektorenhirarchie verwirrend**

Verwirrende Selektorhirarchien kann durch einen sauberen Programmierstil verhindert werden. OOCSS schreibt eine saubere Selektorennutzung vor und in einem Styleguide kann dies einfach definiert werden.

**ANF\_005 – Browser Standartstile beachten**

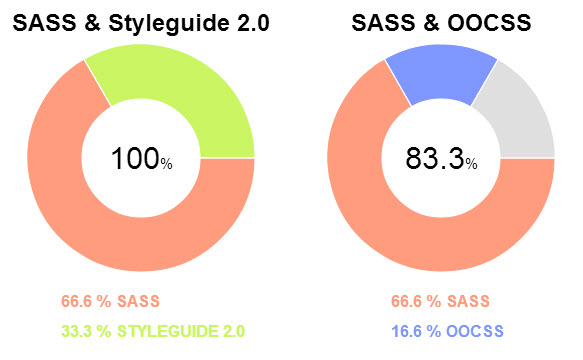
Dieser Fehler kann durch ein CSS Reset Hack behoben werden. Dafür wird ein solcher Reset definiert und aktuell gehalten. Der Styleguide ist dabei der ideale Ort für eine solche Definition.

**ANF\_006 – Fallback von neuen CSS Eigenschaften**

Ein Fallback kann ebenso wie die browserspezifischen CSS Eigenschaften über Mixins vom SASS Präprozessor gelöst werden.

**8.3 Lösungsentscheid**

Im Vorherigen Kapitel zeigte sich, dass der SASS Präprozessor zwei Drittel aller Fehler abdecken kann und somit als Sieger hervorgeht. Die Tabelle zeigt aber auch, dass Fehler, welcher der Präprozessor abdeckt nicht dieselben Fehler sind, die mit OOCSS oder dem Styleguide 2.0 abgedeckt werden. Somit ist SASS zu den anderen Techniken divergierend. Es wäre in diesem Fall naheliegend eine Kombination von mehreren Lösungsansätzen anzuschauen, um noch mehr Anforderungen erfüllen zu können. Durch eine sinnvolle Kombination aus Technik und Tool wird die Abdeckung erhöht.



**DIAGRAMM FEHLERABDECKUNG 2**

Mit einer Kombination von SASS und einem Styleguide können alle vorhanden Anforderungen erfüllt werden. Die beiden Lösungsansätze stehen in keinem direkten Einfluss zueinander, was die Nutzung beider Varianten erlauben würde. Auf den ersten Blick könnte OOCSS auch gleich mit in die Lösungsvariante hineingenommen werden, um die CSS Struktur zu modularisieren. Hierbei würde aber der Styleguide 2.0 und die OOCSS Techniken einander in den Weg kommen und im schlimmsten Fall einander durch eine unterschiedliche Strukturvorgabe konkurrieren. Somit steht eine kombinierte Lösung fest, die weder ein KO-Kriterium verletzt und alle Anforderungen erfüllt.

**9 Umsetzung / Proof of Concept**

Für die Umsetzung beziehungsweise das „Proof of Concept“ werden zwei Teile benötigt. Auf der einen Seite muss ein Tool, der SASS Präprozessor, und auf der anderen Seite eine Technik, der Styleguide, umgesetzt werden. In den nachfolgenden Kapiteln sind die einzelnen Schritte der Vorgehensweise zu finden.

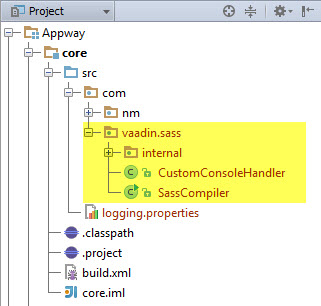
**9.1 Vorgehensweise SASS**

Der standardmässige SASS Präprozessor läuft nur mit der Programmiersprache Ruby. Die Appway Software läuft aber in einem Java Umfeld und ist somit nicht direkt miteinander kompatibel. Die erste Idee des Studenten war Ruby im Java Kontext zu installieren oder zugänglich zu machen. Mit Hilfe von JRuby, eines Ruby Interpreters für Java könnte SASS installiert werden. Jedoch müsste für die Laufzeitumsetzung explizit Ruby Code geschrieben und in einem speziellen Scriptcontainer ausgeführt werden. Diese zwei unterschiedlichen Programmiersprachen so zu durchmischen ist für Appway durch einen möglichen Kontrollverlust ungeeignet und kann daher nicht als Lösung in Betracht gezogen werden.

