HAW HAMBURG

BACHELORARBEIT

Konzeption und Entwicklung einer Single-Page-Website mit scrollbasiertem Animationseffekt

vorgelegt am 4. September 2024

Lan Phuong Nguyen

Erstprüferin: Prof. Dr. Larissa Putza
Zweitprüfer: André Jeworutzki

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Medientechnik
Finkenau 35
22081 Hamburg

Zusammenfassung

Diese Arbeit befasst sich mit der Konzeption und Entwicklung einer Single-Page-Website mit Animationseffekten, die auf dem Scrollverhalten der Nutzer basieren. Der Fokus liegt dabei auf der Gestaltung ansprechender Animationen, die der Website eine lebendige und beeindruckende Benutzererfahrung verleihen sollen. Für die Erstellung der Animationen werden CSS, JavaScript und die externe Bibliothek GSAP verwendet. Zum Schluss wird die Website mit der WebUSE-Methode getestet, die auf Umfragen basiert, um die Usability der Website zu analysieren und zu bewerten.

Abstract

This thesis focuses on the conception and development of a single-page website with animation effects that are based on the user's scrolling behavior. The primary goal is to design visually appealing animations that create a dynamic and engaging user experience. CSS, JavaScript, and the external library GSAP are used for the implementation of these animations. Finally, the website is evaluated using the WebUSE method, which is based on surveys, to analyze and assess the website's usability.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Zielsetzung

Diese Arbeit zielt darauf ab, eine Single-Page-Website zu entwickeln, die sowohl vertikale als auch horizontale Scrollen integriert. Das Hauptziel besteht darin, grundlegendes Wissen über den Aufbau und die Gestaltung einer Website zu erwerben, sowie die Anwendung von CSS und JavaScript für die Aktivierung der Animationen beim Scrollen zu erlernen. Die GSAP-Bibliothek wird verwendet, um horizontale Scrollen und komplexere Animationen zu erstellen. Ein wesentlicher Bestandteil der Arbeit ist die Gestaltung einer ästhetisch ansprechenden Benutzeroberfläche, die sicherstellt, dass der Inhalt leicht lesbar ist und die Animationen korrekt funktionieren. Es ist entscheidend, dass die Animationen flüssig ohne Ruckeln ablaufen, und dass sie richtig aktiviert werden, wenn der Benutzer an eine bestimmte Position scrollt.

Bevor eine Website offiziell genutzt wird, muss ein Usability-Test durchgeführt werden. Dieser Test ist von großer Bedeutung, da ein ansprechendes Design allein nicht garantiert, dass die Website benutzerfreundlich ist. Durch den Usability-Test werden wertvolle Rückmeldungen von den Nutzern gesammelt, um die Stärken und Schwächen der Website zu erkennen. Anhand dieser Erkenntnisse können gezielte Verbesserungen vorgenommen werden, um sicherzustellen, dass die Website sowohl in Bezug auf das Design als auch auf die Benutzerfreundlichkeit optimal gestaltet ist. Die Testergebnisse helfen dabei, geeignete Lösungen zu finden, um die Website weiter zu optimieren und eine durchweg positive Nutzererfahrung zu gewährleisten.

2 Die Geschichte des Webdesigns

Das Webdesign hat sich seit den 1990er Jahren stark weiterentwickelt. Laut Dr. Umapathi Janne lässt sich die Geschichte des Webdesigns in vier Hauptzeitabschnitte unterteilen. Im Folgenden wird eine Zusammenfassung dieser vier Phasen gegeben, die sowohl den technischen Fortschritt als auch die Änderungen der Benutzeroberfläche widerspiegeln.

Die Anfänge des Webdesigns (1991-1994)

Das Webdesign begann mit der Erfindung des World Wide Web (abgekürzt WWW) durch Sir Tim Berners-Lee im Jahr 1989. Berners-Lee veröffentlichte 1991 die weltweit erste Webseite *'info.cern.ch'*. Diese Webseite basierte auf HTML, einer von Berners-Lee erfundenen Markup-Sprache, und war zu 100 % textbasiert. In dieser Zeit war das <table>-Tag die einzige Methode zur Strukturierung von Daten auf Webseiten. Es gab keine grafischen Elemente oder komplexe Layouts, nur Text. Dies unterschied sich stark von dem, was wir heute als „Webdesign“ kennen (Janne, 2024, S. 2).

1993 entwickelten Marc Andreessen und Eric Bina den Browser Mosaic (später Netscape Navigator). Mosaic konnte als erster Webbrowser nicht nur Texte, sondern auch Grafiken in ein und demselben Fenster darstellen (Domin, 2018). Das Grafikdesign und die visuelle Kommunikation waren jedoch sehr begrenzt. Ein Jahr später wurde das *World Wide Web Consortium (W3C)* gegründet, um eine internationale Vereinigung für die Entwicklung des Web zu schaffen (Wilde, 1999, S. 14). In dieser Ära wuchs die Anzahl der Webseiten exponentiell. 1991 gab es nur eine Website. 1994 waren es insgesamt 2738 (Janne, 2024, S. 2).

Die Evolution der Web-Entwicklung (1995–2000)

Die Web-Programmiersprache JavaScript wurde 1995 von der Firma Netscape entwickelt, um interaktive oder dynamische Elemente auf Webseiten zu ermöglichen (Sinner, 2018, S. 32). Später im Jahr 1996 brachte die Einführung von Cascading Style Sheets (CSS) eine Revolution im Webdesign. CSS ermöglicht es, die inhaltliche Gliederung (HTML) von den visuellen Elementen (Layout, Farben und Schriftart) zu trennen. Designer konnten das Erscheinungsbild einer Website flexibler steuern als je zuvor.

Macromedia Flash 1.0 war in den späten 1990er Jahren das führende Tool für Webanimationen, allerdings mit dem Nachteil, dass die Animationen nur angezeigt werden konnten,

wenn das erforderliche Plugin installiert war. Trotz dieser Schwächen leitete Flash eine neue Ära der Web-Animation ein, doch wurde es 2020 offiziell eingestellt (Janne, 2024, S. 3).

2000-2006:

In diesem Zeitraum wurden Webseiten hauptsächlich für Desktop-Browser erstellt. Obwohl das Konzept des *„Responsive Webdesigns“* noch in den Kinderschuhen steckte, begannen Webdesigner bereits damit, verschiedene Versionen von Webseiten zu erstellen, die auf unterschiedlichen Browsern und Geräten zugänglich waren. Dies erwies sich jedoch als zeitaufwendig und ineffizient, da für jede Plattform separate Updates erforderlich waren (Janne, 2024, S. 3–4).

Mit dem Aufstieg des Internets für geschäftliche Zwecke und der Popularität von Blogs gewannen Content Management Systeme (CMS) an Bedeutung. WordPress entwickelte sich von einer einfachen Blogging-Plattform zu einem der weltweit führenden CMS, das heute etwa 35,9 % aller Webseiten betreibt (Janne, 2024, S. 4). Außerdem revolutionierten soziale Netzwerke wie Myspace und Facebook die Internetkommunikation und beeinflussten die Entwicklung des modernen Webdesigns.

Die mobile Ära (2007 bis heute)

Die mobile Ära begann 2007 mit der Einführung des iPhones. Steve Jobs, der damalige CEO von Apple, sagte in seiner Präsentation, dass Safari-Browser “der erste voll nutzbare HTML-Browser auf einem Mobilgerät” war. Das iPhone war das einzigartiges Gerät, das Flash nicht unterstützte. In seinem offenen Brief *„Thoughts on Flash“* im Jahr 2010 nannte Steve Jobs viele Gründe, warum Apple Flash nicht auf dem iPhone zulassen würde. Dazu gehörten Zuverlässigkeit, Sicherheit, Leistung und vor allem die Akkulaufzeit. Dieses Schreiben hatte einen großen Einfluss auf die Erstellung von Webseiten und führte zu einem spürbaren Rückgang von Flash (Janne, 2024, S. 4).

Da die Nutzung von mobilen Geräten stark zunahm, wurde responsives und mobilfreundliches Design zur Notwendigkeit. 2010 stellte der Webdesigner Ethan Marcotte das Konzept des „Responsive Webdesigns“ vor (Janne, 2024, S. 5), das darauf abzielt, Websites so zu gestalten, dass sie auf allen Gerätetypen gut aussehen und funktionieren, unabhängig von der Bildschirmgröße. In den 2010er Jahren setzte sich das flache Design durch, das klare, minimalistische Elemente bevorzugte. Dies stand im Gegensatz zu den vorherigen komplexeren Designs, die zwar realistischer wirkten, aber oft längere Ladezeiten erforderten. Der Wechsel vom Skeuomorphismus[1] zum Flat-Design sorgte für eine effizientere Benutzererfahrung.

[1] Skeuomorphismus ist ein Designkonzept, bei dem Elemente aus der realen Welt digital nachgebildet werden.

3 Website in Zeiten der Digitalisierung

"If your business is not on the Internet then your business will be out of business" [1]-Bill Gate

Dieses Zitat von Bill Gates verdeutlicht eindrucksvoll die essenzielle Bedeutung der Online-Präsenz für Unternehmen. Der zunehmende Einsatz mobiler Geräte und die kontinuierliche Nutzung des Internets führen dazu, dass immer mehr Alltagsdienste in die digitale Welt transformiert und durch das „World Wide Web" global verfügbar gemacht werden (Staab, 2019, S. 21). Unternehmen, die nicht in der Lage sind, sich in diesem digitalen Umfeld zu behaupten, riskieren, ihre Relevanz und letztlich auch ihre Existenz zu verlieren. Websites sind zu einem wichtigen Instrument für Unternehmen geworden, um Kunden anzuziehen und zu halten.

Website als zentrale Kommunikationsplattform und Verkaufsstelle

Im Vergleich zu klassischen Medien wie Fernsehen, Radio, Print etc. haben Websites einen erheblichen Vorteil, dass sie rund um die Uhr verfügbar sind und es den Nutzern ermöglichen, jederzeit auf relevante Informationen zuzugreifen (Streich, 2000, S. 4). Website dient als Kommunikationsplattform, auf der die Kunden sich jederzeit über Produkte und Dienstleistungen informieren oder mit der Firma Kontakt aufnehmen. Mit einem Kontaktformular können Nutzer ohne Aufwand (Telefonbuch konsultieren, Visitenkarte hervorsuchen etc.) direkt mit der Firma kommunizieren (Kielholz, 2008, S. 254).

Für viele Unternehmen sind Websites nicht nur Kommunikationsplattformen, sondern auch Verkaufsstellen. Dank neuester Entwicklungen in der Technologie ist der Transfer sowohl von Informationen als auch von Geld möglich (Herbst, 1999, S. 133). E-Commerce Websites ermöglichen es Unternehmen, ihre Produkte und Dienstleistungen direkt online zu verkaufen. Jeder potenzielle Kunde mit Internetzugang kann von überall auf der Welt auf Websites zugreifen und bequem einkaufen. Unternehmen haben die Möglichkeit, ihre Umsätze zu steigern, ohne auf physische Verkaufsräume angewiesen zu sein.

SEO-Marketing

[1] https://www.tomorrow.city/the-importance-of-business-visibility-and-online-reputation/

In einer zunehmend wettbewerbsorientierten Wirtschaft ist es für Unternehmen entscheidend, online sichtbar zu sein. Suchmaschinenoptimierung (SEO), Content-Marketing und Social-Media-Integration sind einige der Strategien, die durch Websites unterstützt werden. Nach Einschätzung von Microsoft (Southern 2020) ist SEO die wichtigste Marketingdisziplin (Raaf, 2021, S. 3). Laut einer Forschung von BrightEdge im Jahr 2020 werden im Durchschnitt über 50 % des gesamten Website-Traffics durch unbezahlte Suchergebnisse generiert, während 15 % über bezahlte Anzeigen und lediglich rund 4 % der Besucher von Social Media kommen (Raaf, 2021, S. 3). Durch SEO können Unternehmen ihre Websites effektiv bewerben und ihr Ranking in Suchmaschinen wie z.B. Google und Microsoft Bing verbessern. Je besser die Website in den Suchergebnissen sichtbar ist, desto höher sind die Chancen für eine bessere Auffindbarkeit.

Google Analytics ist ein wesentliches Instrument für SEO. Jede Besucher-Interaktion, von Klicks bis hin zu Conversions[2], liefert Website-Statistiken. Mit Hilfe von Google Analytics können Unternehmen viele sehr nützliche Informationen erfahren, wie z.B: wie lange Nutzer insgesamt in dem Shop verweilen, welche Keywords bei einer Suche mit einer Suchmaschine von User benutzt wurden, an welcher Stelle Kunden einen Kaufvorgang abbrechen etc (Süss, 2016, S. 91). Durch regelmäßige Analysen und Updates können Unternehmen Marketingmaßnahmen optimieren und sicherstellen, dass ihre Website immer auf dem neuesten Stand ist und den Erwartungen ihrer Kunden entspricht.

Einsatz moderner Technologien

In der heutigen digitalen Welt setzen immer mehr Unternehmen auf den Einsatz von künstlicher Intelligenz (KI), um ihren Kundenservice zu optimieren. Ein besonders bekanntes Anwendungsbeispiel ist die Integration von Chatbots auf Websites. Chatbots unterstützen Kunden rund um die Uhr bei der Suche nach Informationen über Produkte oder Dienstleistungen. Dabei ersetzen Chatbots zunehmend menschliche Mitarbeiter. Dies führt nicht nur zu einer Entlastung des Personals, sondern auch zu einer erheblichen Kosteneinsparung. Trotz großer Vorteile sind Chatbots kein Allheilmittel für die komplette Automatisierung der Kundeninteraktion, weil sie nur einfache Fragen beantworten können, doch bei komplexeren Kundenanfragen und -problemen stoßen sie häufig an ihre Grenzen (Hadwich, 2022, S. 65). Chatbots können menschliche Mitarbeiter nicht komplett ersetzen. Daher sollten Unternehmen einen hybriden Ansatz verwenden, dabei ein Chatbot Routinefragen und -probleme übernimmt, während die Mitarbeitenden sich auf komplexere Interaktionen mit den Kunden konzentrieren können (Hadwich, 2022, S. 65). Diese Kombination aus Technologie und Menschen ermöglicht es Unternehmen, ihren Kundenservice bestmöglich zu optimieren.

[2]Eine Conversion beschreibt die Umwandlung eines Website-Besuchers zu einem Kunden.

4 Web design trend

Mit der rasanten technologischen Entwicklung hat sich auch das Webdesign in den letzten Jahren stark gewandelt. Dies erfordert Unternehmen und Designern, sich kontinuierlich anzupassen und weiterzuentwickeln, um wettbewerbsfähig zu bleiben und den Erwartungen der Nutzer gerecht zu werden. Dieses Kapitel behandelt die aktuell herausragende Webdesign-Trends.

