Visuelle Aufbereitung qualitativer Merkmale von Website-Code am Beispiel eines Analysetools

Tamara Czakert

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2018

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 25. September 2018

Tamara Czakert

Inhaltsverzeichnis

Kurzfassung

Webentwickler werden damit beauftragt, bestehende Seiten zu überarbeiten oder neue aufzubauen. Dabei sind viele verschiedene Aspekte der Qualitätssicherung von Webanwendungen zu beachten sowie eine breite Anzahl verschiedener Verfahren zur Erstellung von Webinhalten zu beherrschen. Die Arbeit befasst sich mit einer Auswahl solcher Qualitätsmerkmale und den damit einhergehenden Regeln, die vom Entwickler eingehalten werden müssen. Ein in diesem Zuge erstelltes Projekt fasst vier Qualitätsanalysen, anhand voll automatisierter Methoden, zusammen. Hauptaugenmerk wird danach auf die Weiterverarbeitung der erfassten Daten gelegt. Speziell für diese Anwendungsfälle erstellte Informationsvisualisierungen bieten einen schnellen Überblick über die derzeitige Qualität und das Optimierungspotential der untersuchten Webseite. Darüber hinaus werden in einem zusätzlichen Bereich Teilinformationen, die aus dem Quelltext ausgelesen werden, visuell dargestellt. Um dies zu verdeutlichen, werden zwei grundverschiedene Webseiten hinsichtlich dieser Aspekte untersucht.

Abstract

Web developers are tasked with revising and rebuilding internet pages. In this context many different aspects of quality assurance for web applications must be complied with. Also, a broad range of procedures in the establishment of such web content quality features needs to be mastered. This thesis deals with the selection of such quality characteristics and the accompanying rules which need to be adhered by the developer. A project developed in this regard aggregates four quality analyses by means of fully automated methods. Subsequently the focus lies in the further processing of collected data. Information visualizations developed especially for these use cases provide a quick overview on present quality and optimization potentials for the evaluated webpage. Furthermore, partial information gathered from the source code is reflected visually in an additional area. To emphasize this, two fundamentally different webpages are investigated regarding these aspects.

Kapitel 1

Einleitung

Im Jahr 2017 sind mehr als eine Milliarde Webseiten im World Wide Web verfügbar und es werden täglich mehr [26]. Mit jeder öffentlich gestellten Webanwendung werden auch die Anforderungen an diese größer. Zu Beginn war es von Bedeutung überhaupt einen Webauftritt zu haben, dies reicht jedoch schon lange nicht mehr aus. Reine HTML-Seiten, die durch etwas CSS mit Farben und verschiedenen Schrifttypen versehen werden, stellen die Benutzer nicht länger zufrieden. Diese wollen sowohl grafisch als auch technisch mit dem konfrontiert werden, was möglich ist.

Für den Webentwickler ist es wichtig, immer up to date zu bleiben und keinen wichtigen Trend zu verpassen. Fast täglich werden neue Module und Bibliotheken online gestellt. Dabei gilt es jene auszuwählen, die den eigenen Projekten einen Mehrwert bieten können. Neben diesen technischen Errungenschaften, die dem Entwickler fortwährend zur Verfügung stehen, ist es äußerst wichtig, die Qualitätssicherung einer Anwendung nicht zu vergessen. Auch wenn eine Seite auf den ersten Blick modern und grafisch perfekt aussieht, können sich dahinter einige Probleme verstecken, die es zu vermeiden gilt.

1.1 Problemdefinition

Das World Wide Web und die damit einhergehenden Technologien befinden sich in einem stetigen Wandel. Neben den Anforderungen optisch den derzeitigen Erwartungen zu entsprechen, gilt es auch die technischen Möglichkeiten bestmöglich zu nutzen. Webentwickler müssen jederzeit einen Überblick über die aktuell zur Verfügung stehenden Verfahren zur Erstellung von Webanwendungen haben. Darüber hinaus ist es jedoch wichtig, Anwendungen zu bauen, die den qualitativen Ansprüchen entsprechen. Nur Seiten, die neben dem grafischen Auftritt auch qualitativ dem entsprechen, das der Benutzer erwartet, werden erneut Besucher anlocken.

Wie schon bei der Entwicklung von Webanwendungen gibt es auch bei deren Qualitätssicherung eine Vielzahl an Merkmalen, die betrachtet werden sollen. Einige dieser Qualitätsmerkmale können voll automatisch untersucht werden, bei anderen ist es jedoch notwendig, dies von einer speziell geschulten Person durchführen zu lassen.

Online stehen einige Instrumente zur Verfügung, die die Analyse einer Webseite unterstützen. Eine Vielzahl dieser Hilfsmittel legt den Fokus auf ein spezielles Quali-

tätsmerkmal. So gibt es beispielsweise für die Untersuchung des erstellten HTML-Codes einen eigens dafür entwickelten Validator. Um als Entwickler eine Seite überprüfen zu können, bedarf es so mehrerer Analysen in unterschiedlichen Tools. Dies erscheint oft als sehr aufwendig und bietet keinen guten Überblick über die gesamte Auswertung einer Webseite. Zudem ist die Aufbereitung der Ergebnisse der Analysen meist nur eine Liste von Fehlern, die eher unübersichtlich ist.

1.2 Ziele

Um es Webentwicklern einfacher zu machen, die Qualität ihrer Webseiten zu überprüfen, ist es sinnvoll, mehrere Analyseverfahren in einem Tool zur Verfügung zu stellen. Das damit einhergehende Ziel ist, verschiedene Aspekte der Qualitätssicherung zusammenzufügen und dem Entwickler helfen Zeit einzusparen. Durch die Erstellung von Informationsvisualisierungen werden die erfassten Auswertungen grafisch so präsentiert, dass ein schneller Überblick über die Mängel sowie das Optimierungspotential gegeben ist. Zusätzlich soll dem Entwickler mit Hilfe dreier Visualisierungen der erstellte HTML-Code aufgezeigt werden. Informationen zu den im Dokument verwendeten Farben zeigen auf, wie viele unterschiedliche Farben angewendet worden sind und bieten so die Möglichkeit zu überprüfen, ob hier sauber gearbeitet worden ist. Eine weitere Grafik, die die Relationen zwischen den einzelnen HTML-Elementen widerspiegelt, verschafft eine rasche Übersicht über den Aufbau der Anwendung.

1.3 Gliederung

Um dem Leser einen besseren Überblick zu bieten, ist die nachstehende Arbeit in mehrere Bereiche gegliedert. Es wird empfohlen die einzelnen Kapitel in der angeführten Reihenfolge kontinuierlich zu lesen. Bei Interesse können jedoch auch lediglich Teilbereiche betrachtet werden. Das Kapitel 2 beschreibt die Qualitätsmerkmale, die für diese Thesis von Bedeutung sind. Am Beginn wird vorerst der Begriff Qualität geklärt. Nachfolgend werden dann vier wesentliche Merkmale näher beschrieben. Im Kapitel 3 werden Grundlagen zum Thema Visualisierung dargelegt, wobei hauptsächlich auf Informationsvisualisierungen näher eingegangen wird. Das 4. Kapitel gibt einen Überblick über die Technologien, die zum Testen verschiedener Qualitätsmerkmale verfügbar sind und beschreibt deren Prozesse. Informationen zur Implementierung der eigenen Webanwendung zum Untersuchen der Qualität kann im Kapitel 5 gefunden werden. Die Analyse zweier bekannter Webseiten wird im Kapitel 6 ausgeführt. Das letzte Kapitel zeigt die durch diese Arbeit erforschten Aspekte anhand des implementierten Analyse-Tools.

Kapitel 2

Qualitätsmerkmale

Im Folgenden werden einige äußerst wichtige Merkmale in Bezug auf die Qualität von Webseiten beschrieben. Vor der Analyse eines Internetauftrittes ist es notwendig, über die Grundlagen im Bereich Webentwicklung informiert zu sein, um später Verbesserungen an verschiedenen Prozessen durchführen zu können, und zu entscheiden auf welche Optimierungen gegebenenfalls verzichtet werden kann. Zuvor wird jedoch der Qualitätsbegriff näher erläutert.

2.1 Der Begriff Qualität

Qualität ist ein immer wichtiger werdender Faktor in der Wirtschaft. Es reicht schon lange nicht mehr, nur im World Wide Web gefunden zu werden. Der Auftritt im Internet spiegelt die Qualität der Produkte und Dienstleistungen, die über dieses Medium angeboten werden, wieder. Durch den globalen Wettbewerb müssen Kundenerwartungen erfüllt werden, um ein Abwandern zur Konkurrenz zu vermeiden.

Qualität leitet sich vom lateinischen Wort *qualitas* ab und bedeutet Beschaffenheit beziehungsweise Eigenschaft. In der bildungssprachlichen Bedeutung wird Qualität als „Gesamtheit der charakteristischen Eigenschaften (einer Sache, Person); Beschaffenheit“ [33] definiert. Der Begriff wurde auch als ISO-Norm definiert, um ihm eine internationale, einheitliche Bedeutung zuzuschreiben. In der DIN EN ISO 9000 ist Qualität der „Grad, in dem ein Satz inhärenter Merkmale Forderungen erfüllt“ [1].

Der Qualitätsbegriff kann durch folgende Aussagen festgelegt werden [2]:

- Die Qualität einer Sache ist eine relative Kenngröße und charakterisiert die Deckungsgleichheit der zu Beginn ermittelten Anforderungen (Abbildung 2.1).
- Da Qualität kein physikalischer Wert ist, kann lediglich der Grad der Übereinstimmung der Forderungen gemessen werden.
- Qualität ist keine zweiwertige Größe, und kann daher nicht nur „positiv“ bzw. „negativ“ sein. Eine Sache hat nicht keine Qualität, sondern ein gewisses Maß an Ausprägung der qualitativen Merkmale.

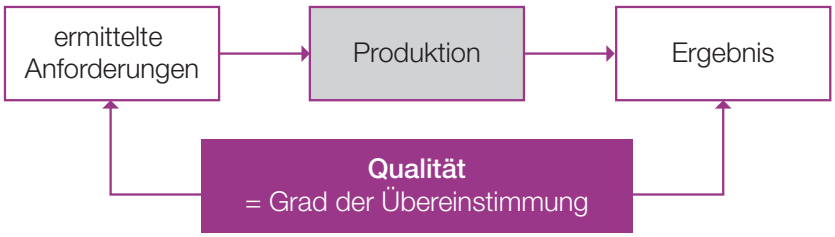


Abbildung 2.1: Qualität als Grad der Übereinstimmung

2.1.1 Qualitätsphilosophien

Im Laufe der Zeit haben sich viele Wissenschaftler und Philosophen mit dem Thema des Qualitätsmanagements auseinandergesetzt und verschiedene Ansätze entwickelt.

Philip B. Crosby hat das sogenannte Null-Fehler-Programm (Zero Defects Concept) entwickelt, welches eine fehlerfreie Produktion beschreibt. Viele Unternehmen akzeptieren eine gewisse Fehlerquote und Nachtbesserungsarbeiten, welche kostenintensiv sein können. Mit vier Punkten soll sein Konzept eingehalten werden können [3]:

- Qualität wird als Übereinstimmung mit Anforderungen definiert.
- Das Grundprinzip der Qualitätserzeugung ist Vorbeugung.
- Null Fehler müssen zum Standard werden.
- Maßstab der Qualität sind die Kosten der Nichterfüllung von Anforderungen.

Der Statistiker und Physiker William Edwards Deming war ein wichtiger Wegbereiter auf dem Gebiet des Qualitätsmanagements. Seine bekanntesten Philosophien stellte er in Japan auf, wohin er eingeladen wurde, um die dortige Wirtschaft zu verbessern. Demings Grundsatz war stets die Verbesserung und Weiterentwicklung von Prozessen. Daraus entwickelte sich der Plan-Do-Check-Act-Zyklus (Abbildung 2.2). Dieser Kreislauf sollte immer wieder durchlaufen werden, um frühzeitig Fehler zu erkennen und die Arbeitsprozesse entsprechend anpassen zu können. Dies spart Zeit und Kosten und kann zu einer effektiveren Entwicklung späterer Projekte führen. Methoden, die die Qualität verbessern, sollen im nächsten Schritt als Standard festgelegt werden [31].

2.1.2 Elementare Qualitätstools

Elementare Qualitätstools sind Instrumente, die es erleichtern sollen, Fehlerquellen zu erkennen, verstehen und beheben. Sie sind in ihrer Anwendung unkompliziert und versuchen durch Visualisierungen Zusammenhänge aufzuzeigen. Zudem wird dabei die Kreativität der Lösungsfindung und die Zusammenarbeit der Mitarbeiter gestärkt. Im -folgenden werden sieben statische Hilfsmittel beschrieben, die zu einer Qualitätssteigerung führen können [2].

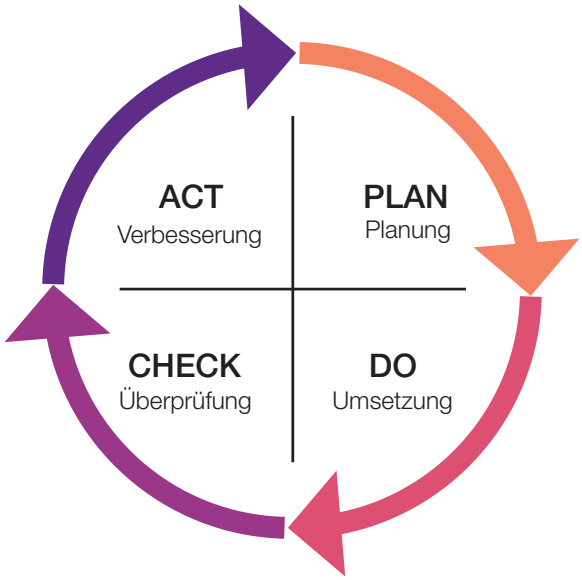


Abbildung 2.2: PLAN-DO-CHECK-ACT-Circle

Fehlersammelkarte

Die Fehlersammelkarte ist eine einfache und schnelle Methode der Fehlererfassung. Dabei werden Fehler kategorisiert und anschließend gezählt. Ein Eintrag in der Fehlersammelkarte kann zum Beispiel die Kategorie HTML-Fehler, mit einer Anzahl von neun aufgetretenen Problemen, sein.

Histogramm

Das Histogramm bietet die Möglichkeit Messwerte in Klassen einzuteilen und deren Häufigkeit mit Hilfe eines Balkendiagramms darzustellen. Auch dieses Werkzeug dient der Fehlererfassung.

Pareto-Analyse

Die Pareto-Analyse soll helfen, die Ursachen die zu den Problemen führen, aufzudecken und entscheiden zu können, welche Fehler schnellstmöglich behoben werden sollen, um eine rasche Verbesserung zu erzielen. Dabei werden die Fehlerarten in einem Säulendiagramm dargestellt. Die Höhe der Säulen wird durch die Häufigkeit des Fehlers definiert. Von links nach rechts werden die Säulen so sortiert, dass sie in ihrer Größe abnehmen. Im nächsten Schritt kann das Diagramm in drei Bereiche eingeteilt werden. Der A-Bereich soll ca. 70% der Fehleranzahl enthalten. Im B-Bereich sind es 20% und im C-Bereich 10%. Somit sind die aufgetretenen Probleme in Prioritäten eingeteilt und es sollte begonnen werden, die Fehler der Kategorie A zu beheben.

Als Grundlage dieser Analyse dient die sogenannte 80/20 Regel, die besagt dass sich

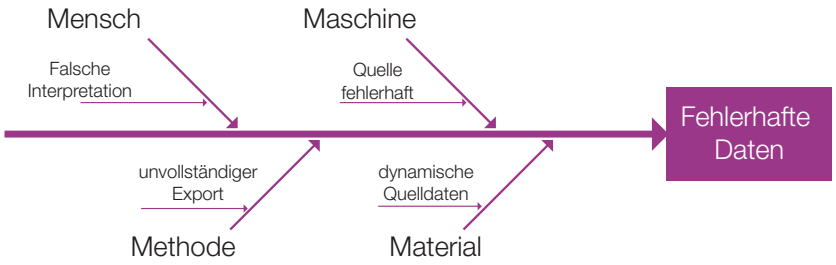


Abbildung 2.3: Ishikawa Diagramm

80% der Probleme auf 20% der Fehlerarten zurückführen lassen.

Korrelationsdiagramm

Bei Korrelationsdiagrammen werden Größen in Abhängigkeit zu einem anderen Parameter aufgezeigt, um so mögliche Fehlerursachen ermitteln zu können. Diese Methode unterstützt die Fehleranalyse.

Ishikawa-Diagramm

Das Ishikawa-Diagramm oder auch Ursache-Wirkungs-Diagramm (Abbildung 2.3) genannt, wurde vom Japaner Kaoru Ishikawa entwickelt und soll auf übersichtliche Art und Weise darstellen in welcher Beziehung Ursache und Wirkung stehen. Dazu wird ein Problem herausgegriffen und im Team näher beleuchtet. Die möglichen Einflussfaktoren werden aufgelistet und untergeordnet die möglichen Ursachen, die später zum Fehler führen können. Auf diese Art können Probleme und ihre Entstehung besser nachvollzogen werden. Die Einbindung mehrerer Mitarbeiter aus unterschiedlichen Produktionsabläufen ist dabei essentiell. Dadurch wird ein besseres Verständnis für Fehlervermeidung geschaffen.

Brainstorming

Da die Fehler nun aufgezeichnet und analysiert worden sind, bedarf es einer Fehlerbehebung. Eine bekannte und weit verbreitete Methode dafür ist das Brainstorming. Dabei werden vorerst Vorschläge gesammelt, ohne diese zu bewerten oder zu kritisieren. Ein reger Austausch kann hierbei zu neuen kreativen Lösungsansätzen führen. Im weiteren Schritt werden die aufgezeigten Lösungsvarianten geordnet und bewertet. Dabei ist es hilfreich einen unparteiischen Moderator im Team zu haben.

2.2 HTML

HTML steht für Hypertext Markup Language. Die erste Version wurde 1992 entwickelt. Diese Sprache wird verwendet, um die Struktur einer Webseite mit Hilfe von HTML-Elementen zu definieren. HTML ist seit der ersten Version nicht *case-sensitive* und

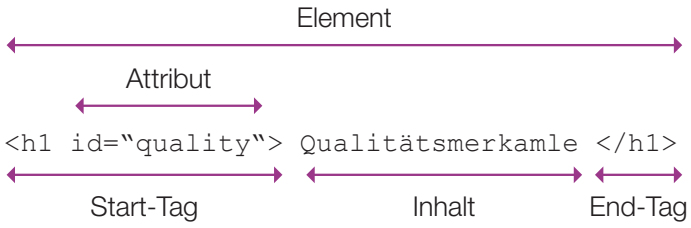


Abbildung 2.4: Aufbau eines HTML-Elements

wurde mit der Version 3.2 erstmals als Empfehlung des W3C veröffentlicht [9]. Die aktuelle Version heißt HTML5, diese ist jedoch nicht nur eine Erweiterung der bisherigen Markupsprache, sondern eine grundlegende Überarbeitung. Sie beinhaltet zahlreiche neue Elemente, die zur besseren Strukturierung der Seite dienen sollen.

Der grundlegende Aufbau einer HTML-Seite wurde nicht verändert. Als *root* Element wird der `html`-Tag verwendet. Darin befinden sich zwei weitere Elemente, `head` und `body`. Im `head` Bereich werden Metadaten angegeben und gegebenenfalls andere Dateien, wie Stylesheets, eingebunden. Der `body` dient der Repräsentation des eigentlichen Inhaltes und kann zahlreiche Elemente enthalten, die wiederum verschachtelt sein können.

Grundlegend wird zwischen zwei verschiedenen Arten von HTML-Elementen unterschieden. Einerseits gibt es Block-level Elemente, die die Eigenschaften für Breite und Höhe sowie Abstände zu anderen Elementen besitzen und standardmäßig in einer neuen Zeile positioniert werden. Die *inline* Elemente haben diese Attribute andererseits nicht und werden im Zeilenfluss angehängt ohne eine neue zu beginnen.

Jedem Tag können Attribute angehängt werden, um die Eigenschaften des Elements besser zu konfigurieren und sein Verhalten anzupassen (siehe Abbildung 2.4). Viele dieser Werte sind speziell für einige Tags definiert (eine vollständige Liste aller zur Verwendung stehender Attribute im Anhang). Darüber hinaus gibt es einige Attribute, die global angewendet werden können: accesskey, class, draggable, hidden, id, lang, style, tabindex, title, usw.

2.3 Stylesheet

Die *Cascading Style Sheets* – CSS – wurden vom W3C eingeführt und dienen der Abgrenzung von Inhalt und Erscheinungsbild. *Cascading* bezieht sich dabei auf das hintereinander Schalten von Prioritäten der anzuwendenden Regeln. Die erste CSS Version wurde 1996 veröffentlicht und danach stets erweitert. Die derzeit aktuelle Version ist CSS3, mit deren Entwicklung bereits 2005 begonnen wurde [9].

Um einem HTML-Element ein Styling zuweisen zu können, bedarf es eines Selektors. Dieser kann bezogen auf den Typ des Elementes gewählt werden. So können beispielsweise alle im Dokument vorkommenden p-Elemente ein gleiches Aussehen erhalten. Sollen nur einige der Elemente gleich dargestellt werden, können Klassen als Auswahl-

kriterium herangezogen werden. Diese werden immer mit einem vorangestellten Punkt gekennzeichnet. Auch IDs sind als Selektor zulässig und sind an einer Raute vor dem Bezeichner erkennbar. Das *-Symbol kann als universeller Selektor verwendet werden und somit allen Elementen gewisse Eigenschaften zuweisen. Auch andere Attribute neben Klassennamen oder IDs können für die Auswahl herangezogen werden. Um auf spezielle Verschachtlungen von Elementen Rücksicht zu nehmen, gibt es sogenannte Kind-, Nachfahr- und Geschwister-Selektoren. Zusätzlich sind Pseudoklassen, wie `:hover`, verfügbar. Diese bieten die Möglichkeit auf verschiedene Zustände des Elements zu reagieren. Pseudoelemente bieten die Möglichkeit nur spezielle Elemente einer Gruppe auszuwählen, wie zum Beispiel `:first-letter`.

2.3.1 Einbindung

Die Styles, die einem Element zugewiesen werden sollen, können auf unterschiedliche Art und Weise eingebunden werden. *Inline styles* werden direkt über das style-Attribut innerhalb des Start-Tags eines Elements eingebaut. Diese Variante ist nur für einmalige und kurz gehaltene Darstellungen relevant und sollte, um mögliche Fehler auszuschließen, eher vermieden werden.

Intern Styles ist ein Block der sämtliche Informationen über das Erscheinungsbild des Dokumentes enthält und wird im `head`-Bereich mit einem eigenen Style-Tag definiert. Diese Variante eignet sich definitiv besser als *inline* Angaben, macht das Dokument jedoch unnötig groß und unübersichtlich.

Externe Style Sheets sind externe Dateien, die im `head`-Bereich verlinkt werden. Großer Vorteil dieser Variante ist, dass ein Stylesheet in mehren Dokumenten verwendet werden kann, ohne den Code mehrfach zu schreiben. Auch Veränderungen in einem global Style Dokument lassen sich leichter durchführen.

2.4 Performance

Die Performance spielt bei Webapplikationen eine äußerst wichtige Rolle, da der User möglichst schnell mit dem System interagieren möchte. Doch wie schnell ist eigentlich schnell genug? Dieser Frage stellte sich der berühmte Usability-Experte Jakob Nielsen und entwickelte 1993 drei wesentliche Grenzwerte, die zu beachten sind und bis heute ihre Gültigkeit haben [8, Kap. 5].

0,1 Sekunde: Dies ist das Limit, bei dem der Benutzer das Gefühl hat, direkt mit den Daten zu interagieren. Er spürt dabei keine Verzögerung des Systems. Daher ist es auch nicht notwendig dem User anzuzeigen, dass das System die Eingabe verarbeitet.

1 Sekunde: Bei einer Sekunde merkt der Benutzer zwar eine leichte Verlangsamung der Applikation, wird davon aber meist nicht irritiert, da er bemerkt, dass das System an seiner Eingabe arbeitet. Feedback ist hier nicht zwingend notwendig, kann aber zum Beispiel durch Veränderung des Mauszeigers oder Ähnlichem angezeigt werden.

10 Sekunden: Benötigt die Anwendung mehr als 10 Sekunden besteht die Gefahr, dass der Benutzer abdriftet und seine Aufmerksamkeit nicht auf dem derzeitigen Task halten

kann. Hier muss dem User unbedingt angezeigt werden, dass das System arbeitet und wann es damit fertig ist. Dies kann mittels Prozent-Anzeige einfach realisiert werden.

2.4.1 Steve Souders – 14 Regeln für bessere Web-Performance

Steve Souders ist ein weltweit bekannter Experte auf dem Gebiet der Web-Performance. Er war lange Zeit für die Performance von Yahoo! verantwortlich und wechselte danach zu Google. In seinem ersten Buch „High Performace Web Sites“ definierte er 14 Regeln, die zur Performance-Steigerung beitragen sollen [10]:

1. Weniger HTTP-Requests verwenden.
2. Ein Content Delivery Network einbinden.
3. Einen Expires Header hinzufügen.
4. Gzip Komprimierung verwenden.
5. Stylesheets am Beginn einfügen.
6. JavaScript am Ende positionieren.
7. CSS Ausdrücke vermeiden.
8. JavaScript und CSS auslagern.
9. DNS Lookups reduzieren.
10. JavaScript minimieren.
11. Weiterleitungen vermeiden.
12. Script Duplikate entfernen.
13. ETags konfigurieren.
14. Ajax cachebar machen.

2.4.2 Drei Grundregeln zur Verbesserung der Performance

Die Optimierung einer Website kann entweder im Backend oder im Frontend passieren. Laut C. Wenz in [5] wird 80% der Zeit, die für das Ausliefern der Website verwendet wird, vom Frontend verbraucht. Daher lohnt es sich definitiv einige Aspekte für die Verbesserung der Performance auf jede Webapplikation anzuwenden.

Daten einsparen

Wenn weniger Daten vom Server zum Client geschickt werden, wird weniger Zeit in Anspruch genommen, da die Daten schneller übertragen und somit Bandbreite und Kosten eingespart werden können. Dabei soll jedoch das bestehende System nicht abgeändert oder gar verringert werden müssen. Es sollen lediglich Optimierungen vorgenommen werden.

Dateien, die Code enthalten, können mit bestimmten Tools minimiert und somit optimiert werden. Dabei gehen die Systeme meist so vor, dass sie Leerräume, wie Leerzeichen, Einrückungen und Zeilensprünge, auf das Nötigste reduzieren. Variablennamen in Scriptteilen werden vereinfacht, und die Reihenfolge von Kommandos wird oft leicht verändert. Bei Stylesheets werden Farbwerte abgekürzt. Im HTML-Code können Attribute, die überflüssig sind, entfernt werden. Der Ersteller solcher Dateien sollte immer

darauf achten, kurze Namen und IDs zu vergeben und zu überprüfen, ob eine Verschachtlung notwendig ist.

Zusätzlich kann eine Komprimierung durch Gzip die Geschwindigkeit der Übertragung erheblich beeinflussen. Der Browser muss in der Lage sein, die komprimierten Dateien zu entpacken. Ist dies der Fall so kann der Server die Inhalte vor dem Senden verpacken.

Viele Daten können auch bei Bildern eingespart werden. Im ersten Schritt sollte das richtige Format ausgewählt werden. Im Web werden drei Grafikformate verwendet:

- GIF,
- PNG,
- JPEG.

Nach der Wahl des Formats kann ein Bild meist zusätzlich verbessert werden. Dazu stehen einige online Tools zur Verfügung, die Grafiken ohne Verlust optimieren können, in dem beispielsweise die Farbpalette reduziert wird. JPEGmini[1] bietet die Möglichkeit Bilder um bis zu 80% in ihrer Größe zu reduzieren, ohne dabei einen Qualitätsverlust zu erleiden.

HTTP-Verbindungen reduzieren

Ältere Browser erlaubten lediglich zwei gleichzeitige Verbindungen zu einem Server, das heißt, dass lediglich zwei Ressourcen zur gleichen Zeit geladen werden konnten. Modernere Webbrowser unterstützen mindestens vier nebeneinander geschaltete Verbindungen und können Daten daher effektiver herunterladen. JavaScript blockiert zudem das downloaden weiterer Ressourcen und sollte daher immer am Ende der Seite eingefügt werden, sodass alle anderen Dateien, die für die Anzeige der Inhalte essentiell sind, bereits vorhanden sind. Stylesheets müssen dagegen am Beginn der Seite eingefügt sein, da sie für das Aussehen des Inhaltes verantwortlich sind und eine spätere Eingliederung den Benutzer verwirren würde. Außerdem ist die Verwendung von CSS-Sprites zu empfehlen. Hier werden Logos und andere Icons in einem Sprite zusammengeführt und mittels CSS Anweisung der jeweilige Teil des Sprites angezeigt.

Auch das Zusammenfügen von mehreren Dateien, egal ob JavaScript oder CSS, vermindert die Anzahl an notwendigen HTTP-Verbindungen. Dabei ist zu beachten, dass JavaScript-Dateien oft Abhängigkeiten zu anderen Dateien haben und diese nicht verloren gehen dürfen. In vielen build Prozessen wird eine einzige resultierende JavaScript Datei generiert, die jedoch bei jeder Veränderung neu heruntergeladen werden muss und daher nur schwer gecached werden kann.

Caching

Das Cachen von Dateien reduziert die nötigen HTTP Anfragen sowie die Datenmenge, die übertragen werden muss. Die Verwendung von Caching bietet jedoch keine sichere Garantie der Performance, da viele User mit einem leeren Cache arbeiten.

Einerseits kann das Cachen von Ressourcen im HTTP-Header definiert werden, andererseits gibt es mit HTML5 auch die Möglichkeit, Dateien in einen lokalen Cache zu

[1] http://www.jpegmini.com/

legen und die Anwendung somit auch offline zur Verfügung stellen. Die Informationen, welche Inhalte gecached werden solle,n werden im sogenannten Manifest definiert.