Wie bereits erwähnt läuft Appway mit Java und daher wäre es ideal, wenn es einen Java Parser gäbe, der die SASS Dateien genau gleich wie der Ruby SASS Präprozessor umschreiben kann. Im Netz gibt es nur wenige Java SASS Parser Implementationen und dazu sind die meisten entweder nicht komplett im Bezug auf Funktionalität oder weisen eine niedrige Codequalität auf. Der einzige komplette und sauber programmierte Java Parser ist im Vaadin Framework zu finden. Vaadin ist ein umfangreiches und komplexes in Java geschriebenes Webframework, welches einen kompletten SASS Parser implementiert hat. Der Einsatz des gesamten Frameworks für Appway steht ausser Frage, aber ein Vaadinentwickler namens Arthur hat sich die Mühe gemacht den Parser aus dem Framework herauszulösen und als eigenständige Bibliothek zur Verfügung zu stellen. Vaadin verwendet zudem die Apache-Lizenz, welches Änderungen und kommerzielle Nutzungen im jedem Umfeld erlaubt. Mit diesem in Java programmierten SASS Parser sollte der Einbindung ins Appway Produkt nichts mehr im Wege stehen.

**9.1.1 Umsetzung - SASS Compiler**

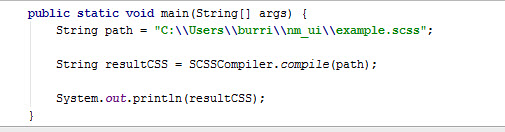
Als erstes wurde das Projekt von Arthur von GitHub heruntergeladen und die entsprechenden Java Dateien in die Appway Dateistruktur gelegt.



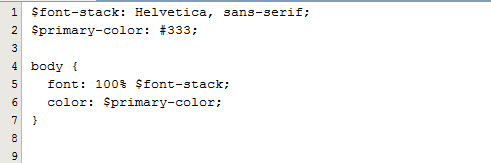
**BILD DATEISYSTEM STRUKTUR**

Die Entwicklungsumgebung IntelliJ zeigt auch gleich an, dass gewisse Bibliotheken fehlen um den die SASS Parser Java Dateien kompilieren zu können. Dabei handelt es sich um SAC und Flute Java Biblotheken. Diese werden von der W3C zur Verfügung gestellt und bieten ein Standard Interface für CSS Parser an. Eine weitere benötigte Bibliothek ist JEXL (Java Expression Language) von Apache Commons. Sie erleichtern das dynamische ausführen von Scriptfunktionen in Java. Diese Bibliotheken sind ebenfalls mit freier Lizenz versehen und können ohne weiteren Probleme in Appway integriert werden.

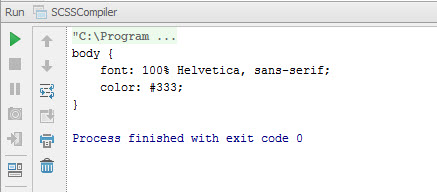
Für einen ersten Test wurde eine eigene SCSSCompiler Java Klasse geschrieben, welche einen Pfad zu einer SASS Datei entgegennimmt. Dieser Pfad wird danach einer weiteren Klasse, dem AppwaySCSSResolver, übergeben, der sich darum kümmert, die Datei einzulesen und als InputSource Objekt dem SCSSCompiler zurückzugeben. Der Compiler ruft, dann auf der InputSource den SASS Parser auf und dessen Resultat, das umgewandelte Stylesheet, wird als Code in der Konsole ausgegeben.