4.1 Animation-Effekt

Animationen auf Websites sind weit mehr als nur ein ästhetisches Element. Sie dienen nicht nur dazu, die Website visuell ansprechender zu gestalten, sondern auch, um Geschichten zu erzählen und Botschaften zu vermitteln. Durch gezielte Animationen können die Inhalte lebendig gemacht werden, wodurch eine tiefere Verbindung zum Nutzer aufgebaut wird. Diese Art des Storytellings ermöglicht es, Aufmerksamkeit zu erlangen oder Emotionen anzuregen (Riempp, 2019, S. 66).

In der Vergangenheit war Flash die bevorzugte Technologie, um Animationen auf Websites zu erstellen. Aufgrund der in Kapitel 2 erwähnten Schwächen wird Flash heute jedoch nicht mehr verwendet. Heute werden Animationen hauptsächlich mit HTML5, CSS3 und JavaScript erstellt. Diese Technologien bieten nicht nur bessere Performance und Kompatibilität über verschiedene Geräte hinweg, sondern auch eine höhere Sicherheit und Flexibilität.

Parallax Scrolling

Ein beliebter und häufig eingesetzter Effekt im Webdesign ist der Parallax-Effekt. Dabei bewegen sich beim Scrollen mindestens zwei oder mehr Elemente mit unterschiedlichen Geschwindigkeiten, was einen dreidimensionalen Eindruck und eine gewisse Tiefe erzeugt (Riempp, 2019, S. 118). Dies macht den Scrollvorgang lebendiger und dynamischer.

Ein anschauliches Beispiel für den Parallax-Effekt im Alltag ist das Autofahren (Siehe Abbildung 4.1). Wenn man aus dem Fenster schaut, scheinen Objekte in der Nähe, wie Bäume am Straßenrand, sich sehr schnell zu bewegen. Gebäude, die weiter entfernt sind, bewegen sich langsamer. Der Mond, der am weitesten von uns entfernt ist,

scheint sich irgendwie an einer festen Stelle zu verharren. Diese Wirkung wird auch als Bewegungsparallaxe bezeichnet (Riempp, 2019, S. 119).

Abbildung 4.1: Parallax-Effekt (Quelle: Swift)

4.2 Verwendung von 3D-Elementen

Ursprünglich wurde minimalistisches Design bevorzugt und 3D-Elemente wurden selten eingesetzt, hauptsächlich aufgrund der technischen Einschränkungen der damaligen Browser und der begrenzten Rechenleistung der Geräte. Dies hat sich jedoch grundlegend geändert. Heute verfügen moderne Browser über integrierte Unterstützung für JavaScript und 3D-Technologien wie WebGL, die es Entwicklern ermöglicht, komplexe 2D- und 3D Computergrafiken direkt im Webbrowser zu rendern. Three.js ist eine leistungsstarke JavaScript-Bibliothek, die zum Erstellen und Anzeigen animierter 3D-Inhalten mit WebGL verwendet wird. Mit Three.js kann man problemlos 3D-Objekte modellieren, Beleuchtungseffekte hinzufügen und sogar Virtual-Reality-Szenen mit nur wenigen Zeilen JavaScript (Dirksen, 2023, S. 3).

Heutzutage werden 3D-Webdesign in vielen Anwendungsbereichen eingesetzt, z.B: E-Commerce, Spielen, Architektur, Automobilindustrie, Trainingssimulatoren. Eine der häufigsten Anwendungen davon ist die Produktpräsentation im Bereich E-Commerce. Durch die Integration von 3D-Modellen können Unternehmen ihre Produkte auf eine interaktive und eindrucksvolle Weise präsentieren. Kunden haben die Möglichkeit, Produktmodelle um beliebigen Winkel drehen oder zoomen. Zudem können 3D-Modelle schnell angepasst werden, wenn sich Bestandteile verändern, wie z.B. Farben und Größe.

Auf diese Weise können potenzielle Kunde das Produkt schon vor dem Kauf besser verstehen, was die Kaufentscheidung beschleunigt.

Abbildung 4.2: Screenshot von Apple (Quelle: Eigene Darstellung)

4.3 Mobile First

Mit dem Anstieg der Smartphone-Nutzung ist die Anzahl der Nutzer, die über mobile Geräte auf Websites zugreifen, stark gestiegen. Weltweit wurden mobile Endgeräte (Mobiltelefon und Tablet) im Jahr 2023 für 55 Prozent, in Afrika sogar 73.8 Prozent aller Seitenaufrufe genutzt[1]. Responsive Webdesign wird nicht mehr als nur ein Trend angesehen, sondern als eine grundlegende Notwendigkeit im modernen Webdesign. Eine Website, die nicht für mobile Geräte optimiert ist, bietet eine schlechte Benutzererfahrung, was zu höheren Absprungraten und potenziellen Umsatzverlusten führen kann.

Wie bereits im Kapitel 1 erwähnt, handelt es sich bei Responsive Webdesign darum, dass die Website auf allen Gerätetypen gut aussieht und funktioniert, unabhängig von der

[1] Anteil mobiler Endgeräte an allen Seitenaufrufen weltweit 2023

Bildschirmgröße. Die Website besitzt nämlich nur eine URL, einen HTML-Code und der Inhalt passt sich automatisch durch die Möglichkeiten von CSS3 den unterschiedlichen Bildschirmgrößen und Geräten an (Michaelis, 2021). Eine responsive Website wird in der Regel durch Media Queries, Fluid Grids und skalierbare Bilder erstellt (Kadlec, 2013, S. 13).

Früher wurden Websites oft zunächst für große Bildschirme wie Desktop und Laptop gestaltet und dann schrittweise auf kleinere Geräte angepasst. Heutzutage wird nun häufig zuerst für mobile Geräte entwickelt – ein Ansatz, der als „Mobile First“ bekannt ist. Dabei werden Websites zunächst für Smartphones entwickelt, bevor sie für größere Geräte wie Tablets und Desktops erweitert werden. Dieser Ansatz ist wichtiger geworden, insbesondere seit Google im Jahr 2019 den Mobile-First-Index eingeführt hat. Mobile-First-Index bedeutet, dass Google bei der Indexierung und Bewertung von Websites die mobile Version bevorzugt. Google passt sich der mobilen Suche an, um sicherzustellen, dass Nutzern keine Informationen angezeigt werden, die auf mobilen Geräten nicht verfügbar sind. Wenn eine Website keine mobile Version hat, verwendet Google zwar weiterhin die Desktop-Version für die Bewertung, stuft diese Informationen jedoch insgesamt niedriger ein (Horster, 2022, S. 129).

Mit dem Mobile First Ansatz gehen zahlreiche Vorteile einher. Einer der größten Vorteile ist die verbesserte Benutzererfahrung. Websites, die mit dem Mobile-First-Ansatz entwickelt werden, sind von Anfang an für kleine Bildschirme und Touch-Bedienung optimiert, was die Bedienbarkeit und Lesbarkeit auf Smartphones und Tablets erheblich verbessert. Zudem führt dieser Ansatz zu schnelleren Ladezeiten, da unnötige Inhalte und Funktionen vermieden werden. Ein weiterer Vorteil ist die positive Auswirkung auf das Suchmaschinenranking. Dies ermöglicht es Unternehmen, potenzielle Kunden zu erreichen.

Bei großen Vorteilen gibt es auch an Mobile First Kritik, da es schwierig sein kann, eine optimale Variante für sowohl Mobilgeräte als auch Desktops zu entwickeln (Höllings, 2022). Dies kann die Nutzerfreundlichkeit beeinträchtigen, indem Inhalte nicht vollständig angezeigt werden. Zudem erfordert Mobile First viel Wissen über mobile Anwendungen, was längere Entwicklungszeiten und zusätzliche Schulungen für Entwickler notwendig macht, was wiederum höhere Kosten verursachen kann. Bei der Entscheidung zwischen Mobile First und Desktop First sollten Unternehmen und Entwickler ihre Zielgruppe genau analysieren, um eine optimale Benutzererfahrung zu gewährleisten.

5 Konzeption einer Single-Page-Website mit scroll basierten Animationseffekt

5.1 Projektidee

Szenario: MeloChoco ist eine neue Schokoladenmarke, die frisch auf dem Markt eingeführt wurde. Um ihre Produkte optimal vorzustellen, plant das Unternehmen die Erstellung einer eigenen Website. Da es sich um ein Start-up handelt, verzichtet MeloChoco zunächst auf eine E-Commerce-Plattform und konzentriert sich stattdessen auf gezieltes Marketing. Daher stellt eine Single-Page-Website mit attraktiven Animationseffekten eine kreative und wirkungsvolle Möglichkeit dar, die Produkte stilvoll zu präsentieren.

Das Layout der Website wird mit Figma erstellt, einem Tool, das von Designern häufig zur Erstellung von Prototypen verwendet wird. Im Rahmen dieses Projekts wurde das Design zunächst für die Desktop-Ansicht erstellt. Im Entwicklungsprozess wird jedoch vorrangig die mobile Version umgesetzt. Der Grund für die ursprüngliche Gestaltung der Desktop-Version besteht darin, einen umfassenden Überblick über das gesamte Layout, die Elemente und die Animationen der Website zu erhalten. Basierend auf diesem Design werden dann die notwendigen Anpassungen vorgenommen, um eine optimale Darstellung auf kleineren Geräten zu gewährleisten.

5.2 Anforderungen

In der Planung und Umsetzung von IT-Projekten sind sowohl funktionale als auch nicht-funktionale Anforderungen von entscheidender Bedeutung, da sie die Grundlage für die erfolgreiche Gestaltung und Implementierung eines Projekts bilden.

Funktionale Anforderungen beschreiben die spezifischen Funktionen, die die Website bieten muss. Dazu gehört:

- Die Website wird als eine einzige Seite gestaltet, auf der alle Inhalte durch Scrollen angezeigt werden.
- Die Website hat verschiedene Animationen, die beim Scrollen der Benutzer durch verschiedene Abschnitte der Seite aktiviert werden.

Nicht-funktionale Anforderungen beziehen sich auf die Eigenschaften und Qualitätsmerkmale der Website. Nachfolgend sind diese aufgelistet:

- Die Website muss schnell laden
- Sie bietet eine flüssige Performance, selbst bei komplexen Animationen.
- Sie sollte mit allen gängigen Browsern und auf verschiedenen Endgeräten (Desktop, Tablet, Smartphone) kompatibel sein.
- Sie soll auf Englisch verfügbar sein, um eine breitere Publikum anzusprechen.
- Sie hat nur kurze, präzise Textabschnitte.

5.3 Struktur der Website

Die Struktur der Website orientiert sich an den spezifischen Inhalten der Seite und ist deshalb in vier Hauptabschnitte gegliedert:

1. **Header**: Das ist ein zentraler Bestandteil jeder Website und befindet sich am oberen Rand der Seite. Er ist das erste Element, das der Nutzer sieht, und beeinflusst den ersten Eindruck sowie die Wahrnehmung der Marke. In diesem Projekt ist der Header minimalistisch gestaltet mit einer Textzeile als Hauptelement, um den Namen der Marke darzustellen.
2. **Main**: Das ist der Hauptteil der Website und enthält die wichtigsten Informationen. Der Inhalt der Main konzentriert sich auf die Marke- und Produktvorstellung und ist in vier kleinere Abschnitte untergliedert:

   Thema 1 - *"Intro"*: Dieser Abschnitt dient als Willkommenheißung und bietet eine kurze Vorstellung der Marke in Form einer Tabelle.

   Thema 2 - *"The Classic"*: In diesem Abschnitt werden zwei bestimmte Schokoladensorten der Marke vorgestellt.

   Thema 3 - *"Full Collection"*: Kunden erhalten einen Überblick über die gesamte Produktkollektion der Marke. Neben den bereits vorgestellten zwei Schokoladensorten werden hier auch drei neue Sorten angedeutet.

Thema 4 - *"The Unique"*: In diesem Abschnitt werden drei neue Schokoladensorten der Marke präsentiert.

3. **Footer**: Das ist der abschließende Bereich der Website und befindet sich am unteren Rand der Seite. Zu den typischen Inhalten eines Footers gehören Kontaktinformationen, rechtliche Hinweise wie Impressum und Datenschutzbestimmungen sowie Links zu weiteren Seiten oder Ressourcen. Der Footer ist in diesem Projekt minimalistisch nur mit Texte gestaltet und wird hauptsächlich mit dem Ziel erstellt, das Ende der Seite zu markieren und dem Nutzer zu signalisieren, dass keine weiteren Inhalte mehr folgen.

5.4 Gestaltung der Benutzeroberfläche und Animationen

Header:

Im Header-Bereich stehen eine Textzeile und zwei horizontal abgerundete Balkens (Siehe Abbildung 5.1). Diese Elemente nehmen eine Breite von 90 % der Viewport-Breite ein und sind horizontal zentriert auf der Website platziert. Wichtig ist, dass die Schriftgröße so eingestellt werden soll, dass die Texte immer in einer Zeile bleiben und beim Verkleinern oder Vergrößern des Browserfensters nicht umbrechen. Auch bei einer Anpassung der Bildschirmgröße soll die Textzeile ihre Breite von 90 % der Viewport-Breite beibehalten.

Abschnitt „Intro“:

In diesem Bereich befindet sich eine abgerundete Tabelle mit einer Breite von 90 % der Viewport-Breite. Diese Tabelle hat drei Spalten, deren Breite im Verhältnis zur Gesamtbreite der Tabelle wie folgt verteilt ist: 50 %, 30 % und 20 % (Siehe Abbildung 5.1).

- Die erste Spalte enthält drei Kindelemente: einen kurzen Absatz, ein Bild und einen Text. Der Absatz ist linksbündig ausgerichtet, das Bild befindet sich zentriert in einem quadratischen Container, und der Text „Premium“ läuft kontinuierlich von rechts nach links.
- Die zweite Spalte besteht aus sechs vertikal angeordneten Textzeilen, die in der Mitte platziert sind und gleichmäßig voneinander getrennt sind.
- Die dritte Spalte enthält drei Kindelemente, von denen jedes ein Bild und einen Text umfasst, die beide zentriert angeordnet sind.