2.5 Barrierefreiheit

Die oben beschriebenen Qualitätsmerkmale sind mehr oder weniger für alle Menschen spürbar und werden meist auch mit großer Sorgfalt optimiert. Die Barrierefreiheit im täglichen Leben wird immer wichtiger, so muss es Menschen mit Beeinträchtigung möglich sein, öffentliche Gebäude betreten zu können. Dass diese Grundlage auch für Informationen im Web gegeben sein muss, wird oft vergessen. Werden bestimmte Personengruppen von neuen Medien ausgeschlossen, nennt man dies „Digital Divide“ [15]. Doch bevor näher auf die Umsetzung von Barrierefreiheit eingegangen werden kann, muss definiert werden, was eine Barriere ist. Eine Barriere ist laut Duden eine „Absperrung, die jemanden, etwas von etwas fernhält“ [14]. Diese Einschränkungen können visuelle, auditive, motorische, sprachliche, kognitive, Sprach-, Lern- und neurologische Behinderungen sein.

2.5.1 Gesetzliche Lage in Österreich

Um Menschen mit besonderen Bedürfnissen nicht auszuschließen, wurden auf EU-Ebene in den E-Government-Strategien die WAI-Leitlinien[2] aufgenommen, die die Mitgliedstaaten einhalten müssen, und auch die Ziele die dadurch erreicht werden sollen definiert. Die Mitgliedstaaten müssen folgende Punkte einhalten:

- die Anpassungsprozesse von Web-Inhalten an die WAI-Leitlinien beschleunigen,
- die WAI-Konformität auf allen Ebenen föderal, regional und lokal erreichen,
- die WAI-Konformität bei externer Beauftragung zur Erstellung von Web-Inhalten berücksichtigen,
- den Dialog mit Interessengruppen wie Behindertenorganisationen oder Seniorinnen und Seniorenverbänden stärken,
- den Zugang von Menschen mit Behinderung zur Wissensgesellschaft verbessern,
- technische, rechtliche und andere Schranken für eine wirkliche Beteiligung an der wissensbasierten Wirtschaft und Gesellschaft beseitigen.

Zusätzlich zu den Verordnungen auf EU-Ebene hat Österreich dieses Thema auch in die Rechtsgrundlage aufgenommen. Die Bundesverfassung enthält im Artikel 7 den Gleichheitsgrundsatz und definiert ein Diskriminierungsverbot von eingeschränkten Personen. Bund, Länder und Gemeinden sorgen für die Gleichstellung aller Menschen in allen Bereichen des täglichen Lebens.

Im Behindertengleichstellungspaket wurden Kriterien erstellt, die die Zumutbarkeit definieren und eine mögliche Rechtsfolge von Diskriminierung festlegen. Seit 1. Jänner

[2]WAI – Die Web Accessibility Initiative (WAI) innerhalb des W3C entwickelt und veröffentlicht Empfehlungen (Synonyme: Richtlinien, Leitlinien), um Internetangebote und -techniken für Menschen mit Behinderungen, ältere Menschen und Benutzerinnen und Benutzer im Allgemeinen barrierefrei, zumindest zugänglich, zu gestalten. in https://www.ag.bka.gv.at/at.gv.bka.wiki-bka/index.php/Barrierefrei:WAI-Leitlinien

2006 ist die Barrierefreiheit gesetzlich festgelegt und somit müssen auch alle nichtbehördlichen Onlineangebote den Richtlinien entsprechen. Hier ist jedoch die jeweilige Zumutbarkeit zu prüfen. Im Falle eine Klage kann die betroffene Person Schadenersatz fordern [15].

2.5.2 Web Content Accessibility Guidelines

Das World Wide Web Consortium (W3C) definierte bereits 1999 die erste Version der WCAG 1.0, um Inhalte für alle Menschen zugänglich zu machen. 2008 wurden diese Richtlinien überarbeitet und die neue Fassung – WCAG 2.0 – wurde veröffentlicht. In der neuen Version gibt es 4 Richtlinien mit einigen untergeordneten Leitsätzen, die zur Verbesserung der Barrierefreiheit im Web dienen sollen. Die Maßnahmen, die aufgrund dieser Vorschläge gesetzt werden können, unterscheiden sich in den Konformitätslevels A, AA und AAA. Webseiten sollten zumindest dem Konformitätslevel A entsprechen.

Wahrnehmbar

> „Informationen und Bestandteile der Benutzerschnittstelle müssen den Benutzern so präsentiert werden, dass diese sie wahrnehmen können.“ [36]

Textalternativen: Alle Elemente, die keine Textelemente sind, müssen mit einer Textalternative ausgestattet sein, die der jeweilige Benutzer verarbeiten kann. Hier sind vor allem Bilder und Grafiken betroffen, die mit einem alt-Attribut ausgestattet werden müssen. Dieses kann dann von einem Screenreader vorgelesen werden, um dem Rezipienten keine Informationen vorzuenthalten. Dabei ist es wichtig zu beachten, dass Alternativtexte sinnvoll und aussagekräftig sind.

Zeitbasierte Medien: Auch für zeitbasierte Medien sollen Alternativen bereitgestellt werden. Dies umfasst Untertitel, sowie Audio-Deskriptionen für Audio- bzw. Videomaterial. In der höchsten Konformitätsstufe wäre es zusätzlich wünschenswert, Gebärdensprache anzubieten. Bei den zeitbasierten Medien ist die Unterscheidung zwischen Live- und aufgezeichnetem Material zu beachten.

Anpassbar: Die angezeigten Inhalte sollen auf unterschiedliche Art und Weise dargestellt werden können, ohne dabei die Struktur zu verlieren. Elemente zur Strukturierung sollen richtig eingesetzt werden, um Systemen eine Orientierung zu ermöglichen und um die Inhalte in der richtigen Reihenfolge zu interpretieren.

Unterscheidbar: Der Anwender soll die dargestellten Inhalte leicht wahrnehmen und dabei Vorder- und Hintergrund voneinander unterscheiden können. Wichtig ist hierbei, dass bestimmte Elemente nicht nur mit Hilfe von Farbe hervorgehoben sind, um die entsprechende Information zu vermitteln. Außerdem ist auf ein ausreichendes Kontrastverhältnis von Text zum Hintergrund zu achten. Dabei gilt die Formel [5]

$$\frac{L_1 + 0,05}{L_2 + 0,05}, \tag{2.1}$$

wobei L_1 für den relativen Luminanz-Wert der helleren und L_2 für den der dunkleren Farbe steht. Um den Luminanz-Wert einer Farbe zu berechnen, werden die RGB-Werte zuerst normiert, indem sie in ihrer 8-Bit-Darstellung durch 255 geteilt werden. Das jeweilige Ergebnis wird durch R, G und B repräsentiert. Daraus ergibt sich

$$L = 0,2126 \cdot R + 0,7152 \cdot G + 0,0722 \cdot B. \tag{2.2}$$

Ob Texte bezüglich ihrer Farbe und dem entsprechenden Hintergrund den Richtlinien entsprechen, kann mit einigen Online-Tools überprüft werden. Der von WebAIM[3] zur Verfügung stehende *Color Contrast Checker*[4] bietet die Möglichkeit eine Farbe für Text und eine für den Hintergrund zu wählen. Mittels Slider kann die Farbe so abgewandelt werden, dass sie den Levels AA oder AAA entspricht.

Bedienbar

> „Bestandteile der Benutzerschnittstelle und Navigation müssen bedienbar sein.“ [36]

Per Tastatur zugänglich: Alle in der Webapplikation zur Verfügung stehenden Funktionalitäten müssen mit der Tastatur ausgelöst werden können. Dabei spielt die Bedienung des Menüs per Tastatur die wohl größte Rolle. Alle Hauptpunkte sowie die Unterseiten im Navigationsbereich müssen in einer korrekten und sinnvollen Reihenfolge mit den Tasten erreichbar sein. Der aktuelle Fokus muss dabei für den Benutzer immer klar ersichtlich sein.

Ausreichend Zeit: Der User soll genügend Zeit haben, Inhalte zu lesen und mit dem System interagieren zu können. Daher müssen für alle zeitlich begrenzten Inhalte Bedienelemente vorhanden sein, die es ermöglichen die zeitliche Begrenzung abzuschalten, anzupassen und auszuweiten. Für alle Inhalte, die sich bewegen muss die Möglichkeit des Pausierens, Beendens oder Ausblendens gegeben sein.

Anfälle: Die dargestellten Inhalte dürfen nicht zu Anfällen führen. Flackern oder aufblitzende Elemente sollen vermieden werden. Dabei ist die Grenze von drei Blitzen in einer bestimmten Zeitspanne zu beachten.

Navigierbar: Die Navigation durch die Seite soll vereinfacht werden. Dabei können Seitentitel vergeben werden, Links mit ihrem Zweck ausgestattet und Formularfelder mit Labels versehen sein. Die Fokus-Reihenfolge spielt dabei eine wichtige Rolle und soll den Benutzer durch die Seite begleiten.

Verständlich

> „Informationen und Bedienung der Benutzerschnittstelle müssen verständlich sein.“ [36]

[3] Center for Persons with Disabilities

[4] https://webaim.org/resources/contrastchecker/

Lesbar: Die auf der Seite präsentierten Inhalte sollen lesbar und verständlich formuliert sein. Die Dokumentsprache soll von Systemen automatisch erkannt werden können.

Vorhersehbar: Der Benutzer soll die Art und Weise, wie sich Elemente verändern nachvollziehen können. So sollen Teile auf denen der Fokus liegt immer gleich erscheinen. Die Eingabe in ein Formularfeld löst immer ein gleiches Erscheinungsbild aus. Navigationselemente sind auf allen Seiten gleich oder zumindest ähnlich und verändern ihr Aussehen, die Reihenfolge und die Position nicht.

Hilfestellung bei der Eingabe: Die Verwendung von richtigen Formularfeldern ist essentiell wichtig. Somit kann das System erkennen, ob wirklich eine Email-Adresse eingegeben worden ist. Hilfestellungen zur Eingabe sowie erkannte Fehler sind dem Benutzer zu stellen.

Robust

> „Inhalte müssen robust genug sein, damit sie zuverlässig von einer großen Auswahl an Benutzeragenten einschließlich assistierender Technik interpretiert werden können.“ [36]

Kompatibel: Der enthaltene Code soll den Richtlinien entsprechen, sodass Systeme damit einwandfrei umgehen können. Dies beinhaltet vor allem Start- und End-Tags, sowie die Zuordnung von Name, Rolle und Wert-Attributen durch das System.

Kapitel 3

Visuelle Grundlagen

3.1 Informationsvisualisierung

Um aus dem Ergebnis der vorangegangenen Analyse einen Nutzen für den Anwender zu generieren, bedarf es einer geeigneten Repräsentation der Daten. Informationsvisualisierung versucht mittels zahlreicher Richtlinien, Informationen aufzubereiten, sodass diese für den Endnutzer zu Wissen verarbeitet werden können. Der Begriff Information lässt sich aus dem lateinischen Wort „informatio“ bzw. „informare“ ableiten. Dies bedeutet einerseits soviel wie „durch Unterweisung bilden, unterrichten“ und andererseits „eine Gestalt geben, formen“ [24]. Werner Gitt beschreibt Information als „diejenige Unsicherheit, die durch das Ersetzen des betreffenden Zeichens beseitigt wird“ [28].

3.1.1 Grundlagen

Der Mensch ist tagtäglich einer unüberschaubaren Menge an Daten ausgesetzt. Durch Informationsvisualisierung werden Daten geordnet und strukturiert, sodass diese als Information wahrgenommen werden können. Dabei ist es wichtig, dass der Ersteller solcher Visualisierungen den Empfänger der Darstellung möglichst gut kennt, denn nur wenn dieser die Information auch lesen und verstehen kann, kann ein Mehrwert daraus gezogen werden [11]. Der Prozess, bei dem aus Daten Information und aus Information Wissen generiert wird, beschreibt die Abbildung 3.1

Wichtig ist es zu beachten, dass Design die Aufnahme von Information nicht nur wesentlich unterstützen kann, es kann durchaus auch zu Problemen führen. Oft müssen Details weggelassen werden, da sich sich in die Visualisierung nicht einarbeiten lassen. Von zu vielen schönen Akzenten kann der Rezipient abgelenkt werden und die Daten werden verschleiert. Die Aufgabe des Grafikers ist daher eine optimale Balance zwischen Ästhetik und Nutzen zu finden.

3.1.2 Entstehung

Die Entwicklung von Informationsvisualisierung reicht weit zurück in die Anfänge der Menschheit und deren Höhlenmalerei. Die am ehesten zu vergleichenden Werke stammen aus der Statistik bzw. der Kartografie. Karten versuchen Daten aufgrund ihrer räumlichen Position anzuordnen und verhelfen dem Leser somit zu Orientierung in einem für

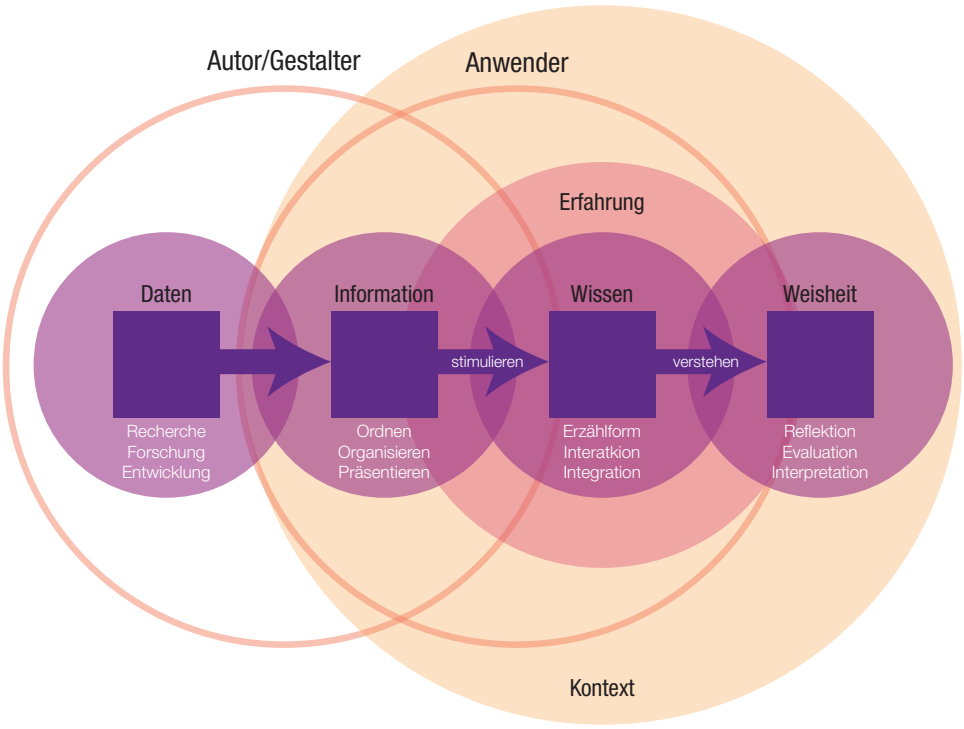


Abbildung 3.1: Transformieren von Daten, Informationen und Wissen [11]

ihn noch unbekanntem Gebiet. Wie vor allem bei Karten offensichtlich ist, sind Visualisierungen immer nur eine Repräsentation der aktuellen Informationslage und können zu einem späteren Zeitpunkt, mit anderem Ausgangspunkt, komplett verworfen oder verändert werden.

Der gelernte Kartograf Jacques Bertin erstellte 1967 eine sogenannte Zeichensystemtheorie auf, welche vor allem für analoge Darstellungen auch heute noch von Bedeutung ist. Für Punkt, Linie und Fläche im zweidimensionalen Raum bestimmte er, wie auch in Abbildung 3.2 zu sehen, die sechs nachstehende Werte:

- Größe,
- Helligkeit,
- Muster,
- Farbe,
- Richtung,
- Form.

3.1.3 Gestaltungsprinzipien

Die von Bertin entwickelte Zeichensystemtheorie war anfangs ausreichend. Im 20 Jahrhundert wurden aus den Erkenntnissen der Gestaltpsychologie die sogenannten Gestalt-

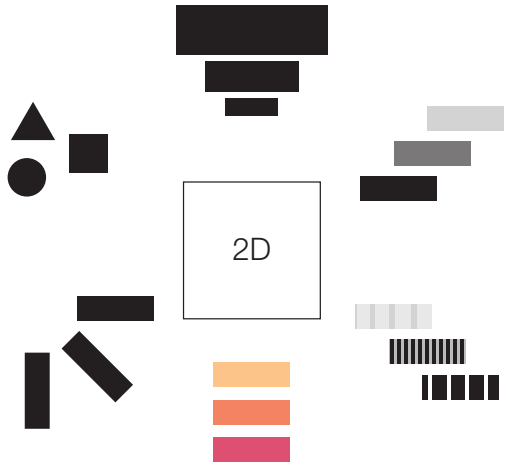


Abbildung 3.2: Zeichensystemtheorie von Jacques Bertin [11]

gesetze formuliert. Der Mensch versucht stets Form und Hintergrund zu unterscheiden. In der Grafikgestaltung kann dieses Wissen genutzt und durch Abweichungen des Normalfalls Aufmerksamkeit generiert werden. Die nachstehenden sieben Gesetze stammen aus dem von Bühler, Schlaich und Sinner veröffentlichten Werk „Visuelle Kommunikation“und wurde durch das Gesetz der guten Fortsetzung ergänzt.

Gesetz der einfachen Gestalt

Das menschliche Gehirn versucht das Wahrgenommene auf einfache geometrische Formen zu reduzieren. Dies erklärt auch, warum Kleinkinder bereits sehr früh Kreise von Quadraten und Dreiecken unterscheiden können.

Gesetz der Nähe

Elemente, die nahe beieinander liegen, werden als zusammengehörig empfunden. Vergrößert sich der Abstand zu anderen Teilen wird diese Entfernung als Grenze bzw. Trennung interpretiert. Dadurch lassen sich Objekte strukturieren und schaffen Orientierung für den Rezipienten (Abbildung 3.3).

Gesetz der Gleichheit

Nicht nur die Nähe von Objekten zueinander ist ausschlaggebend für eine Gruppierung, auch die äußere Erscheinung spielt eine wichtige Rolle. Eigenschaften wie Form und Farbe haben größeres Gewicht als das Gesetz der Nähe. Elemente mit gleichem Aussehen werden als Gruppe interpretiert, auch dies kann für die Strukturierung von Objekten genutzt werden (Abbildung 3.4).

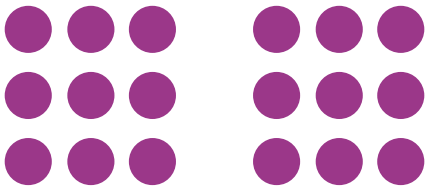


Abbildung 3.3: Gesetz der Nähe

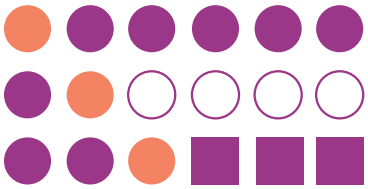


Abbildung 3.4: Gesetz der Gleichheit

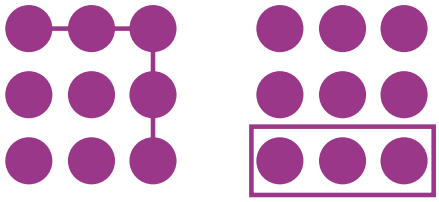


Abbildung 3.5: Gesetz der Geschlossenheit

Gesetz der Geschlossenheit

Noch intensiver werden Elemente als Gruppe angesehen, wenn diese auf einem gleichen Hintergrund stehen oder durch eine Rahmenlinie von anderen Objekten abgegrenzt werden. Das Gesetz der Geschlossenheit wird oft für Layouts verwendet, um Kopf- und Fußbereich vom Hauptinhalt zu trennen (Abbildung 3.5).

Abbildung 3.6: Gesetz der guten Fortsetzung

Gesetz der Erfahrung

Das Gesetz der Erfahrung baut auf das Wiedererkennen von bereits erlernten Formen und Gestalten, selbst wenn diese stark stilisiert oder abstrahiert sind. Dies macht die Verwendung von Icons möglich, welche meist nur sehr gering in ihrem Informationsgehalt sind, jedoch beim Betrachter automatisch Assoziationen auslösen.

Gesetz der Konstanz

Konstanz ist ein sehr wesentlicher Bestandteil für das Gestalten von digitalen Medien. Der Mensch nimmt Objekte mit ihrer Gestalt immer in dessen Umfeld wahr. Ein Nutzer erlernt auf einer Seite, wie ein Button aussieht und wo er ungefähr positioniert ist. Wenn er nun die Seite wechselt, sollte eine Schaltfläche mit gleicher Funktion für den Benutzer schnell erkennbar, also konstant in seinem Aussehen und der Umgebung sein.

Gesetz der Figur-Grund-Trennung

Die Möglichkeit Elemente vom Hintergrund zu unterscheiden ist notwendig und wird durch Konturen, Kontrast, Texturen und Farben gegeben. Ist dies nicht der Fall, so verschwimmt Vordergrund mit Hintergrund und die Wahrnehmung wird getrübt.

Gesetz der guten Fortsetzung

Das Gesetz der guten Fortsetzung beschreibt den natürlichen Weg unserer Wahrnehmung. Linien werden immer in ihrer natürlichsten Richtung verfolgt. Starke Veränderungen durch Ecken oder Knicke eines Pfades werden von unserem Gehirn vermieden. Auch bei nicht durchgehenden Linien kann der Mensch eine Form erkennen und als solche interpretieren (Abbildung 3.6).

3.1.4 Datenabstraktion

Datentypen

Bevor eine Informationsvisualisierung erstellt werden kann, müssen die vorliegenden Daten analysiert werden. Dazu bedarf es einiger wichtiger Unterscheidungsmerkmale

von Daten Typen. Tamara Munzner unterscheidet in ihrem Buch „Visualization Analysis and Design“ fünf verschiedene Arten von Daten [6].

Item: Das Item beschreibt eine individuelle, abgegrenzte Einheit. Dabei kann es sich um einen Knoten in einem Netzwerk handeln. Diese können Lebewesen, Objekte, Länder oder Ähnliches repräsentieren.

Attribut: Ein Attribut ist eine Eigenschaft eines Objektes, die messbar ist, beobachtet oder aufgezeichnet werden kann. Dabei handelt es sich meist um Werte, die mit Zahlen dargestellt werden können. Temperaturen, Preise, oder die Anzahl an Objekten werden mittels Attributen definiert.

Links: Beziehungen zwischen einzelnen Items werden mit Hilfe von Links dargestellt und können mit zusätzlichen Informationen, wie Richtung oder Wertigkeit ausgestattet werden.

Position: Die Position definiert eine örtliche Lage und kann im zweidimensionalen sowie im dreidimensionalen Raum gegeben sein. Koordinaten mit den beiden Werten für Breite (Latitude) und Länge (Longitude) definieren die Position eines Items.

Raster: Raster helfen, kontinuierliche Daten mit geometrischen und topologischen Beziehungen zu repräsentieren.

- Item Das Item beschreibt eine individuelle, abgegrenzte Einheit. Dabei kann es sich um einen Knoten in einem Netzwerk handeln. Diese können Lebewesen, Objekte, Länder oder Ähnliches repräsentieren.
- Ein Attribut ist eine Eigenschaft eines Objektes, die messbar ist, beobachtet oder aufgezeichnet werden kann. Dabei handelt es sich meist um Werte, die mit Zahlen dargestellt werden können. Temperaturen, Preise, oder die Anzahl an Objekten werden mittels Attributen definiert.
- Beziehungen zwischen einzelnen Items werden mit Hilfe von Links dargestellt und können mit zusätzlichen Informationen, wie Richtung oder Wertigkeit ausgestattet werden.
- Die Position definiert eine örtliche Lage und kann im zweidimensionalen sowie im dreidimensionalen Raum gegeben sein. Koordinaten mit den beiden Werten für Breite (Latitude) und Länge (Longitude) definieren die Position eines Items.
- Raster helfen, kontinuierliche Daten mit geometrischen und topologischen Beziehungen zu repräsentieren.

Datensatztypen

Die Erfassung der oben beschriebenen Datentypen resultiert in einem Datensatz. Die Verfügbarkeit der Daten lässt eine Unterscheidung zwischen einem statischen und einem dynamischen Datenset zu. Beim statischen sind zu Beginn alle notwendigen Daten vorhanden und zusammengefasst. Das dynamische Set wird über die Zeit verändert, es können neue Datensätze eingefügt, bestehende verändert oder existierende gelöscht

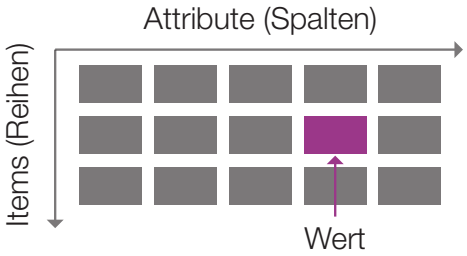


Abbildung 3.7: Tabellen

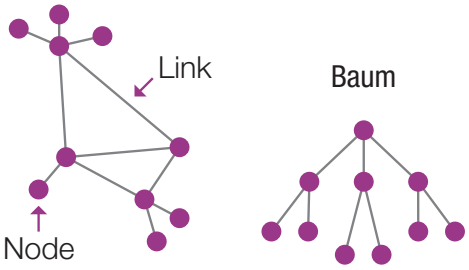


Abbildung 3.8: Netzwerke und Bäume

werden. Je nach Informationslage werden auch hier wiederum fünf Arten voneinander unterschieden. Jeder Datensatztyp kennzeichnet sich durch die in den Daten vorliegenden Datentypen.

Tabellen Tabellen sind die wahrscheinlich üblichste Form der Datensätze und sind aus Tabellenkalkulationsprogrammen vertraut. Sie besitzen Zeilen, welche Items repräsentieren, Spalten, die für deren Attribute stehen und Zellen, in denen die jeweiligen Werte, die sich durch Item und Attribut ergeben, gespeichert sind. Sie benötigen daher sowohl Items als auch deren Attribute und können im zweidimensionalen bzw. im dreidimensionalen Raum angewendet werden. Bei diesen Daten handelt es sich um diskrete Daten (Abbildung 3.7).

Netzwerke Netzwerke werden verwendet, um die Beziehungen zwischen Items, in diesem Zusammenhang auch oftmals Nodes - also Knotenpunkt - genannt, darzustellen. Die Verbindungen zwischen den Items werden als Links bezeichnet. Sowohl Links als auch Nodes können Attribute enthalten. Eine übliche Anwendung finden Netzwerke bei der Abbildung von Sozialen Netzwerken, in denen Personen und deren Beziehung dargestellt werden soll. Eine Sonderform des Netzwerkes ist die hierarchische Darstellung als Baum. Wesentlichster Unterschied ist hier, dass jeder Knotenpunkt im Baum nur jeweils einen Elternknoten hat und keine zyklischen Beziehungen möglich sind (Abbildung 3.8).

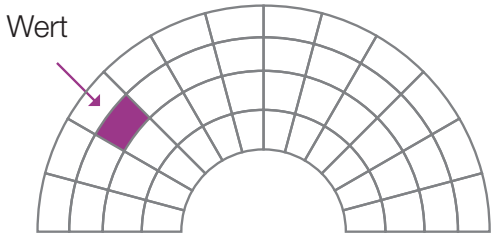


Abbildung 3.9: Felder

Abbildung 3.10: Geometrie

Felder Ähnlich wie bei der Tabelle befindet sich der Wert eines Attributes in einer Zelle. Anders, als bei den zuvor beschriebenen Typen, handelt es sich bei Feldern um kontinuierliche und nicht um diskrete Daten. Jede hier dargestellte Zelle beschreibt einen Messwert im kontinuierlichen Bereich. Daher ist hier nur eine Teilerhebung möglich. Die Aufzeichnung von Temperatur würde eine solche Einteilung erfordern. Wichtig ist es hierbei, ein geeignetes Maß zu finden, um Missverständnissen vorzubeugen. Es ist mit viel mathematischem Aufwand verbunden, eine geeignete Interpolation zwischen den darzustellenden Werten gewährleisten zu können (Abbildung 3.9).

Geometrie Die Geometrie hat die Aufgabe Auskunft über die Form eines Items zu geben. Punkte können dabei sowohl im zwei- sowie im dreidimensionalen Raum positioniert werden. Dieser Typ ist oft mit hierarchisch aufgebauten Daten verbunden. Attribute zu den jeweiligen Items sind nicht zwingend notwendig (Abbildung 3.10).

Cluster, Sets und Listen Neben den zuvor beschriebenen Typen, welche die am meisten verwendeten sind, gibt es noch andere Datensatztypen die hier kurz erwähnt werden sollen.

- Ein Set ist eine Untergruppe an Items.
- Die Liste beinhaltet Items in einer bestimmten Reihenfolge.

- Items mit gleichen Attributen werden zu Clustern zusammengeführt.

Attribut-Typen

Attribute werden grundsätzlich in zwei verschiedene Typen unterteilt. Es gibt kategorische und geordnete Attribute.

Kategorische Attribute Diese Attribute besitzen keine Ordnung im direkten Sinn, sind jedoch oft mit Hierarchie verbunden. Mögliche Attribute können Namen oder Formen sein. Hier ist es wichtig zwei Dinge voneinander unterscheiden zu können oder gegebenfalls bestimmen zu können, dass es sich um das gleiche handelt. Durch Ordnungsformen, wie das Alphabet, können auch diese Attribute in eine Reihung gebracht werden.

Geordnete Attribute Geordnete Attribute besitzen eine implizite Ordnung und können wiederum in ordinale und quantitative Daten unterscheidet werden. Ordinale Daten können zum Beispiel Kleidergrößen sein, diese besitzen eine vordefinierte Ordnung. Mit diesen Kenngrößen kann jedoch keine arithmetische Aufgabe gelöst werden, sie werden eher als eine Art Ranking behandelt. Die quantitativen Attribute sind messbare Werte und können so auch mathematisch in Beziehung zueinander gestellt werden. Geordnete Daten können sequenziell, auseinander laufend oder zyklisch angeordnet sein.