**BILD TEST PARSER**

****

**BILD vorher**

****

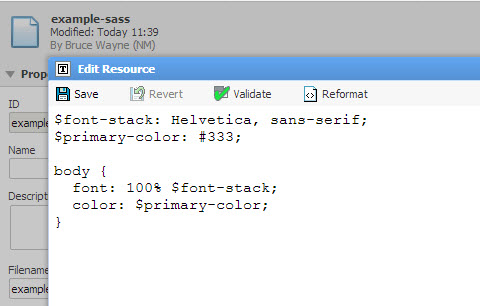
**BILD KONSOLE nachher**

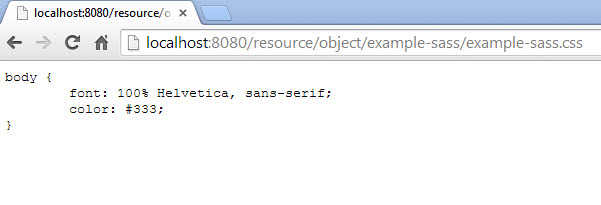
Als dies funktionierte ist die Klasse so angepasst worden, dass eine .scss Datei eingelesen und einen neue .css Datei im gleichen Verzeichnis erstellt wird. Der SASS Parser funktioniert und kann mit ein paar weiteren kleinen Anpassungen in einen Build-Vorgang integriert werden. Mit dieser Änderung könnten alle Appway Entwickler in Zukunft die SASS Syntax für CSS Programmierung verwenden.

**9.1.2 Umsetzung - Laufzeit Compiler**

Anhand des bisherigen SASS Compilers können Appway Studio Anwender noch nicht auf die Funktionen vom Parser zurückgreifen. Im Appway Studio können eigene Ressourcen, dazu gehören auch CSS Stylesheets, erstellt und bearbeitet und im Nachhinein in die Appway Screens eingebaut werden. Damit dies gelingt werden bei jedem Request, den ein Appway Screen macht, alle SASS Dateien erkannt und zur Laufzeit in CSS umgewandelt.

Einen bestehenden Request abzufangen und zu bearbeiten kann über einen Servlet Filter realisiert werden. Wie bereits im Kapitel „IST Analyse“ gezeigt werden bereits gewisse Begriffe wie der CONTEXT\_PATH zur Laufzeit ersetzt. Diesen ContextPathFilter wird als Vorlage genommen und ein neuer Filter, die SCSSFilter Java Klasse erstellt. Für den Proof of Concept wird der Servlet Filter so konfiguriert, dass er nur Dateien berücksichtigt, die mit „-sass.css“ enden. Der Filter ruft nach erkannter Datei den bereits programmierten Parser auf und schreibt in die HTTP Response das resultierende CSS. Der Studio Anwender hat in diesem Fall nur eine einzige Datei in welche mit der SCSS Syntax Code geschrieben wird und im Browser wird aber letztendlich nur noch reines CSS angezeigt. Ein wichtiger Vorteil für die Studio Anwender ist, dass die SASS Datei auch in der Vorschau eines Screens korrekt geparsed wird.