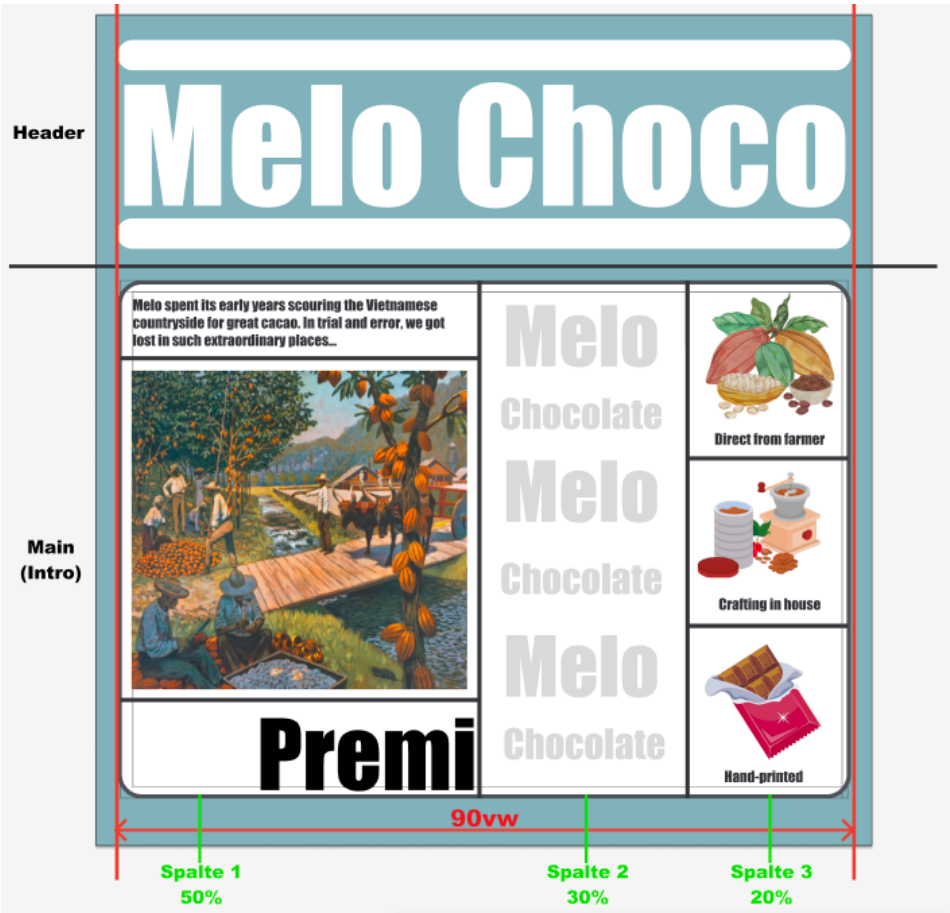

Abbildung 5.1: Header und „Intro“ Layout (Quelle: Eigene Darstellung)

Um die Aufmerksamkeit der Nutzer zu gewinnen, werden Animationseffekte in diesem Abschnitt erstellt und diese Effekte werden automatisch aktiviert, sobald der Nutzer die Website betritt. Zu Beginn wird nur der Header angezeigt, während die Tabelle im *“Intro”* ausgeblendet bleibt. Anschließend erscheinen die drei Spalten der Tabelle nacheinander von links nach rechts. Nachdem alle drei Spalten sichtbar sind, erscheinen der Absatz der ersten Spalte sowie die Texte und Bilder der dritten Spalte gleichzeitig. Während sich der Absatz und die Texte von links nach rechts bewegen, bewegt sich die Bilder der dritten Spalte von rechts nach links. Die Animation-Ablauf wird in Abbildung 5.2 dargestellt.

Abschnitt „The Classic“:

In diesem Bereich gibt es zwei Unterabschnitte, die jeweils die volle Viewport-Breite und -Höhe von 100 % einnehmen. In *“The Classic-1”* befindet sich ein vertikaler Balken, der leicht nach links versetzt ist, sowie eine Textzeile, die zentriert platziert ist. Sobald dieser Abschnitt beim Scrollen den gesamten Viewport ausfüllt, bleibt er fixiert und bewegt

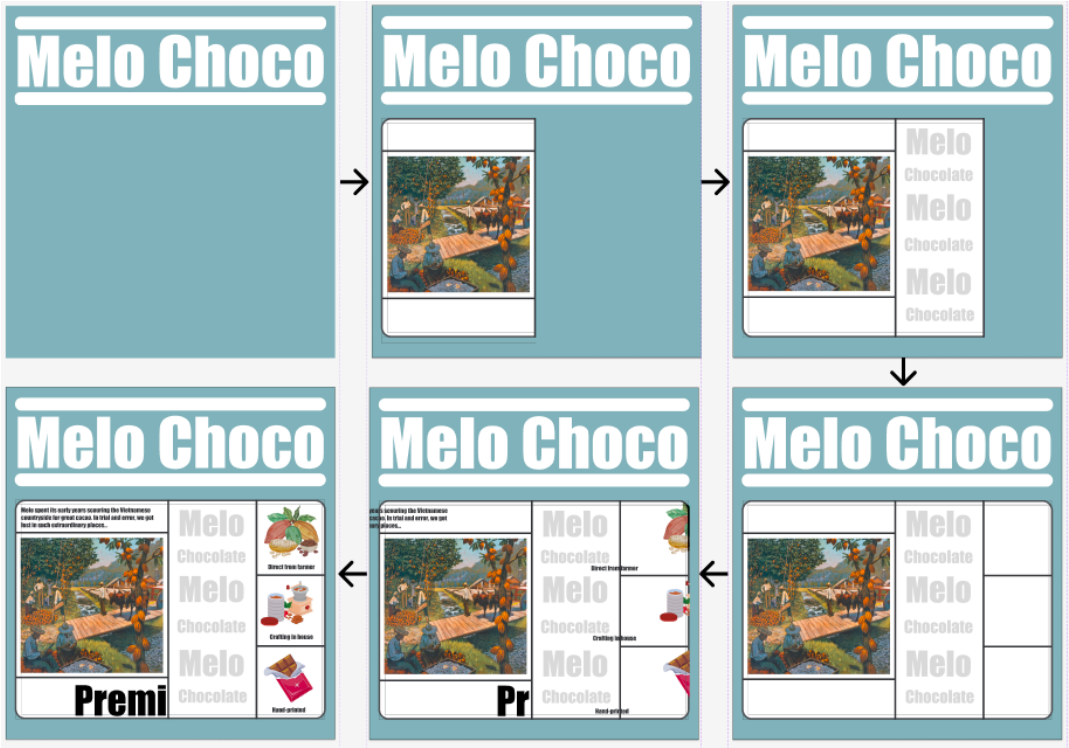

Abbildung 5.2: „Intro“ Animation-Ablauf (Quelle: Eigene Darstellung)

sich nicht weiter nach oben. Stattdessen wird der folgende Abschnitt *“The Classic-2”* nach oben gescrollt und überlagert *“The Classic-1”*. Außerdem wird sich die Breite des Balkens vergrößen. Dieser Prozess sowie das Layout der beiden Unterabschnitt wird in Abbildung 5.3 dargestellt.

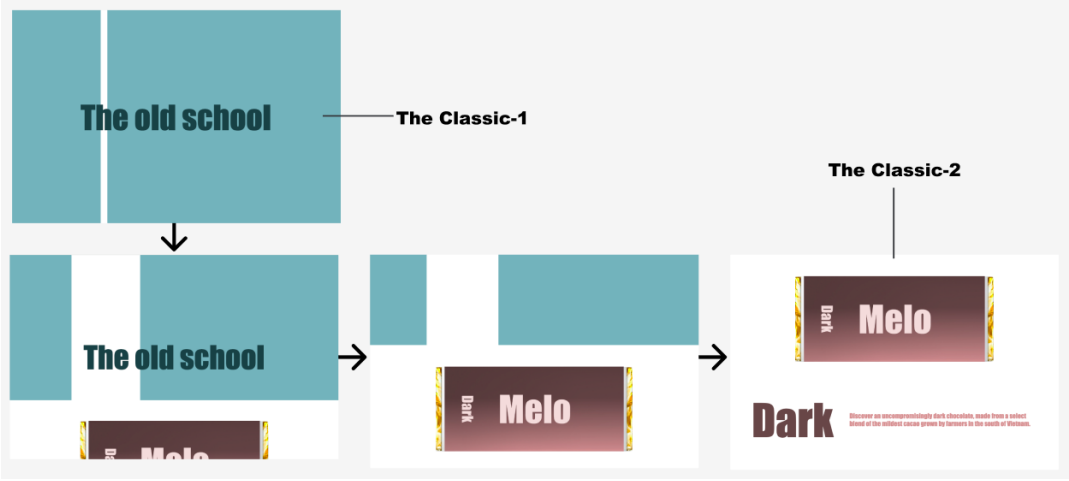

Abbildung 5.3: The Classic Animation-Ablauf (Quelle: Eigene Darstellung)

Im Abschnitt *„The Classic-2“* wird ein Bild einer Schokoladentafel mit brauner Verpackung zusammen mit einem Textblock angezeigt. Ähnlich wie *“The Classic-1”* wird der Abschnitt *“The Classic-2”* fixiert, sobald er beim Scrollen am oberen Browserrand ist. Während der Nutzer weiter scrollt, bewegt sich der Textblock nach links und schafft Platz für einen weiteren Textblock. Gleichzeitig verwandelt sich die Verpackung der Schokoladentafel in eine andere Verpackung (Siehe Abbildung 5.4).

Abbildung 5.4: The Classic Animation-Ablauf (Quelle: Eigene Darstellung)

Um diesen Effekt zu erzielen, sollte die Anordnung der Elemente sorgfältig berücksichtigt werden und wie folgt umgesetzt werden:

- Das Schokolade-Element besteht aus drei separaten Bildern: einem Schokoladentafel und zwei Verpackungen. Diese Bilder sollten in einem Container platziert werden, der genau auf die Größe des Schokoladentafel-Bilds abgestimmt ist. Die Verpackungsbilder werden über die Schokoladentafel gelegt, wobei die blaue Verpackung unter die braune platziert und zunächst verborgen wird (Siehe Abbildung 5.5).
- Die beiden Textblöcke befinden sich in einem Container mit einer Breite von 200 % der Viewport-Breite, wobei der zweite Textblock am Anfang außerhalb des Sichtfeldes verborgen bleibt (Siehe Abbildung 5.6).

Abschnitt „Full Collection“:

In diesem Abschnitt gibt es eine Textzeile und eine Liste von Bildern, die auf fünf horizontalen Linien gleichmäßig verteilt sind. Anfangs besteht die Liste aus zwei Schokoladen und drei Musiknoten. Diese Liste wird auf dem Bildschirm durch die Aktivierung einer Bounce-Animation angezeigt, bei der die Elemente wiederholt nach unten und oben springen (Siehe Abbildung 5.7). Anschließend verwandeln sich die drei Musiknoten nacheinander von links nach rechts in drei verschiedene Schokoladen. Dieser Prozess wird in Abbildung 5.8 dargestellt.

Nachdem alle Schokoladen eingeblendet wurden, taucht der Text „New“ unter den letzten drei auf (Siehe Abbildung 5.9). Schließlich bewegen sich alle Schokoladen unendlich auf und ab.

Abschnitt „The Unique“:

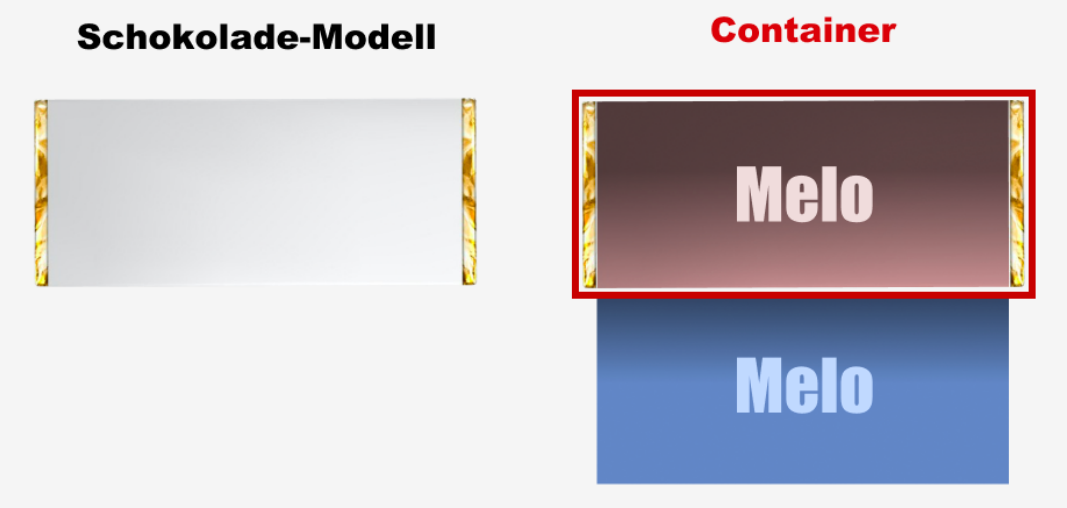

Abbildung 5.5: Schokolade-Modell (Quelle: Eigene Darstellung)

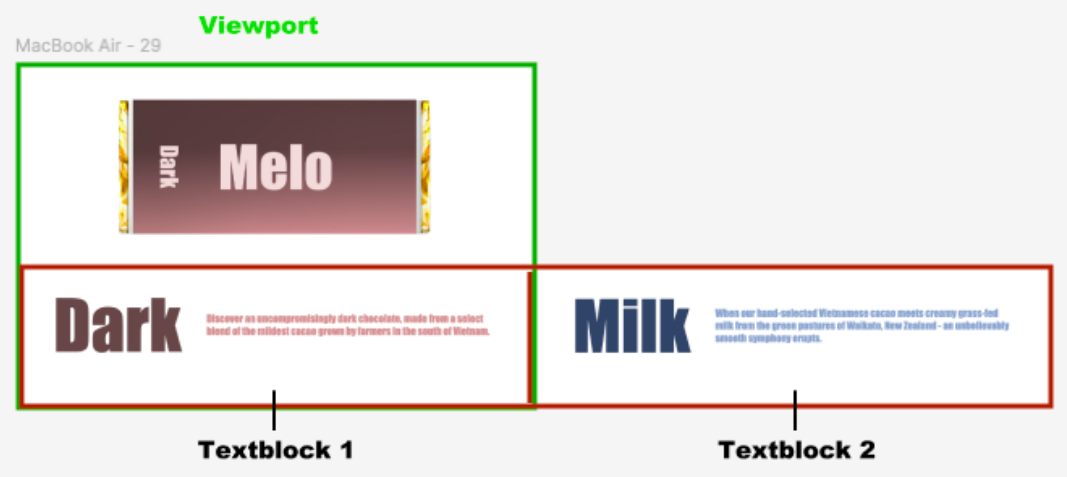

Abbildung 5.6: Textblöcke (Quelle: Eigene Darstellung)

Dieser Abschnitt hat eine Höhe von 100vh und eine Breite von 400vw. Er besteht aus vier Unterabschnitten, die jeweils eine Höhe und Breite von 100 % der Viewport-Größe einnehmen und horizontal angeordnet sind. Sobald der Abschnitt *„The Unique“* den oberen Rand des Browsers erreicht, wird von Usern von links nach rechts gescrollt (Siehe Abbildung 5.10).