3.1.5 Zeichen und Kanäle

Zeichen sind primitive geometrische Formen, um Items darzustellen. Dabei kann es sich um Punkte, Linien oder Bereiche handeln. Kanäle werden verwendet, um den Items eine Bedeutung zuzuschreiben. Kanäle können einerseits eine Position (horizontal oder vertikal) näher beschreiben, eine Farbe aufweisen, eine bestimmte Form, eine Neigung und Größe (Länge, Fläche oder Volumen) anzeigen.

Kapitel 4

Stand der Technik

Im folgenden Kapitel werden einige Hilfsmittel sowie deren Funktionsweise für die Analyse der Qualitätsmerkmale für Webseiten beschrieben. Dabei ist zu beachten, dass dies nur ein Auszug der vorhandenen Anwendungen ist, um einen kurzen Einblick in den aktuellen Stand der Technik zu geben. Darüber hinaus werden verwandte Arbeiten und deren Vorgehensweise beschrieben.

4.1 HTML Validatoren

Bei der Validierung von HTML-Sourcecode geht es vorrangig um die Überprüfung der Einhaltung der Grammatikregeln der Markup Sprache. Als wichtigste Regel gilt das Schließen eines jeden geöffneten Tags. Ausnahmen dafür gibt es lediglich wenige, diese Elemente haben den Slash des schließenden Tags meist am Ende des Start-Tags. Wenn auch zu jedem Element ein End-Tag vorhanden ist, muss überprüft werden, ob die Verschachtlung der Elemente korrekt ist. Dabei wird analysiert, ob eine Überlappung der Elemente passiert ist. In der Spezifikation von HTML5 wird genau beschrieben, welche Elemente in welchen verschachtelt werden dürfen. So ist es nicht erlaubt, Block-Level Elemente innerhalb von inline-Elementen anzubringen.

Sind die verwendeten Elemente angesichts der oben genannten Punkte korrekt, werden weiters die angewendeten Attribute überprüft. Dabei ist wichtig zu beachten, dass nicht jedes Element jedes Attribut enthalten darf.

Für die Validierung wird der sogenannte DOCTYPE ausgelesen, anhand dessen die verwendete HTML Version erkannt werden kann [9].

4.1.1 W3C Markup Validator

Das World Wide Web Consortium bietet eine freie Online-Lösung für die Validierung von HTML. Dabei werden die oben beschriebenen Aspekte überprüft. Der Benutzer kann entscheiden, ob eine URL validiert werden soll, eine einzelne Datei oder nur ein kurzes Codesnippet durch Direkteingabe. Darüber hinaus können zusätzliche Optionen für die Validierung gesetzt werden. Der Dokumenttyp kann automatisch erkannt oder vom User festgelegt werden. Somit ist es möglich zu überprüfen, ob eine ältere Seite auch den Anforderungen neuer Dokumenttypen entspricht.

Nach der Analyse wird eine Liste an Fehlern und Warnungen ausgegeben. Neben der Fehlerbezeichnung wird außerdem angegeben in welcher Codezeile der Fehler aufgetreten ist. Zusätzlich sind Verlinkungen zu Hilfestellungen zur Lösung der Probleme enthalten. Über ein kleines Menü im oberen Bereich können die Fehler gefiltert und nach Kategorien aufgelistet werden.

4.1.2 Firefox Addons

Neben verschiedensten Online-Tools gibt es auch einige Addons für Browser. Nach deren Installation können Validierungsfehler in den Entwicklertools des jeweiligen Webbrosers gefunden werden. Auch hier wird eine Liste mit Querverweisen zu den zugrunde liegenden Regeln sowie den möglichen Hilfestellungen angezeigt. Einige dieser Addons basieren auf dem Validierungsverfahren des W3C und liefern somit die gleichen Ergebnisse. Dies bestätigt die Qualität der Überprüfung durch das World Wide Web Consortium.

4.1.3 Programme

Viele Editoren unterstützen den Entwickler während der Erstellung von HTML-Code. Je nach Programm können zusätzliche Features installiert und aktiviert werden. Das System kann automatisch die schließenden Tags erstellen und durch farbliche Hinterlegung auf mögliche Fehler hinweisen. Dies bietet eine gute Möglichkeit bereits ab Beginn der Programmierung auf einen sauberen Code zu achten und weniger Fehler zu produzieren.

4.2 CSS Validatoren

Entscheidend bei der Überprüfung der Stylesheets ist die verwendete Spezifikation. Valides CSS des Levels 1 ist ebenfalls valide im Level 2. Lediglich einige wenige Regeln wurden in ihrer Schreibweise oder den Eigenschaften leicht verändert. Mit neuen Versionen wurden zusätzliche Style-Eigenschaften definiert, die wiederum mit älteren Versionen nicht kompatibel sind.

Bei der Validierung der Stylesheets wird auf die Grammatik der jeweiligen Version und die richtige Verwendung der verschiedenen Werte geachtet.

Eine automatische Erkennung der Version ist im CSS nicht möglich, da es keine Deklaration am Beginn der Datei gibt und die verwendbaren Eigenschaften versionsübergreifend sind [9].

4.2.1 W3C CSS Validator

Ähnlich wie für die Validierung von HTML stellt das W3C auch ein Tool zur Überprüfung der Stylesheets zur Verfügung. Der Benutzer hat wiederum die Möglichkeit eine URL, eine Datei oder einen Ausschnitt per Direkteingabe zu analysieren.

4.3 Performance Analyse

Das wohl bekannteste Tool zur Untersuchung der Performance einer Website ist PageSpeed Insights[1] von Google. PageSpeed Insights erstattet einen genauen Bericht über die Leistung der Seite unter Berücksichtigung von mobilen bzw. Desktop-Geräten. Im Juli 2018 wurde das Tool erneuert und stellt den Entwicklern noch mehr Informationen zur Optimierung von Seitenladezeiten zur Verfügung.

Die Ladegeschwindigkeit wird in drei Bereiche unterteilt:

- Fast – schnell,
- Average – durchschnittlich,
- Slow – langsam.

Um den jeweiligen Wert berechnen zu können, werden zwei wichtige Kenngrößen herangezogen. Der First Contentful Paint, kurz FCP, misst die Zeit bis zu dem Punkt an dem für den Benutzer erste Inhalte, wie Text oder Bilder, sichtbar sind. Messungen unter 1,6 Sekunden werden dabei als schnell verzeichnet. Benötigt die Seite länger als 3 Sekunden wird diese als langsam eingestuft. Der zweite Wert ist der DCL und steht für DOM Content Loaded. Dieser beschreibt die Zeit, die benötigt wird, um das gesamte HTML Dokument vollständig zu laden ohne dabei CSS-Informationen und Grafiken zu berücksichtigen. Aus den beiden Kenngrößen wird der Median gebildet, um die jeweilige Einstufung vornehmen zu können.

Beim Optimierungsfaktor ist darauf zu achten, dass dieser für mobile Anwendungen anders berechnet wird als für Desktop-Geräte. Hier können Werte von 0 bis 100 erreicht werden, wobei darauf zu achten ist, dass man zumindest auf einen Wert von über 80 kommt [35].

Im Bereich Optimierungsvorschläge befinden sich nach Kategorien geordnet Vorschläge, die zu einer Verbesserung der Ladegeschwindigkeit führen sollen. Jeder Entwickler sollte dabei darauf achten, welches Optimierungspotenzial die Seite hat, da man keine qualitativen und funktionalen Abstriche machen sollte.

Zum Schluss wird dem Benutzer eine Übersicht über die Optimierungen geboten, die bereits in die Anwendung eingeflossen sind.

4.4 Analyse der Barrierefreiheit

Die automatische Überprüfung von HTML-Code ist einfach, da es genau definierte Regeln gibt auf die das System überprüft werden kann. Bei der Untersuchung einer Seite auf Barrierefreiheit ist dies ein schwierigeres Unterfangen. Viele Richtlinien können nicht von einer Software erkannt werden und bedürfen einer menschlichen Analyse. Nichtsdestotrotz gibt es einige Hilfsmittel, die es dem Entwickler leichter machen auch auf Barrierefreiheit zu testen und Anwendungen dahingehend zu verbessern.

[1] https://developers.google.com/speed/pagespeed/insights/?hl=de

4.4.1 WAVE

Das Web Accessibility Evaluation Tool WAVE[2] wurde bereits 2001 von WebAIM[3] entwickelt und steht frei zur Verfügung. Es bietet eine Online-Version sowie Zugriff auf eine API. Verwendet man die Browser-Methode, so wird man vorerst aufgefordert eine URL einzugeben. Danach untersucht das System die angegebene Seite auf Richtlinien der Barrierefreiheit. Die resultierende Seite teilt sich in zwei Bereiche (siehe Abbildung 4.1). Rechts wird die Seite, die untersucht wurde, mit einigen zusätzlichen Informationen angezeigt. Links daneben befindet sich eine Art Menü, in dem die aufgetretenen Probleme in folgende sechs Bereiche aufgeteilt werden:

- Errors,
- Allerts,
- Features,
- Structural Elements,
- HTML5 and ARIA,
- Contrast Errors.

In einer Liste werden diese Probleme dann genauer beschrieben und Hilfestellungen angeboten. Wird ein Fehler angeklickt, versucht das System den Bereich in der Website (rechter Teil) zu fokussieren.

Mit Hilfe des Kontrast-Registers können Hintergrund- und Vordergrundfarben auf ihre Tauglichkeit getestet werden. Die Anwendung bietet einen Colorpicker mit dessen Hilfe die Farben leicht abgewandelt werden können, damit sich ein ausreichendes Kontrastverhältnis ergibt.

4.4.2 tenon.io

Tenon.io[4] hat es sich zum Ziel gesetzt, Webseiten für jeden bedienbar zu gestalten. Daher bieten sie neben der online verfügbaren Überprüfung auch eine API an, die bereits in der Entwicklung von Webapplikationen zum Einsatz kommen soll. Dadurch können von Beginn an Probleme vermieden und dadurch Kosten gespart werden.

In der Dokumentation wird ausdrücklich darauf hingewiesen, dass auch dieses Tool automatische Testverfahren einsetzt und daher nicht alle Aspekte der Barrierefreiheit überprüfen kann. Eine vom Menschen durchgeführte Analyse ist stets zusätzlich erforderlich. Das System testet je nach Konformitätslevel auf unterschiedliche Aspekte. Im WCAG Level A werden 99 Tests an der Webapplikation ausgeführt. Für das Level AA kommen weitere fünf Testfälle hinzu. Will man die eigene Webanwendung auf das höchste WCAG Level testen, werden nochmals 23 Überprüfungen vorgenommen. Die Testszenarien werden stetig erweitert und abgeändert, um ein bestmögliches Ergebnis zu erzielen.

Das Testergebnis (siehe Abbildung 4.2) dieser Anwendung ist anders aufgebaut als das zuvor beschriebene Tool. Im oberen Bereich der Ergebnisseite werden drei Diagramme dargestellt. Das erste Ringdiagramm gibt Auskunft darüber, wie viele Testfälle

[2] http://wave.webaim.org/
[3] https://webaim.org/
[4] https://tenon.io/index.php

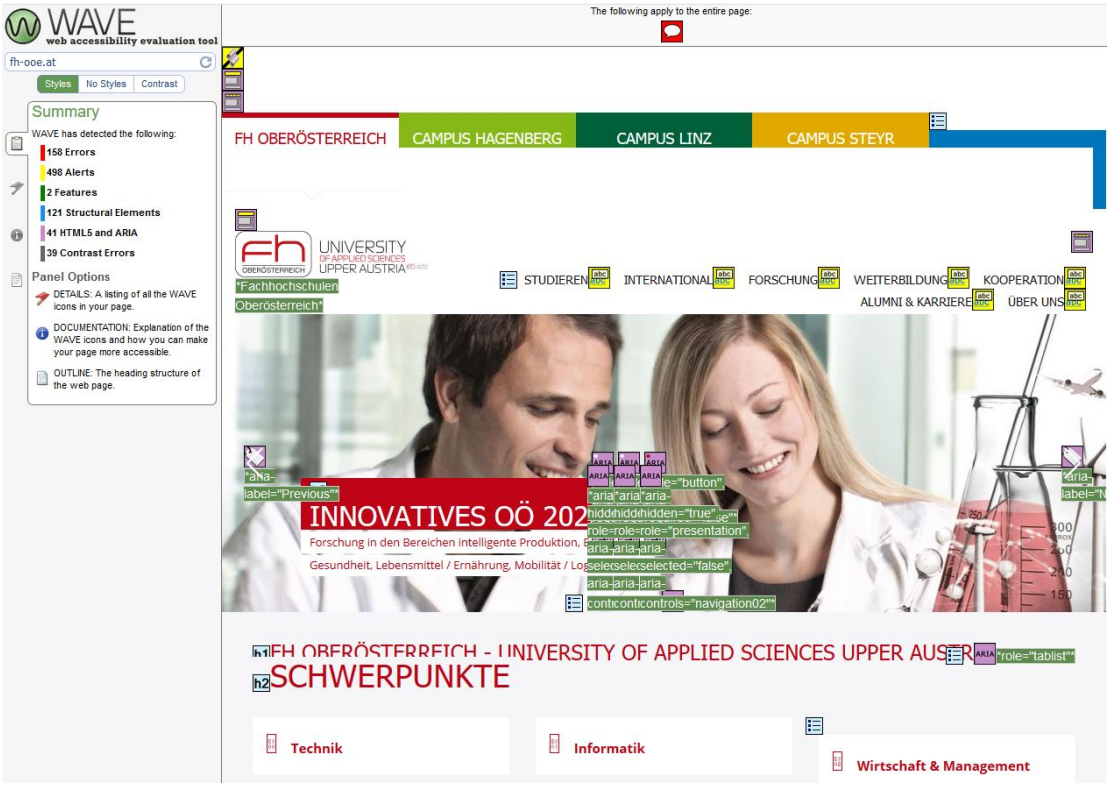


Abbildung 4.1: WAVE Report für www.fh-ooe.at

angewendet wurden und wie viele davon Fehler ausgeworfen haben bzw. positiv verlaufen sind. Das Kreisdiagramm stellt die aufgetretenen Fehler anhand ihrer Test-ID mengenmäßig ins Verhältnis. So ist in diesem Fall zu erkennen, dass 73,6% der aufgetretenen Fehler den selben Ursprung haben, durch dessen Veränderung ein weitaus besseres Ergebnis erzielt werden kann.

Im darunter liegenden Bereich wird eine Liste aller gefundenen Fehler ausgegeben. Links stehen die Code-Zeilen in denen das Problem aufgetreten ist sowie die Test-ID, die auf dieses Problem hinweist. In der Beschreibung wird angezeigt, ob es sich dabei um einen Fehler oder nur um eine Warnung handelt. Gleich daneben kann man erkennen welche Priorität der Lösung dieses Problems zugesprochen wird. Neben dem Titel des Fehlers wird zusätzlich angegeben, welcher WCAG Richtlinie dieser Testfall zugeordnet wird. Über einen Link können zusätzliche Hilfestellungen zur Problembehebung in Anspruch genommen werden.

4.5 Verwandte Arbeiten

Aufgrund des stetigen Wachstums des World Wide Web und die damit einhergehenden Anforderungen, wurde das Thema der Qualitätsüberprüfung von Webseiten bereits mehrfach erarbeitet. Als Auszug werden im folgenden Abschnitt zwei unterschiedliche Projekte vorgestellt, deren Grundlagen auch in die hier erläuterte Arbeit eingeflossen

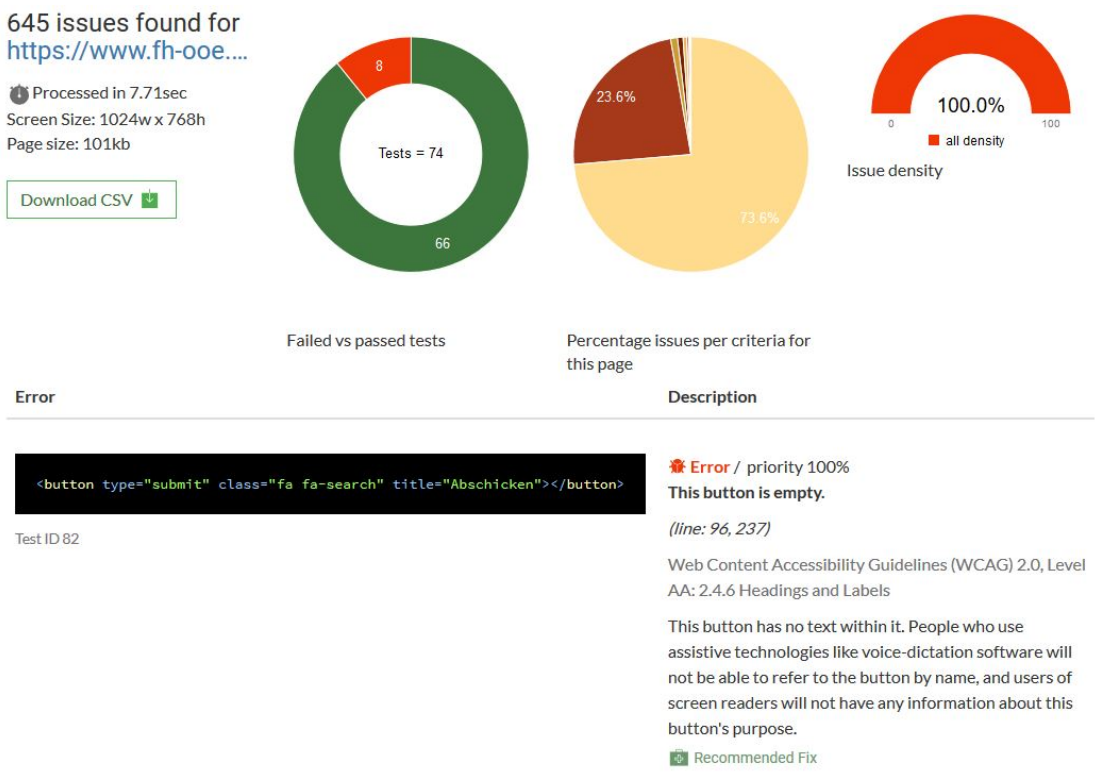


Abbildung 4.2: Tenon.io Testergebnis für www.fh-ooe.at

sind. Ziel beider Projekte ist es, die Qualität einer Webanwendung zu verbessern, um so den Erfolg einer Organisation zu steigern.

4.5.1 4-dimensionale Kriterien

Layla Hasan und Emad Abuelrub [4] untersuchten zahlreiche Ansätze, wie die Qualität einer Webseite untersucht werden kann. Während ihrer Recherchen hat sich gezeigt, dass es viele verschiedene Möglichkeiten gibt, verschiedene Webseiten hinsichtlich der Erwartungen zu analysieren. Doch jede Überprüfung konzentriert sich lediglich auf ein Problem aus einem bestimmten Blickwinkel. Das Ziel der beiden Forscher war es, ein Grundgerüst zu entwickeln, welches für jegliche Art von Webanwendung herangezogen werden kann. Mit dem Wissen der bereits vorhandenen Tools entwickelten sie vier Qualitätsdimensionen:

- Inhalt,
- Design,
- Organisation,
- Benutzerfreundlichkeit.

Zu jeder dieser Kategorien gibt es mehrere Unterpunkte, die zu beachten sind:

Inhalt

- **Zeitbezug**: Ist die Seite up to date? Wann wurden die Inhalte zuletzt bearbeitet?
- **Sachbezug**: Entspricht der Inhalt dem richtigen Maß an Details?
- **Mehrsprachigkeit**: Ist der Inhalt in mehreren Sprachen verfügbar?
- **Vielfalt der Präsentation**: Werden Inhalte in unterschiedlicher Form angeboten (Text-, Video- und Audioinhalte)?
- **Richtigkeit**: Sind Rechtschreib- oder Grammatikfehler enthalten?
- **Objektivität**: Enthalten die Inhalte eine politische, religiöse oder kulturelle Ausrichtung?
- **Kompetenz**: Ist eine physikalische Adresse der Organisation vorhanden? Sind Manager und Sponsoren klar ersichtlich?

Design

Aufgrund der visuellen Aufbereitung des Inhalts entscheiden sich viele Benutzer, wie lange sie eine Webseite verwenden und ob sie diese gegebenenfalls erneut besuchen.

- **Attraktivität**: Entspricht die Seite einem innovativem Design? Löst das Design positive Emotionen beim Benutzer aus? Werden Farben, Text und Bilder in einem ausgewogenen Maß eingesetzt?
- **Farbe**: Sind die Hintergrund- und Vordergrundfarben sinnvoll gewählt?
- **Bilder, Ton, Video**: Gibt es Text-Alternativen zu diesen Inhalten? Werden die Download-Zeiten negativ beeinflusst?
- **Text**: Ist der Text in einem einheitlichen Stil? Befinden sich abweichende Schriftgrößen nur in den Überschriften?

Organisation

- **Index**: Gibt es Verlinkungen zu allen Seiten einer Website?
- **Zuordnung**: Eine Navigation ist auf jeder Seite verfügbar?
- **Konsistenz**: Wird ein generelles Layout auf allen Seiten verwendet?
- **Links**: Rufen die Verlinkungen die gewünschte Funktion auf?
- **Logo**: Ist das Logo der Organisation klar und auf jeder Seite sichtbar?

Benutzerfreundlichkeit

- **Bedienbarkeit**: Ist die Seite einfach zu bedienen?
- **Verlässlichkeit**: Sind die Download-Zeiten gering? Ist die Adresse leicht zu merken?
- **Interaktivität**: Gibt es Hilfestellungen? Sind Fehlermeldungen klar formuliert?
- **Sicherheit**: Werden die aktuellen Sicherheitsstandards zur Übertragung von Daten verwendet?
- **Anpassung**: Kann die Seite für spezielle Bedürfnisse angepasst werden?

Die Ergebnisse dieses Projektes können mit Hilfe eines Fragebogens auf eine spezielle Webseite angewendet werden. Anhand dieser Merkmale können Webseiten miteinander verglichen, Optimierungen vorgenommen und Richtlinien für die Erstellung neuer Anwendungen festgelegt werden.

4.5.2 Qualitätsfaktoren

Auch für Doaa Nabil [7] stellte sich die Frage, wie die Qualität von Webanwendungen überprüft und verbessert werden kann. Es ist naheliegend, dass die Qualität einer Applikation als Erfolgsfaktor für die Organisation oder das Unternehmen gilt. Doch welche Eigenschaften muss eine Webanwendung mit sich bringen, um qualitativ hochwertig zu sein? Arbeiten aus der Vergangenheit konzentrieren sich meist nur auf einige wenige Eigenschaften, die aus einem speziellen Blickwinkel betrachtet werden. Doaa Nabil setzte sich das Ziel, Qualitätsfaktoren festzulegen, die für jegliche Webanwendung herangezogen werden können und so viele Blickwinkel wie möglich abdecken.

Als Grundlage seiner Arbeit dient die ISO-Norm 9126. Es handelt sich dabei um ein Qualitätsmodell, das sich an das von Jim McCall [34] im Jahr 1977 entwickelte Modell zur Analyse von Softwareprodukten anlehnt. McCall's Intention war es, die unterschiedlichen Bedürfnisse von User und Entwickler abzudecken. Er definierte daher drei Perspektiven, welche jeweils eigene Qualitätsfaktoren beinhalten:

- **Produkt-Überarbeitung**: Qualitätsfaktoren zum Verändern einer bestehenden Software.
- **Produkt-Wechsel**: Qualitätsfaktoren, zum Anpassen an ein neues Environment.
- **Produkt-Operationen**: Qualitätsfaktoren, die den Umfang der spezifizierten Forderungen messen.

Die ISO-Norm 9126 [27] wurde im Jahr 2011 von einer neuen Definition, der Norm 25010 abgelöst. Dennoch wird hier die für die beschriebene Arbeit verwendete, alte Norm erläutert. Diese besteht aus sechs Merkmalen, die die Qualität beeinflussen:

- Abänderbarkeit,
- Effizienz,
- Übertragbarkeit,
- Zuverlässigkeit,
- Funktionalität,
- Benutzbarkeit.

Diese Qualitätsmerkmale besitzen jeweils zahlreiche untergeordnete Faktoren. Manche dieser Faktoren stehen im Widerspruch zueinander. In solchen Fällen ist es die Aufgabe des Entwicklers, die für ein spezielles Projekt vorrangigen Faktoren zu wählen. Doaa Nabil verwendete diese sechs Faktoren und wies sie den zuvor identifizierten Blickwinkeln einer Webanwendung zu. Diese setzen sich aus der Sicht des Eigentümers, der Sicht des Entwicklers und der Sicht des Benutzers zusammen. Nach der Aufteilung der Hauptfaktoren entwickelte Nabil zusätzliche Subkategorien, die eine erfolgreiche Webanwendung mit sich bringen sollten.

Kapitel 5

Implementierung

Dieses Kapitel beschreibt die technische Aufbereitung des Analyse-Tools. In dieses Projekt wurden verschiedene Qualitätsmerkmale, die in den vorherigen Kapitel beschrieben worden sind, eingearbeitet und grafisch dargestellt. Der Mehrwert dieses Hilfsmittels besteht darin, dass Webentwickler auf einen Blick mehrere Aspekte einer Webseite überprüfen und gegebenenfalls auf Stärken und Schwächen einer Seite genauer eingehen können.

5.1 Anforderungen

Die Anforderungen an Webentwickler verändern sich rasant und sind sehr breit gestreut. Daher ist es wichtig einen Überblick zu behalten und erstellte Webseiten regelmäßig zu überprüfen. Durch neue Versionen von HTML, CSS und JavaScript werden Ausdrücke obsolet oder möglicherweise durch andere ersetzt.

5.2 Node.js

Als Laufzeitumgebung für diese Projekt wurde *node.js* verwendet[1]. Es eignet sich hervorragend für skalierbare Anwendungen und bietet mit dem *node package manager*, kurz npm, eine Vielzahl an Open-Source-Paketen. Durch die Vielzahl an Anleitungen und Dokumentationen, sowie den einfachen Aufbau, kann der Umgang mit *node.js* schnell erlernt werden [32]. Für dieses Projekt wurde *node.js* in der Version 6.10.0 verwendet, um auch mit allen verwendeten Modulen kompatibel zu sein.

5.3 React und Redux

Das Frontend der Applikation wurde mit *React*[2] realisiert. Dies ist eine JavaScript Bibliothek für die einfache Erstellung von interaktiven Benutzeroberflächen. Die Anwendung kann mit Hilfe von *React* in einzelne voneinander unabhängige Komponenten

[1] https://nodejs.org/en/
[2] https://reactjs.org/

gegliedert werden. Jede dieser Komponenten hat einen eigenen State und kann bei Veränderungen gezielt neu gerendert werden. Dabei werden stets nur die notwendigen Teile des DOM[3] verändert.

Durch die Einbindung von *Redux*[4] kann ein global verfügbarer Store herangezogen werden. Durch spezielle Methoden werden Daten in diesen gespeichert und in den einzelnen Komponenten ausgelesen und für das Rendering verwendet.

5.4 Analysen

Die unterschiedlichen durchzuführenden Analysen werden jeweils in eigenen Komponenten und Unterkomponenten durchgeführt. Somit bleibt das Projekt auch bei umfangreicher Komplexität übersichtlich und nachvollziehbar. In jeder einzelnen Komponente werden Teile der resultierenden Anwendung gerendert und in der übergeordneten Komponente an die richtige Stelle gesetzt. Wichtig ist, dass die jeweiligen Komponenten zum richtigen Zeitpunkt generiert bzw. neu durchgeführt werden. Dafür werden von *React* spezielle Methoden zur Verfügung gestellt. In diesem Projekt wurden hauptsächlich die beiden Funktionen `componentDidMount` und `componentDidUpdate` verwendet. Diese werden in einer von *React* definierten Reihenfolge bei der Erstellung bzw. Veränderung einer Instanz automatisch ausgeführt [23].

Viele Komponenten greifen auf den global verfügbaren *store* von *Redux* zu. Zusätzlich werden in den in jeder Instanz verfügbaren *state* Informationen gespeichert, die lediglich in dieser expliziten Komponente von Bedeutung sind.

5.4.1 HTML Codeanalyse

Für die Validierung der HTML-Information der angegebenen Website werden zwei Komponenten verwendet. Die erste Komponente ist für die grundsätzliche Analyse zuständig, die zweite für die Erstellung des dazugehörigen Balkendiagramms. Für die Validierung muss vorerst die verwendete URL aus dem globalen *store* ausgelesen werden. Damit wird die Funktion `validateHTML` aufgerufen. Diese beinhaltet ein Objekt mit Optionen sowie ein *Promise* für die asynchrone Abwicklung der Validierung. Dieses sogenannte *Promise* kann sich in drei verschiedenen Zuständen befinden [30]:

- *pending*: initialer Status,
- *fulfilled*: Aktion wurde erfolgreich durchgeführt,
- *rejected*: Aktion wurde nicht erfolgreich durchgeführt.

Für die Validierung des HTML-Codes weist das *Promis* folgenden Aufbau auf:

```
1 htmlValiator(options)
2         .then((data) => {
3           this.setState({
4             data: data,
5             url: this.props.url,
6           });
7           this.countErrors(data);
8         })
```

[3] Document Object Model
[4] https://redux.js.org/

```
9         .then(() => this.generateDataForBarchart())
10        .catch((error) => {
11          this.setState({
12            error: error,
13          });
14        });
```

Kann dieses Promise aus dem npm Modul *html-validator* [13] in den *fulfilled* Zustand übergehen, so werden die daraus erhaltenen Daten in den lokalen *state* gespeichert und die Funktion `countErrors` wird aufgerufen. Ein weiteres *then* sorgt dafür, dass nach Erhalt der Daten eine außerhalb definierte Methode zum Generieren der Datensätze für das Balkendiagramm ausgeführt wird. Kann die Operation nicht erfolgreich durchgeführt werden, wird im *catch*-Bereich eine Fehlermeldung gespeichert.