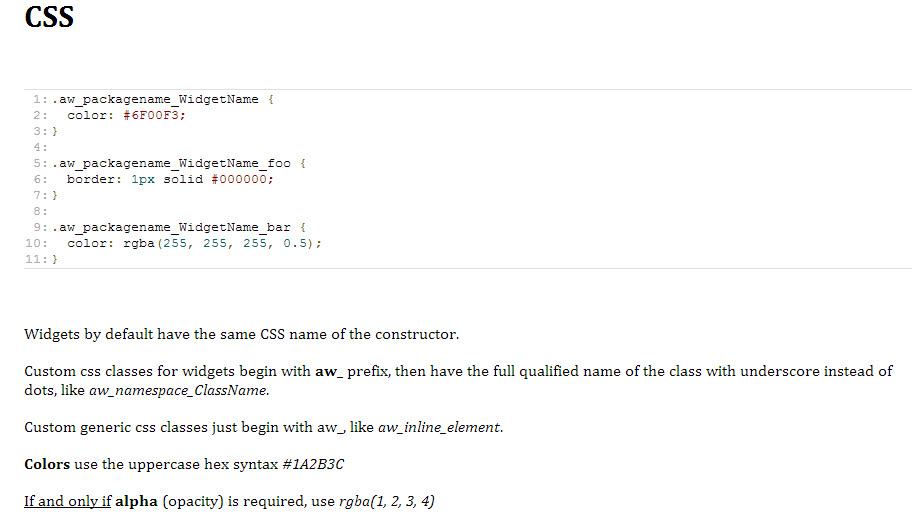
**BILD LAUFZEIT Vorher / Nachher**

**9.1.3 Einschränkungen**

Die Endresultat des Java SASS Parsers ist sehr zufriedenstellen. Es gibt jedoch einzelne Punkte die nicht oder im Rahmen des Proof of Concepts nicht umgesetzt werden. Zum Einen handelt es sich um die Nutzung des Compass Frameworks. Diese umfangreiche Bibliothek fällt durch das weglassen einer Ruby Programmierumgebung weg. Vaadin ist sich das fehlen von einer Compass Integration bewusst und arbeitet daran\*. Für diese Arbeit wird dies aber nicht mehr genutzt werden können. Weiter ist die Eigenschaft SASS Dateien in andere SASS Dateien einzubinden nicht umgesetzt. Der Java Parser bietet zwar diese Funktionalität ebenfalls an, aber die Integration mit Appway würde den Rahmen dieser Arbeit sprengen und wurde deshalb hier weggelassen.

**9.2 Vorgehensweise Stylguide**

Appway Entwickler haben bereits angefangen einen kleinen Styleguide für Front-End Konventionen zu schreiben in dem auch der Umgang mit CSS festgelegt ist. Dies umfasst aber nur eine kleine Sammlung mit Regeln.



**BILD STYLEGUIDE WIKI**

Im Bild ist zu sehen, dass hauptsächlich die Benennung von CSS Klassen oder die Schreibweise von Farben definiert sind. Mit den Ergänzungen im neuen Styleguide sollen vor allem die noch offenen Anforderungen abgedeckt werden können.

**9.2.1 Umsetzung - Stylguide 2.0**

Im neuen Styleguide 2.0 wurde als erstes ein CSS Reset in Angriff genommen. Das Ziel ist die von den Browsern verwendeten Standardstile zurück zusetzten damit die darauf folgenden CSS Regeln auf allen Browsern die gleiche visuellen Effekte erzeugen. Obwohl es diverse CSS Resets gibt, ist eines davon auf fast allen Webseiten wiederzufinden. Dabei handelt es sich um denn CSS Reset von Eric Meyer, welcher eine kleine und solide Basis aufweist. Eric Meyer aber sagt explizit, dass dies nicht ein fertiger CSS Reset ist und dieser auf die eigenen Bedürfnisse angepasst werden sollte. Aus der Sicht von Appway AG sollten natürlich bereits bestehende Lösungen nicht gleich auseinanderfallen, wenn der neue CSS Reset eingeführt wird. Deswegen wurden einige Modifikationen unternommen und führen letztendlich zu untenstehenden Code. Im Sytelguide wird dieser CSS Reset ebenfalls eingetragen und entsprechend kommentiert.

**TODO: BILD / CODE CSS RESET**

Die Anforderung die Selektorhirarchie zu strukturieren ist weder messbar noch ein Punkt der als Abgeschlossen angesehen werden kann. Die Stylesheets sollten immer einen logischen Ablauf wiedergeben und für die Leser Sinn ergeben. Der Styleguide wurde um mehr als die Hälfte durch den Input dieser Bachelorarbeit vergrössert. Viele Richtlinien sind dabei dazugekommen und sollen dafür sorgen, dass die Fehler in Zukunft nicht mehr auftreten sollen. Folgend ist ein paar wichtige Punkte die auch im Styleguide 2.0 zu finden sind.

Eine Regel ist zum Beispiel, dass die Stile dir Reihenfolge des HTML Dokuments gleichen. Das heisst, wenn im HTML eine Tabelle zu sehen ist, hat sie Zeilenelemente und die wiederum Zellenelemente als Kinder. Im CSS sollte daher auch zuerst der Stil für die Tabelle definiert werden und danach erst die der Zeilen und der Zellen. Diese logische Abfolge hilft auch damit, dass Stile nicht ungewollt überschrieben werden.

Weiter sind Statusänderungen wie Ein- oder Ausblenden von Inhalt separat im Stylesheet zu führen. Ebenfalls wird ersichtlich welche Änderungen an bestehenden Stiles gemacht werden kann und dadurch wird die Kontrolle über alle Statusänderungen beibehalten.

Es gibt viele Regeln, die helfen die Struktur eines Stylesheets zu verbessern. Das Endresultat findet sich im finalen angehängten Styleguide 2.0.

**9.3. Nicht relevante Fehler**

Im Fehlerkatalog sind natürlich noch weitere Fehler zu finden, welche aber durch geringe Kritikalität oder Häufigkeit als nicht relevant eingestuft wurden. Es gibt aber gewisse Fehler die automatisch durch die neue Lösungsvariante behoben werden.