Auf *“Page 1”* befinden sich ein Text und ein horizontaler Balken, beide werden vertikal zentriert. Der Text ist linksbündig ausgerichtet und auf der linken Seite platziert. Die Länge des Balkens wird sich vergrößern, sobald *Page 1* am oberen Rand des Browsers ist (Abbildung 5.11).

„Page 2“ enthält einen vertikalen Balken *(Line)*, einen großen Block *(BigBlock)* mit einer

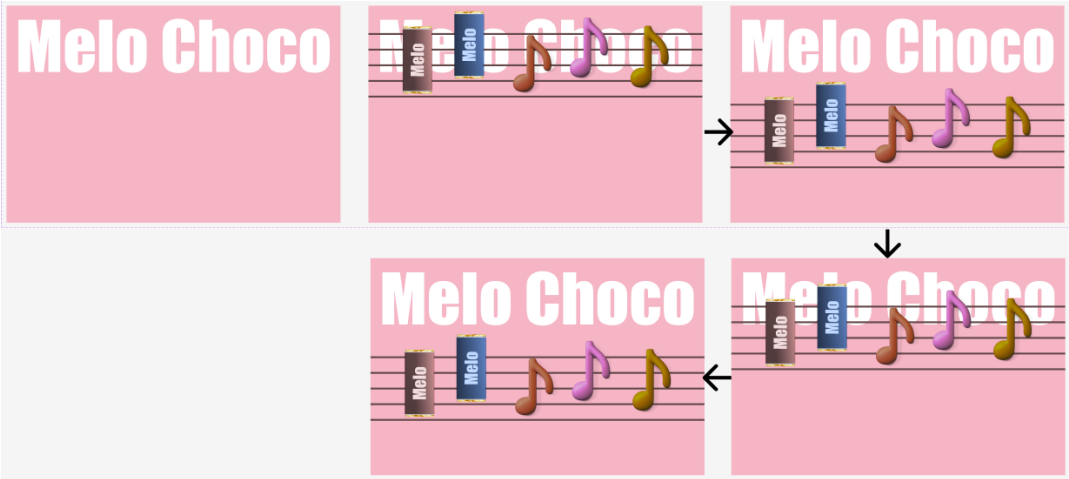

Abbildung 5.7: Bounce Animation (Quelle: Eigene Darstellung)

Abbildung 5.8: Choco List Animation (Quelle: Eigene Darstellung)

Breite von 50vw, eine Überschrift *(Title)*, ein großes Bild *(MainImg)*, einen kurzen Absatz *(Description)*, ein kleines Bild in der unteren linken Ecke des Bildschirms *(DecoImg)* und ein kreisförmiges Element in der unteren rechten Ecke *(RoundElement)*. Sobald *Page 2* in den Viewport gelangt, wird eine Reihe von Animationen für mehrere Elemente gleichzeitig ausgeführt: *BigBlock* bewegt sich von links nach rechts, bis er den rechten Rand der Seite erreicht, *Titel* und *Description* bewegen sich langsam von oben nach unten, *MainImg* wandert von unten nach oben, und *DecoImg* wird langsam um 60 Grad rotiert. Dieser Prozess endet, wenn *Page 2* den linken Rand des Viewports erreicht. Das Layout der Seite sowie der Animation-Prozess werden in Abbildung 5.12 dargestellt.

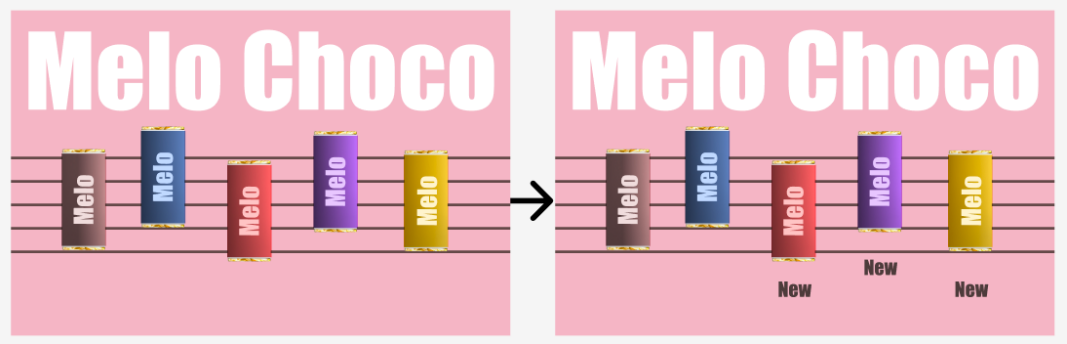

Abbildung 5.9: Choco List (Quelle: Eigene Darstellung)

Abbildung 5.10: The Unique Layout (Quelle: Eigene Darstellung)

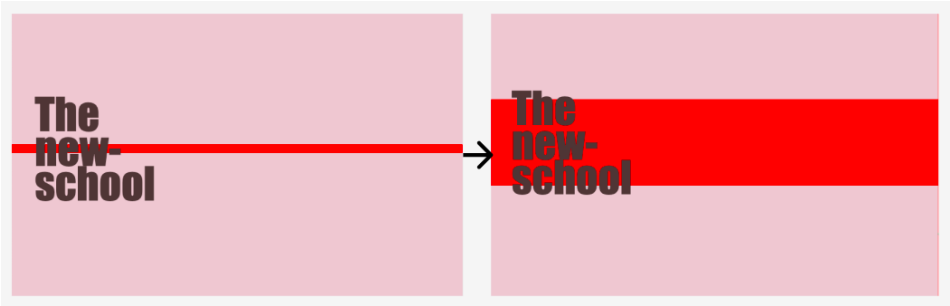

Abbildung 5.11: Page 1 Animation (Quelle: Eigene Darstellung)

„Page 3“ und *„Page 4“* haben ein ähnliches Layout und Animationseffekte wie *„Page 2“*. Der einzige Unterschied liegt im Animationseffekt des Elements “DecoImg”. Auf *Page 3* hat das *DecoImg* keinen Rotations-Effekt, sondern einen Fade-In-Effekt. Auf *Page 4* bewegt sich das *DecoImg* von der oberen linken Ecke des Bildschirms nach unten (Siehe Abbildung 5.13).

Footer:

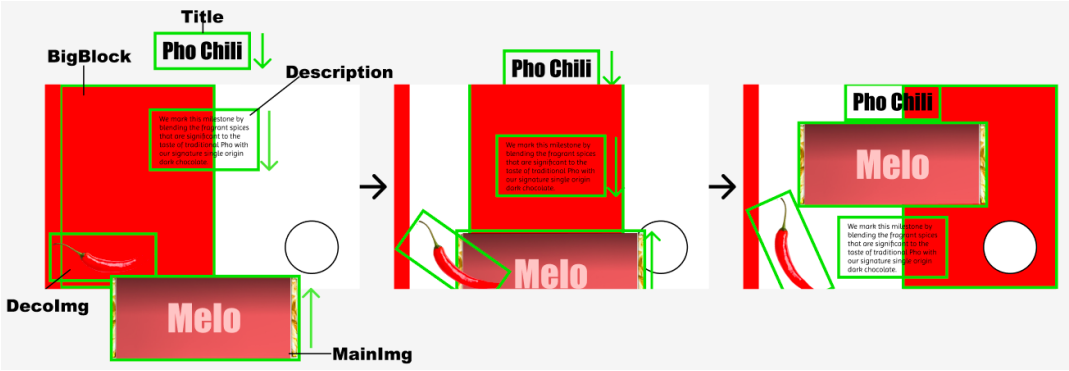

Abbildung 5.12: Page 2 Animation (Quelle: Eigene Darstellung)

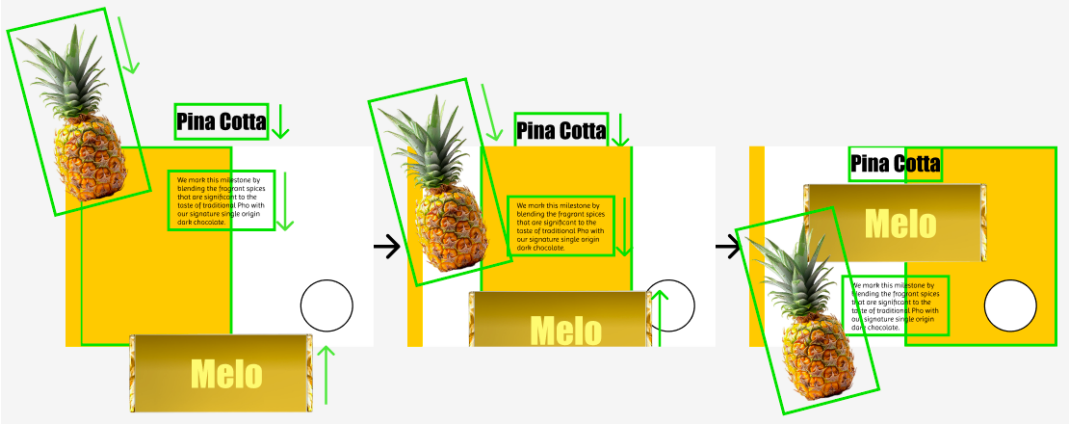

Abbildung 5.13: Page 4 Animation (Quelle: Eigene Darstellung)

Dieser Abschnitt ist minimalistisch gestaltet, nur mit drei Texten und keine Animationseffekt. Die drei Texte sind horizontal angeordnet, wie in Abbildung 6.5 dargestellt.

Abbildung 5.14: Footer (Quelle: Eigene Darstellung)

6 Entwicklung der Website

6.1 Entwicklungsumgebung

Im Rahmen dieses Projekt wird Visual Studio Code als IDE verwendet. Eine besonders nützliche Extension in Visual Studio Code ist Live Server. Diese Extension ermöglicht es, eine lokale Entwicklungsumgebung zu starten, die Änderungen am Code automatisch in Echtzeit im Browser aktualisiert.

Google Chrome wird als bevorzugter Browser eingesetzt. Mit Chrome DevTools kann die Darstellung der Webseite auf verschiedenen Bildschirmgrößen und Geräten simuliert werden. Es ist besonders nützlich für Entwickler zu testen, dass das Responsive Design auf allen Plattformen optimal funktioniert.

6.2 Technische Hintergrund

6.2.1 HTML5

HTML dient als Auszeichnungssprache, die den Inhalt der Website definiert und organisiert. Es legt dabei die semantische Struktur der Webseite fest, indem es verschiedene Tags verwendet, um unterschiedliche Inhaltsbereiche zu markieren.

Codeblock 6.1: Aufbau des HTML-Dockument

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
        initial-scale=1.0">
        <meta name="description" content="New chocolate brand
        from Vietnam">
        <title>Melochoco</title>
```

```
10      </head>
11      <body>
12          <!--INHALT DER WEBSITE-->
13      </body>
14  </html>
```

- <!DOCTYPE html> ist kein HTML-Tag, sondern eine Anweisung für den Webbrowser, dass das Dokument HTML5 nutzt.
- <html lang =„en“>-Tag ist das Wurzelelement und erhält die Information, dass die Sprache des Seiteninhalts Englisch ist.
- <head>-Tag ist das Kopftelement und enthält Metadaten, Dokumenttittel und verlinkte CSS-Datei.
- <meta charset=„utf-8“>: definiert die Zeichenkodierung des Dokuments. UTF-8 ist ein gängiges Format, das die Darstellung fast aller Schriftzeichen aus den meisten Sprachen der Welt ermöglicht. Es ist nicht zwingend erforderlich, aber empfohlen, dieses Tag zu verwenden.
- Viewport Meta Tag: stellt sicher, dass die Webseite auf mobilen Geräten korrekt angezeigt wird.
- Description Meta Tag: liefert eine kurze Beschreibung des Inhalts der Webseite. Suchmaschinen verwenden diese Beschreibung oft, um einen kurzen Textausschnitt anzuzeigen, wenn die Seite in den Suchergebnissen erscheint. Es wird empfohlen, eine prägnante und relevante Beschreibung anzugeben, weil sie das Ranking in Suchmaschinen verbessern kann.
- <body>-Tag enthält den gesamten sichtbaren Inhalt der Webseite wie Texte und Bilder. Alles, was innerhalb dieses Tags steht, wird auf der Webseite angezeigt.

6.2.2 CSS3

CSS ist die Gestaltungs- und Formatierungssprache, die für das visuelle Erscheinungsbild von HTML-Inhalten zuständig ist. Die Einbindung eines CSS-Codes ist durch drei Methoden möglich:

- Inline-Style: die Gestaltungsanweisungen werden mit style-Attribut direkt innerhalb des HTML-Tags angewendet. Diese Methode ist nützlich, wenn ein einzelnes Element schnell und spezifisch angepasst werden muss.

```
1 <h1 style="color:white">MeloChoco</h1>
```

- Interne CSS: die Gestaltungsanweisungen werden von dem <style>-Tag eingefasst und in den Header des HTML-Dokuments geschrieben.

```
<head>
    <style>
        h1 {color:white}
    </style>
</head>
<body>
    <h1>MeloChoco</h1>
</body>
```

- ausgelagert in externer CSS-Datei: die Gestaltungsanweisungen werden in einer separaten Datei mit der Endung *.css* geschrieben. Diese Datei wird dann mit dem HTML-Dokument verknüpft, indem das <link>-Tag im Header verwendet wird. Diese Methode ermöglicht eine saubere Trennung zwischen Inhalt und Design und erleichtert die Wartung.

```
<head>
    <link rel="stylesheet" href="main.css">
</head>
```

6.2.3 JavaScript

Für dieses Projekt wird JavaScript verwendet, um CSS-Animationen basierend auf den Scroll-Interaktionen der Benutzer zu aktivieren. Javascript kann als Code mit dem <script>-Element sowohl im <head> als auch im <body> des HTML-Dokuments integriert werden. Eine weitere Möglichkeit ist die externe Einbindung, bei der die Skripte in einer separaten Datei mit der Endung *.js* gespeichert werden und mit dem <script>-Tag in HTML-Dokument eingebunden werden. Diese Methode hat den Vorteil, dass JavaScript-Code von der Struktur des HTML-Dokuments getrennt wird und der Code auf mehreren Seiten verwendet werden kann.

```
<body>
    <!--INHALT DER WEBSITE-->
    <script src="script.js"></script>
</body>
```

Ein wichtiger Aspekt, der berücksichtigt werden sollte, ist die Position des Script Tags. Grundsätzlich kann das Script Tag entweder im Header oder ans Ende des body-Elementes

stehen. Es wird empfohlen, das Script Tag am Ende des Body zu platzieren, um sicherzustellen, dass das HTML-Dokument vollständig geladen ist, bevor die Skripte ausgeführt werden. Wenn das Script Tag im <head>-Bereich eingebunden wird, können Fehler auftreten, da der Browser den Dokumenteninhalt noch nicht fertig gestellt hat und die benötigten HTML-Elemente zu diesem Zeitpunkt möglicherweise noch nicht verfügbar sind. Eine Lösung dafür ist, das *DOMContentLoaded-Event* zu verwenden. Dieses Event stellt sicher, dass das Skript erst dann ausgeführt wird, wenn das gesamte HTML-Dokument geladen und geparst wurde, wodurch Fehler vermieden werden.