Die Methode `countErrors` durchschreitet die erhaltenen Datensätze und erhöht je nach Typ den Counter für Fehler bzw. Warnungen. Zusätzlich werden zwei Objekte in den lokalen *state* gespeichert, die jeweils nur Meldungen zu einem Fehlertyp enthalten. Dies bildet die Grundlage für die Generierung der Datensätze der Visualisierung.

In der Funktion `gererateDataForBarchart` werden die Fehler untersucht und anhand der Fehlerkategorie verglichen. Da es in diesem Fall keine einheitliche Kategoriebezeichnung gibt, werden Textteile miteinander verglichen und als eine Kategorie gewertet – falls die Übereinstimmung erfolgreich ist. Die dafür zuständige Schleife sieht folgendermaßen aus:

```
1   for (let i = 0; i < errors.length; i++) {
2       currentErrorMessage = errors[i].message;
3       for (let index in errorTypes) {
4         if (stringSimilarity.compareTwoStrings(currentErrorMessage, index) === 1) {
5           errorTypes[index] = errorTypes[index] + 1;
6           found = true;
7         }
8       }
9       if (!found) errorTypes[currentErrorMessage] = 1;
10
11      found = false;
12
13      if (i === errors.length - 1) {
14        this.setState({
15          errorTypes: errorTypes,
16        });
17      }
18    }
```

Am Ende befindet sich die für *React* notwendige `render`-Funktion, die für die tatsächliche Veränderung des DOM verantwortlich ist und ebenfalls vom System automatisch ausgeführt wird. Darin werden die Filteroptionen für Fehler und Warnungen erstellt und die Komponente für die Generierung des Balkendiagramms aufgerufen. Danach wird die Liste an gefundenen Fehlern ausgegeben.

5.4.2 Stylesheet Analyse

Die Untersuchung der Stylesheets der angegebenen Seite erfolgt auf ähnliche Art und Weise wie die Analyse des HTML-Codes. Die vom npm-Modul w3c-css [12] zur Verfügung gestellte Funktion `validate` bietet zwar nicht die Möglichkeit der Verwendung

eines *Promise*, doch kann eine ähnliche Funktionalität erreicht werden. Zuerst wird überprüft, ob es bei der Durchführung zu Fehlern gekommen ist. Diese werden gegebenenfalls in der Konsole ausgegeben. Ist dies nicht der Fall, so werden die erhaltenen Daten in den lokalen *state* gespeichert. Die manuelle Erfassung der Fehler und Warnungen ist nicht notwendig, da die Informationen bereits in dieser Form geliefert werden. Der nachstehende Code beinhaltet den beschriebenen Ablauf:

```
cssValiator.validate(url, (err, data) => {
      if (err) {
        console.log('crashed', err);
      } else {
        this.setState({
          errors: data.errors,
          errorcount: data.errors.length,
        });
        this.setState({
          warnings: data.warnings,
          warningcount: data.warnings.length,
          url: url,
        });
        this.generateDataForBarchart();

      }
    });
```

Danach werden in der Funktion für die Erstellung der Datensätze für das Balkendiagramm lediglich die Fehler einer Kategorie gezählt und mit der Fehlerbezeichnung in ein neues JavaScript Objekt gespeichert, das später für die Visualisierung herangezogen werden kann.

Auch beim Rendern der CSS-Validierung werden Filteroptionen für Fehler und Warnungen angeboten. Darunter befindet sich wiederum das Balkendiagramm in Anlehnung an die *Pareto*-Analyse, sowie eine ausführliche Liste der gefundenen Mängel. Jedes gefundene Problem wird, sowohl bei der HTML- als auch bei der CSS-Analyse, so angeführt, dass links ein Codesegment zu sehen ist, in dem der Fehler gefunden worden ist. Rechts daneben wird dann erläutert, wo das Problem liegt und wie dieses gelöst werden kann. Gegebenenfalls werden auch zusätzliche Hilfestellungen mittels Verlinkungen angeboten.

5.4.3 Performance-Test

Für die Geschwindigkeitsanalyse der Webseite wird das von Google entwickelte Tool *PageSpeed Insights* [21] herangezogen. Wie auf Seite 26 beschrieben, gibt es hier die Unterscheidung zwischen mobilen Geräten und Desktop-Anwendungen. Die von der API kommenden Daten werden getrennt im globalen *store* abgespeichert, um später weiter verarbeitet werden zu können. Als zusätzliche Information werden die verwendeten Ressourcen, wie JavaScript, Bilder, CSS und HTML, in einem Tortendiagramm anhand ihrer Datenmenge dargestellt. Um zu den erforderlichen Daten zu gelangen wird folgendes Code-Segment benötigt:

```
fetch(pagespeedUrl)
        .then(res => res.json())
        .then((out) => {
```

```
4           this.setState({
5             response: out,
6             pageStats: [
7               {label: 'JavaScript', color: 'rgb(251, 201, 141)', count: out.
    pageStats.javascriptResponseBytes},
8               {label: 'Images', color: 'rgb(239, 129, 96)', count: out.pageStats.
    imageResponseBytes},
9               {label: 'CSS', color: 'rgb(219, 71, 106)', count: out.pageStats.
    cssResponseBytes},
10              {label: 'HTML', color: 'rgb(159, 47, 127)', count: out.pageStats.
    htmlResponseBytes},
11            ],
12          })
13        })
14        .then(() => {
15          this.props.dispatch(setPagespeeddata(this.state.response));
16        })
17        .catch(err => {
18          throw err
19        });
```

In der Hauptkomponente der Performance-Analyse wird lediglich die API aufgerufen und die resultierenden Daten gespeichert. Dies erfolgt mit Hilfe der *Fetch API* [29], die für den Aufruf der `pagespeedUrl` zuständig ist. Die `fetch`-Methode liefert ein *Promise* [30], welches die Antwort auf die Anfrage beinhaltet. Beim Auftreten eines Fehlers wird der `catch`-Block ausgeführt und eine Fehlermeldung generiert.

In der `render`-Funktion werden zwei Komponenten für die Generierung der Visualisierung für mobile bzw. Destkop-Geräte aufgerufen. Jede dieser beiden Teile erstellt mit den entsprechenden Datensätzen ein Säulendiagramm mit Hilfe von *d3.js*, welches die gefundenen Probleme der Größe nach geordnet darstellt.

5.4.4 Untersuchung der Barrierefreiheit

Die Analyse der Barrierefreiheit wird, wie auf Seite 27 mit der von tenon.io zur Verfügung gestellten API, durchgeführt. Für die Verwendung dieser API werden zwei Objekte benötigt. Einerseits ein `data`-Objekt, welches die zu untersuchende Adresse sowie einen API-Key enthält und andererseits ein Objekt mit einigen Optionen:

```
1   const data = querystring.stringify({
2     url: url,
3     key: '***',
4   });
5
6   const options = {
7     host: 'tenon.io',
8     port: 443,
9     path: '/api/',
10    method: 'POST',
11    headers: {
12      'Content-Type': 'application/x-www-form-urlencoded',
13      'Content-Length': Buffer.byteLength(data)
14    }
15  };
```

Mittels `http`-Request und den zuvor definierten Objekten wird eine Anfrage an die API gestellt. Auch hier wird wiederum ein *Promise* verwendet und bei erfolgreicher Abfrage werden die resultierenden Informationen im lokalen *state* zwischengespeichert:

```
1  this.httpRequest(options, data)
2        .then(function (myData) {
3          self.setState({
4            data: JSON.parse(myData),
5            url: self.props.url,
6          });
7          this.countByType(JSON.parse(myData).resultSet)
8        })
9        .catch(err => {
10         throw err
11 });
```

Nach Erhalt der Daten werden diese in der Methode `countByType` anhand ihrer Test-ID gezählt, um später wie auch bei der HTML und CSS-Analyse ein Balkendiagramm in Anlehnung an die Pareto-Analyse erstellen zu können.

In der `render`-Methode dieser Komponente wird eine Unterkomponente aufgerufen, die ein kleines Balkendiagramm erstellt, welches die Information enthält, wie viele Testfälle positiv bzw. negativ ausgefallen sind. In einer weiteren Komponente wird das komplexere Balkendiagramm mit den zuvor ermittelten Daten erstellt. Darunter folgt die Auflistung aller von der API resultierenden Fehler. Auch hier wird links ein Code-Ausschnitt angezeigt, um den Fehler möglichst schnell finden zu können. Rechts daneben werden Informationen zum Fehlertyp sowie Hilfestellungen angezeigt. All diese Daten werden aus dem *JSON*-Objekt, das die API-Anfrage geliefert hat, ausgelesen.

5.5 Visuelle Aufbereitung

Die Analyse einer Webapplikation auf verschiedene Qualitätsmerkmale bildet einen wichtigen Grundstein für die Optimierung dieser Anwendung. Resultierende Listen an Fehlern wirken oft sehr verwirrend, vor allem für Laien und Entwickler, die noch nicht lange in diesem Bereich tätig sind. Um einen guten Überblick zu schaffen, bieten sich Visualisierungen der Daten an. Durch ein solches Verfahren können ähnliche Informationen zu Fehlergruppen zusammengeschlossen werden und die Informationsflut auf die wichtigsten Fakten heruntergebrochen werden.

5.5.1 D3.js

Um Daten im Browser visuell darstellen zu können, wurde die JavaScript Bibliothek d3.js von Mike Bostock herangezogen. Mit Hilfe dieser Bibliothek können Daten mittels HTML, SVG und CSS direkt in ein bestehendes Webprojekt eingebunden werden. D3.js benötigt keine weiteren Abhängigkeiten und kann direkt im Browser verwendet werden. Durch Animationen und Veränderungen in den Datensätzen können diese zum Leben erweckt werden, wobei das Document Object Model direkt zur Laufzeit verändert wird. Aufgrund des einfachen Aufbaus der Bibliothek kann d3.js äußerst schnell arbeiten und mühelos auch große Datensätze behandeln. Die Herangehensweise an Visualisierungen mit d3.js ist vorerst etwas ungewohnt, doch dank einer Vielzahl an Beispielen mit

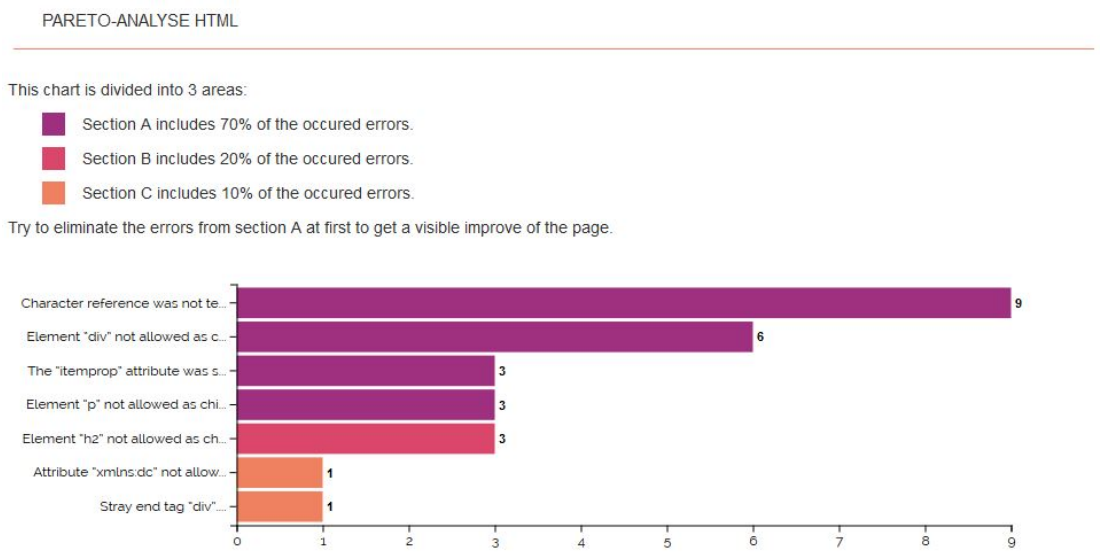


Abbildung 5.1: Balkendiagramm mit Hilfe der Pareto-Analyse für HTML-Validierung

Programmcode und Informationen zu den verwendeten Daten, können schnell eigene Visualisierungen erstellt werden [16].

5.5.2 Visuelle Darstellung mittels der Pareto-Analyse

Die in Kapitel 2.1.2 beschriebene Pareto-Analyse wird für die Darstellung der Fehlerdiagnose der HTML- und CSS-Validierung sowie der Überprüfung der Barrierefreiheit der Webseite verwendet. Für die Datengrundlage werden die Fehler kategorisiert und anschließend gezählt. Anhand dieser Information kann ein Balkendiagramm mit abnehmenden Längen erstellt werden. Nach der Sortierung der Datensätze anhand der errechneten Häufigkeiten wird das Diagramm in drei Bereiche eingeteilt. Der Bereich A enthält dabei rund 70% der aufgetretenen Fehler, der B-Bereich entsprechend 20% und der Teil C 10%. Diese Eingrenzung wird mit Hilfe der drei aufgezeigten Farben, in denen die Balken hinterlegt werden, vorgenommen (siehe Abbildung 5.1). Die Möglichkeit, die Anzeige des Validierungsergebnisses anhand der Filteroption für Fehler und Warnungen zu verändern, spiegelt sich in Echtzeit auch im Diagramm wider. Der Webentwickler sieht durch diese Darstellung auf einen Blick, welche Probleme bei Behebung die größte Verbesserung mit sich bringen. In vielen Fällen schleicht sich ein Fehler mehrmals ein, durch dessen Richtigstellung das Ergebnis der Validierung maßgeblich beeinflusst werden kann.

5.5.3 Säulendiagramm zur Performance-Beurteilung

Die erhaltenen Informationen zur Ladegeschwindigkeit der untersuchten Webseite beinhalten ein Objekt mit einer Reihe an Unterobjekten, die jeweils ein erkanntes Problem schildern sowie Lösungsansätze zur Optimierung bieten. Im Folgenden wird ein Ausschnitt eines gefundenen Optimierungspotenzials aufgezeigt:

```
"OptimizeImages": {
    "localizedRuleName": "Optimize images",
    "ruleImpact": 2.5435,
    "groups": [
      "SPEED"
    ],
    "summary": {
      "format": "Properly formatting and compressing images can save many bytes of data."
    },
    "urlBlocks": [
      {
        "header": {
          "format": "{{BEGIN_LINK}}Optimize the following images{{END_LINK}} to reduce their size by {{SIZE_IN_BYTES}} ({{PERCENTAGE}} reduction).",
          "args": [
            {
              "type": "HYPERLINK",
              "key": "LINK",
              "value": "https://developers.google.com/speed/docs/insights/OptimizeImages"
            },
            {
              "type": "BYTES",
              "key": "SIZE_IN_BYTES",
              "value": "24.8KiB"
            },
            {
              "type": "PERCENTAGE",
              "key": "PERCENTAGE",
              "value": "19%"
            }
          ]
        },
        "urls": [
          {
            "result": {
              "format": "Compressing {{URL}} could save {{SIZE_IN_BYTES}} ({{PERCENTAGE}} reduction).",
              "args": [
                {
                  "type": "URL",
                  "key": "URL",
                  "value": "https://www.fh-ooe.at/typo3conf/ext/theme/Resources/Public/Frontend/assets/img/partners/bizup.png"
                },
                {
                  "type": "BYTES",
                  "key": "SIZE_IN_BYTES",
                  "value": "5.1KiB"
                },
                {
                  "type": "PERCENTAGE",
                  "key": "PERCENTAGE",
                  "value": "19%"
                }
              ]
```

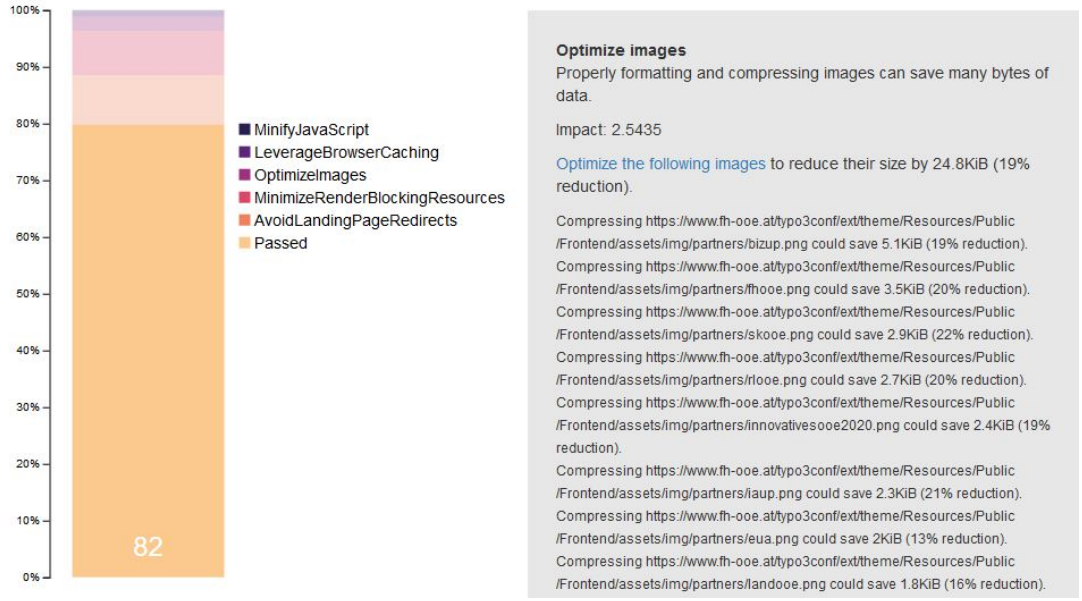


Abbildung 5.2: Gestapeltes Säulendiagramm für Pagespeed-Daten

```
53              }
54            },...
55          ]
56        }
57      ]
58 }
```

Neben dem Titel der gefundenen Richtlinie wird auch der Grad der Auswirkung auf die Gesamtauswertung der Geschwindigkeit angegeben. In der Zusammenfassung ist eine Kurzdefinition der Regel hinterlegt. Danach folgt in diesem Fall eine Liste an Bildern inklusive URL, die optimiert werden können und somit Speicherkapazität einsparen können. Bei jeder der aufgelisteten Grafiken wird angegeben wie viel durch eine Kompression eingespart werden kann.

Anhand der Informationen über den Einfluss zur Gesamtbewertung wird ein gestapeltes Säulendiagramm erstellt. Die Gesamthöhe der Säule repräsentiert den maximal erreichbaren Geschwindigkeitslevel von 100. Die einzelnen Teile des Diagramms werden anhand ihrer Größe absteigend übereinander gestapelt (siehe Abbildung 5.2).

Wird ein Teilbereich der Säule mit der Maus berührt oder ein Titel in der Legende, so wird im rechten Bereich angezeigt, um welche Richtlinie es sich handelt. Hier werden die Daten aus dem JavaScript-Objekt so zusammengebaut, dass sie für den Benutzer lesbar sind. Eine Hilfestellung zur Optimierung wird als externer Link hinterlegt. Im Beispiel werden alle Bilder, die komprimiert werden sollen, aufgelistet.

Diagramm der verwendeten Ressourcen

Google PageSpeed Insights liefert auch Informationen über die für die Webseite verwendeten Datenmengen mit. In einem Ringdiagramm werden die benötigen Byte für

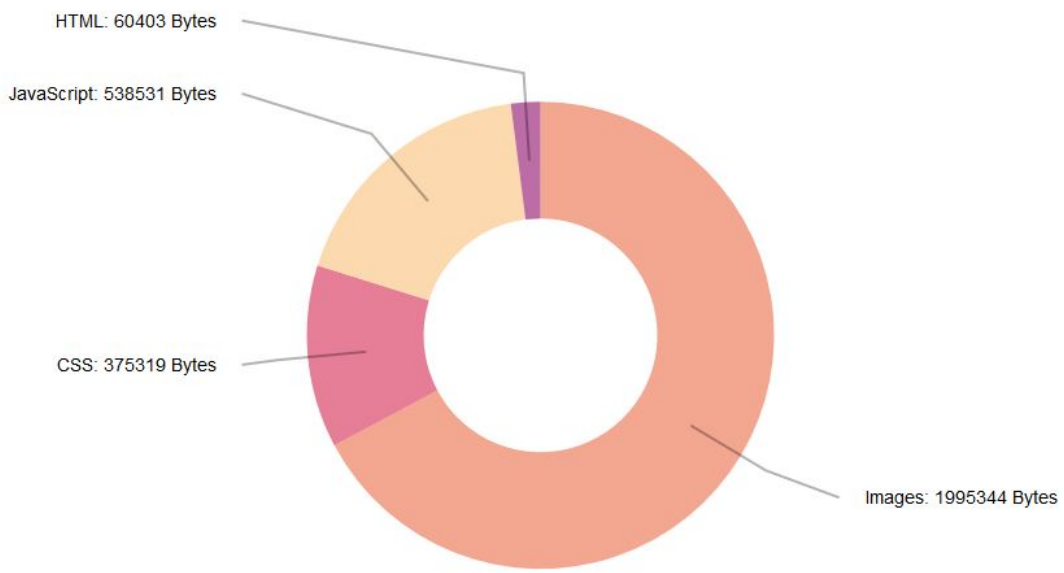


Abbildung 5.3: Ressourcenaufteilung in Bytes

HTML, CSS, JavaScript und Bild-Ressourcen dargestellt (siehe Abbildung 5.3).

5.5.4 Visualisierung der verwendeten CSS-Klassen

In diesem Projekt wird neben der Visualisierung der Fehlerinformationen einzelner Analysen auch versucht, Informationen aus den vorhandenen Dateien zu generieren und diese grafisch darzustellen. In einem separaten Parsingprozess werden alle verwendeten Klassennamen der HTML-Elemente ausgelesen und deren Häufigkeit gezählt. Mit Hilfe eines Bubble-Diagramms (siehe Abbildung 5.4) werden die erfassten Daten so dargestellt, dass die Größe sowie die Farbe der Kreise auf die Anzahl der Verwendung des Namens schließen lässt. Zusätzlich wird der Klassenname und diese Anzahl in den Kreis eingetragen. Da bei einer Vielzahl an verschiedenen Namen die Durchmesser der Kreise sehr klein ausfallen können, wird neben der Darstellung eine Liste aller vorkommenden Klassennamen ausgegeben. In eckigen Klammern kann auch hier die Anzahl abgelesen werden. Für eine bessere Übersicht wurden die Klassen aufgrund ihrer Häufigkeit geordnet.

Die Darstellung der Klassennamen soll dem Entwickler helfen, eigenständig zu überprüfen, ob tatsächlich alle Namen notwendig sind und ob ähnlich benannte Klassen möglicherweise zusammengefasst werden können. Auch die unterschiedliche Schreibweise eigentlich gleicher Klassennamen durch Tippfehler oder verschiedene Entwickler kann auf einen Blick erkannt und ausrangiert werden.

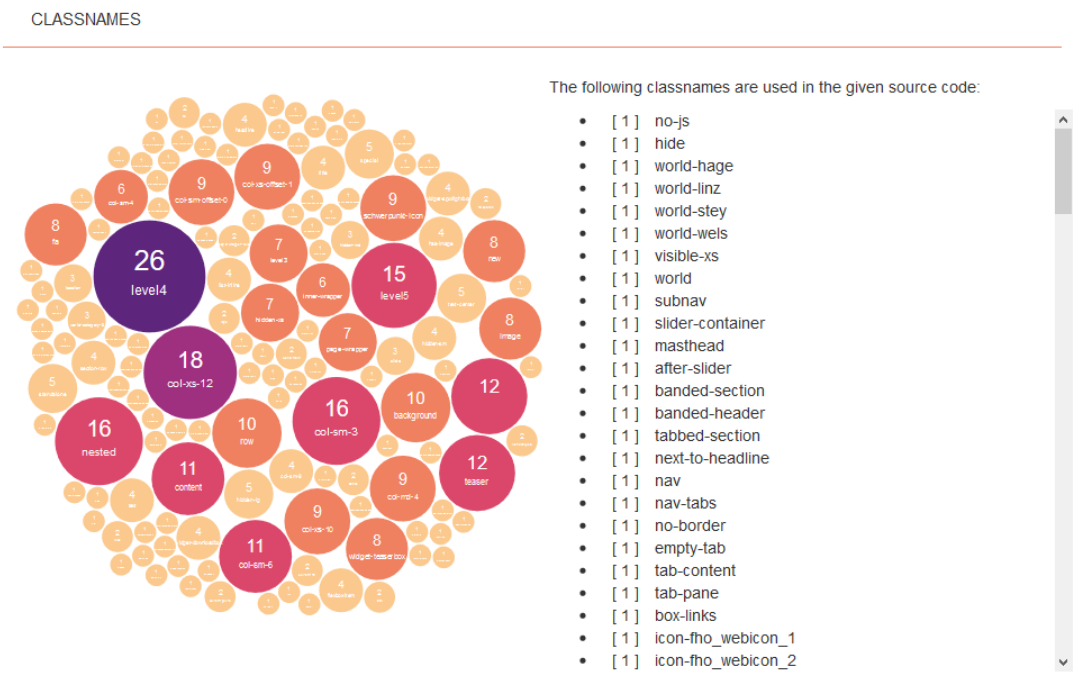


Abbildung 5.4: Verwendete Klassennamen

#RRGGBB

Abbildung 5.5: Hexadezimale Farbdefinition

5.5.5 Verwendete Farben

In einem weiteren Schritt werden auch die im HTML eingebundenen Stylesheets genauer betrachtet. Mit Hilfe eines CSS-Parsers kann eine Datei als JavaScript-Objekt geladen und anschließend weiter verarbeitet werden. Dieses Objekt wird auf drei verschiedene Eigenschaftstypen untersucht und speichert Farbcodierungen für Hintergrund, Schriften und Rahmen. Dabei werden die unterschiedlich verwendeten Farbdefinitionen in die hexadezimale Codierung übertragen. Diese Schreibweise besteht aus drei aufeinander folgende, zweistellige Hexadezimalzahlen. Jedes Ziffernpaar ist dabei für einen Farbbereich des RGB-Farbraums zuständig (siehe Abbildung 5.5) und kann 256 verschiedene Werte repräsentieren.

Dem Farbcode wird ein Rautensymbol vorangestellt, um ihn eindeutig als solchen interpretieren zu können.

Nach der Vereinheitlichung der Farbdefinitionen können die verwendeten Farben

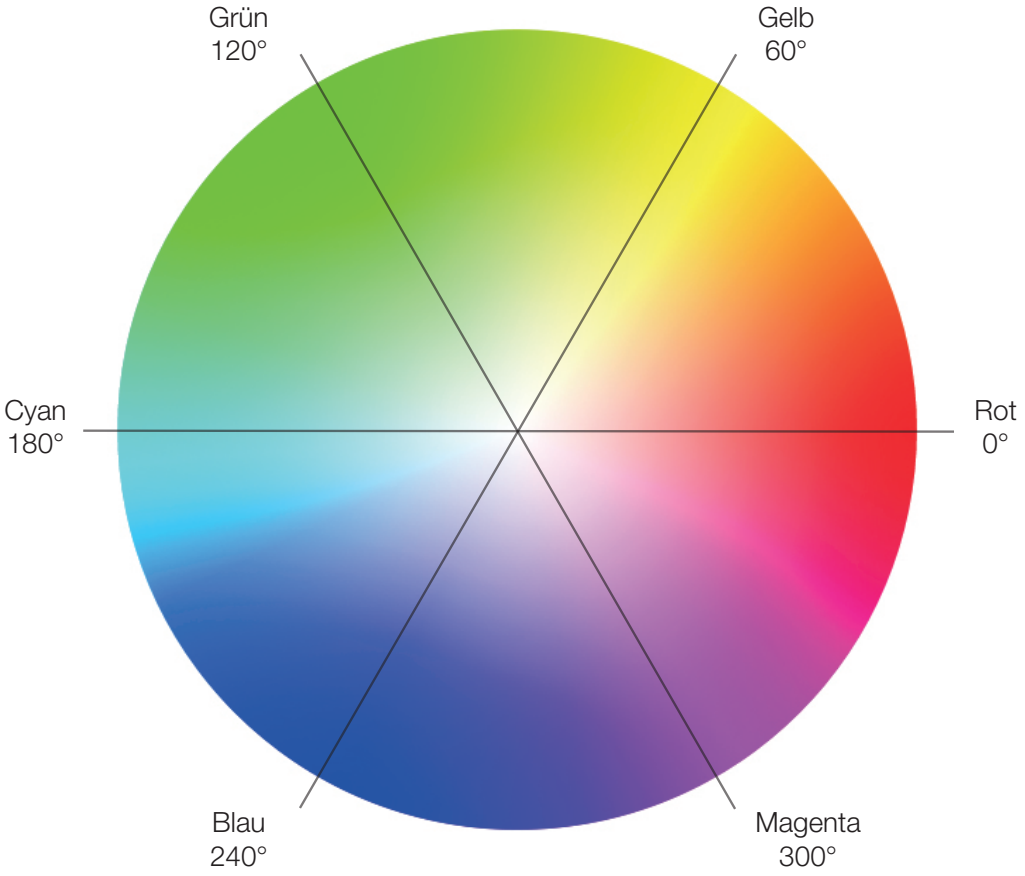


Abbildung 5.6: Farbkreis

auch anhand ihrer Häufigkeit gezählt werden. Diese gesammelten Informationen werden in einer Übersichtsgrafik für den Benutzer dargestellt. Da komplexere Webseiten oft eine Vielzahl an Farben verwenden, werden diese vor dem Anzeigen für den User sortiert. Die Sortierung basiert auf dem Farbwert jeder einzelnen Farbdefinition. Der Farbwert (englisch hue) beschreibt den Farbwinkel am Farbkreis (siehe Abbildung 5.6) [6].