**ERR\_004 - Redundanz im CSS Code**

Stylesheets welche mit der SCSS Syntax geschrieben werden haben durch die Verschachtelung von Selektoren eine bessere Struktur und verhindern somit die viele Codeduplikationen.

**ERR\_010 - Shorthand CSS Darstellung fördern**

Dieser Fehler kann durch eine weitere Regel im Styleguide 2.0 verhindert werden.

**ERR\_012 - Farbenwertdarstellung im Browser**

Dieser Fehler kann durch eine weitere Regel im Styleguide 2.0 verhindert werden.

**9.4 Evaluation der Effizienzsteigerung**

Die effektive Messung der Effizienzsteigerung kann erst ab dem Zeitpunkt der Einführung von SASS und dem Styleguide 2.0 bei Appway AG erfolgen. Im Voraus kann aber berechnet werden wie viel Zeit oder Aufwand durch die Fehlerabdeckung eingespart werden kann. Jeder Fehler aus dem Fehlerkatalog besitzt einen Wert Häufigkeit und Kritikalität. Aus der Kritikalität lässt sich der Zeitaufwand berechnen den ein Entwickler zur Behebung des Fehlers durchschnittlich benötigt. Hierbei wird die Zeit für das Auffinden, Beheben sowie das Testen des Fehlers gezählt.

|  |  |
| --- | --- |
| **Kritikalität** | **Zeit** |
| 0 | Nicht relevant |
| 1 | Ca. 30 Minuten |
| 2 | Ca. 4 Stunden (halber Tag) |
| 3 | Ca. 8 Stunden (ganzer Tag) |

Die behandelten Fehler aus dem Fehlerkatalog kommen alle sehr häufig vor. Drei der Fehler tauchen mehrmals in der Woche und 3 weitere im Monatsrhythmus auf. Die Berechnung der Dauer für die Fehlerbehebung sieht folgendermassen aus:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fehler** | **Häufigkeit** | **Kritikalität** | **Ø Zeitaufwand** | **Ø Min / Tag** |
| ERR\_001 | 3 | 1 | 30 Minuten pro Woche | 4,286 |
| ERR\_002 | 2 | 2 | 4 Stunden pro Monat | 4,286 |
| ERR\_003 | 0 | 1 | Nie aufgetreten | 0 |
| ERR\_004 | 3 | 0 | Nicht relevant | 0 |
| ERR\_005 | 3 | 2 | 4 Stunden pro Woche | 34,286 |
| ERR\_006 | 2 | 2 | 4 Stunden pro Monat | 4,286 |
| ERR\_007 | 0 | 1 | Nie aufgetreten | 0 |
| ERR\_008 | 0 | 1 | Nie aufgetreten | 0 |
| ERR\_009 | 3 | 1 | 30 Minuten pro Woche | 4,286 |
| ERR\_010 | 2 | 0 | Nicht relevant | 0 |
| ERR\_011 | 2 | 0 | Nicht relevant | 0 |
| ERR\_012 | 1 | 1 | 1 Stunde pro Jahr | 0,179 |
| ERR\_013 | 1 | 2 | 8 Stunden pro Jahr | 1,429 |

Summe der Zeit pro Tag über alle Fehler = tTotal

tTotal= 34,286 + (4 x 4,286) + 1,429 + 0,179 = **53,038** Minuten

Eingesparte Zeit pro Tag durch Lösungsvariante = tAbgedeckt

tAbgedeckt = (0,5 x 34,286) + (4 x 4,286) + 1,429 = **35,716** Minuten

Bei der eingesparten Zeit wird nur die Hälfte von Fehler ERR\_005 genommen, da die Zeit für das Debuggen nie 0 wird sondern lediglich reduziert werden kann.

tNeu = tTotal - tAbgedeckt = 53,038 – 35,716 = **17.322** Minuten

**TODO: FORMELN SCHÖNER MACHEN**

Durchschnittlich benötigt daher ein Appway Entwickler 53 Minuten pro Tag um CSS Fehler zu beheben. Durch die Nutzung von SASS und dem Styleguide 2.0 können 35 Minuten pro Tag an Zeit eingespart werden. Die Effizienzsteigerung verringert also den Täglichen Aufwand eines Appway Entwicklers von 53 auf 17 Minuten. Bei einem Front-End Team mit einer Grösse von 5 Entwicklern macht dies hochgerechnet im Monat einen Gewinn von 10 Arbeitstagen aus was ein durchaus beachtliches Resultat ist.