```
<head>
  <script>
    document.addEventListener("DOMContentLoaded", function () {
      //Code
    });
  </script>
</head>
```

6.2.4 GSAP

GSAP (GreenSock Animation Platform) ist eine Javascript-Bibliothek für (zeitleistenbasierte) Animationen - die in allen gängigen Browsern funktionieren (Steinberg, 2020, S. 4). GSAP wird auf über 8 Millionen Websites verwendet, darunter viele preisgekrönte Websites (Bei Awwwards sind heute über 3500 Websites mit GSAP-Animation gelistet). Es gibt viele bekannte Unternehmen, die GSAP in ihren Websites und Projekten einsetzen, wie z.B. Google, Intel, Coca Cola, Gucci etc. GSAP zeichnet sich insbesondere durch seine Geschwindigkeit (bis zu 20-mal schneller als jQuery) (Steinberg, 2020, S. 5) und Vielseitigkeit aus, da es in der Lage ist, nahezu jedes Element auf einer Website zu animieren.

GSAP bietet viele optionale Plugins zur Erstellung fortgeschrittener Effekte und ein hervorzuhebendes davon ist ScrollTrigger, das es ermöglicht, komplexe Scroll-Animationen mit minimalem Codeaufwand zu erstellen. Außerdem bietet es ein Debugging-Feature (*markers: true*), mit dem der Animationsbereich (Start- und End-Position) angezeigt wird und Entwickler das Animationsverhalten direkt im Browser kontrollieren können. Diese Funktionalitäten machen GSAP nicht nur zu einer mächtigen Bibliothek für erfahrene Entwickler, sondern auch zu einer zugänglichen Option für Anfänger. Aus diesem Grund wurde GSAP in dem vorliegenden Projekt verwendet.

GSAP sowie das zusätzliche Plugin ScrollTrigger können über das CDN importiert werden:

```
<script src="https://cdn.jsdelivr.net/npm/gsap@3.12.5/dist/gsap.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/gsap@3.12.5/dist/ScrollTrigger.min.js"></script>
```

Abbildung 6.1: Einbindung von GSAP über CDN

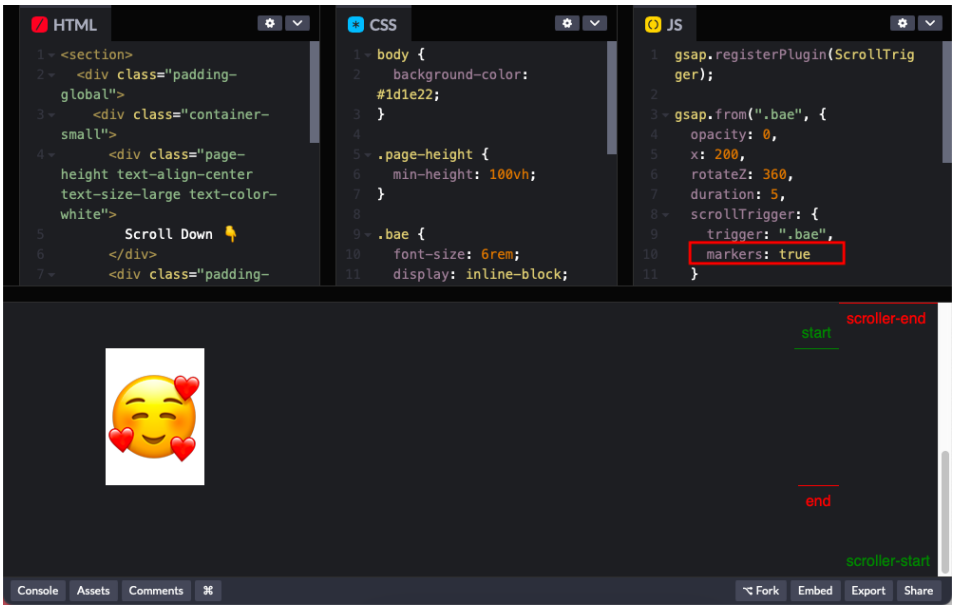

Abbildung 6.2: Beispiel GSAP markers (Quelle: CODEPEN
)

6.3 Layout-Umsetzung

6.3.1 Media Queries für responsive Website

Im Rahmen dieses Projektes wird der Mobile-First-Ansatz verwendet. Media Queries in CSS werden benutzt, um Stylesheets für verschiedene Ausgabemedien anzupassen. Es wurden zwei Media Queries angelegt: eine für Tablets und eine für Laptops. Der Code, der sich außerhalb der Media Queries befindet, ist für Handy-Layout.

Die Media Query *@media only screen and (min-width: 768px) and (max-width: 992px) and (orientation: portrait)* wird verwendet, um CSS-Regeln für Tablets anzuwenden. Sie greift nur, wenn die Breite des Geräts zwischen 768px und 992px liegt, und wenn sich das Gerät im Hochformat (portrait) befindet.

Die Media Query *@media only screen and (orientation: landscape)* wurde für Laptops angewendet. Da dies für Geräte gilt, deren Bildschirmbreite größer ist als ihre Höhe, also für das Querformat.

6.3.2 Mobile-First

Header:

Der Text im Header repräsentiert den Markennamen und fungiert als die wichtigste Überschrift der Webseite, weshalb er als *h1* genannt wurde. Um sicherzustellen, dass diese Textzeile eine feste Breite relativ zur Viewport-Breite beibehält, wurde 'vw' als Einheit für die Schriftgröße gewählt. 'vw' passt die Schriftgröße dynamisch an die Breite des Viewports an, was besonders bei responsiven Designs vorteilhaft ist. Im Gegensatz zu 'rem' und 'em', die auf der Schriftgröße des Root-Elements (html) oder Parent-Elements basieren, ist 'vw' unabhängig von der hierarchischen Struktur des HTML-Dokuments und reagiert direkt auf Änderungen der Fensterbreite. 'px' ist ebenfalls eine häufig verwendete Einheit im Webdentwicklung. Allerdings handelt es sich dabei um eine absolute Einheit, bei der die Schriftgröße immer fest bleibt und sich nicht ändert, wenn sich die Bildschirmgröße verändert. Damit die Textzeile 90 % der Viewport-Breite entspricht, wurde eine Schriftgröße von 19vw festgelegt.

Die zwei horizontale Balken wurden mit einer Höhe von 20px und einer Breite von 90vw festgelegt. Durch *'border-radius'* werden die Ecken der Balken mit einem Radius von 20px abgerundet.

Abschnitt *„Intro“*:

Die Tabelle innerhalb des Abschnitts *"Intro"* hat im Desktop-Design drei Spalten, die den drei Bereichen im HTML-Code entsprechen: * *, * * und * *. In der mobilen Ansicht wird diese Tabelle jedoch auf eine Spalte reduziert, wobei *'.firstCol'*, *'.secondCol'* und *'.thirdCol'* nacheinander von oben nach unten angeordnet werden.

'.firstCol' enthält drei Container. Im ersten Container befindet sich ein Absatz, der mit *'text-align: left'* linksbündig ausgerichtet ist. Der zweite Container wird durch Setzen von Höhe und Breite auf jeweils 90vw zu einem Quadrat geformt. Innerhalb dieses Containers befindet sich ein Bild, das auf 95 % der Höhe und Breite des Containers skaliert ist. Durch die Verwendung von *'display: flex'*, *'justify-content: center'* und *'align-items: center'* in diesem Container wird das Bild dabei horizontal und vertikal zentriert. Der dritte Container hat eine Höhe von 3vh und enthält einen Text mit der Aufschrift *„Premium“*, deren eine Schriftgröße mit 3em.

'.secondCol' besteht aus drei Containern, die jeweils eine Breite von 30vw hat und zwei Textzeilen enthält. Im Gegensatz zu *'.firstCol'* sind die Container in *'.secondCol'* horizontal angeordnet. Durch die Verwendung von *'display: flex'* gilt *'.secondCol'* als ein Flexbox-Container mit dem Standardwert *„row“* für *'flex-direction'*. Dies sorgt dafür, dass die direkten Kind-Elemente in einer Reihe angeordnet werden.

'.thirdCol' hat auch drei Container, die von oben nach unten angeordnet sind. Jeder Container hat eine Höhe von 40vh und enthält ein Bild sowie einen Text. Die Bilder sind auf eine Höhe und Breite von 40vw eingestellt.

Ergebnis: Siehe Abbildung 6.3

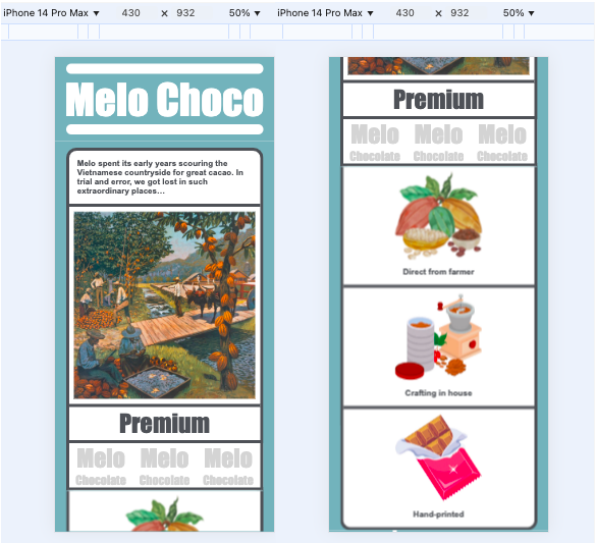

Abbildung 6.3: Header und „Intro“ Layout für Mobile (Quelle: Eigene Darstellung)

Abschnitt *"The Classic“*:

"The Classic" besteht aus zwei Unterabschnitte: *„The Classic-1“* und *„The Classic-2“*. In *"The Classic-1"* gibt es einen Text, der über einer vertikalen Linie liegt. Diese Linie hat eine Höhe von 100vh und eine Breite von 2vw. Durch *'transform: translateX(-20vw)'* wird sie um 20 % der Breite des Viewports nach links verschoben. Der Text überlappt die Linie, weil ihm die Eigenschaft *'position: absolute'* zugewiesen wurde. Durch eine Absolute Positionierung wird ein Element aus dem normalen Layoutfluss herausgenommen und relativ zu dem nächstgelegenen Vorfahren positioniert, der eine Positionierung (relative, absolute, fixed oder sticky) aufweist. In diesem Fall hat der Text kein solches Vorfahren, er wird unabhängig von anderen Elementen und relativ zum Viewport positioniert.

„The Classic-2“ besteht aus zwei Hauptbestandteile *„Schokolade“* und *„Textblock“*, die den zwei Bereichen im HTML-Code entsprechen: * *, * *. In der Mitte des *'.mainContent'* befindet sich ein Schokolade-Objekt, das als *'.wrapper'* in HTML-Code genannt wird. Wie bereit erwähnt, das Schokolade-Objekt besteht aus 3 Bildern: ein Schoko-Modell und zwei Verpackungen. Wie in Codeblock 6.2 dargestellt, werden die zwei Verpackungen in dem Container *'.photos'* platziert. Da das *„Label2“* in der Reihenfolge nach *„Label1“* im HTML-Code angeordnet ist, überlagert es das *„Label1“* durch absolute Positionierung. *„Label2“*

wird um 100 % seiner eigenen Höhe nach unten verschoben durch *'transform: translateY(100 %)'*, wodurch es sich unterhalb des Elements *'Label1'* positioniert. Um das *„Label2"* außerhalb des Sichtbereichs zu verstecken, wird *'overflow: hidden'* auf den Container *'.photos'* angewendet.

Codeblock 6.2: HTML Struktur des Objekts SchokoModell