So steht 0° für Rot und 120° beispielsweise für Grün. Anhand dieser Farbinformation können die Farbwerte mit folgendem Programmcode und dem npm-Modul *color-js* [17] aufsteigend sortiert werden:

```
1 //sort by color
2     dataColor.sort((a, b) => {
3       colorA = colorJs(a.COLOR).getHue();
4       colorB = colorJs(b.COLOR).getHue();
5       return colorA - colorB;
6     });
```

Im letzten Schritt werden die sortierten Farbwerte mit Hilfe kleiner, passend eingefärbter Quadrate in den drei Kategorien Hintergrund, Schrift und Rahmen grafisch

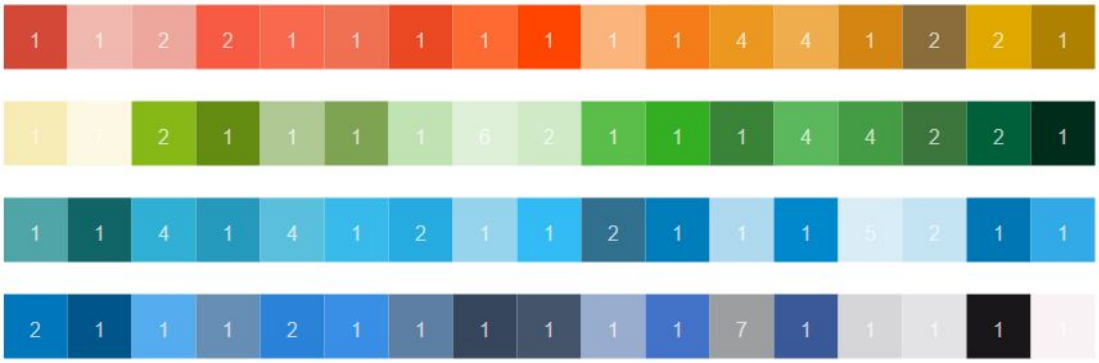


Abbildung 5.7: Ausschnitt der verwendeten Hintergrundfarben der Seite www.fh-ooe.at

dargestellt. Im Zentrum des Quadrates steht die Anzahl der Verwendungen der jeweiligen Farbe. Durch diese Visualisierung ist es dem Entwickler möglich, zu prüfen wie oft eine Farbe eingesetzt worden ist. Zudem kann überprüft werden, ob nahezu identische Farben mit unterschiedlichen Farbcodierungen angewendet worden sind (siehe Abbildung 5.7).

5.5.6 Darstellung der HTML-Struktur

In den vorhergehenden Abschnitten wurden Teilaspekte der untersuchten Webseite grafisch repräsentiert. Eine einen Überblick verschaffende Darstellung fehlt jedoch noch. Das Grundgerüst einer Webseite wird durch die Hierarchie in den HTML-Elementen definiert und dadurch meist als Baum visualisiert. Bei großen, komplexen Seitengerüsten kann dieser Baum eine Unzahl an Verzweigungen mit mehreren wiederum verschachtelten Elementen enthalten. Bei dieser Informationsdichte den Überblick zu bewahren, ist für Laien nahezu unmöglich und selbst für Experten eine zeitintensive Aufgabe. Durch Abstraktion der Daten und einen anderen Ansatz können die Datensätze jedoch auch auf eine ganz andere Art und Weise veranschaulicht werden. Im Zuge dieser Arbeit wurde ein sogenanntes Chord-Diagramm gewählt. Dieser Diagrammtyp visualisiert die internen Verbindungen zwischen Elementen. Entlang eines Kreises werden alle in den Daten vorkommenden Elemente arrangiert. Die Beziehungen werden mit Hilfe von Kurven, die von einem zum anderen Element verlaufen, dargestellt. In Abbildung 5.8 wird ein vereinfachtes Modell eines solchen Diagramms gezeigt. Durch Verwendung von Farben können gegebenenfalls zusätzliche Informationen vermittelt werden [19].

Grundlage für die Erstellung einer solchen Darstellung sind die Daten, die in einer bestimmten Form vorliegen müssen, um danach die Visualisierung zu generieren. Für den Anwendungsfall der HTML-Datenstruktur werden bei jedem Datensatz zwei HTML-Elemente gespeichert. Zum einen das Eltern-Element und zum anderen dessen Kind-Element. Somit kann später eine Verbindung vom übergeordneten Element zum Kind-Element gezogen werden.

Wie ein solches Diagramm für eine HTML-Seite aussehen kann, sieht man in Abbildung 5.9. Hier wird der Aufbau der Startseite der Fachhochschule Oberösterreich her-

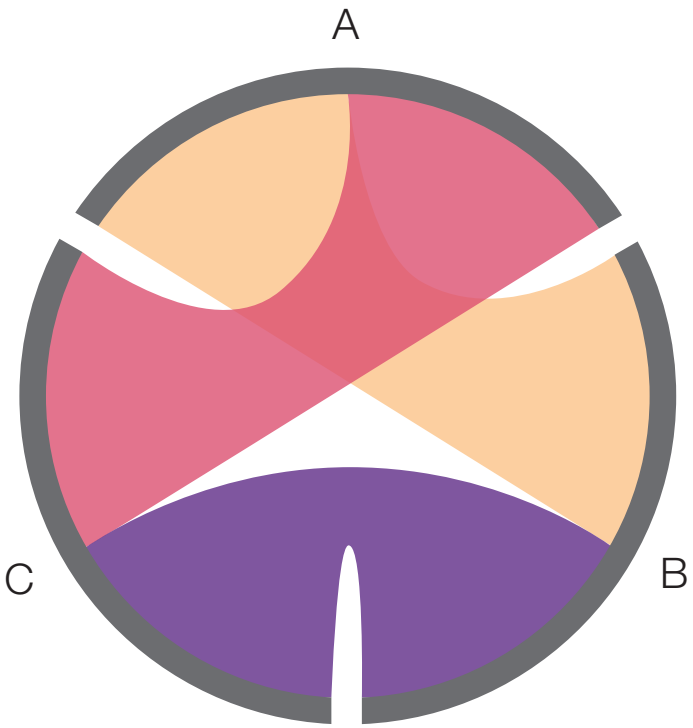


Abbildung 5.8: Aufbau Chord-Diagramm[19]

angezogen. Alle in diesem HTML-Dokument verwendeten Elemente werden im Kreis angeordnet. Danach werden die Beziehungen zwischen den Elementen gezeichnet. Mit Hilfe der Maus kann der Benutzer Detailinformationen aus der Grafik einblenden. Im oberen Bereich wird in diesem Beispiel angezeigt, dass sich insgesamt 89 `div`-Elemente im Dokument befinden. Zusätzlich kann die Verschachtlung der `div`-Elemente abgelesen werden. Es wird angezeigt, dass 3 `span`-Elemente in `div`-Elementen enthalten sind. Webentwickler werden bemerken, dass auch 6 `div`-Elemente in `span`-Elementen enthalten sind. Dies ist in den Richtlinien der Verschachtlung von HTML-Elementen aber untersagt, da es sich bei `div`-Elementen um Blockelemente handelt und diese nicht in *inline*-Elementen, wie einem `span`-Element, enthalten sein dürfen. Durch diesen Diagrammtyp können also auf schnellem Wege die Hierarchie der HTML-Datei überprüft und Fehler in der Struktur bereinigt werden.

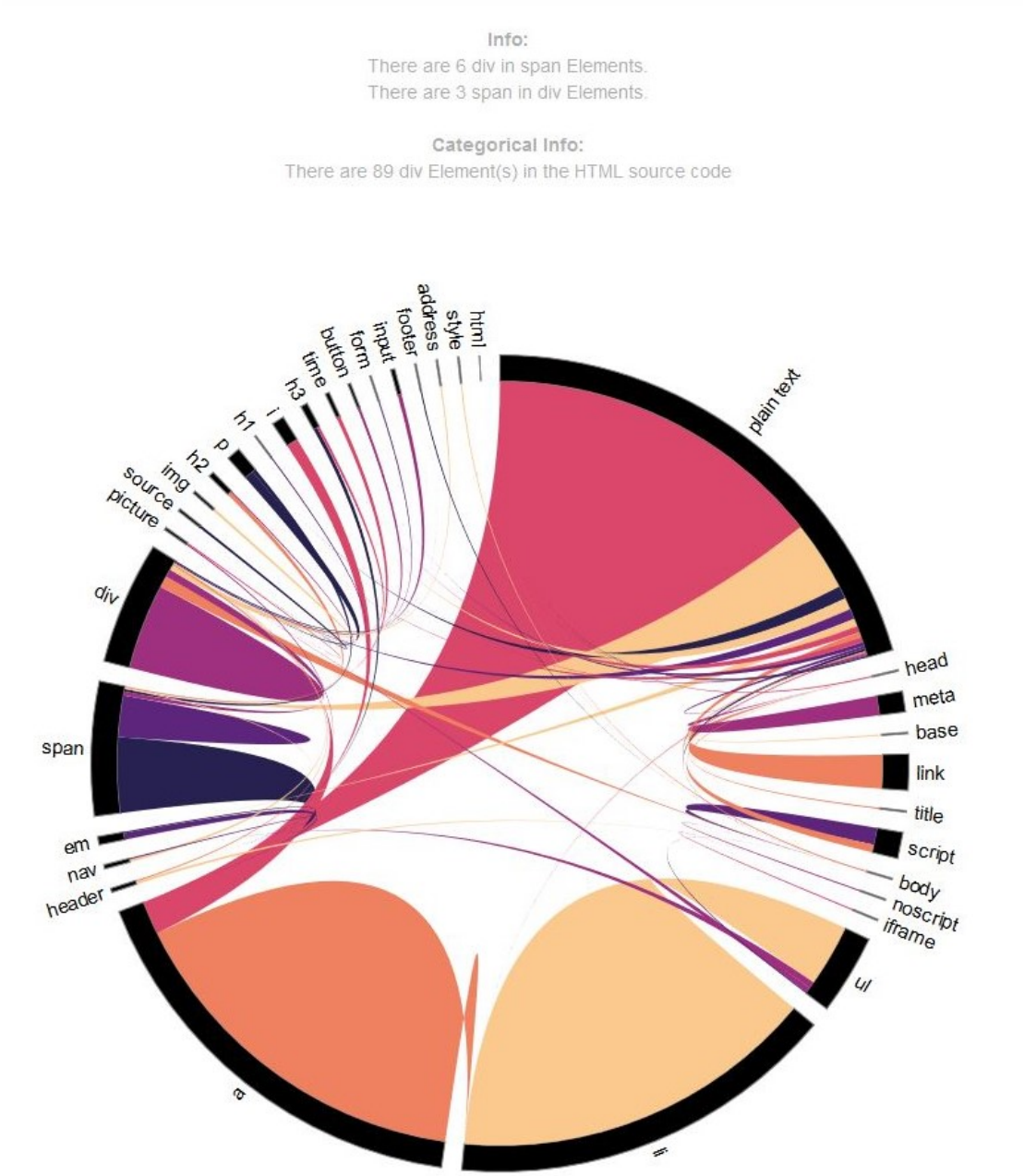


Abbildung 5.9: Chord-Diagramm für www.fh-ooe.at

Kapitel 6

Analyse

In diesem Kapitel werden anhand des beschriebenen Analysetools zwei bekannte Webseiten des öffentlichen Interesses auf die zuvor beschriebenen Qualitätsmerkmale untersucht. Die beiden Seiten wurden zufällig ausgewählt und dienen lediglich als Referenzen. Es besteht weiters die Möglichkeit, dass zum jetzigen Zeitpunkt bereits Veränderungen an den Internetauftritten vorgenommen worden sind und daher die Analyse nicht mehr gültig ist. Aus organisatorischen Gründen wird bei beiden Beispielen lediglich die Startseite genauer betrachtet und analysiert. Auf den Unterseiten können weitere andere Probleme auftauchen, die hier nicht berücksichtigt wurden. Die Analyse der Seiten soll die Funktionalität des Tools und die möglichen Optimierungen aufzeigen, jedoch keineswegs die Webseiten kritisieren oder in Frage stellen.

6.1 Analyse der Webseite www.orf.at

Der Österreichische Rundfunk bietet auf der Seite *www.orf.at* Nachrichten aus Österreich und der ganzen Welt. Diese werden mehrmals täglich upgedatet und stehen den Nutzern in Bild, Text oder Video zur Verfügung. Im Ranking der meistbesuchten österreichischen Webseiten steht diese auf Rang sechs und wird von Besuchern durchschnittlich fünfeinhalb Minuten genutzt. Die Hauptzielgruppe sind Nutzer aus Österreich, aber auch aus Deutschland, Italien und der Schweiz wird die Website aufgerufen [25].

6.1.1 Ergebnisse der einzelnen Bereiche

Nachstehend werden die Resultate der Validierungen dargelegt und näher erläutert. Ein ausführlicher Bericht der Analyse kann im Anhang eingesehen werden.

HTML-Validierung

Die Untersuchung der HTML Struktur der Seite www.orf.at liefert ein ausgesprochen positives Ergebnis. Wie in Abbildung 6.1 zu sehen ist, werden lediglich ein Fehler und elf Warnungen gefunden. Bei genauerer Betrachtung des Diagramms, ist zu erkennen, dass von den elf Warnungen acht in die gleiche Kategorie fallen und somit einfach und schnell zu beheben wären.

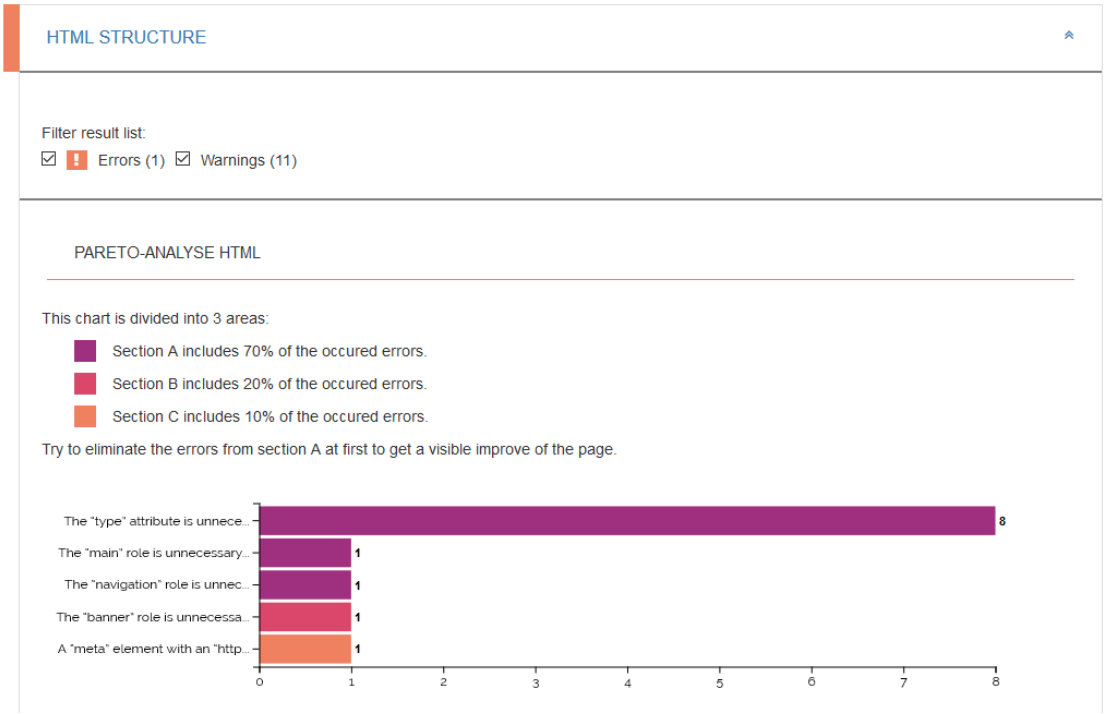


Abbildung 6.1: HTML-Validierung www.orf.at

CSS-Validierung

Die Validierung der CSS Informationen zeigt eine längere Liste an Problemen auf. Jedoch auch hier sind bloß 19 Fehler entdeckt worden. Die Häufigkeit des Fehlers *no existance at all* deutet darauf, dass hier eine Eigenschaft verwendet wurde, die in einer späteren CSS-Version nicht mehr mit auf genommen wurde. Für neue Benutzer möglicherweise erschreckend wirkt die Anzahl der Warnungen in den Stylesheets. Bei genauerer Betrachtung kann aber Entwarnung gegeben werden, da die Validierung hier sehr pingelig ist und auch erneute Definitionen von Eigenschaften aufzeigt und diese in Frage stellt.

Performance

Die Analyse der Seitenladezeiten liefert für die Desktop-Variante eine Auswertung von 71, welche somit im mittleren Bereich liegt. Für mobile Geräte ist die Seite besser optimiert und erzielt einen Wert von 93. Die einflussreichste Optimierung für die Desktop-Auswertung kann durch die Kompression einiger Bilder erreicht werden. Dadurch kann die Bewertung um zahlreiche Punkte verbessert werden. In einer Liste werden zahlreiche Bilder aufgezeigt, bei denen eine Kompression vorgenommen werden sollte. Eine solche Optimierung der Bilder stellt einen Mehraufwand dar und wird daher oftmals ignoriert. Für Inhalte, die längere Zeit online stehen, können Online-Tools zur Bildoptimierung herangezogen werden. Diese reduzieren die Dateigröße um bis zu 80% ohne dabei Qualität einbüßen zu müssen.

Enable Gzip Compression weist darauf hin, dass durch die spezifische Kompression Daten für die Übertragung eingespart werden können. Die Reduzierung durch *gzip* wird mittlerweile in allen modernen Browsern unterstützt und bietet die Möglichkeit Ressourcen schneller zum Client übertragen zu können [20]. Bei der Analyse der Webseite des ORF kann festgestellt werden, dass die Verwendung von *gzip* einen Einfluss von 4,8 auf das Gesamtergebnis hat. Die Größe der zu übertragenden Dateien könnte um 75% verringert werden.

Das Laden benötigter Ressourcen von einem Server belastet die Geschwindigkeit einer Seite. Daher ist es zu empfehlen, für jede Datei eine Caching-Strategie anzulegen und so zu definieren, in welchen Fällen der Client auf bereits früher geladene Ressourcen zurückgreifen kann. Dabei ist es möglich zu definieren, wer eine Datei chachen darf und für wie lange diese gespeichert werden kann [22]. Durch das Cachen einiger Dateien ergibt sich in der Gesamtwertung der Performance-Analyse für die Seite des ORF eine Verbesserung um 3,4.

Des Weiteren werden Optimierungen für JavaScript und CSS-Daten empfohlen und darauf hingewiesen, dass die Seite Ressourcen beinhaltet, die das Laden anderer blockiert. Dies kann zu erheblichen Verzögerungen führen und sollte stets vermieden werden.

Barrierefreiheit

Im Abschnitt der Barrierefreiheit kann abgelesen werden, dass von den 74 durchgeführten Testfällen 63 positiv abgeschlossen werden konnten. Lediglich bei elf Szenarien gibt es Probleme. Im Balkendiagramm werden diese Probleme nach ihrer Häufigkeit gereiht dargestellt. In diesem Fall ist zu erkennen, dass der Fehler *This link has a 'title' attribute that's the same as the text inside the link.* 99 mal aufgetreten ist und somit den größten Teil der Problematik einnimmt. Durch die Eliminierung dieses Fehlers kann eine 70%ige Verbesserung der Barrierefreiheit erreicht werden. Insgesamt können auf dieser Seite 140 Fehler behoben werden.

Visualisierungen

Im letzten Bereich werden allgemeine Informationen aus den Ressourcen der Website *www.orf.at* mittels grafischer Darstellungen aufbereitet. Als erstes werden die verwendeten Klassennamen im Bubble-Diagramm visualisiert. In dieser Seite enthalten die HTML-Elemente 75 verschiedene Klassennamen. 45 dieser Namen werden lediglich einmal im Dokument verwendet. Hier ist der Entwickler aufgefordert, zu überprüfen, ob diese notwendig sind und möglicherweise auf anderen Seiten wieder verwendet werden oder ob hier Klassennamen zusammengefasst werden können. Zehn Klassennamen werden 33 mal verwendet und weisen daher auf ein gleichbleibendes Strukturmuster im Aufbau der einzelnen Artikel der Seite hin.

Die Aufschlüsselung der in der Seite verwendeten Farben zeigt ein sehr einheitliches Bild. Für Textinhalte werden 26 unterschiedliche Farben eingesetzt. Bei den Hintergrundfarben sind es zwei Farben mehr. Für Texte werden mehr als zehn verschiedene Grautöne verwendet. Hier stellt sich die Frage, ob diese wirklich notwendig sind, denn auch wenn diese nebeneinander liegen fällt eine Unterscheidung mit dem bloßen Auge teilweise schwer. Ein ähnliches Bild weisen auch die zahlreichen Blautöne auf.

Die Visualisierung der Beziehungen zwischen den einzelnen HTML-Elementen macht es möglich einige Fakten über die Struktur einer Seite in kürzester Zeit abzuleiten. Wie es für einem korrekten HTML-Aufbau notwendig ist, gibt es genau ein `html`-Element, welches wiederum genau ein `head`- und ein `body`-Element besitzt. Zusätzlich ist erkennbar, dass sich richtigerweise alle `meta`-Deklarationen im `head`-Bereich befinden. Der `body` wird mit Hilfe der in HTML5 neu definierten Strukturelemente in drei Bereiche eingeteilt: `header, main, footer`. Die Gliederung der Navigation ist einfach zu erkennen, da auch hier das spezifische `nav`-Element eingesetzt wurde. Dieses beinhaltet eine `ul`, eine so genannte ungeordnete Liste, die wiederum zahlreiche `li`, also Listeneinträge, verwaltet. Im Diagramm kann auch die Verwendung der Überschriften überprüft werden. So fällt auf, dass bei den Überschriften-Levels kein Sprung vorgenommen worden ist. Es befinden sich Überschriften von Level eins bis drei im Dokument. Außerdem ist ersichtlich, dass Überschriften-Elemente keine weiteren Elemente enthalten, stattdessen befindet sich lediglich blanker Text darin. Zu erkennen ist auch, dass sich 18 `h1`-Elemente im Dokument befinden. Aus Gründen der Suchmaschinenoptimierung sollte darauf geachtet werden, nur eine Überschrift des Levels eins pro Seite zu beinhalten.

6.2 Analyse der Webseite www.fh-ooe.at

Die Webseite der Fachhochschule Oberösterreich beinhaltet Informationen zum Studienangebot an den vier Standorten: Hagenberg, Linz, Steyr und Wels. Darüber hinaus werden auch Fakten zu Forschungsprojekten, Weiterbildungen und alles rund um das Leben am Campus angeboten. Im Ranking der meistbesuchten Webseiten Österreichs liegt die Seite der FH Oberösterreich an 334. Stelle. Interessanterweise kommen die Besucher der Webseite nicht nur aus dem deutschsprachigen Raum, sondern z.B. auch aus Indien oder Nigeria.

6.2.1 Ergebnisse der einzelnen Bereiche

Die nachstehenden Punkte werden aus der Analyse der Seite www.fh-ooe.at mit Hilfe des Analyse-Tools gezogen. Die genauen Ergebnisse der Untersuchung befinden sich im Anhang.

HTML-Validierung

Bei der Analyse der HTML Struktur wurden bei der Seite der Fachhochschule Oberösterreich 26 Fehler und 17 Warnungen gefunden. Dieses Ergebnis ist als zufriedenstellend – aber verbesserungswürdig – einzustufen. Betrachtet man die Fehler genauer, sieht man dass neun mal das gleiche Problem gefunden wurde. Dabei handelt es sich um folgenden Fehler: *Character reference was not terminated by a semicolon.* Dies bedeutet, dass ein Strichpunkt nach einem für ein Zeichen spezifischen Code fehlt. Des weiteren kann man erkennen, dass die Verschachtelung der HTML-Elemente teilweise nicht richtig angewendet wurde. Es werden Blockelemente, wie `div`, `p` und Überschriften innerhalb eines inline-Elements, dem `span`, verwendet. Außerdem wurde ein `div`-Element geschlossen, obwohl kein öffnender Tag dazu gefunden wurde.

CSS-Validierung

Die Validierung der Stylesheets findet 66 Fehler und fast 4000 Warnungen. Wobei auch hier darauf zu achten ist, dass ein Großteil dieser Warnungen unter drei Kategorien fällt: Erneute Definition, keine Vordergrundfarbe zur Hintergrundfarbe und umgekehrt. Mehr Aufmerksamkeit sollte jedoch den Fehlern gewidmet werden. Dabei fallen 34 Missachtungen in die Kategorie *Zero*. Dies bedeutet, dass lediglich der Wert Null ohne eine Einheit angegeben werden darf, alle anderen benötigen eine eindeutige Definition der Einheit, wie zum Beispiel `px`. Der zweite große Teil der aufgeführten Verstöße weist darauf hin, dass keine negativen Werte verwendet werden dürfen. Weitere acht Fehler wurden entdeckt, durch die Verwendung von nicht existenten Eigenschaften kreiert. Hier kann durch wenige Veränderungen des CSS-Codes ein beachtlicher Teil der Fehler ausgebessert werden.

Performance

Die Analyse durch die von *Google PageSpeed Insights* angebotene Schnittstelle liefert eine Geschwindigkeitsanalyse von 56 für Desktop-Geräte und 77 für die mobile Version. Während die Punktzahl der mobilen Version im mittleren Bereich liegt, findet sich die der Desktop-Ansicht im unteren Bereich. Die größtmögliche Verbesserung der Auswertung kann durch die Optimierung der verwendeten Bilder erzielt werden. Dadurch kann deren Größe um bis zu 82% verringert werden.

Auch in diesem Beispiel gibt es Ressourcen, die die Ladenzeiten negativ beeinflussen. Dabei handelt es sich um zwei JavaScript- und fünf CSS-Dateien. Diese können bei Bedarf später oder asynchron geladen werden, um andere Ressourcen nicht zu blockieren.Es wird auch darauf hingewiesen Browser-Caching zu verwenden, um so statische Daten beim User zu speichern und nicht bei jedem Aufruf neu vom Server anfragen zu müssen. Durch die Minimierung von JavaScript Ressourcen kann deren Größe um bis zu 29% reduziert werden.

Barrierefreiheit

Die Untersuchung der Barrierefreiheit der Seite zeigt, dass von den 74 durchgeführten Testszenarien in diesem Fall 66 positiv abgeschlossen werden konnten und bei lediglich 8 ein Problem aufgetreten ist. Der hier am häufigsten aufgetretene Fehler ist interessanterweise der gleiche wie bei der zuvor untersuchten Seite des ORF: *This link has a 'title' attribute that's the same as the text inside the link.* In diesem Fall wurde er jedoch etwas häufiger gefunden, genauer gesagt 475 mal. Diese wortwörtliche Wiederholung ist gegenstandslos für die Unterstützungstechnologien und bringt auch für den User keinen Mehrwert.

Darüber hinaus sind 152 Links ohne Textinhalt vorhanden. Der enthaltene Text in einem Link wird von technischen Hilfsmitteln als Referenz zur Ankündigung dieses Verweises herangezogen. Fehlt dieser, so kann der Link nicht betitelt werden. Für den Fall, dass kein Text enthalten sein soll gibt es `aria-labels` mit denen ein Titel vergeben werden kann.

Die Verwendung von *Accesskeys* soll dabei helfen, bestimmte Links mittels Tastaturkürzel schneller zu erreichen. Leider hat sich herausgestellt, dass diese Abkürzungen

viele Risiken und Nachteile bergen. Im Bereich der Barrierefreiheit ist vor allem anzuführen, dass Screenreader eigene Tastenkombinationen haben, die durch die Verwendung von Accesskeys in Konflikt zueinander geraten können. Außerdem haben Browser sowie Betriebssysteme ihre Eigenheiten hinsichtlich Tastaturbefehlen [18].

Sorgfältig erstellte Formulare können es beeinträchtigten Benutzern wesentlich erleichtern diese auszufüllen. Daher sollte immer darauf geachtet werden, nicht nur Hilfestellungen für Personen ohne Sehbeeinträchtigung zu stellen. Die Anwendung von Labels ist äußerst wichtig. Nur so kann ein Screenreader die Zusammengehörigkeit von Titel und Formularfeld richtig interpretieren und an den Benutzer weitergeben. In der untersuchten Seite tritt diese Fehler lediglich zweimal auf, sollte jedoch trotzdem ehestmöglich behoben werden.

Visualisierungen

Der letzte Bereich enthält drei Informationsvisualisierungen deren Daten aus den vorhandenen Ressourcen gefiltert werden. Zu Beginn werden sämtliche im HTML-Dokument verwendete Klassennamen als Blasendiagramm dargestellt. Dabei werden die 139 Namen so gezeigt, dass der Radius einer Blase die Häufigkeit repräsentiert. Daraus kann entnommen werden, das hier 83 Klassennamen lediglich einmal verwendet worden sind. Auch hier sollte vom Entwickler überprüft werden, ob diese wirklich notwendig sind.

Die Farbpaletten für Texte, Hintergrund und Rahmenlinien zeigen eine wesentlich größere Vielfalt als bei der zuvor untersuchten Seite des ORF. Dies lässt sich vor allem durch die farbliche Kennzeichnung der vier Standorte der Fachhochschule Oberösterreich erklären(siehe Abbildung 6.2). Für den Inhalt werden 85 verschiedene Farben verwendet, während für Hintergründe 108 Farbtöne zum Einsatz kommen. Obwohl ein erhöhtes Maß an Farben durch die inhaltliche Unterscheidung nachzuvollziehen ist, sollte dennoch geprüft werden, ob alle Nuancen wirklich benötigt und vom Benutzer unterschieden werden können.

Aus dem Chord-Diagramm können zahlreiche Beziehungen zwischen den verwendeten HTML-Elementen abgelesen werden. So steht natürlich als erstes fest, dass genau ein `html`-Element mit einem `head`- und einem `body`-Element eingesetzt wurde. Hervorzuheben ist auch, dass neue in HTML5 definierte Elemente, die die Semantik und Struktur von Webseiten verbessern sollen eingesetzt wurden. Die Überschriften sind so gewählt, dass lediglich eine Überschrift des Levels 1 vorkommt. Außerdem wird kein Überschriftenlevel übersprungen. Alle vorkommenden `li`-Elemente befinden sich innerhalb eines `ul`-Elementes, eine fälschliche Verschachtlung ist daher auszuschließen. Lediglich bei der genaueren Betrachtung des `form`-Elementes entstehen Zweifel. Im Dokument befindet sich genau ein solches Element, welches jedoch keinerlei `input`-Elemente als direkte Kindelemente besitzt. Dies liegt jedoch nicht daran, dass hier ein Fehler unterlaufen ist, es wurde lediglich zwischen `form`- und `input`-Elementen ein `div`-Element eingeschoben. Das im Abschnitt der Barrierefreiheit aufgewiesene Nichtvorhandensein der `label`-Elemente zum entsprechenden `input`-Element kann ebenfalls aus diesem Diagramm abgelesen werden.