**10 Testing**

Im Rahmen dieser Arbeit wurden nur Tests für das Proof of Concept durchgeführt. Dabei wurde in erster Linie der SASS Präprozessor getestet. Der Parser muss die SCSS Syntax korrekt und gemäss SASS Dokumentation umwandeln. Für diesen Test wurden die Codebeispiele von der SASS Webseite genommen und geprüft ob das Ergebnis mit dem eingebauten Java Parser übereinstimmt.

**TODO: BILD DOKU SASS**

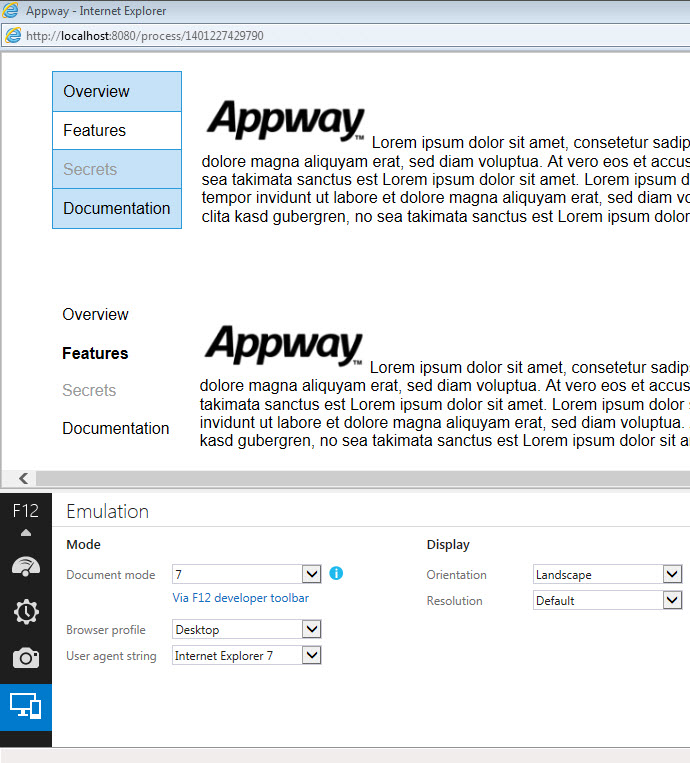
Bis auf die noch nicht unterstützte Import Funktionalität wurden alle Beispiele erfolgreich getestet.

**10.1 Akzeptanztests**

Bei Akzeptanztests werden alle vorhandenen Anforderungen auf Ihre Erfüllung geprüft. Wird das Akzeptanzkriterium einer Anforderung erfüllt gilt diese als erledigt.

|  |  |  |
| --- | --- | --- |
| Anforderung | Akzeptanzkriterium | Erfüllt |
| ANF\_001 | Werte können in Variablen gespeichert und im Stylesheet wiederverwendet werden. | **C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\greencheck.jpg** |
| ANF\_002 | Fehlende Browserpräfixe werden automatisch ergänzt. | **C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\greencheck.jpg** |
| ANF\_003 | Der Entwickler wird beim Debuggen von CSS Code unterstützt. | **C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\greencheck.jpg** |
| ANF\_004 | Die CSS Selektoren sind übersichtlich sortiert. | **C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\greencheck.jpg** |
| ANF\_005 | Ein Clearfix / CSS Reset wird angewendet. | **C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\greencheck.jpg** |
| ANF\_006 | Es gibt mögliche Fallbacks für ältere Browser. | **C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\greencheck.jpg** |
| ANF\_007 | CSS Funktionen sind auf Internet Explorer 7 und neuer lauffähig. | **C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\greencheck.jpg** |
| ANF\_008 | Das Appway Design ist nach der Implementation der Lösung nicht verändert. | **C:\Users\burri\elburro\zhaw\Bachelorarbeit\images\greencheck.jpg** |

**TODO: CODE : ANF: 1, 2, 4 im Stylesheet (NUR AUSSCHNITT)**



**BILD: IE7 Anforderung**

**11 Konzept Roll-Out**

Der Proof of Concept war erfolgreich und somit ist bewiesen, dass die Lösungsvariante für Appway AG produktiv umgesetzt werden kann. Das Roll-Out des neuen SASS Präprozessors und des neuen Styleguides wird nach Beendigung der Bachelorarbeit, ab August, in Angriff genommen und eingeführt. Bevor aber die Umstellung stattfinden kann müssen noch weitere Punkte berücksichtigt werden.