```
<div class="wrapper">
    <img src="/img/classic/chocobar.png" alt="Chocobar"/>
    <div class="photos">
        <img class="photo dark" src="/img/classic/label1.png"
        alt="Label1"/>
        <img class="photo milk" src="/img/classic/label2.png"
        alt="Label2"/>
    </div>
</div>
```

Während der Container *'.photos'* absolut positioniert ist, besitzt der Container *'.wrapper'* eine relative Positionierung. Die Position des Containers *'.photo'* wird relativ zum *'.wrapper'* bestimmt, was sicherstellt, dass die Verpackungen stets über Schokomodell liegen und sich mit ihm verschieben, wenn Schokomodell seine Position ändert. Um die Größe der Verpackungen an Schokomodell anzupassen, wurde der Container *'.photos'* mit einer Höhe von 99 % und einer Breite von 95 % im Verhältnis zur Größe des Container *'.wrapper'* festgelegt.

Unter dem Schoko-Modell befinden sich zwei Textblöcke, von denen jeder eine Breite von 100vw hat und jeweils eine Überschrift sowie einen Absatz enthält. Diese beiden Textblöcke sind in dem Flex-Container *'.bottomContent'* deren Breite von 200vw platziert. Um sicherzustellen, dass der zweite Textblock vom Bildschirm verborgen wird, muss die Eigenschaft *'overflow-x: hidden'* im *body* gesetzt werden. Im Desktop-Design sind Überschrift und Absatz nebeneinander angeordnet. Für die mobile Ansicht wird die Überschrift hingegen oberhalb des Absatzes angezeigt. Der Abstand zwischen dem Inhalt und den beiden Seiten des Bildschirms wurde durch *'padding-left'* und *'padding-right'* von jeweils 10vw festgelegt.

Ergebnis: Siehe Abbildung 6.4

Abschnitt „Full Collection":

Am Anfang dieses Abschnittes befindet sich eine Textzeile mit dem Namen der Marke, die dieselben Eigenschaften wie die *h1*-Überschrift im Header aufweist. Unter dieser Textzeile befindet sich ein Container mit einer Höhe von 40vh. Dieser Container enthält fünf Linien und eine Liste von Objekt Schokolade. Die fünf Linien werden durch *<hr>*-Elemente

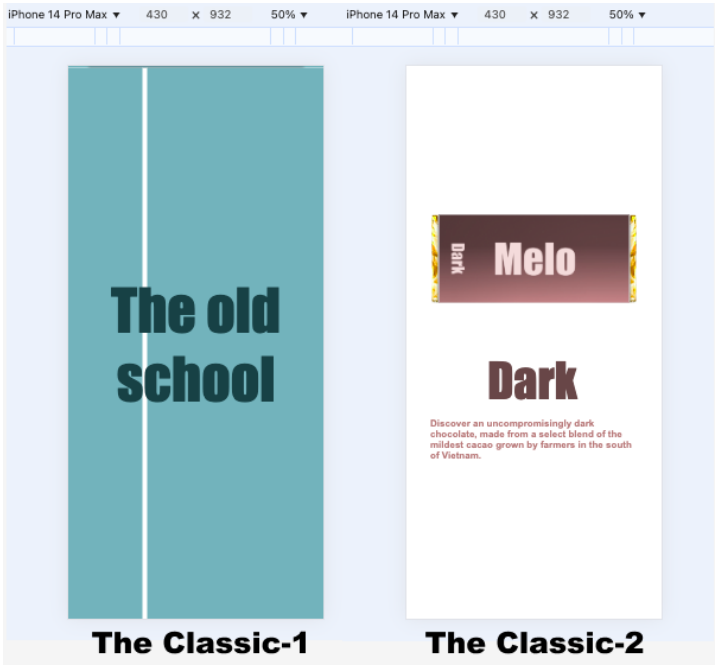

Abbildung 6.4: The Classic Layout für Mobile (Quelle: Eigene Darstellung)

mit einer Breite von 100vw und einer Höhe von 3px erstellt. Die fünf Schokolade-Objekte werden in HTML-Code als *'.choco-container'* bezeichnet und sind in dem Flex-Container *'.chocoList'* platziert. Durch *'justify-content: space-around'* ist der Abstand zwischen den Objekten sowie zu den Rändern des Containers gleichmäßig verteilt ist.

Es gibt insgesamt fünf *'.choco-container'*. Die ersten zwei enthalten nur ein Schokolade-Bild. Die verbleibenden drei Container bestehen aus einer Kombination von einem Schokolade-Bild, einem Musiknote-Bild und einem Text. Zur Vorbereitung auf die Animation-Prozess werden Musiknote und Schokolade in einem weiteren Container *'.img-wrapper'* platziert. Dieser Container hat die Eigenschaft *'position: relative'*, während die Musiknote absolute positioniert wird. Das Schokolade-Bild und der Text werden durch die Eigenschaft *'opacity: 0'* verborgen.

Die Position der fünf *'.choco-container'* wird mithilfe der Eigenschaft *align-self* festgelegt. Das erste und letzte Container werden durch *'align-self: center'* vertikal zentriert. Das zweite und vierte werden mit *'align-self: flex-start'* positioniert, sodass sie am oberen Rand des Eltern-Containers ausgerichtet sind. Das dritte wird mit *'align-self: flex-end'* versehen, wodurch es am unteren Rand des Eltern-Container ist.

Abschnitt „The Unique“:

Dieser Abschnitt umfasst vier horizontal angeordnete Unterabschnitte, die jeweils 100vh hoch und 100vw breit sind. Daher wurde *"The Unique"* als *'display: flex'* definiert, und

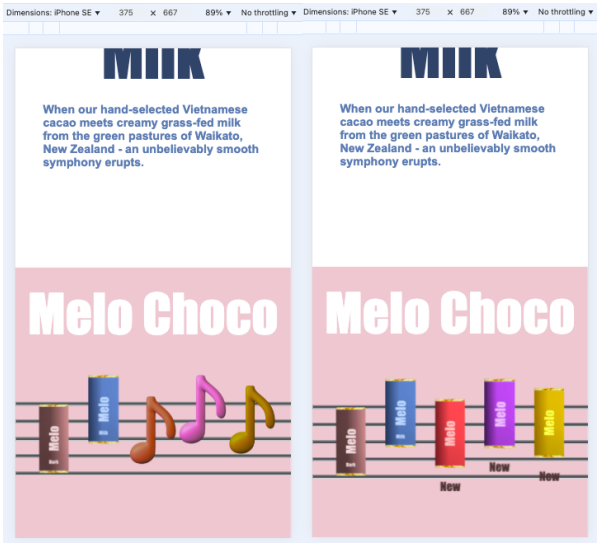

Abbildung 6.5: „Full Collection“ Mobile Layout (Quelle: Eigene Darstellung)

seine Breite beträgt 400 % der Viewport-Breite.

“Page 1” wurde mit *flex* und der Eigenschaft *‘align-items: center’* versehen, um den Balken und den Text vertikal zu zentrieren. Der Balken wurde auf eine Breite von 100vw und eine Höhe von 1 rem festgelegt. Der Text ist mit *‘position: absolute’* positioniert, damit er über dem Balken liegt. Wie im Design dargestellt, wird der Text nicht auf einer einzigen Zeile, sondern auf drei Zeilen angezeigt. Dies wird durch das Einfügen von $
$ -Element in HTML erreicht, das den Text an den gewünschten Stellen umbricht. Der Abstand zwischen den Zeilen wurde mit *‘line-height: 0.8’* verkleinert. Der Text ist linksbündig mit *‘text-align: left’* ausgerichtet und befindet sich 5 % vom linken Rand entfernt, durch die Verwendung von *‘left: 5vw’*.

Die Layouts von *‘Page 2’*, *‘Page 3’* und *‘Page 4’* sind identisch. Wie im Design dargestellt, besteht jede Seite aus sieben Bestandteilen: *Line*, *BigBlock*, *Title*, *MainImg*, *Description*, *DecoImg* und *RoundElement*. *‘Line’* und *‘BigBlock’* sind dabei absolut positioniert, sodass sie aus dem normalen Layoutfluss herausgenommen werden und sich relativ zu dem Viewport positionieren. *‘Line’* erstreckt sich über die gesamte Höhe der Seite mit einer Länge von 100vh und einer Breite von 5vw. Sie wird mit *‘left: 0’* direkt an den linken Rand der Seite gesetzt. *‘BigBlock’* ist ebenfalls 100vh hoch und nimmt 50vw der Seitenbreite ein, wobei er mit *‘right: 0’* an den rechten Rand der Seite verschoben wird.

Die übrigen Bestandteile sind in jeweils eigenen Containern untergebracht, und ihre Positionen werden mithilfe der CSS-Eigenschaft *‘grid-template-areas’* festgelegt. Diese Eigenschaft ermöglicht eine präzise Definition der Platzierung von Elementen innerhalb

eines CSS-Grids, indem für jedes Element eine spezifische Position im Raster mit *'grid-area'* definiert wird.

Der CSS-Code für die Klasse *'.choco'* in Codeblock 6.3 definiert ein Grid-Layout für *Page 2*, *Page 3* und *Page 4*. Für die Mobile-Ansicht ist das Layout der Seite in fünf Zeilen und zwei Spalten unterteilt. Die erste Zeile enthält keine Inhalte, weil der Punkt *'.'* für einen leeren Bereich steht).

Codeblock 6.3: CSS Grid Template Areas

```
.choco{
    display: grid;
    grid-template-columns: 50% 50%;
    grid-template-rows: auto 10% auto auto 30%;
    grid-template-areas:
        ".              ."
        "title          title"
        "bigImg         bigImg"
        "description    description"
        "decoImg        container-round";
}
```

Nachdem die Bereiche definiert wurden, sollten die einzelnen Grid-Items diesen Bereichen zugeordnet werden. In dem Beispiel in Codeblock 6.4 wird das Grid-Item mit der Klasse *'.title'* in den *„title“*-Bereich des Rasters gesetzt.

Codeblock 6.4: Zuweisen der Bereiche zu Grid-Items

```
.title {
    grid-area: title;
}
```

Ergebnis:

Footer:

Im Unterschied zum Desktop-Design sind im mobilen Design die drei Texte im Footer auf zwei Zeilen verteilt. Ganz oben befindet sich ein Container, der den Text *„Melo“* enthält. Dieser Text hat eine Schriftgröße von 4 rem und ist durch *'text-align: center'* zentriert ausgerichtet. Die beiden anderen Texte sind in einem Flex-Container nebeneinander angeordnet. Sie sind durch *'justify-content: center'* horizontal zentriert. Der Abstand zwischen den beiden Texten wird durch die Eigenschaft *'gap'* auf 10vw festgelegt.

Ergebnis:

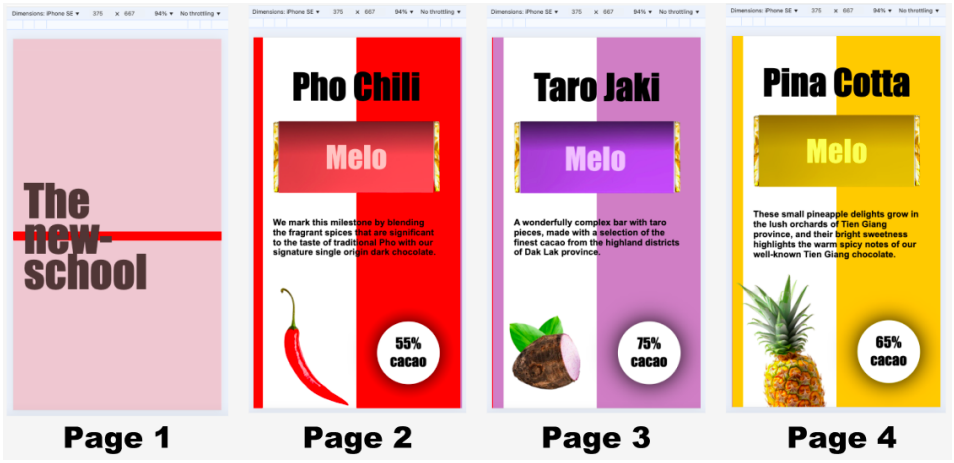

Abbildung 6.6: *„The Unique“* Mobile-Layout (Quelle: Eigene Darstellung)

Abbildung 6.7: Footer Mobile-Layout (Quelle: Eigene Darstellung)

6.3.3 Tablet-Layout

Die Benutzeroberfläche der Website für Tablets unterscheidet sich nur geringfügig von der mobilen Version. Die Hauptunterschiede liegen im Abschnitt *“Intro”* und im Footer.

Abschnitt *“Intro”*

Während die Tabelle in der mobilen Ansicht nur eine einzelne Spalte hat, wird das Layout der Tabelle in der Tablet-Ansicht mithilfe von *‘display: grid’* und *‘grid-template-columns: 60vw 30vw’* neu organisiert, sodass zwei Spalten entstehen. Dabei nimmt *‘.firstCol’* die erste Spalte mit einer Breite von 60vw ein *(grid-column: 1 / 2)*, *‘.secondCol’* die zweite Spalte mit 30vw *(grid-column: 2 / 3)*. *‘.thirdCol’* erstreckt sich über die gesamte Breite und liegt unter den beiden anderen Spalten *(grid-column: 1 / 3)*.

Durch Hinzufügen der Eigenschaft *‘flex-direction: column’* zu *‘.secondCol’* werden deren Inhalte vertikal ausgerichtet. Die drei Container in *‘.thirdCol’* werden mithilfe von Flexbox horizontal angeordnet, und ihre Breite wird auf 30vw reduziert.

Footer:

Abbildung 6.8: Abschnitt „Intro“ Tablet-Layout (Quelle: Eigene Darstellung)

Im Tablet-Design ist die Höhe des Footers auf 15vh festgelegt. Die zwei Textblöcke werden durch die Verwendung von *‘display: flex’* in einer einzigen Reihe angeordnet. Sie sind vertikal zentriert und gleichmäßig verteilt, mittels der Eigenschaften *‘align-items: center’* und *‘justify-content: space-evenly’*.

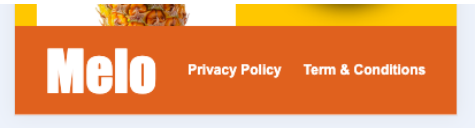

Abbildung 6.9: Footer Tablet-Layout (Quelle: Eigene Darstellung)

6.3.4 Desktop-Layout:

Die Unterschiede zwischen der Desktop-Version und den beiden anderen Versionen liegen in den Bereichen *„Intro“*, *„The Classic 2“*, *„The Unique“* (*Page 2*, *Page 3*, *Page 4*) und Footer.

Abschnitt „Intro“:

Die Tabelle wird durch die Verwendung von Flexbox in drei horizontale Spalten umgewandelt. Mit *‘display:flex’* werden diese drei Spalten immer gleich hoch und passen sich besonders gut an ihren umfassenden Container an. Die Breite der Spalten *‘.firstCol’*,

'.secondCol' und *'.thirdCol'* werden dabei mit der Eigenschaft *'width'* von jeweils 50 %, 30 % und 20 % festgelegt.

Im mobilen Layout wird die Höhe des dritten Containers innerhalb von *'.firstCol'*, der den Text „Premium“ enthält, auf 10vh festgelegt. Im Desktop-Version wird die Größe dieses Containers automatisch an den verbleibenden freien Raum seines Parent-Containers - *'.firstCol'* angepasst. Durch die Verwendung von Flexbox wird *'.firstCol'* zu einem Flex-Container, und der dritte Container gilt als ein Flex-Element. Wenn ein Flex-Element die Eigenschaft *'flex-grow: 1'* hat, bedeutet das, dass es sich proportional zum verfügbaren Platz im Flex-Container ausdehnen kann.

Abbildung 6.10: Abschnitt „Intro“ Desktop-Layout (Quelle: Eigene Darstellung)

Abschnitt „The Classic 2“:

Die beiden Texte werden nicht mehr vertikal, sondern horizontal angeordnet, indem Flexbox auf ihre Parent-Container angewendet wird. Dadurch befinden sich die Texte in einer Reihe und haben einen Abstand von 5vw zueinander, der durch die Eigenschaft *'gap'* festgelegt wird.

Abschnitt „The Unique“:

Das Layout von *'Page 2'*, *'Page 3'*, *'Page 4'* wurde in drei Reihen und drei Spalten umgewandelt, wobei die Größenverhältnisse der Bereiche wie in Codeblock 6.5 beschrieben aufgeteilt sind. Mit diesem Layout befinden sich *'DecoImg'* und *'RoundElement'* nicht mehr unter *'Description'*, sondern zu beiden Seiten davon.

Codeblock 6.5: CSS Grid Template Areas Desktop

```
1 .choco{
2     grid-template-columns: 30% 40% 30%;
3     grid-template-rows: 20% 40% 40%;
4     grid-template-areas:
```