Abbildung 6.2: Farbanalyse FH Oberösterreich

Kapitel 7

Resümee

Webentwickler müssen während der Erstellung oder Überarbeitung einer Webanwendung unzählige Dinge beachten und viele verschiedene Technologien beherrschen. Um einen gewissen Qualitätsstandard sicherstellen zu können, bedarf es einer eingehenden Prüfung der erstellten Dateien. Im Web werden Hilfsmittel für die Analyse angeboten, die sich jeweils auf ein Spezialgebiet konzentrieren. Andere Qualitätsmerkmale bedürfen einer individuell, von speziell ausgebildeten Personen durchgeführten, Überprüfung.

Der erstellte Prototyp konzentriert sich auf vier Qualitätsmerkmale, die dieser in nur einem Instrument vereint. Die Validierung des HTML-Codes sowie der angewendeten Stylesheets bilden den Grundstein für eine hochwertige Webanwendung. Treten Mängel bereits in diesem Bereich auf, sind sie meist auch in anderen Analysen auffindbar. Aus diesem Grund werden die Ergebnisse der Untersuchung des Codes im oberen Bereich des Prototypen dargestellt. Darüber hinaus wird die Geschwindigkeit der Seite untersucht. Dieser Faktor ist nicht nur aus technischer Sicht wichtig, sondern vorrangig auch für das Benutzererlebnis. Seiten mit zu langen Ladezeiten werden vom Benutzer binnen kürzester Zeit wieder verlassen und nur selten erneut besucht. Ein weiteres und somit das vierte der hier untersuchten Qualitätsmerkmale bildet die Barrierefreiheit einer Webseite. Nicht nur im täglichen Leben ist es wichtig, Informationen und Gebäude für alle Menschen zugänglich zu machen, auch Inhalte im Internet sollen allen zur Verfügung stehen. Dieser Bereich der Optimierung ist besonders aufwendig, da jeder Mensch und jede Einschränkung andere Bedürfnisse an eine Webanwendung steht. Mittels der vom W3C erstellten Richtlinien kann ein Großteil der damit einhergehenden Forderungen abgedeckt und systematisch überprüft werden.

Die verschiedenen durchgeführten Analysen liefern eine große Menge an Daten, die für den Nutzer des Tools aufbereitet werden müssen. In dieser Arbeit wurde neben der grundlegenden Überprüfung einer Webseite, ein großes Augenmerk auf die visuelle Darstellung der erfassten Daten gelegt. Mit Hilfe der in der Qualitätssicherung bekannten Pareto-Analyse werden Diagramme erstellt, die die gefundenen Fehler in Kategorien einteilen und in einer Prioritätenliste reihen. Für die Visualisierung der Performance-Analyse wurde ein gestapeltes Säulendiagramm gewählt. Ergänzend dazu gibt es drei verschiedene Visualisierungen, die Informationen aus dem Quelltext repräsentieren. Ein Blasendiagramm stellt alle im HTML verwendeten Klassennamen und die damit verbundene Häufigkeit dar. In einem weiteren sind alle angewendeten Farben der Stylesheets

aufgelistet. Das komplexeste Diagramm befindet sich am Ende der Anwendung. Ein so genanntes Sehnendiagramm zeigt die Beziehungen zwischen allen im Dokument auftretenden HTML-Elementen auf.

Die anschließend, mit dem entwickelten Prototypen, durchgeführte Analyse der Seite des ORF sowie der Fachhochschule Oberösterreich zeigt, dass auch moderne Seiten großer Unternehmen ein gewisses Optimierungspotential aufweisen können. Qualität bei Webanwendungen kann also durchaus überprüft und gegebenenfalls verbessert werden. Ein 100%iges Maß an Qualität ist jedoch kaum zu erreichen. Der Webentwickler hat stets die Entscheidung zu treffen, ob eine Optimierung hingehend der beschriebenen Qualitätsmerkmale für den speziellen Anwendungsfall sinnvoll erscheint oder zu ungewollten Einschränkungen führt. Wichtig ist aber, diese Eigenschaften schon während der Erstellung einer Webseite im Hinterkopf zu behalten und regelmäßige Überprüfungen, auch nach dem Online-Stellen, durchzuführen.

Die Auswahl der vier zu analysierenden Qualitätsmerkmale wurde dahingehend getroffen, dass diese auch die Grundlage anderer Überprüfungen bilden. Es besteht jedoch die Möglichkeit den derzeitigen Prototypen um weitere Merkmale zu ergänzen. So können beispielsweise Testverfahren zum Thema Benutzerfreundlichkeit, Suchmaschinenoptimierung und eine Analyse der verwendeten Technologien hinzugefügt werden.

Anhang A

Inhalt der CD-ROM/DVD

Format: CD-ROM, Single Layer, ISO9660-Format

A.1 Masterarbeit

Pfad: /

Czakert_Tamara_2018.pdf Masterarbeit mit Instruktionen (Gesamtdokument)

A.2 Literatur

Pfad: /OnlineSources

*.pdf . . . . . Kopien der Online-Ressourcen

A.3 Bilder

Pfad: /Images

*.png, *.jpg, *.pdf . . . Verwendete Bilder

A.4 Projektdateien

Pfad: /SourceCode

Projekt-Prototype.zip . Source code des entwickelten Prototypen

Anhang B

Auswertung ORF

CHECK QUALITY

orf.at

Here are the results for the given page:
http://orf.at

HTML STRUCTURE

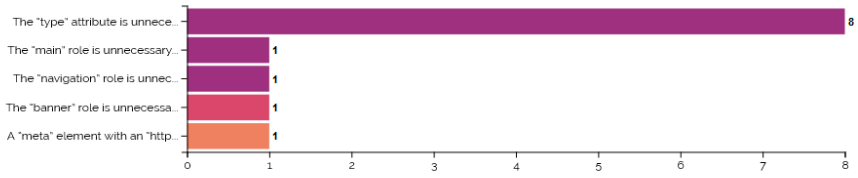
Filter result list:
☑ Errors (1) ☑ Warnings (11)

PARETO-ANALYSE HTML

This chart is divided into 3 areas:

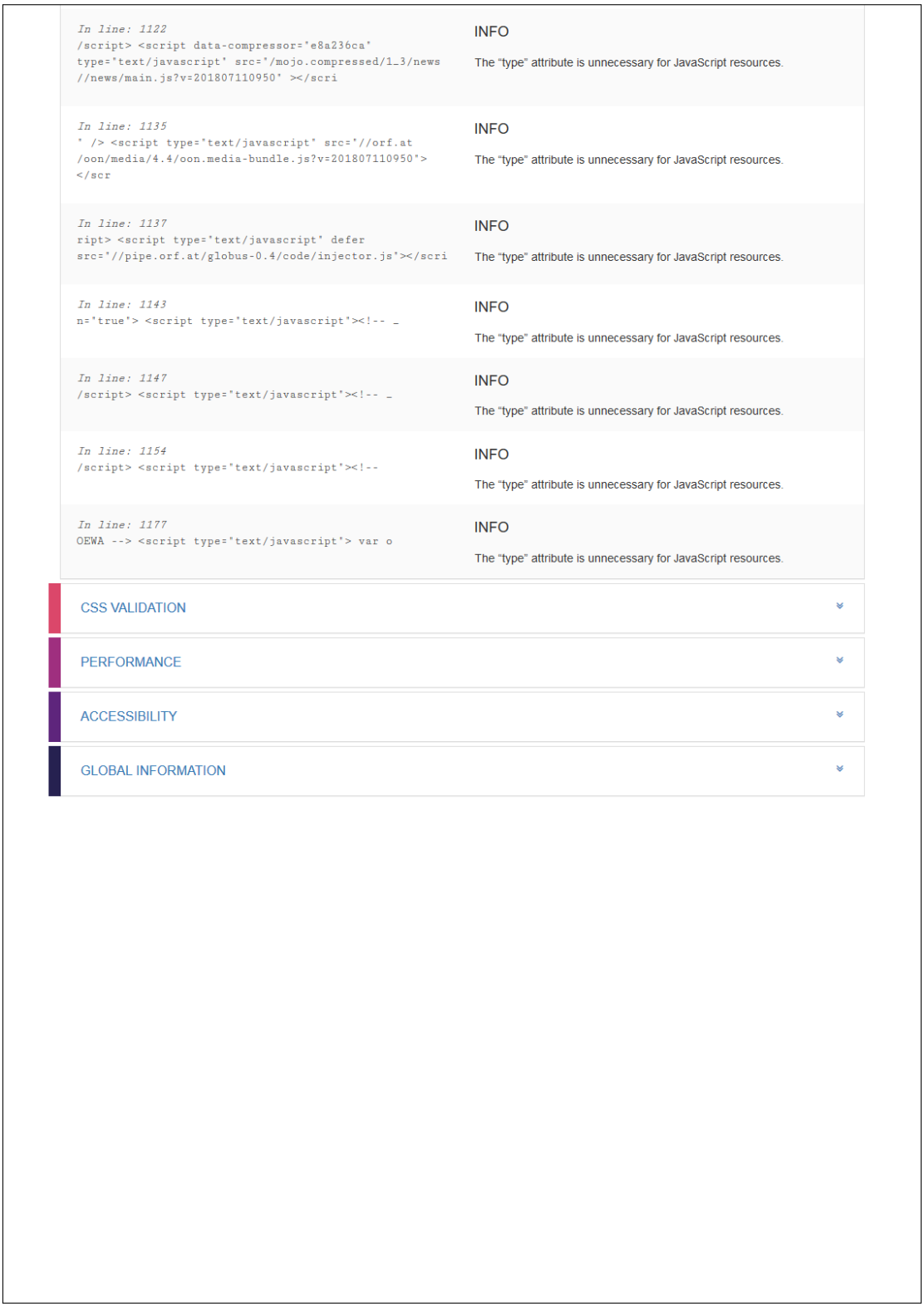
- Section A includes 70% of the occured errors.
- Section B includes 20% of the occured errors.
- Section C includes 10% of the occured errors.

Try to eliminate the errors from section A at first to get a visible improve of the page.



ERROR-LIST

In line: 5 `-8' /> <meta http-equiv='X-UA-Compatible' content='IE=edge,chrome=1' /> <m`	**ERROR** A “meta” element with an “http-equiv” attribute whose value is “X-UA-Compatible” must have a “content” attribute with the value “IE=edge”.
In line: 95 `'> <header class='header' role='banner'> <`	**INFO** The “banner” role is unnecessary for element “header”.
In line: 138 `ipt> <nav role='navigation' id='nnav'> <u`	**INFO** The “navigation” role is unnecessary for element “nav”.
In line: 177 `<main class='content clearfix front' id='content' role='main'>`	**INFO** The “main” role is unnecessary for element “main”.
In line: 1090 `div> <script type='text/javascript'> va`	**INFO** The “type” attribute is unnecessary for JavaScript resources.



In line: 1122 /script> <script data-compressor="e8a236ca" type="text/javascript" src="/mojo.compressed/1_3/news //news/main.js?v=201807110950" ></scri	INFO The "type" attribute is unnecessary for JavaScript resources.
In line: 1135 " /> <script type="text/javascript" src="//orf.at /oon/media/4.4/oon.media-bundle.js?v=201807110950"> </scr	INFO The "type" attribute is unnecessary for JavaScript resources.
In line: 1137 ript> <script type="text/javascript" defer src="//pipe.orf.at/globus-0.4/code/injector.js"></scri	INFO The "type" attribute is unnecessary for JavaScript resources.
In line: 1143 n="true"> <script type="text/javascript"><!-- _	INFO The "type" attribute is unnecessary for JavaScript resources.
In line: 1147 /script> <script type="text/javascript"><!-- _	INFO The "type" attribute is unnecessary for JavaScript resources.
In line: 1154 /script> <script type="text/javascript"><!--	INFO The "type" attribute is unnecessary for JavaScript resources.
In line: 1177 OEWA --> <script type="text/javascript"> var o	INFO The "type" attribute is unnecessary for JavaScript resources.

CSS VALIDATION

PERFORMANCE

ACCESSIBILITY

GLOBAL INFORMATION

CHECK QUALITY

orf.at

Here are the results for the given page:
http://orf.at

HTML STRUCTURE

CSS VALIDATION

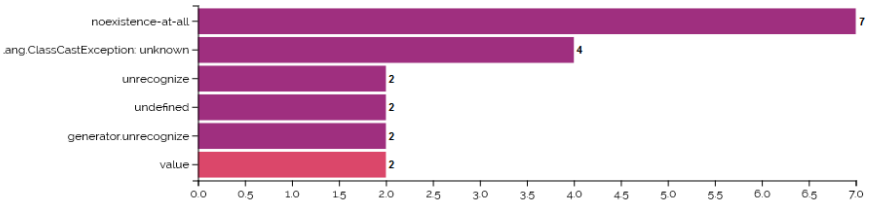
Filter result:
Errors (19) Warnings (605)

PARETO-ANALYSE CSS

This chart is divided into 3 areas:

- Section A includes 70% of the occured errors.
- Section B includes 20% of the occured errors.
- Section C includes 10% of the occured errors.

Try to eliminate the errors from section A at first to get a visible improve of the page.



In line: 74
Used value: none

JAVA.LANG.CLASSCASTEXCEPTION: UNKNOWN

Einlese-Fehler

In line: 107
Used value: none

VALUE

Ungültige Nummer : background-color (nullcolors.html#propdef-background-color) "none" ist kein "background-color"-Wert :

In line: 117
Used value: none

NOEXISTENCE-AT-ALL

Die Eigenschaft "pointer-events" existiert nicht :

In line: 197
Used value: none

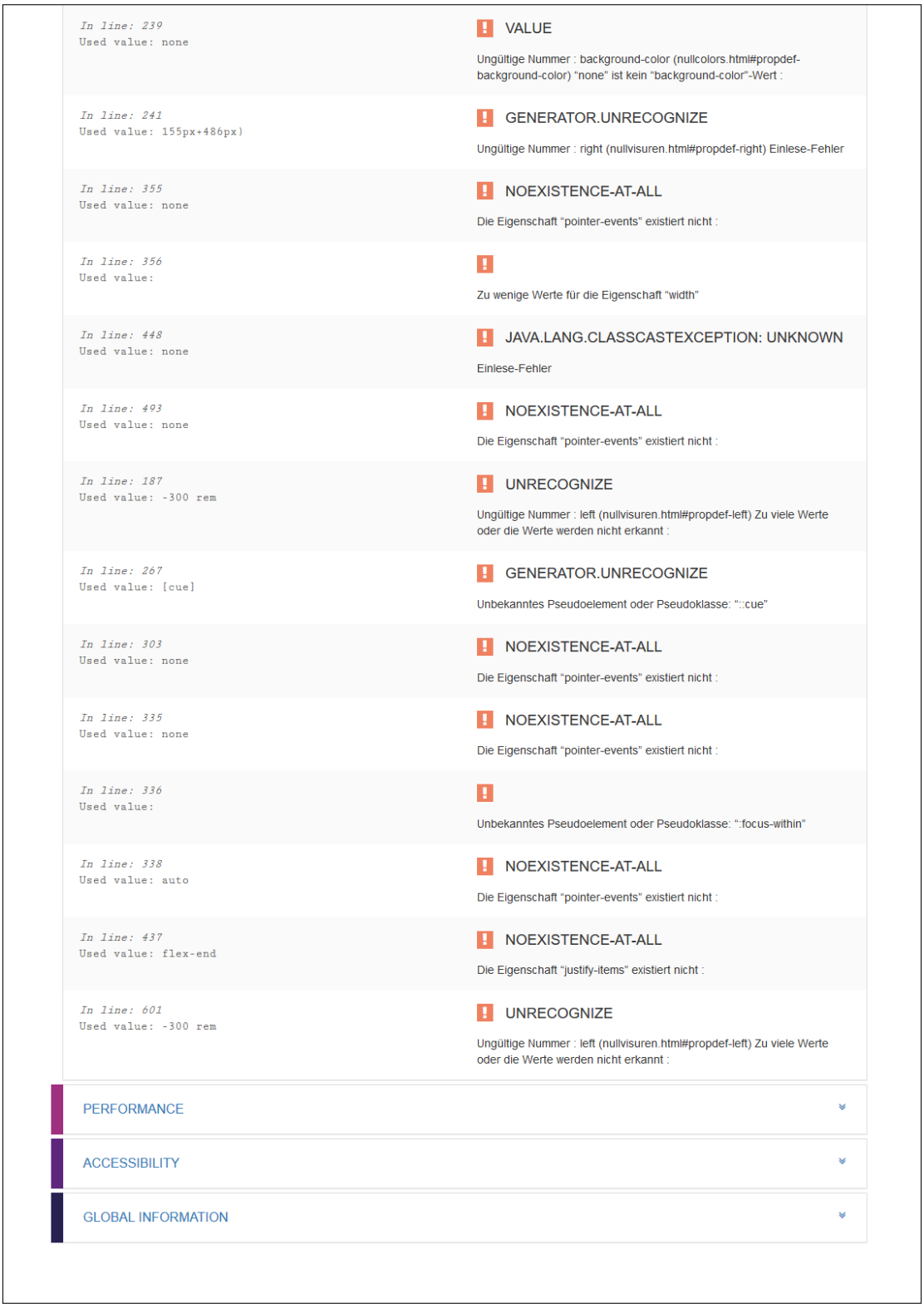
JAVA.LANG.CLASSCASTEXCEPTION: UNKNOWN

Einlese-Fehler

In line: 198
Used value: none

JAVA.LANG.CLASSCASTEXCEPTION: UNKNOWN

Einlese-Fehler

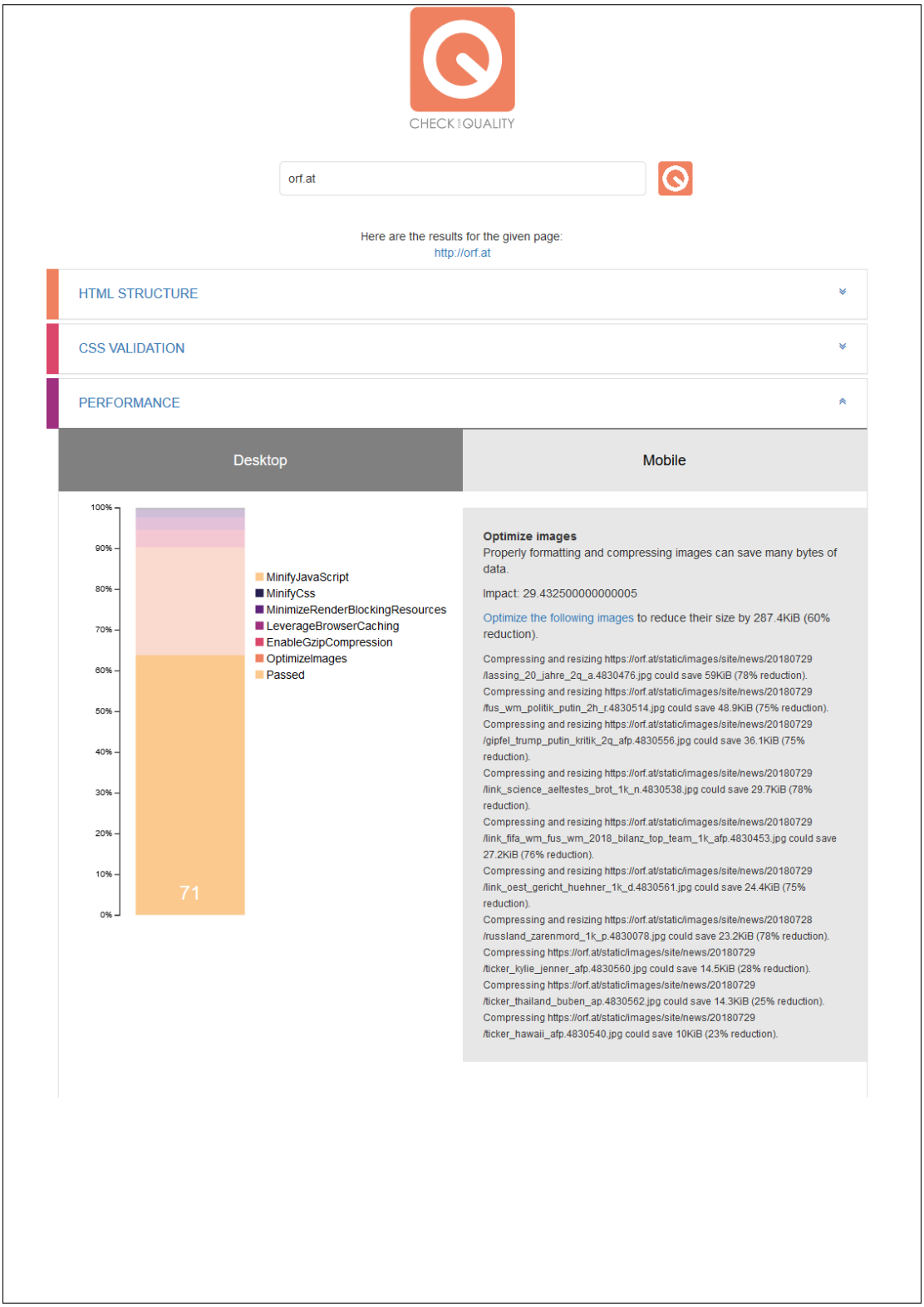


In line: 239 Used value: none	VALUE Ungültige Nummer : background-color (nullcolors.html#propdef-background-color) "none" ist kein "background-color"-Wert :
In line: 241 Used value: 155px+486px)	GENERATOR.UNRECOGNIZE Ungültige Nummer : right (nullvisuren.html#propdef-right) Einlese-Fehler
In line: 355 Used value: none	NOEXISTENCE-AT-ALL Die Eigenschaft "pointer-events" existiert nicht :
In line: 356 Used value:	 Zu wenige Werte für die Eigenschaft "width"
In line: 448 Used value: none	JAVA.LANG.CLASSCASTEXCEPTION: UNKNOWN Einlese-Fehler
In line: 493 Used value: none	NOEXISTENCE-AT-ALL Die Eigenschaft "pointer-events" existiert nicht :
In line: 187 Used value: -300 rem	UNRECOGNIZE Ungültige Nummer : left (nullvisuren.html#propdef-left) Zu viele Werte oder die Werte werden nicht erkannt :
In line: 267 Used value: [cue]	GENERATOR.UNRECOGNIZE Unbekanntes Pseudoelement oder Pseudoklasse: "::cue"
In line: 303 Used value: none	NOEXISTENCE-AT-ALL Die Eigenschaft "pointer-events" existiert nicht :
In line: 335 Used value: none	NOEXISTENCE-AT-ALL Die Eigenschaft "pointer-events" existiert nicht :
In line: 336 Used value:	 Unbekanntes Pseudoelement oder Pseudoklasse: ":focus-within"
In line: 338 Used value: auto	NOEXISTENCE-AT-ALL Die Eigenschaft "pointer-events" existiert nicht :
In line: 437 Used value: flex-end	NOEXISTENCE-AT-ALL Die Eigenschaft "justify-items" existiert nicht :
In line: 601 Used value: -300 rem	UNRECOGNIZE Ungültige Nummer : left (nullvisuren.html#propdef-left) Zu viele Werte oder die Werte werden nicht erkannt :

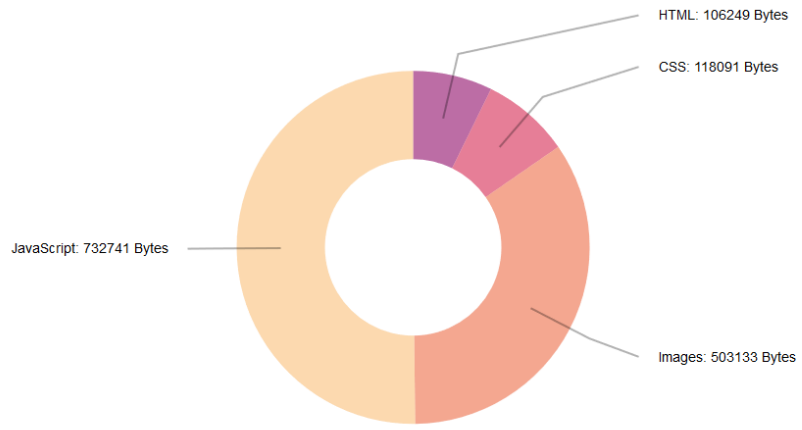
PERFORMANCE

ACCESSIBILITY

GLOBAL INFORMATION

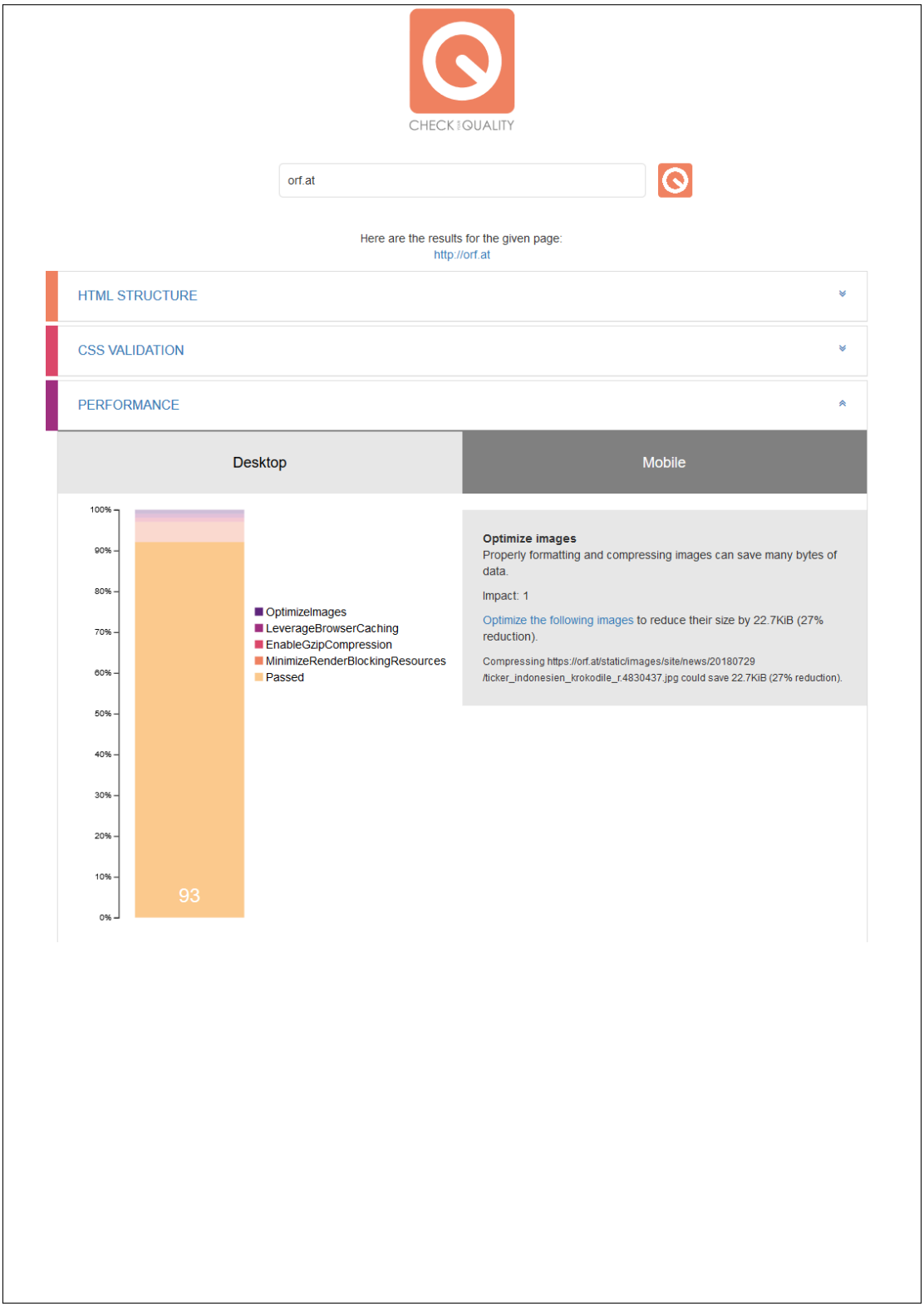

CHECK QUALITY
orf.at
Here are the results for the given page:
http://orf.at
HTML STRUCTURE
CSS VALIDATION
PERFORMANCE
Desktop
Mobile
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
71
MinifyJavaScript
MinifyCss
MinimizeRenderBlockingResources
LeverageBrowserCaching
EnableGzipCompression
OptimizeImages
Passed
Optimize images
Properly formatting and compressing images can save many bytes of data.
Impact: 29.432500000000005
Optimize the following images to reduce their size by 287.4KiB (60% reduction).
Compressing and resizing https://orf.at/static/images/site/news/20180729/lassing_20_jahre_2q_a.4830476.jpg could save 59KiB (78% reduction).
Compressing and resizing https://orf.at/static/images/site/news/20180729/fus_wm_politik_putin_2h_r.4830514.jpg could save 48.9KiB (75% reduction).
Compressing and resizing https://orf.at/static/images/site/news/20180729/gipfel_trump_putin_kritik_2q_afp.4830556.jpg could save 36.1KiB (75% reduction).
Compressing and resizing https://orf.at/static/images/site/news/20180729/link_science_aeltestes_brot_1k_n.4830538.jpg could save 29.7KiB (78% reduction).
Compressing and resizing https://orf.at/static/images/site/news/20180729/link_fifa_wm_fus_wm_2018_bilanz_top_team_1k_afp.4830453.jpg could save 27.2KiB (76% reduction).
Compressing and resizing https://orf.at/static/images/site/news/20180729/link_oest_gericht_huehner_1k_d.4830561.jpg could save 24.4KiB (75% reduction).
Compressing and resizing https://orf.at/static/images/site/news/20180728/russland_zarenmord_1k_p.4830078.jpg could save 23.2KiB (78% reduction).
Compressing https://orf.at/static/images/site/news/20180729/ticker_kylie_jenner_afp.4830560.jpg could save 14.5KiB (28% reduction).
Compressing https://orf.at/static/images/site/news/20180729/ticker_thailand_buben_ap.4830562.jpg could save 14.3KiB (25% reduction).
Compressing https://orf.at/static/images/site/news/20180729/ticker_hawaii_afp.4830540.jpg could save 10KiB (23% reduction).