**11.1 Auswirkungen auf Stakeholder**

Ein Roll-Out hat Einfluss auf jeden einzelnen Stakeholder dieser Arbeit. Aus diesem Grund ist es wichtig zu wissen welche Rolle die einzelnen Stakeholder einnehmen werden. Einige Stakeholder werden laufen über den Fortschritt benachrichtigt und Andere sind wiederum aktiv am Roll-Out beteiligt.

|  |  |
| --- | --- |
| **Stakeholder** | **Rolle beim Roll-Out** |
| Ramón Burri (Student) | Leitet das Roll-Out über den ganzen Zeitraum und bereitet allfällige Schulungsunterlagen vor und führt diese durch. |
| Entwicklerteam (Appway AG) | Wird 1:1 von Ramón Burri im Umgang mit SASS und dem neuen Styleguide geschult. |
| CTO, Oliver Brupbacher (Appway AG) | Überwacht den Roll-Out und wird über kritische Punkte informiert. |
| Studio Anwender | Werden keine spürbare Änderung miterleben. Haben die Möglichkeit eine SASS Dokumentation vom Appway Developer Portal zu erhalten. |
| Beat Seelinger  Experte  ZHAW | Alle Stakeholder von der ZHAW werden nach Abschluss der Bachelorarbeit nicht mehr involviert sein. Weitere Zusammenarbeit ist auf freiwilliger Basis. |

**11.2 Risikoanalyse**

Jedes Projekt trägt gewisse Risiken die abgewägt werden müssen. Das Roll-Out für Appway AG hat verschiedene Risikopunkte die beachtet werden müssen. Dabei reicht es völlig die Risiken und deren Auswirkungen zu kennen.

**11.2.1 Eigensinnige Appway Entwickler**

Fehlerreduktion kann nur erreicht werden, wenn das Appway Entwicklerteam sich auch an den neuen Styleguide hält und sich mit der neuen SCSS Syntax auseinander setzt. Geschieht dies nicht werden weiterhin Fehler auftreten. Dieses Risiko kann mit guter Kommunikation und Schulung im Team reduziert werden.

**11.2.2 Codeintegration bricht bestehende Lösungen**

Die Einführung eines CSS Resets und der langsame Umbau bestehender Stylesheets zu SCSS Dateien kann dazu führen, dass Änderungen im Layout einer Webseite entstehen. Für die Kunden von Appway AG ist dies ein KO-Kriterium, wenn es um die Migration einer neuen Softwareversion geht. Damit dies nicht passiert müssen alle zukünftigen Änderungen sehr detailliert getestet werden.

**11.2.3 SASS Community schwindet**

Die Internettechnologien gehen meist so schnell wie sie kommen. Es ist unwahrscheinlich, dass die bestehende SASS Community so stark zurück geht, dass die Weiterentwicklung eingestellt und die Akzeptanz von Präprozessoren verloren geht.

**11.3 Zeitplan Roll-Out**

Der Start des Roll-Out findet anfangs August statt und soll bis Ende 2014 abgeschlossen sein. Welche Meilensteine geplant sind ist im folgenden Zeitplan ersichtlich.

**TODO: ZEITPLAN ROLL OUT**

* Integration SASS Parser
* Schulungsunterlagen vorbereiten
* Schulung Entwicklerteam
* Schulungsunterlagen weitergeben an Appway Academy
* Umbau der bestehenden CSS Stylesheets zum neuen SCSS Syntax
* Testphase