```
        "title     title         title"
        "bigImg    bigImg        bigImg"
        "decoImg   description   container-round"
}
```

Footer:

Der Footer hat eine Höhe von 20vh. Die beiden Textblöcke sind, ähnlich wie in der Tablet-Version, ebenfalls nebeneinander auf einer horizontalen Linie angeordnet, was durch die Verwendung von Flexbox ermöglicht wird. Allerdings wird ihre Position durch die Eigenschaft *'justify-content: space-between'* festgelegt, wodurch die Textblöcke an den gegenüberliegenden Enden des Containers ausgerichtet werden. Durch *'padding-left'* und *'padding-right'* im Footer kleben diese Textblöcke nicht direkt am Rand, sondern einen Abstand von 5vw hat.

6.4 Erstellen von Animationen

Abschnitt „Intro“:

In diesem Abschnitt wird die CSS-Animation verwendet. Jede Animation umfasst zwei wesentliche Schritte:

- Definition der Animation mit der *@keyframes-Regel*: Hier wird festgelegt, wie sich die Animation über die Zeit entwickeln soll. Die @keyframes-Regel definiert die verschiedenen Phasen der Animation, indem sie angibt, welche CSS-Eigenschaften sich zu bestimmten Zeitpunkten ändern.
- Verwendung der *@keyframes-Regel* im CSS-Selektor des HTML-Elements: Nachdem die Animation definiert wurde, wird sie dem Ziel-HTML-Element zugewiesen. Dies erfolgt durch die Angabe der Eigenschaft *'animation'* im CSS-Selektor des betreffenden Elements, wobei der Name der @keyframes-Regel, die Dauer und andere Eigenschaften der Animation festgelegt werden.

Zum Beispiel, die Animation des Element mit der Klasse *'.firstCol'* wird durch @keyframes-Regel mit dem Namen *'appear'* definiert. Die Definition der *'appear-Animation'* sieht wie folgt aus:

```
@keyframes appear {
    0% { opacity: 0;}
    100% { opacity: 1; }
}
```

In dieser Definition wird die Anfangs- und Endphase der Animation so beschrieben: Zu Beginn, bei 0 %, hat das Element eine *opacity* von 0, was bedeutet, dass es komplett ausgeblendet ist. Am Ende der Animation, bei 100 %, wird die *opacity* auf 1 erhöht, wodurch das Element vollständig sichtbar ist.

Nachdem die Animation definiert wurde, wird sie dem Element mit der Klasse *'firstCol'* zugewiesen. Die Zuweisung erfolgt wie folgt:

```
.firstCol {
    animation: appear 2s 1s forwards;
}
```

Dies bedeutet, dass die Animation mit dem Namen *'appear'* ausgeführt wird. Bei zwei Zeitangaben ist die erste Zeit (2s) die Dauer der Animation, die zweite (1s) die Verzögerung. Durch das Schlüsselwort *forwards* bleibt der Endzustand der Animation erhalten, nachdem sie abgeschlossen ist, anstatt zum ursprünglichen Zustand zurückzukehren.

Die Elemente *'.firstCol'*, *'.secondCol'* und *'.thirdCol'* teilen sich dieselbe Animation und haben die gleiche Dauer. Der einzige Unterschied besteht darin, dass sie nicht gleichzeitig, sondern nacheinander in einem Abstand von 0,3 Sekunden erscheinen. Deshalb haben sie die folgende Zuweisung:

```
.secondCol {
    animation: appear 2s 1.3s forwards;
}
.thirdCol {
    animation: appear 2s 1.6s forwards;
}
```

Die Texte in dem ersten Container von *'.firstCol'* und die texte in *'.thirdCol'* bezitzen die *fadeInLeft-Animation*, die ein Objekt von links nach recht einblenden lässt. Zu Beginn der Animation sind die Texte noch nicht sichtbar und befinden sich um 50 % nach links verschoben (transform: translateX(-50 %)). Während die Animation fortschreitet, wird die opacity auf 1 erhöht und die Texte bewegen sich zurück in ihre Ausgangsposition (transform: translateX(0)).

```
@keyframes fadeInLeft {
    0% { opacity: 0; transform: translateX(-50%); }
    100% { opacity: 1; transform: translateX(0); }
}
.paragraph p, .thirdCol p {
    animation: fadeInLeft 2s 1.9s forwards;
}
```

Gleichzeitig bewegen sich die drei Bilder in '.thirdCol' in die entgegengesetzte Richtung, also von rechts nach links. Diese Animation wird durch das folgende @keyframes definiert:

```
@keyframes fadeInRight {
    0% { opacity: 0;transform: translateX(50%); }
    100% { opacity: 1;transform: translateX(0); }
}
.box1 img, .box2 img, .box3 img {
    animation: fadeInRight 2s 1.9s forwards;
}
```

Im Gegensatz zu den anderen Animationen, die nur einmal ausgeführt werden, handelt es sich bei dem Text *„Premium“* um eine unendlich laufende Animation. Der Text wird durch *'transform: translateX(100 %)'* außerhalb des sichtbaren Bereichs auf der rechten Seite positioniert. Er bewegt sich von rechts nach links, bis er den sichtbaren Bereich auf der linken Seite verlässt (*transform: translateX(-100 %)*). Dabei bleibt die Sichtbarkeit des Textes (opacity: 1) konstant. Die Eigenschaft *'infinite'* in der Anweisung legt fest, dass die Animation unendlich wiederholt wird. Sobald die Animation einmal durchgelaufen ist, beginnt sie sofort wieder von vorne. *'linear'* sorgt dafür, dass die Animation gleichmäßig und ohne Beschleunigung oder Verzögerung abläuft. Die Animation-Ablauft sowie die Anweisung erfolgen wie folgt:

```
@keyframes slidingText {
    from { opacity: 1; transform: translateX(100%);}
    to { opacity: 1; transform: translateX(-100%);}
}
.slidingText span {
    animation: slidingText 8s 1.9s linear infinite;
}
```

Damit die Tabelle vor Beginn der Animationen unsichtbar ist, wird die Eigenschaft 'opacity: 0' zu allen Klassen hinzugefügt, die eine Animation enthalten.

Abschnitt „The Classic“:

Um die Animationen für diesen Abschnitt zu erstellen, wurde die Bibliothek GSAP verwendet. ScrollTrigger wird aktiviert, indem die Zeile *'gsap.registerPlugin(ScrollTrigger)'* in die js-Datei eingefügt wird.

Durch die Verwendung von der *ScrollTrigger.create()*-Methode wird eine GSAP-Animation definiert, die bei bestimmten Scroll-Positionen ausgelöst wird. Der Parameter *'trigger'* spezifiziert das Element, bei dem die Animation aktiviert werden soll. Die *start*- und *end*-Parameter legen fest, an welchen Scroll-Positionen die Animation beginnen und

enden soll. Im folgenden Code wird ein ScrollTrigger für den Abschnitt *"The Classic 1"* mit der entsprechenden Klasse *'.classicIntro'* erstellt:

```
ScrollTrigger.create({
    trigger: '.classicIntro',
    start: 'top top',
    pin: true,
    pinSpacing: false,
});
```

Im vorliegenden Beispiel wird das trigger-Element auf *'.classicIntro'* gesetzt, was bedeutet, dass die Animation ausgelöst wird, sobald dieses Element den definierten Bereich erreicht. Der Parameter *'start'* mit dem Wert *'top top'* legt fest, dass die Animation beginnt, wenn die obere Kante des trigger-Elements mit der oberen Kante des Viewports übereinstimmt. Das *pin*-Attribut ist auf *true* gesetzt, wodurch das *trigger*-Element an seiner Position fixiert wird. In diesem Beispiel gibt es keinen Endpunkt, die Animation wird nicht automatisch gestoppt oder verändert, wenn der Benutzer weiter scrollt. Stattdessen bleibt das Element in der fixierten Position, bis der Benutzer das Element aus dem Viewport heraus scrollt.

'pinSpacing' ist ein zusätzliches Attribut für das Pinning. Sollte dieses Attribut nicht angegeben ist oder auf *true* gesetzt wird, fügt ScrollTrigger automatisch einen zusätzlichen Raum im Layout unter dem gepinnten Element hinzu (Siehe Abbildung 6.11). Dieser Raum hat die gleiche Größe wie das gepinnte Element und verhindert, dass die nachfolgende Elemente über das gepinnte Element hinweg scrollen. Wird *'pinSpacing'* auf *false* gesetzt, wird der zusätzlichen Raum entfernt, sodass die nachfolgende Abschnitte den gepinnten Abschnitt (*"The Classic 1"*) überdecken können.

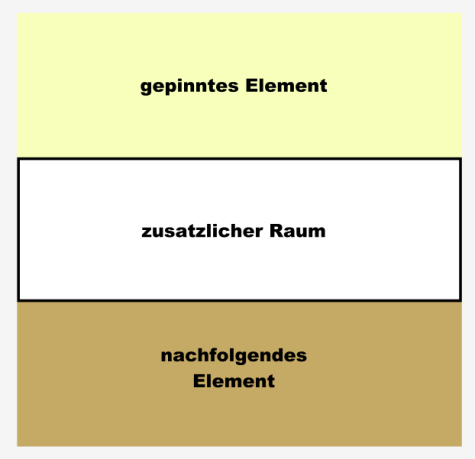

Abbildung 6.11: Layout mit *'pinSpacing: true'* (Quelle: Eigene Darstellung)

Um eine Animation zu erstellen, bei der ein Element von seinem aktuellen Zustand zu einem definierten Endzustand animiert wird, wird die Methode gsap.to() verwendet. Nachfolgend ist ein Beispiel für eine Animation, bei der die Breite des vertikalen Balkens im Abschnitt *„The Classic 1“* mit der Klasse *'.vertical-line'* sich vergrößert wird.

```
gsap.to('.vertical-line', {
    width: '20vw',
    scrollTrigger: {
        trigger: '.classic',
        scrub: 0.5,
        start: 'top bottom',
        end:'top bottom',
    }
});
```

Am Anfang hat der Balken eine Breite von 2vw. Durch diese Methode wächst seine Breite auf 20vw. Die Animation wird durch das Scrollen ausgelöst und gesteuert, indem *scrollTrigger* verwendet wird. Das *trigger*-Element ist *'.classic'* („The Classic 2“), und die Animation beginnt, wenn die Oberkante dieses Elements die Unterkante des Viewports erreicht (*start: 'top bottom'*). Der Parameter *'end'* hat die gleiche Wert mit *'start'*, was bedeutet, dass die Animation keine sichtbare Dauer hat, sie wird sofort abgeschlossen, sobald der Triggerpunkt erreicht wird. Der Parameter *'scrub'* sorgt dafür, dass die Animation synchron zum Scrollen verläuft und eine flüssige Bewegung entsteht, anstatt abrupt zu starten oder zu stoppen.

Im Abschnitt *„The Classic 2“* laufen zwei Animationen gleichzeitig ab. Die erste Animation betrifft die Bewegung der blauen Verpackung mit der Klasse *'.blue'* (Siehe Abbildung 5.4), die von unten nach oben bewegt. Die zweite Animation ist für die horizontale Bewegung der zwei Textblöcke (Siehe Abbildung 5.6) zuständig, die beide die Klasse *'.contentBox'* haben. Diese zweite Animation wird als Hauptanimation betrachtet und in Codeblock 6.6 definiert:

Codeblock 6.6: Animation für Textblöcke

```
const panels = gsap.utils.toArray(".contentBox");
const ani = gsap.to(panels, {
    xPercent: -100 * (panels.length - 1),
    ease: "none",
    scrollTrigger: {
        trigger: ".classic",
        scrub: true,
        start:'top top',
```

```
        pin: '.classic',
        end: "+=" + window.innerWidth,
    }
});
```

Zunächst wird ein Array von den beiden Textblöcken erstellt und in der Variable 'panels' gespeichern. Anschließend wird diese Variable in einer *gsap.to()*-Methode eingesetzt, um eine Animation zu erstellen. Die Eigenschaft *'xPercent: -100 * (panels.length - 1)'* sorgt dafür, dass die Elemente so weit nach links verschoben werden, dass das letzte Element des Arrays am Ende des Animationsbereichs erscheint. Die Animation beginnt, wenn die obere Kante des Trigger-Elements (*"The Classic 2"*) die obere Kante des Viewports erreicht, und endet, sobald die Scroll-Position um die volle Breite des Viewports verschoben wurde. *„The Classic 2"* ist während des gesamten Ablaufs der Animation durch *pin* fixiert.

Im Codeblock 6.7 wird eine Animation für das Element mit der Klasse *'.blue'* (blaue Verpackung) definiert.

Codeblock 6.7: Animation für die blaue Verpackung