YOUR WEBPAGE RESSOURCES IN BYTES:

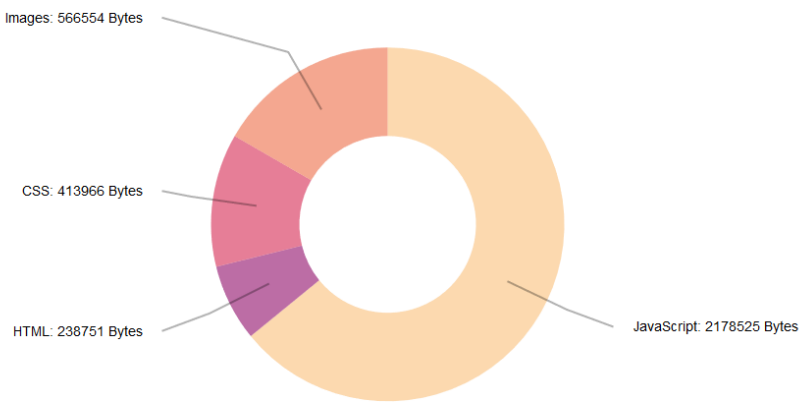


ACCESSIBILITY

GLOBAL INFORMATION

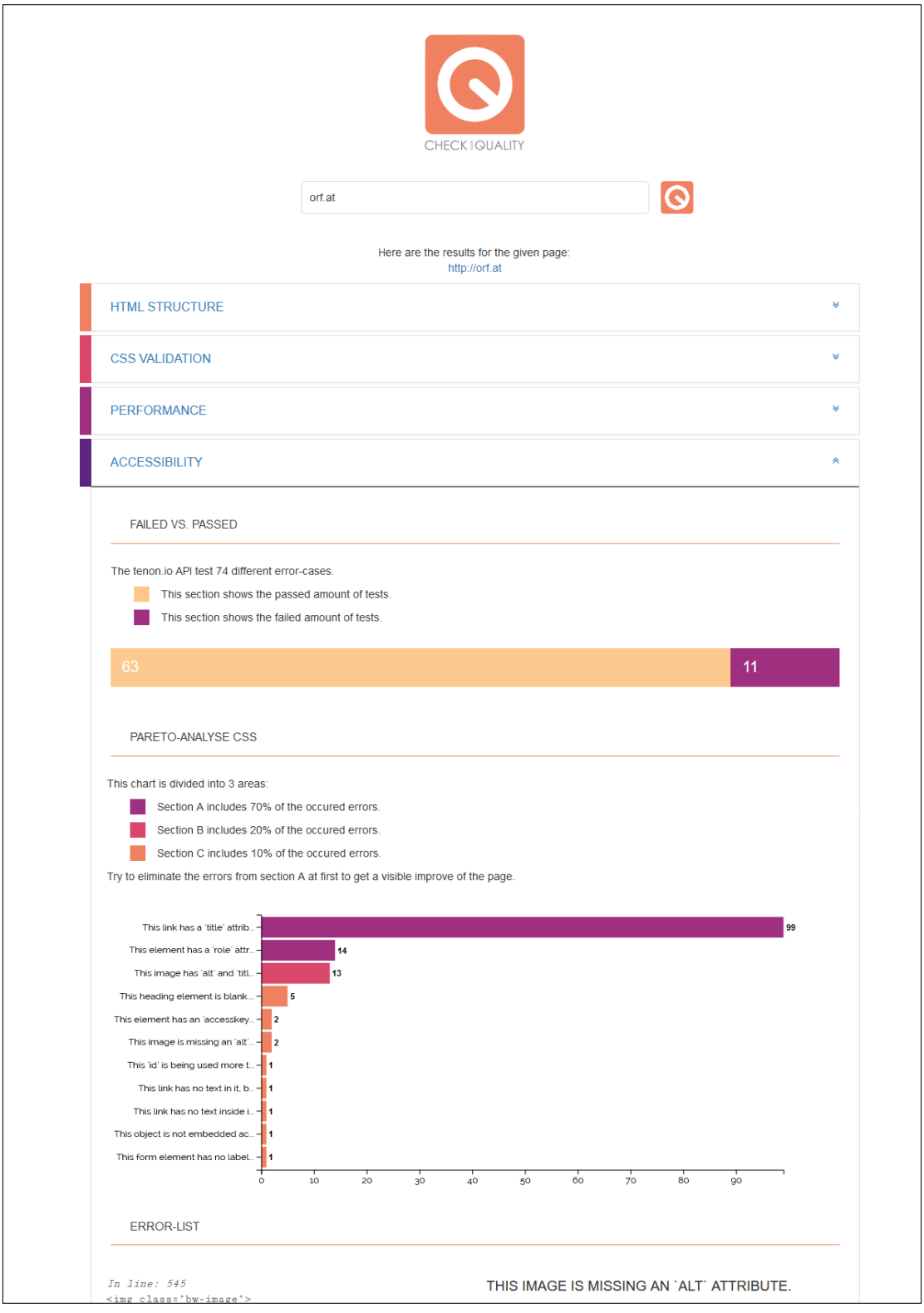

CHECK QUALITY
orf.at
Here are the results for the given page:
http://orf.at
HTML STRUCTURE
CSS VALIDATION
PERFORMANCE
Desktop
Mobile
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
93
OptimizeImages
LeverageBrowserCaching
EnableGzipCompression
MinimizeRenderBlockingResources
Passed
Optimize images
Properly formatting and compressing images can save many bytes of data.
Impact: 1
Optimize the following images to reduce their size by 22.7KiB (27% reduction).
Compressing https://orf.at/static/images/site/news/20180729/ticker_indonesien_krokodile_r.4830437.jpg could save 22.7KiB (27% reduction).

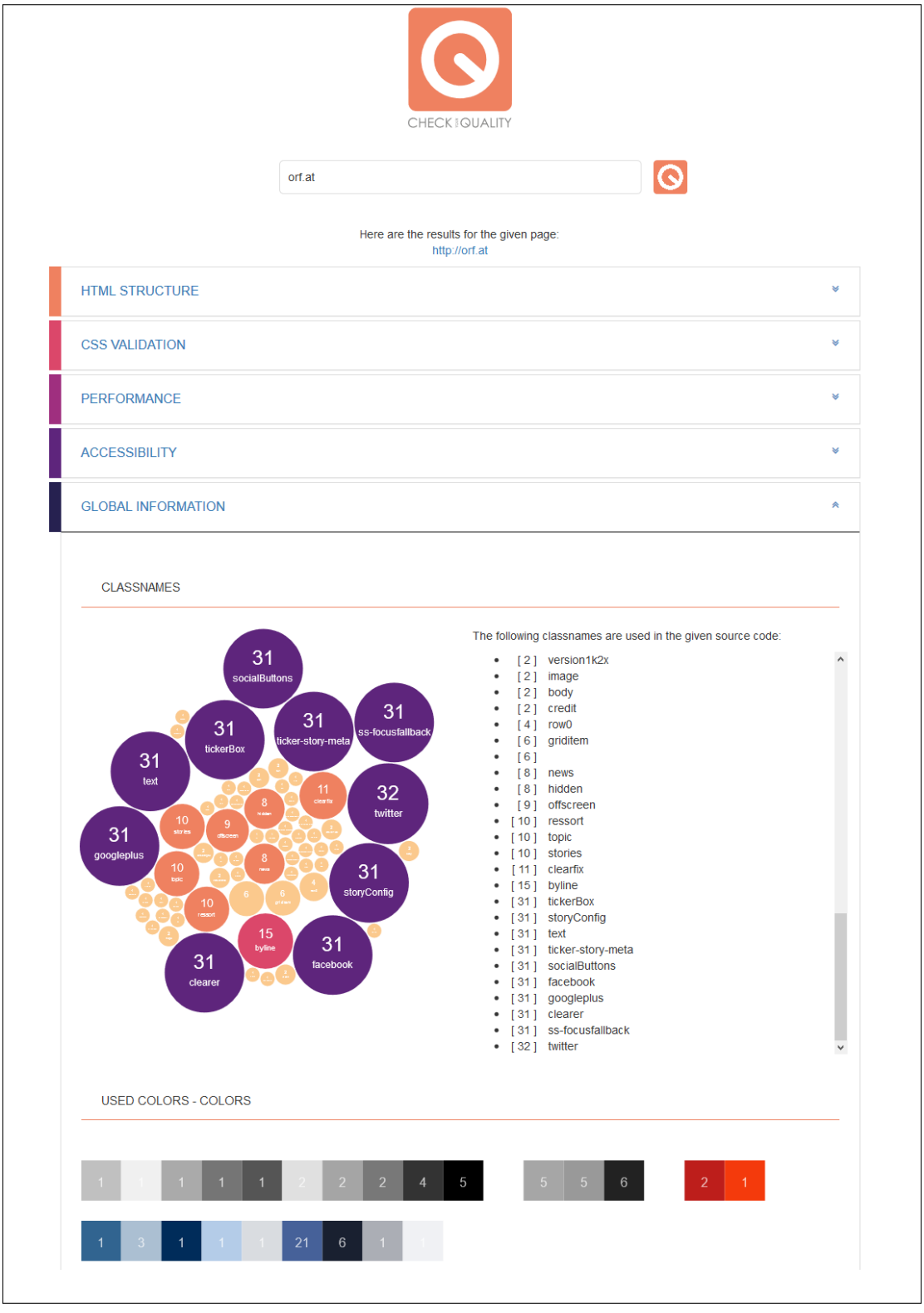
YOUR WEBPAGE RESSOURCES IN BYTES:



ACCESSIBILITY

GLOBAL INFORMATION


CHECK QUALITY
orf.at
Here are the results for the given page:
http://orf.at
HTML STRUCTURE
CSS VALIDATION
PERFORMANCE
ACCESSIBILITY
FAILED VS. PASSED
The tenon.io API test 74 different error-cases.
This section shows the passed amount of tests.
This section shows the failed amount of tests.
63
11
PARETO-ANALYSE CSS
This chart is divided into 3 areas:
Section A includes 70% of the occured errors.
Section B includes 20% of the occured errors.
Section C includes 10% of the occured errors.
Try to eliminate the errors from section A at first to get a visible improve of the page.
This link has a 'title' attrib...
This element has a 'role' attr...
This image has 'alt' and 'titl...
This heading element is blank...
This element has an 'accesskey...
This image is missing an 'alt'...
This 'id' is being used more t...
This link has no text in it, b...
This link has no text inside i...
This object is not embedded ac...
This form element has no label...
99
14
13
5
2
2
1
1
1
1
1
0
10
20
30
40
50
60
70
80
90
ERROR-LIST
In line: 545
<img class="bw-image">
THIS IMAGE IS MISSING AN `ALT` ATTRIBUTE.


CHECK QUALITY
orf.at
Here are the results for the given page:
http://orf.at
HTML STRUCTURE
CSS VALIDATION
PERFORMANCE
ACCESSIBILITY
GLOBAL INFORMATION
CLASSNAMES
31 socialButtons
31 tickerBox
31 ticker-story-meta
31 ss-focusfallback
31 text
11 clearfix
32 twitter
8 hidden
10 stories
9 offscreen
31 googleplus
8 news
10 topic
31 storyConfig
6
6 griditem
10 ressort
15 byline
31 facebook
31 clearer
The following classnames are used in the given source code:
[2] version1k2x
[2] image
[2] body
[2] credit
[4] row0
[6] griditem
[6]
[8] news
[8] hidden
[9] offscreen
[10] ressort
[10] topic
[10] stories
[11] clearfix
[15] byline
[31] tickerBox
[31] storyConfig
[31] text
[31] ticker-story-meta
[31] socialButtons
[31] facebook
[31] googleplus
[31] clearer
[31] ss-focusfallback
[32] twitter
USED COLORS - COLORS
1 1 1 1 1 2 2 2 4 5
5 5 6
2 1
1 3 1 1 1 21 6 1 1

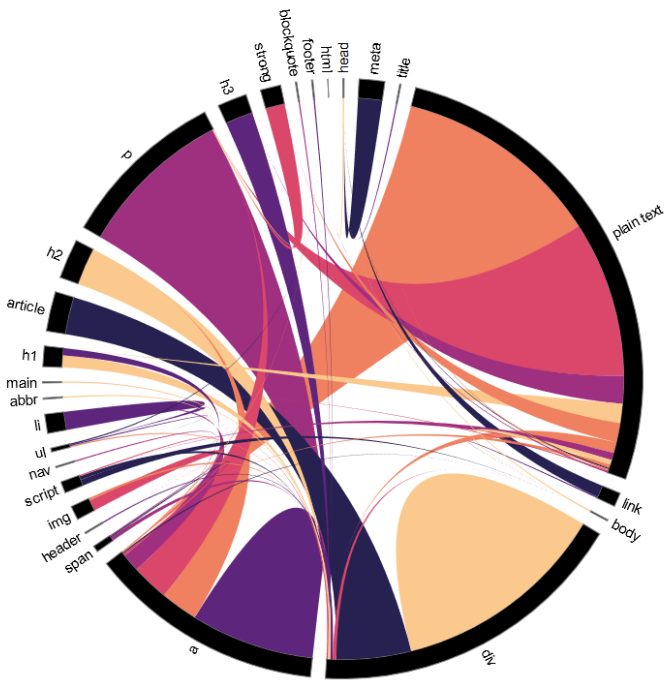
USED COLORS - BACKGROUNDS

USED COLORS - BORDER

RELATIONS BETWEEN HTML ELEMENTS:

Info:
There are 62 div in article Elements.
There are 31 article in div Elements.

Categorical Info:
There are 250 div Element(s) in the HTML source code



Anhang C

Auswertung FH Oberösterreich

CHECK QUALITY

fh-ooe.at

Here are the results for the given page:
http://fh-ooe.at

HTML STRUCTURE

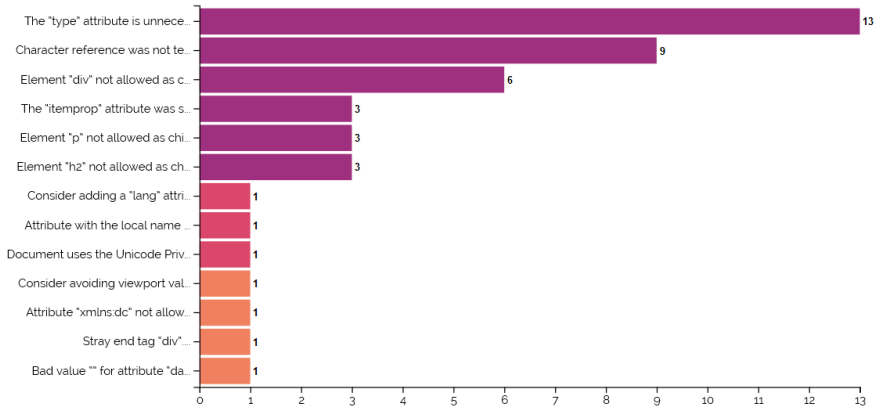
Filter result list:
Errors (27) Warnings (17)

PARETO-ANALYSE HTML

This chart is divided into 3 areas:

- Section A includes 70% of the occured errors.
- Section B includes 20% of the occured errors.
- Section C includes 10% of the occured errors.

Try to eliminate the errors from section A at first to get a visible improve of the page.



ERROR-LIST

In line: 1
`ch</title><meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no, minimal-ui"><link`

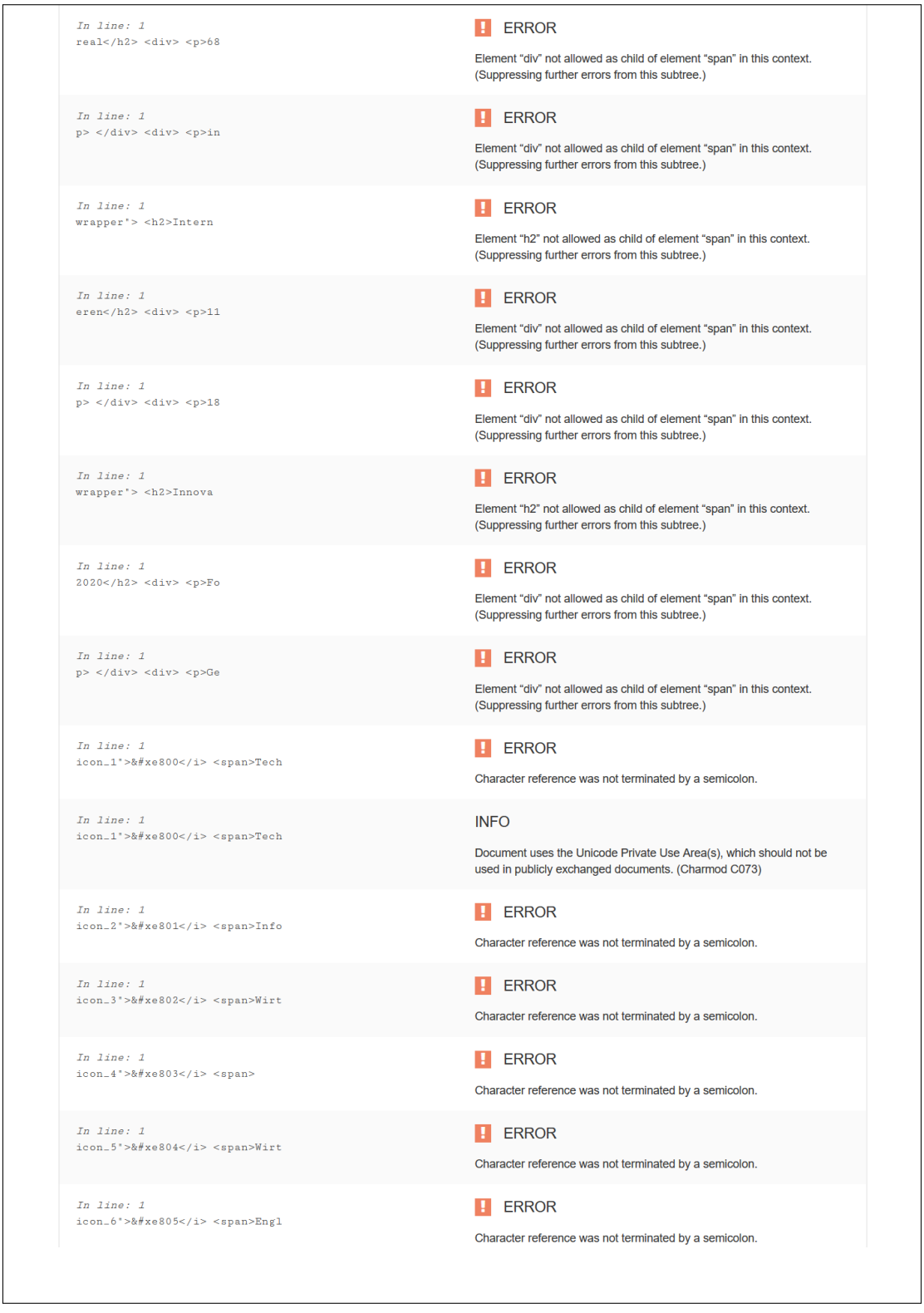
INFO

Consider avoiding viewport values that prevent users from resizing documents.

In line: 1
`wrapper"> <h2>Make i`

ERROR

Element “h2” not allowed as child of element “span” in this context. (Suppressing further errors from this subtree.)



In line: 1 `real</h2> <div> <p>68`	ERROR Element “div” not allowed as child of element “span” in this context. (Suppressing further errors from this subtree.)
In line: 1 `p> </div> <div> <p>in`	ERROR Element “div” not allowed as child of element “span” in this context. (Suppressing further errors from this subtree.)
In line: 1 `wrapper"> <h2>Intern`	ERROR Element “h2” not allowed as child of element “span” in this context. (Suppressing further errors from this subtree.)
In line: 1 `eren</h2> <div> <p>11`	ERROR Element “div” not allowed as child of element “span” in this context. (Suppressing further errors from this subtree.)
In line: 1 `p> </div> <div> <p>18`	ERROR Element “div” not allowed as child of element “span” in this context. (Suppressing further errors from this subtree.)
In line: 1 `wrapper"> <h2>Innova`	ERROR Element “h2” not allowed as child of element “span” in this context. (Suppressing further errors from this subtree.)
In line: 1 `2020</h2> <div> <p>Fo`	ERROR Element “div” not allowed as child of element “span” in this context. (Suppressing further errors from this subtree.)
In line: 1 `p> </div> <div> <p>Ge`	ERROR Element “div” not allowed as child of element “span” in this context. (Suppressing further errors from this subtree.)
In line: 1 `icon_1">&#xe800</i> <span>Tech`	ERROR Character reference was not terminated by a semicolon.
In line: 1 `icon_1">&#xe800</i> <span>Tech`	INFO Document uses the Unicode Private Use Area(s), which should not be used in publicly exchanged documents. (Charmod C073)
In line: 1 `icon_2">&#xe801</i> <span>Info`	ERROR Character reference was not terminated by a semicolon.
In line: 1 `icon_3">&#xe802</i> <span>Wirt`	ERROR Character reference was not terminated by a semicolon.
In line: 1 `icon_4">&#xe803</i> <span>`	ERROR Character reference was not terminated by a semicolon.
In line: 1 `icon_5">&#xe804</i> <span>Wirt`	ERROR Character reference was not terminated by a semicolon.
In line: 1 `icon_6">&#xe805</i> <span>Engl`	ERROR Character reference was not terminated by a semicolon.

CHECK THE QUALITY

fh-ooe.at

Here are the results for the given page:
http://fh-ooe.at

HTML STRUCTURE

CSS VALIDATION

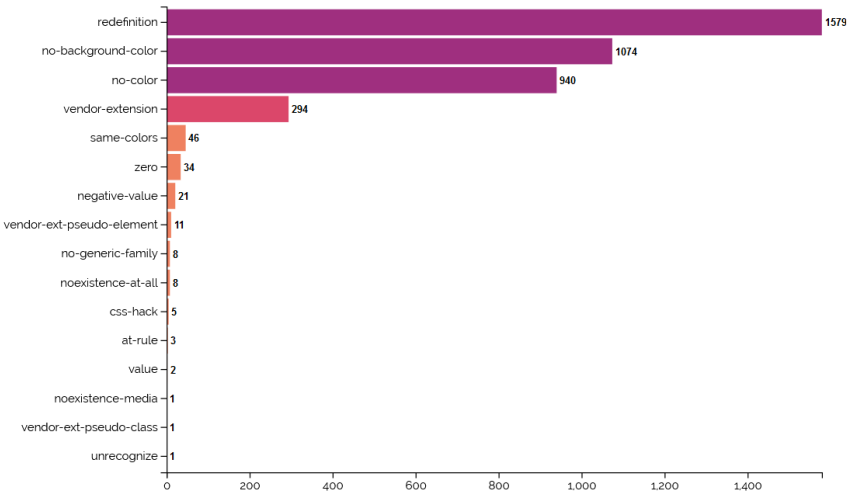
Filter result:
Errors (66) Warnings (3962)

PARETO-ANALYSE CSS

This chart is divided into 3 areas:

- Section A includes 70% of the occured errors.
- Section B includes 20% of the occured errors.
- Section C includes 10% of the occured errors.

Try to eliminate the errors from section A at first to get a visible improve of the page.



In line: 4
`Used value: auto`

NOEXISTENCE-AT-ALL

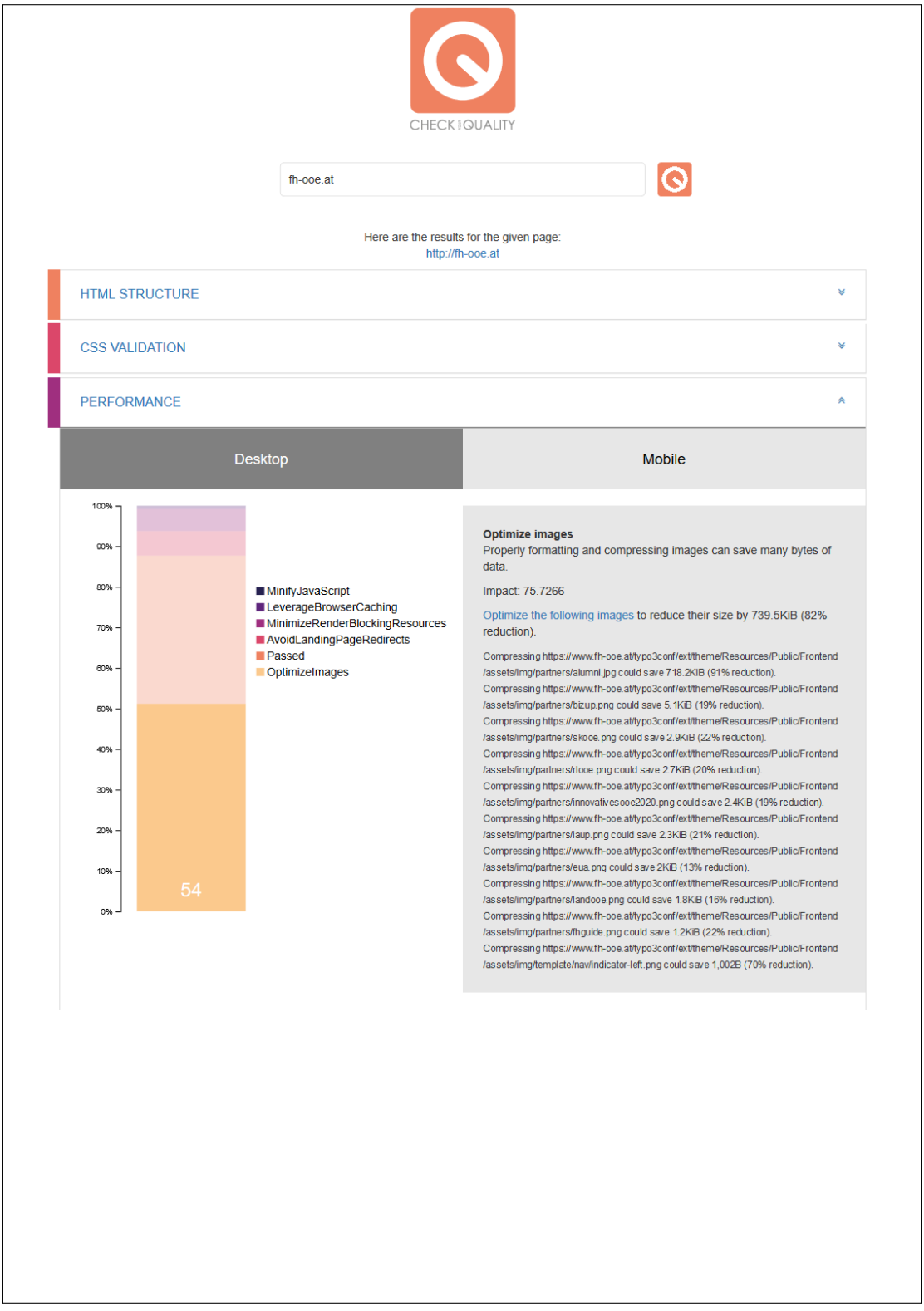
Die Eigenschaft "text-rendering" existiert nicht :