**12 Fazit**

Die Appway AG Front-End Entwickler hatten bisher fast keine einheitliche Regelung wie mit der Programmierung von Cascading Style Sheets (CSS) umgegangen wird. Die Tatsache, dass CSS viel weniger Möglichkeiten bietet als eine reifere Programmiersprache wie Java führt dazu, dass sich viele Fehler einschleichen, welche nur schwer zu beheben sind. Die kommende Einführung des in das Appway Produkt eingebauten SASS Präprozessor gibt den Entwicklern die Chance den Umgang mit CSS zu überdenken und effektiven sowie fehlerfreien Code zu schreiben. Der Umfang der die SCSS Syntax bietet eliminiert die meisten Fehlerquellen und sorgt automatisch für mehr Produktivität der Mitarbeiten von Appway AG. Die dazukommende Erweiterung des bestehenden Styleguides merzt bei befolgen der enthaltenen Richtlinien die restlichen grossen Fehlerquellen auch noch aus. Die kombinierte Lösungsvariante bringt nach der Einführung ein Effektivitätssteigerung von über 60% im Umgang mit CSS. Das Resultat dieser Bachelorarbeit ist aber nicht nur ein Gewinn in der Effektivität sondern soll den Mitarbeitern von Appway AG aufzeigen, dass in Allen Bereichen immer Potential für Verbesserung besteht.

Ab August beginnt der Roll-Out der neuen Lösungsvariante und bis Ende Jahr soll diese in den Arbeitsalltag integriert werden. Es gibt jedoch noch weitere Punkte die im Rahmen dieser Arbeit nicht mehr gemacht wurden, in der Zukunft aber noch hinzukommen werden. Die Import Eigenschaft von SASS, um Dateien in weitere SASS Dateien zu laden, ist ein wichtiges Werkzeug für einen gegliederten Aufbau mit CSS. Der Import muss aber für Appway noch angepasst werden, da innerhalb des Produkts die Stylesheets von diversen Quellen kommen können und nicht wie von SASS erwartet in derselben Dateistruktur liegen. Ein weiteres Tool auf welches im Moment noch verzichtet wird ist das Framework Compass, welches SASS mit unzähligen nützlichen Zusatzfunktionen erweitert. Die Integration von Compass in das Vaadin Framework ist zurzeit im Gange und sobald dies erfolgreich abgeschlossen wurde, wird über die Integration in Appway nachgedacht. Abgesehen von der eingeführten Lösungsvariante arbeiten die Appway Entwickler gleichzeitig an einem neuen CSS Editor. Dies wird den Umgang für die Studio Anwender in erster Linie um ein vielfaches vereinfachen und gleichzeitig die Nutzung der SCSS Synatx fördern.

Wie die Zukunft von CSS im Allgemeinen aussieht ist völlig offen. Die Präprozessoren haben die Arbeit mit Stylesheets wesentlich vereinfacht und sie werden nicht die letzten Tools sein, welche solch eine Popularität erreichen. Dies heisst für Appway AG die Augen nach Neuerungen und Ideen im Web offen zu lassen und stetig neugierig bleiben.

**13 Glossar**

* RnD
* Fogbugz
* Prozessengine
* Rendering
* Appway Studio
* HTML ID
* W3C
* IED
* Paradigmenwechsel
* Depricated
* Cleafix
* Gridsysteme
* Syntax
* MIT-LIZENZ

**14 Literatur**

- Scalable and Modular Architecture for CSS – A felxible guide to developing sites small and large (by Jonathan Snook)

- An introduction to object oriented css (http://coding.smashingmagazine.com/2011/12/12/an- introduction-to-object-oriented-css-oocss/)

- Every time you call a proprietary feature “CSS3” (http://alistapart.com/article/every-time-you-call-a- proprietary-feature-css3-a-kitten-dies)

-Learning to love the boring bits of CSS (http://alistapart.com/article/love-the-boring-bits-of-css)

- Why Sass (http://alistapart.com/article/why-sass)

- Working with CSS Preprocessors (https://developers.google.com/chrome-developer-tools/docs/css- preprocessors)

* Give your CSS some room to breathe (http://www.impressivewebs.com/css-readability/)
* SMACCS
* http://coding.smashingmagazine.com/2011/12/12/an-introduction-to-object-oriented-css-oocss/
* http://code.tutsplus.com/tutorials/object-oriented-css-what-how-and-why--net-6986

Compass

http://www.apache.org/licenses/LICENSE-2.0

https://vaadin.com/home

https://vaadin.com/book/vaadin7/-/page/themes.sass.html

**15 Quellen**

* <http://www.w3.org/Style/CSS/>

In ruhe nochmals en Text durchgehen und alles Links und Bücher nachführen!

**16 Bestätigung**

**17 Anhang**

- Styleguide Appway

- JAVA KLASSEN