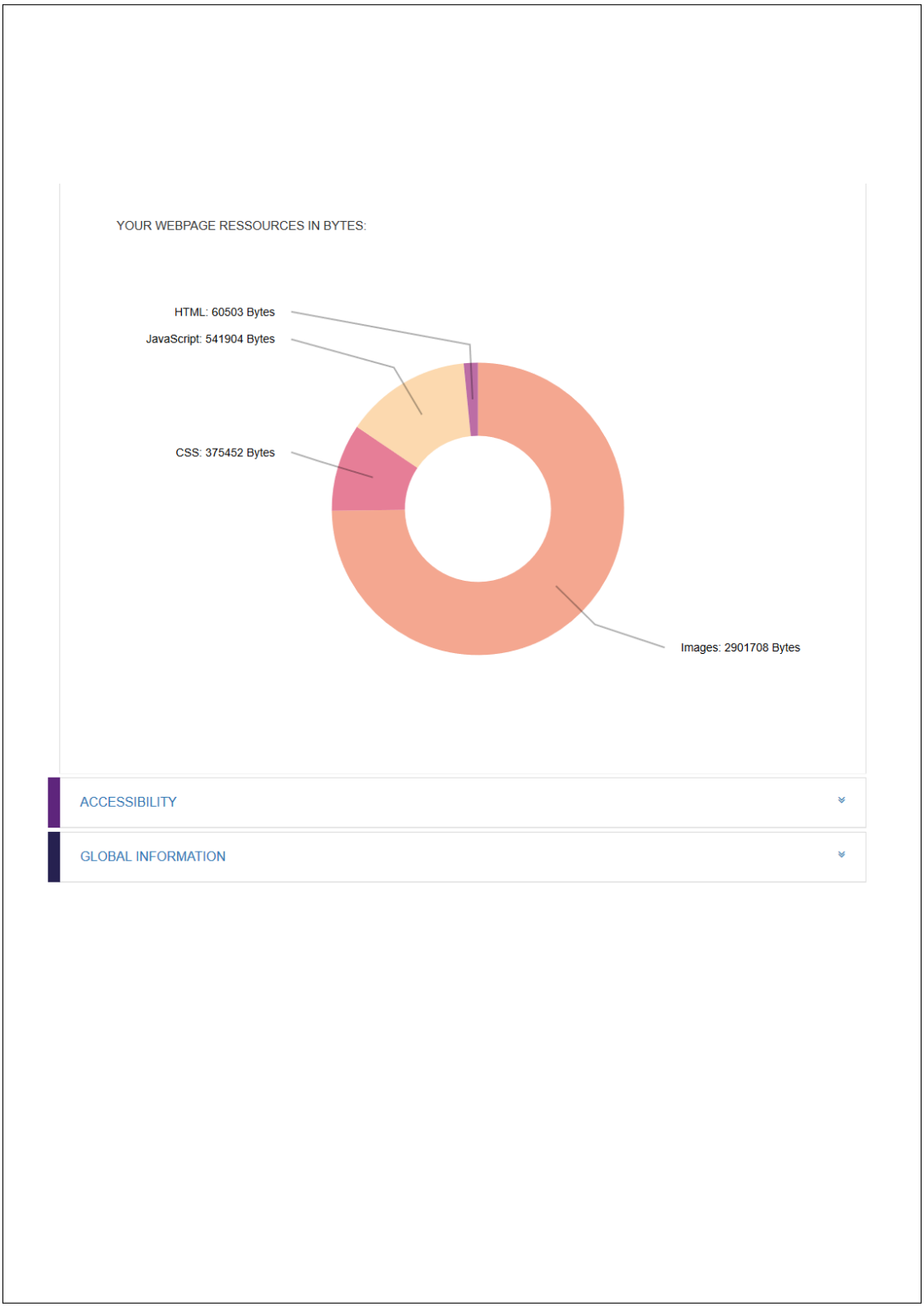
In line: 5
`Used value: 1.5`

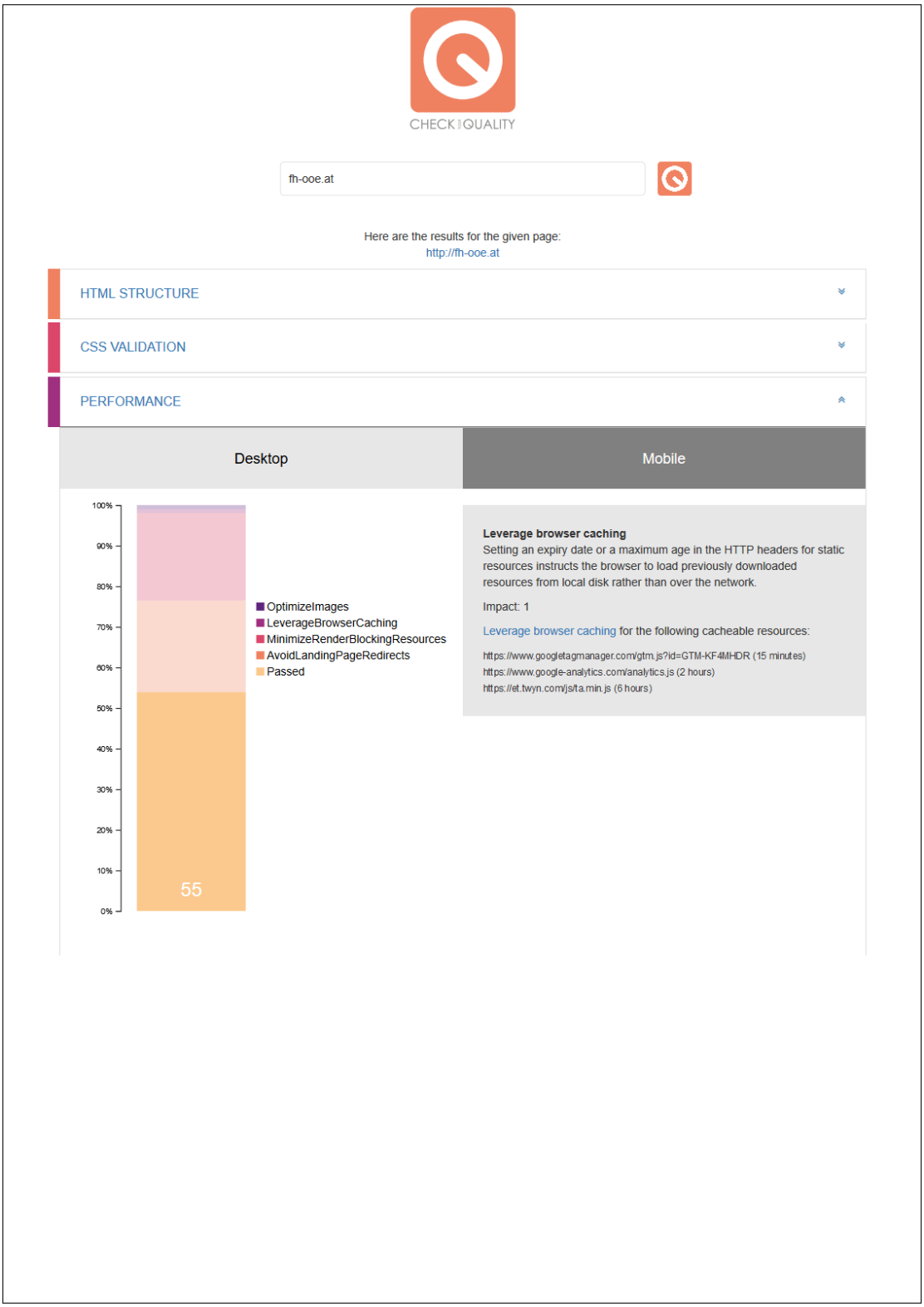
ZERO

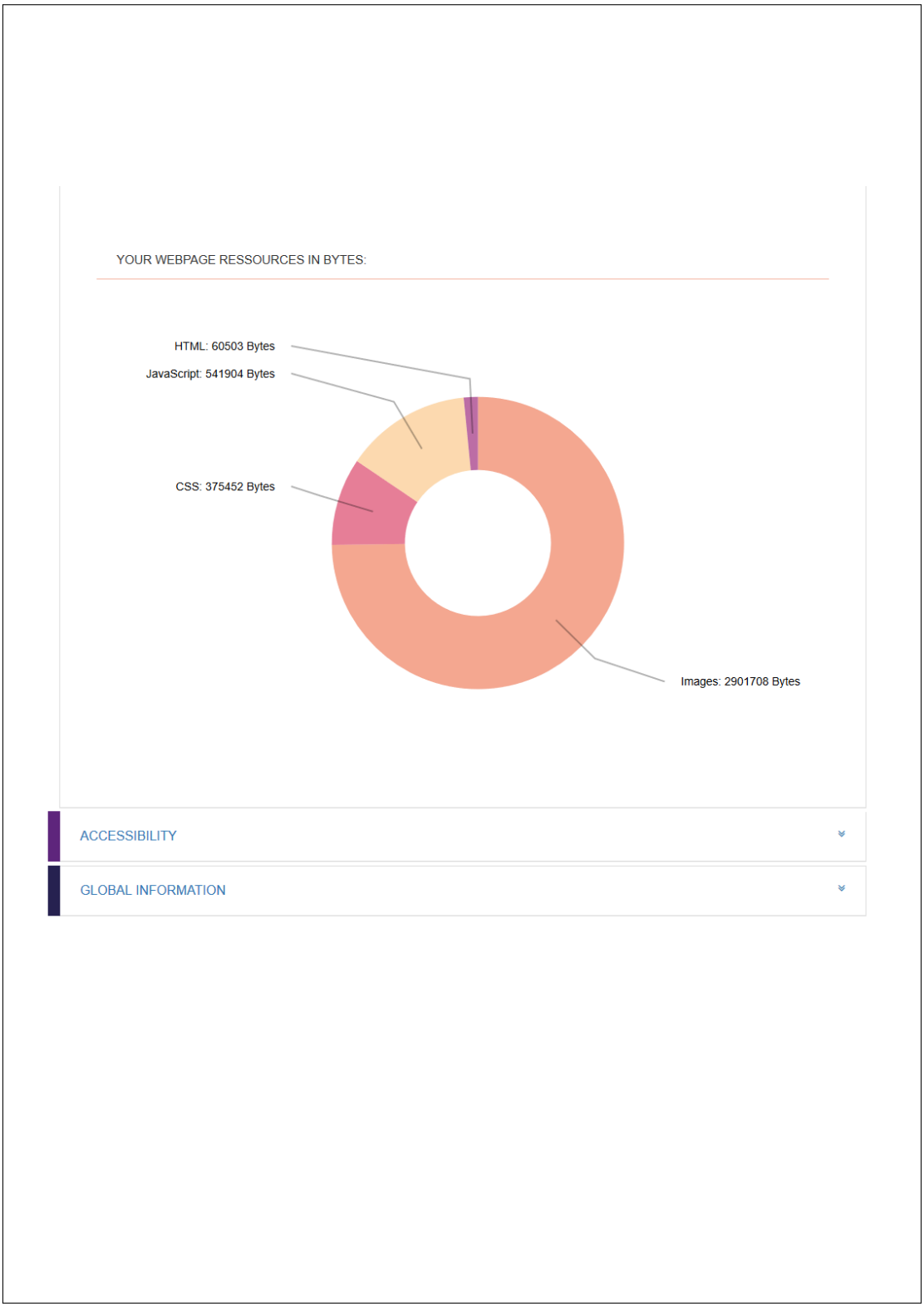
Ungültige Nummer : margin-top (nullbox.html#propdef-margin-top) Nur 0 kann ein "unit" sein. Nach der Zahl muss eine Einheit stehen. :

In line: 5 Used value: 1.5	ZERO Ungültige Nummer : margin-bottom (nullbox.html#propdef-margin-bottom) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: 1.5	ZERO Ungültige Nummer : margin-top (nullbox.html#propdef-margin-top) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: 0.75	ZERO Ungültige Nummer : margin-bottom (nullbox.html#propdef-margin-bottom) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: 0.75	ZERO Ungültige Nummer : margin-top (nullbox.html#propdef-margin-top) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: 0.75	ZERO Ungültige Nummer : margin-bottom (nullbox.html#propdef-margin-bottom) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: 0 0 0.75	ZERO Ungültige Nummer : margin (nullbox.html#propdef-margin) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: 1.5	ZERO Ungültige Nummer : margin-bottom (nullbox.html#propdef-margin-bottom) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: -0.25	ZERO Ungültige Nummer : padding-bottom (nullbox.html#propdef-padding-bottom) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: 3 0 1.5	ZERO Ungültige Nummer : margin (nullbox.html#propdef-margin) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: 0.75	ZERO Ungültige Nummer : margin-bottom (nullbox.html#propdef-margin-bottom) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: 1.5	ZERO Ungültige Nummer : margin-bottom (nullbox.html#propdef-margin-bottom) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: 0.75 1.5	ZERO Ungültige Nummer : padding (nullbox.html#propdef-padding) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: 0 0 1.5	ZERO Ungültige Nummer : margin (nullbox.html#propdef-margin) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :
In line: 5 Used value: 1.5	ZERO Ungültige Nummer : margin-bottom (nullbox.html#propdef-margin-bottom) Nur 0 kann ein “unit” sein. Nach der Zahl muss eine Einheit stehen. :


CHECK QUALITY
fh-ooe.at
Here are the results for the given page:
http://fh-ooe.at
HTML STRUCTURE
CSS VALIDATION
PERFORMANCE
Desktop
Mobile
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
54
MinifyJavaScript
LeverageBrowserCaching
MinimizeRenderBlockingResources
AvoidLandingPageRedirects
Passed
OptimizeImages
Optimize images
Properly formatting and compressing images can save many bytes of data.
Impact: 75.7266
Optimize the following images to reduce their size by 739.5KiB (82% reduction).
Compressing https://www.fh-ooe.at/typo3conf/ext/theme/Resources/Public/Frontend/assets/img/partners/alumni.jpg could save 718.2KiB (91% reduction).
Compressing https://www.fh-ooe.at/typo3conf/ext/theme/Resources/Public/Frontend/assets/img/partners/bizup.png could save 5.1KiB (19% reduction).
Compressing https://www.fh-ooe.at/typo3conf/ext/theme/Resources/Public/Frontend/assets/img/partners/skooe.png could save 2.9KiB (22% reduction).
Compressing https://www.fh-ooe.at/typo3conf/ext/theme/Resources/Public/Frontend/assets/img/partners/rlooe.png could save 2.7KiB (20% reduction).
Compressing https://www.fh-ooe.at/typo3conf/ext/theme/Resources/Public/Frontend/assets/img/partners/innovativesooe2020.png could save 2.4KiB (19% reduction).
Compressing https://www.fh-ooe.at/typo3conf/ext/theme/Resources/Public/Frontend/assets/img/partners/iaup.png could save 2.3KiB (21% reduction).
Compressing https://www.fh-ooe.at/typo3conf/ext/theme/Resources/Public/Frontend/assets/img/partners/eua.png could save 2KiB (13% reduction).
Compressing https://www.fh-ooe.at/typo3conf/ext/theme/Resources/Public/Frontend/assets/img/partners/landooe.png could save 1.8KiB (16% reduction).
Compressing https://www.fh-ooe.at/typo3conf/ext/theme/Resources/Public/Frontend/assets/img/partners/fhguide.png could save 1.2KiB (22% reduction).
Compressing https://www.fh-ooe.at/typo3conf/ext/theme/Resources/Public/Frontend/assets/img/template/nav/indicator-left.png could save 1,002B (70% reduction).


YOUR WEBPAGE RESSOURCES IN BYTES:
HTML: 60503 Bytes
JavaScript: 541904 Bytes
CSS: 375452 Bytes
Images: 2901708 Bytes
ACCESSIBILITY
GLOBAL INFORMATION


CHECK QUALITY
fh-ooe.at
Here are the results for the given page:
http://fh-ooe.at
HTML STRUCTURE
CSS VALIDATION
PERFORMANCE
Desktop
Mobile
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
55
OptimizeImages
LeverageBrowserCaching
MinimizeRenderBlockingResources
AvoidLandingPageRedirects
Passed
Leverage browser caching
Setting an expiry date or a maximum age in the HTTP headers for static resources instructs the browser to load previously downloaded resources from local disk rather than over the network.
Impact: 1
Leverage browser caching for the following cacheable resources:
https://www.googletagmanager.com/gtm.js?id=GTM-KF4MHDR (15 minutes)
https://www.google-analytics.com/analytics.js (2 hours)
https://et.twyn.com/js/ta.min.js (6 hours)


YOUR WEBPAGE RESSOURCES IN BYTES:
HTML: 60503 Bytes
JavaScript: 541904 Bytes
CSS: 375452 Bytes
Images: 2901708 Bytes
ACCESSIBILITY
GLOBAL INFORMATION

CHECK QUALITY

fh-ooe.at

Here are the results for the given page:
http://fh-ooe.at

HTML STRUCTURE

CSS VALIDATION

PERFORMANCE

ACCESSIBILITY

FAILED VS. PASSED

The tenon.io API test 74 different error-cases.

This section shows the passed amount of tests.

This section shows the failed amount of tests.

66 | 8

PARETO-ANALYSE CSS

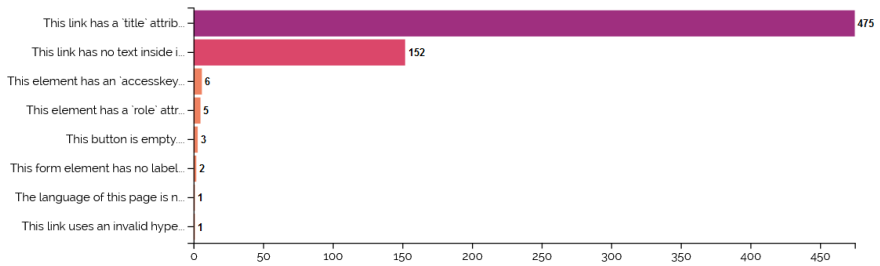
This chart is divided into 3 areas:

Section A includes 70% of the occured errors.

Section B includes 20% of the occured errors.

Section C includes 10% of the occured errors.

Try to eliminate the errors from section A at first to get a visible improve of the page.



CHECK QUALITY

fh-ooe.at

Here are the results for the given page:
http://fh-ooe.at

HTML STRUCTURE

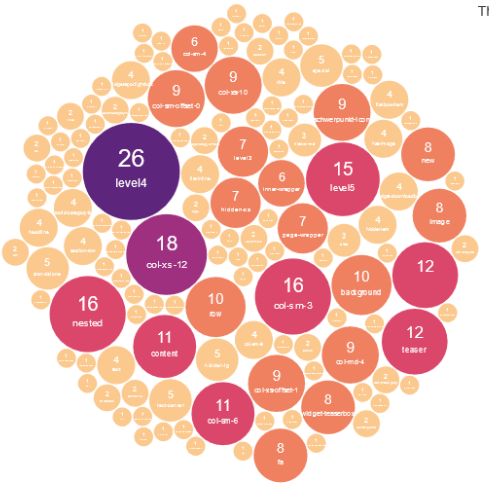
CSS VALIDATION

PERFORMANCE

ACCESSIBILITY

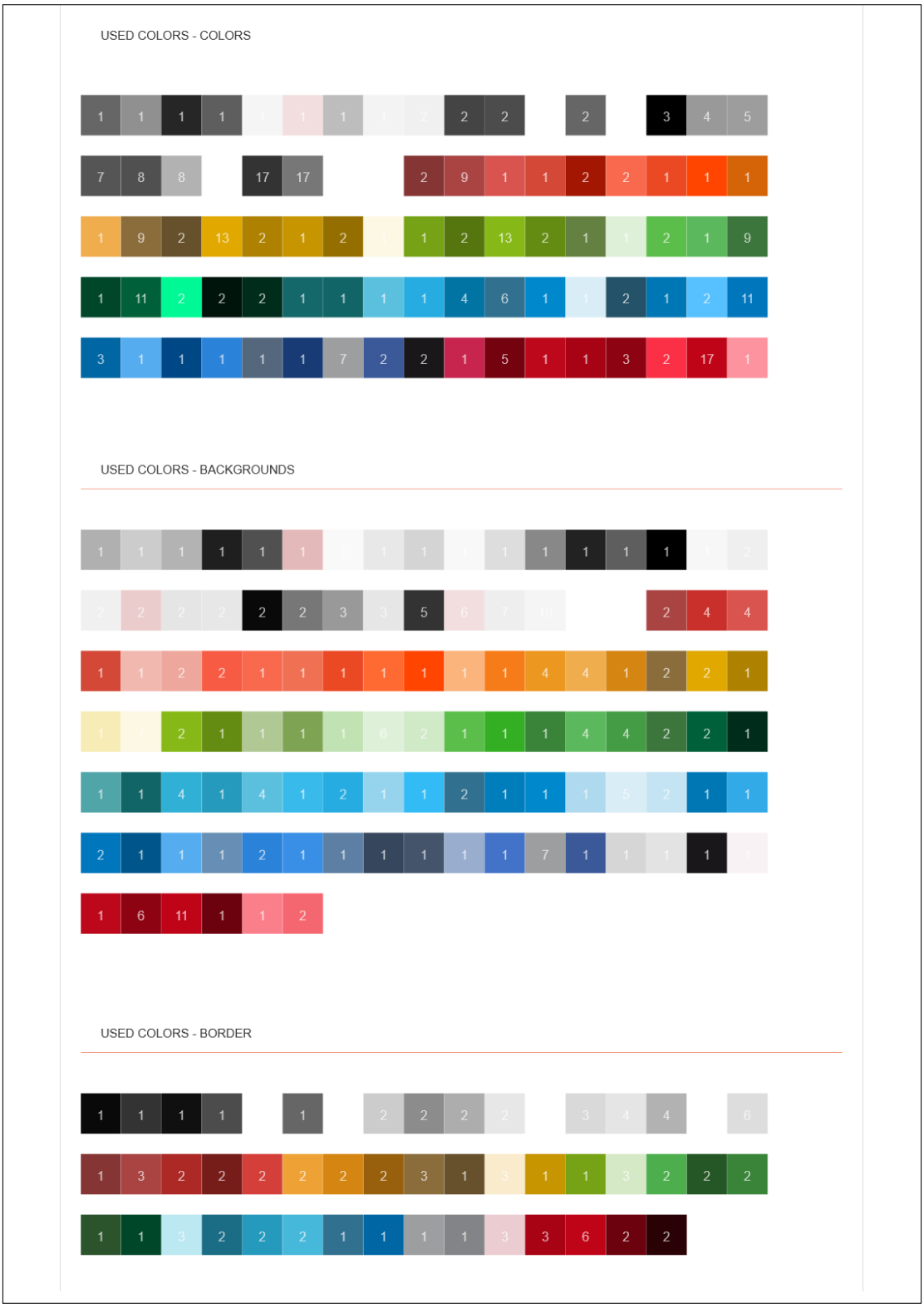
GLOBAL INFORMATION

CLASSNAMES



The following classnames are used in the given source code:

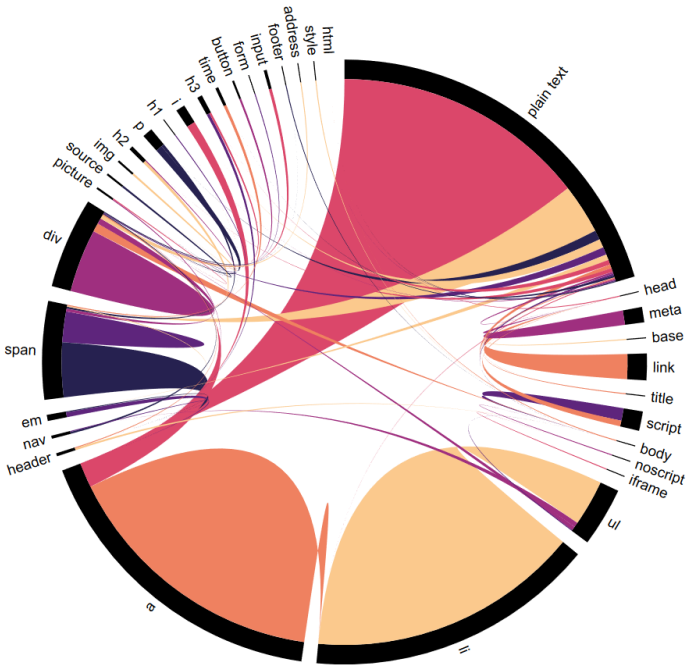
- [1] no-js
- [1] hide
- [1] world-hage
- [1] world-linz
- [1] world-stey
- [1] world-wels
- [1] visible-xs
- [1] world
- [1] subnav
- [1] slider-container
- [1] masthead
- [1] after-slider
- [1] banded-section
- [1] banded-header
- [1] tabbed-section
- [1] next-to-headline
- [1] nav
- [1] nav-tabs
- [1] no-border
- [1] empty-tab
- [1] tab-content
- [1] tab-pane
- [1] box-links
- [1] icon-fho_webicon_1
- [1] icon-fho_webicon_2


USED COLORS - COLORS
USED COLORS - BACKGROUNDS
USED COLORS - BORDER

RELATIONS BETWEEN HTML ELEMENTS:

Info:
There are 3 input in div Elements.
There are 0 div in input Elements.

Categorical Info:
There are 3 input Element(s) in the HTML source code



Quellenverzeichnis

Literatur

[1] DIN EN ISO 9000. *Qualitätsmanagementsysteme – Grundlagen und Begriffe*. Berlin: Beuth Verlag, 2005 (siehe S. 3).

[2] Holger Brüggemann und Peik Bremer. *Grundlagen Qualitätsmanagement. Von den Werkzeugen über Methoden zum TQM*. Wiesbaden: Springer Vieweg, 2012 (siehe S. 3, 4).

[3] Philip B. Crosby. *Qualität kostet weniger: Handbuch der Fehlerverhütung für Führungskräfte*. Holz, 1972 (siehe S. 4).

[4] Layla Hasan und Emad Abuelrub. „Assessing the quality of web sites“. *Applied Computing and Informatics* 9.1 (2011), S. 11–29 (siehe S. 29).

[5] Tobias Hauser und Christian Wenz. *Websites optimieren - Das Handbuch*. 2. Aufl. Wiesbaden: Springer, 2015 (siehe S. 9, 12).

[6] Tamara Munzner. *Visualization Analysis and Design*. CRC Press Taylor und Francis Group, 2014 (siehe S. 20, 43).

[7] Doaa Nabil, Abeer Mosad und Hesham A Hefny. „Web-Based Applications quality factors : A survey and a proposed conceptual model“. *Egyptian Informatics Journal* 12.3 (2011), S. 211–217 (siehe S. 31).

[8] Jakob Nielsen. *Usability Engineering*. London: Morgan Kaufmann, 1993 (siehe S. 8).

[9] Leslie F. Sikos. *Web Standards. Mastering HTML5, CSS3, and XML*. 2. Aufl. Apress, 2014, S. 524 (siehe S. 7, 24, 25).

[10] Steve Souders. *High Performance Web Sites: Essential Knowledge for Frontend Engineers*. 1. Aufl. Sebastopol: O'Reilly Media, 2007, S. 1–170 (siehe S. 9).

[11] Torsten Stapelkamp. *Informationsvisualisierung - Web - Pint - Signaletik. Erfolgreiches Informationsdesign: Leitsysteme, Wissensvermittlung und Informationsarchitektur*. 2. Aufl. Heidelberg: Springer, 2013 (siehe S. 15–17).

Software

[12] *CSS Validator*. URL: https://www.npmjs.com/package/w3c-css (siehe S. 34).

[13] *HTML Validator*. URL: https://www.npmjs.com/package/html-validator (siehe S. 34).

Online-Quellen

[14] *Barriere*. Apr. 2018. URL: https://www.duden.de/rechtschreibung/Barriere (besucht am 03. 04. 2018) (siehe S. 11).

[15] *Barrierefreies Web – Zugang für alle - Digitales Österreich*. 2018. URL: https://www.digitales.oesterreich.gv.at/barrierefreiheit (besucht am 03. 04. 2018) (siehe S. 11, 12).

[16] Mike Bostock. *d3.js*. 2017. URL: https://d3js.org/ (besucht am 11. 07. 2018) (siehe S. 38).

[17] Andrew Brehaut. *color-js*. 2017. URL: https://github.com/brehaut/color-js (besucht am 10. 07. 2018) (siehe S. 43).

[18] webAIM - Center for Persons with Disabilities. *Keyboard Accessibility - Accesskey*. Jan. 2016. URL: https://webaim.org/techniques/keyboard/accesskey (besucht am 23. 07. 2018) (siehe S. 52).

[19] The Data und Visualisation Catalogue. *Chord Diagram*. 2018. URL: https://datavizcatalogue.com/methods/chord_diagram.html (besucht am 05. 08. 2018) (siehe S. 44, 45).

[20] Google. *Enable Compression - Google PageSpeed insights*. Jan. 2018. URL: https://developers.google.com/speed/docs/insights/EnableCompression (besucht am 17. 07. 2018) (siehe S. 49).

[21] Google. *Google PageSpeed Insights*. Jan. 2018. URL: https://developers.google.com/speed/docs/insights/about?hl=de-DE&utm_source=PSI&utm_medium=incoming-link&utm_campaign=PSI (besucht am 14. 08. 2018) (siehe S. 35).

[22] Google. *Leverage Browser Caching - Google PageSpeed insights*. Jan. 2018. URL: https://developers.google.com/speed/docs/insights/LeverageBrowserCaching (besucht am 17. 07. 2018) (siehe S. 49).

[23] Facebook Inc. *React.Component*. 2018. URL: https://reactjs.org/docs/react-component.html (besucht am 09. 08. 2018) (siehe S. 33).

[24] *informieren*. Apr. 2018. URL: https://www.duden.de/rechtschreibung/informieren (besucht am 03. 04. 2018) (siehe S. 15).

[25] Alexa Internet. *Traffic Statistics - orf.at*. URL: https://www.alexa.com/siteinfo/orf.at (besucht am 17. 07. 2018) (siehe S. 47).

[26] InternetLiveStats. *Total number of Websites*. 2018. URL: http://www.internetlivestats.com/total-number-of-websites/ (besucht am 05. 08. 2018) (siehe S. 1).

[27] *ISO/IEC 9126*. 2018. URL: https://de.wikipedia.org/wiki/ISO/IEC_9126 (besucht am 03. 08. 2018) (siehe S. 31).

[28] Helmut Krcmar. *Unterscheidung: Daten, Information, Wissen*. 2012. URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/daten-wissen/Informationsmanagement/Information-/index.html/?searchterm=Unterscheidung:%20daten%20information%20wissen (besucht am 03. 07. 2018) (siehe S. 15).

[29] Mozilla und einzelne Mitwirkende. *Fetch API*. Juli 2018. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (besucht am 14. 08. 2018) (siehe S. 36).

[30] Mozilla und einzelne Mitwirkende. *Promise*. Mai 2018. URL: https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Global_Objects/Promise (besucht am 09. 08. 2018) (siehe S. 33, 36).

[31] Ronald Moen und Clifford Norman. *Evolution of the PDCA Cycle*. 2009. URL: https://www.researchgate.net/publication/228475044_Evolution_of_the_PDCA_cycle (besucht am 03. 08. 2018) (siehe S. 4).

[32] *node.js*. 2018. URL: https://nodejs.org/en/ (besucht am 03. 07. 2018) (siehe S. 32).

[33] *Qualität*. Apr. 2018. URL: https://www.duden.de/rechtschreibung/Qualitaet (besucht am 26. 04. 2018) (siehe S. 3).

[34] selvims. *McCall's Quality Model - 1977*. 2011. URL: https://de.scribd.com/doc/55502678/McCall-s-Quality-Model-1977 (besucht am 05. 08. 2018) (siehe S. 31).

[35] Splendid Internet. *Was ist neu an Googles PageSpeed Insights?* 2018. URL: https://www.splendid-internet.de/blog/was-ist-neu-an-googles-pagespeed-insights/ (besucht am 03. 07. 2018) (siehe S. 26).

[36] W3C. *Web Content Accessibility Guidelines (WCAG) 2.0 - Autorisierte deutsche Übersetzung*. 2009. URL: http://www.w3.org/Translations/WCAG20-de/ (besucht am 03. 07. 2018) (siehe S. 12–14).