```
gsap.to('.blue', {
    y: 0,
    ease: 'none',
    scrollTrigger: {
        containerAnimation: ani,
        trigger: '.milk',
        start: 'left 90%',
        end: 'left left',
        scrub: 0.5,
    }
});
```

Die blaue Verpackung wurde entlang der Y-Achse nach unten verschoben. Durch diese Animation bewegt sich diese Verpackung entlang der Y-Achse zurück auf den Wert 0 (y: 0), ohne eine Verzögerung oder Beschleunigung (ease: *'none'*). Das Element mit der Klasse *'.milk'* (der zweite Textblock) fungiert als Trigger für die Animation, wobei die Animation startet, sobald 10 % des Trigger-Elements den rechten Rand des Viewports erreicht, und sie endet, wenn 90 % desselben Elements den rechten Rand des Viewport berührt. Mit hilfe von *'containerAnimation'* wird diese Animation mit einer anderen Animation (*'ani'*) verknüpft. Der ScrollTrigger steuert nicht nur seine eigene Animation, sondern auch mit der in Codeblock 6.6 definierten Animation synchronisiert wird. Das

bedeutet, dass der User während des Scrollens gleichzeitig sowohl die blaue Verpackung als auch die Textblöcke steuert.

Abschnitt „Full Collection“:

In diesem Abschnitt werden CSS-Animationen verwendet. Zusätzlich kommt der *'IntersectionObserver'* zum Einsatz, der dazu dient, das Eintreten eines bestimmten Elements in den sichtbaren Bereich des Viewports zu erkennen und daraufhin eine Animation zu aktivieren.

Wie in Abbildung 5.7 dargestellt, erscheinen die fünf Linien zusammen mit den fünf Schokolade-Objekten zunächst mit einem Bouncing-Effekt. Dieser Effekt wird durch das Keyframe *„bounceAnim“* in der CSS-Datei definiert und dem Element mit der Klasse *'.container'* zugewiesen. Diese Animation besteht aus zehn Phasen, die in der Abbildung 6.12 dargestellt sind.

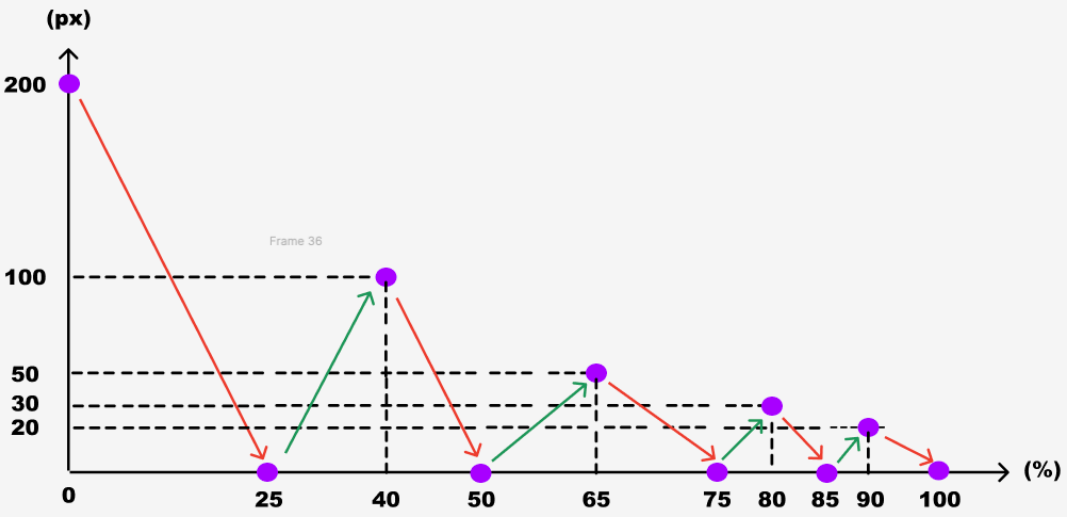

Abbildung 6.12: Bounce-Animation Grafik (Quelle: Eigene Darstellung)

In dieser Grafik steht der lila Kreis für das betreffende Element. Auf der Y-Achse ist die Position des Elements in Pixeln (px) dargestellt, während die X-Achse die bestimmte Zeitpunkte der Animation in Prozent (%) angibt. Zu Beginn (0 %) fällt das Element von einer Höhe von 200px steil ab und erreicht den Boden bei 0px bei 25 % der Animation. Danach prallt das Element wieder auf, erreicht eine Höhe von 100px bei 40 % , fällt erneut ab, und prallt dann mehrmals mit abnehmender Höhe, bis es schließlich bei 100 % der Animation in seiner Endposition bei 0px bleibt. Die Phasen des Abpralls und Aufpralls werden mit abwechselnd roten und grünen Pfeilen dargestellt, wobei die Abprallhöhen in jeder Phase geringer werden.

Nach diesem Bounce-Effekt folgt der Übergang, bei dem die Musiknoten in Schokolade-Objekte verwandelt werden. Das Ausblenden der Musiknoten wird durch *'@keyframes fadeOut'* definiert, wobei die Sichtbarkeit des Elements durch die Eigenschaft *'opacity'* von 1 auf 0 reduziert wird. Die Animation der Schokolade-Objekte wird durch *'@keyframes fadeIn'* definiert, bei der ein Element sowohl in der Sichtbarkeit als auch in der Position auf der Y-Achse verändert wird. Zu Beginn (0 %) ist das Element unsichtbar (*'opacity: 0'*) und befindet sich in seiner Ausgangsposition. 10 % wird es sichtbar und bleibt noch an dieser Stelle. Dann bewegt es sich bei 25 % um 180px nach oben (*'transform: translateY(-180px)'*), bevor es bei 75 % durch *'transform: translateY(30px)'* nach unten wandert. Am Ende, bei 100 %, bleibt das Element sichtbar und kehrt in seine ursprüngliche Position zurück.

Nachdem alle fünf Schokoladen erschienen sind, wird unter den letzten drei Schokoladen der Text *'New'* mit Klasse *'.text'* angezeigt. Diese Animation wird wie folgt definiert:

Codeblock 6.8: Scale-Animation

```
@keyframes textAnim {
    0% { opacity: 1; transform:scaleX(0);
        transform-origin: top left;
    }
    100% { opacity: 1; transform: scaleX(1);
        transform-origin: top left;
    }
}
```

Bei dieser Animation wächst das Element von einer unsichtbaren, schmalen Form zu seiner vollen Breite. Durch die Funktion *'scaleX(0)'* wird das Element am Anfang entlang der X-Achse auf eine Breite von 0 verkleinert. Es ist dann unsichtbar, obwohl die Opazität immer auf 1 bleibt. Am Ende wird seine horizontale Skalierung auf 1 zurückgesetzt, wodurch es seine volle Breite hat. Der Ursprungspunkt für die Skalierung liegt dabei in der oberen linken Ecke des Elements (*'transform-origin: top left'*).

Am Ende bewegen sich alle Schokolade-Objekte entlang der Y-Achse. Die Objekte 1, 3 und 5 bewegen sich durch *'@keyframes container-odd-Anim'* 15px nach oben und kehren dann in ihre Ausgangsposition zurück. Die Objekte 2 und 4 bewegen sich durch *'@keyframes container-odd-Anim'* 15px nach unten und kehren ebenfalls zurück. Dieser Vorgang wird kontinuierlich wiederholt, da die Animation mit der Eigenschaft *'infinite'* den betreffenden Elementen zugewiesen wird.

Die Dauer und Verzögerung der Animation aller betreffenden Elementen sind In Tabelle 6.1 aufgelistet.

Tabelle 6.1: Dauer und Verzögerung der Animation

CSS-Selector	animation-name	duration (s)	delay (s)
.container	bounceAnim	1	0
.note3	fadeOut	1	1.4
.choco3	fadeIn	1	1.6
.note4	fadeOut	1	1.6
.choco4	fadeIn	1	1.8
.note5	fadeOut	1	1.8
.choco5	fadeIn	1	2
.text	textAnim	0.5	3
.choco-container:first-child	container-odd-Anim	3	3.5
.choco-container:nth-child(2)	container-even-Anim	3	3.5
.choco-container:nth-child(3)	container-odd-Anim	3	3.5
.choco-container:nth-child(4)	container-even-Anim	3	3.5
.choco-container:nth-child(4)	container-odd-Anim	3	3.5

Wie bereits erwähnt, werden diese Animationen ausgeführt, nur wenn der Nutzer zu einer bestimmten Position scrollt. Dabei kommt der IntersectionObserver zum Einsatz, der erkennt, wann ein Element in den sichtbaren Bereich des Viewports eintritt.

Im Codeblock 6.9 wird der Abschnitt *„Full Collection“* mit der Klasse *'.collection'* ausgewählt, um eine Animation zu steuern. Eine Variable *'animationPlayed'* wird auf *'false'* gesetzt, um sicherzustellen, dass die Animation nur einmal ausgeführt wird. Es werden IntersectionObserver-Optionen definiert, wobei *'root'* auf *'null'* gesetzt ist, sodass der Viewport als Bezugspunkt dient, und *'threshold'* auf 0,8 festgelegt ist, was bedeutet, dass 80 % des Abschnitts *„Full Collection“* sichtbar sein müssen, um die Animation zu starten. Der IntersectionObserver überwacht den Abschnitt und aktiviert die Animation, indem die Klasse *'animate-full'* hinzugefügt wird. Nachdem die Animation gestartet wurde, wird *'animationPlayed'* auf *'true'* gesetzt und der Observer hört auf, das Element zu überwachen, um die Animation nicht erneut auszulösen.

Codeblock 6.9: IntersectionObserver (

```
1 const collection = document.querySelector('.collection');
2 let animationPlayed = false;
3 const options = {
4     root: null,
5     threshold: 0.8
6 };
```

```
const observer = new IntersectionObserver((entries, observer)=>{
    entries.forEach(entry => {
        if (entry.isIntersecting && !animationPlayed) {
            collection.classList.add('animate-full');
            animationPlayed = true;
            observer.unobserve(entry.target);
        }
    });
}, options);
observer.observe(collection);
```

Abschnitt „The Unique“

Das horizontale Scrollen in diesem Abschnitt wird mit gsap.to()-Method (Codeblock 6.10) erzeugt, auf ähnliche Weise wie im Abschnitt *„The Classic“*.

Codeblock 6.10: Horizontales Scrollen

```
const sections = gsap.utils.toArray('#section');
const scrollTween = gsap.to(sections, {
    xPercent: -100 * (sections.length - 1),
    ease: 'none',
    scrollTrigger: {
        trigger: '.unique',
        pin: true,
        scrub: 1,
        start: 'top top',
        end: document.querySelector(".unique").offsetWidth *
            (sections.length),
    }
});
```

Zunächst wird ein Array von allen Unterabschnitte (*Page 1*, *Page 2*, *Page 3*, *Page 4*) erstellt und in der Variable 'sections' gespeichern. Das horizontale Scrollen beginnt, wenn die obere Kante des Abschnitts „The Unique“ die obere Kante des Viewports erreicht, und endet, wenn das Scrollen die Breite von *'.unique'* multipliziert mit der Anzahl der Unterbschnitte erreicht hat. Dies bedeutet: Die Animation endet, nachdem der Benutzer weit genug gescrollt hat, dass alle Unterabschnitte einmal durch den Sichtbereich gescrollt sind.

Auf *Page 1* gibt es nur eine Animation: Die Länge der horizontalen Balken wächst von 1rem auf 10rem. Diese Animation beginnt und endet gleichzeitig, sobald die Oberkante von *Page 1* die Oberkante des Viewports erreicht.

Auf *Page 2*, *Page 3* und *Page 4* gibt es 5 Animationen. Der vertikale Balken (*Line*) mit der Klasse *'.block-left'* dient als Trigger für diese 5 Animationen. Die Animationen beginnen, wenn die linke Kante von *'.block-left'* die rechte Kante des Viewports berührt, und enden, wenn die linke Kante von *'.block-left'* 20 % des Viewports erreicht.

Das Element BigBlock mit der Klasse *'.big-block'* wurde durch *'transform: translateX(-100 %)'* nach links verschoben. Wenn die Animation aktiviert wird, bewegt sich das Element wieder zurück in seine Ausgangsposition, da *'x: 0'* in der gsap.to()-Methode festgelegt ist.

Das Element *Title* mit der Klasse *'.title'* und das Element *Description* mit der Klasse *'.description'* wurden durch *'transform: translateY(-400 %)'* bzw. *'transform: translateY(-200 %)'* nach oben verschoben. Das Element *MainImg* mit der Klasse *'.mainImg'* wurden durch *'transform: translateY(200 %)'* nach unten verschoben. Diese Elemente bewegt sich wieder zurück in die Ausgangsposition, da *'y: 0'* in der gsap.to()-Methode festgelegt ist.

Das Element mit der Klasse *'.chili-img'* auf *Page 2* dreht sich um 60 Grad durch der Eigenschaft *'rotate: 60'*.

Das Element mit der Klasse *'.taro-img'* auf *Page 3* besitzt eine Fade-in-Animation, bei der die Opazität auf 1 erhöht wird.

Das Element mit der Klasse *'.pineapple-img'* auf Page 4 wird zunächst durch *'transform: rotate(-10deg) translateY(-200 %)'* gedreht und nach oben verschoben, was bedeutet, dass es um 10 Grad entgegen Uhrzeigersinn gedreht und um das Doppelte seiner Höhe nach oben verschoben wurde. Anschließend kehrt es durch die Angabe von *'x: 0'* und *'y: 0'* in der gsap.to()-Methode zu seiner ursprünglichen Position zurück.

7 Usability Test

7.1 Testmethod

Nachdem die Website vollständig entwickelt wurde, wurde ein Usability-Test durchgeführt, um sicherzustellen, dass die Website benutzerfreundlich ist und den Anforderungen der Nutzer entspricht. Usability-Tests sind wichtig, da sie aufzeigen, wie effektiv, effizient und zufriedenstellend eine Website für die Nutzer ist. Um diesen Test durchzuführen, wurde die Webuse-Methode verwendet. Dabei wird ein subjektiver Ansatz verwendet, der auf einer Fragebogen-Methode basiert, um die zu evaluierende Website zu bewerten. Die Bewertungskriterien wurden in vier Usability-Kategorien zusammengefasst, die in Abbildung 7.1 dargestellt werden.

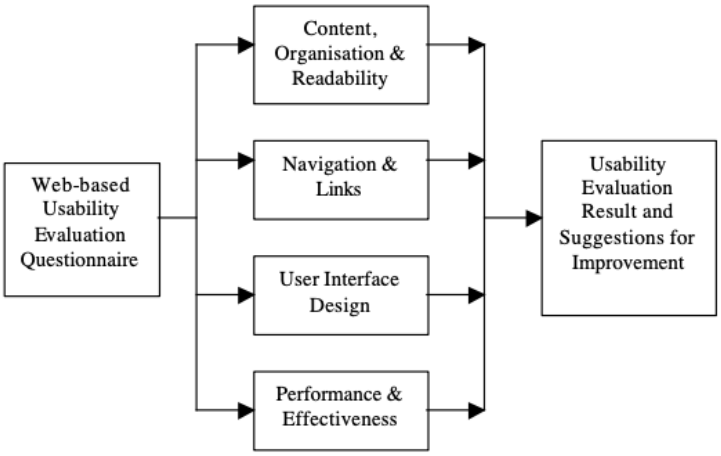

Abbildung 7.1: WebUse Modelle (Quelle: (Chiew, 2003, S. 53))

Jede Kategorie umfasst sechs formulierte Aussagen, die jeweils im Fragebogen mit einer Likert-Skala bewertet wird. Die Likert-Skala bietet fünf Antwortoptionen: *strongly agree*, *agree*, *neutral*, *disagree* and *strongly disagree*. Jede Option entspricht einem bestimmten Merit-Wert, der zur Gewichtung von Umfragedaten verwendet wird. Die Likert-Skala und der Merit-Wert sind in Tabelle 7.1 dargestellt.

Tabelle 7.1: Likert-Skala und Merit-Wert

Option	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Merit	1.0	0.75	0.5	0.25	0

Um den Usability-Wert jeder Kategorie in der Webuse-Methode zu berechnen, wird die folgende Formel verwendet:

$$x = \frac{\sum(\text{Merit-Wert für jede Frage der Kategorie})}{\text{Anzahl der Fragen}} \tag{7.1}$$

Die Formel zur Berechnung des gesamten Usability-Werts der Website sieht wie folgt aus:

$$x = \frac{\sum(\text{Usability-Werte aller Kategorien})}{\text{Anzahl der Kategorien}} \tag{7.2}$$

Nachfolgend ist die Bewertungstabelle der Webuse-Methode, die den Usability-Point und das entsprechende Usability-Level darstellt.

Tabelle 7.2: Usability-Point mit entsprechenden Usability Level

Option	**Usability Level**
0.8<= x <=1.0	Excellent
0.6<= x <=0.8	Good
0.4<= x <=0.6	Moderate
0.2<= x <=0.4	Poor
0.0<= x <=0.2	Bad

Fragebogen

In der Webuse-Methode enthält der Fragebogen insgesamt 24 Aussagen, die auf diese vier Kategorien verteilt sind, wobei jede der vier Kategorien 6 Aussagen umfasst (Jannah, 2022).

Content, Organisation, Readibility (W1):

(W1.1) The website has provided the latest information (updates) as I needed

(W1.2) On this website, I can easily find the information I need

(W1.3) The content on the website is well organized

(W1.4) I can easily read the contents of the website

(W1.5) I feel comfortable and familiar with the language applied to the website

(W1.6) I don't have to swipe left and right when reading the contents on the website

Navigation, Links (W2):

(W2.1) I can easily find out which position or on what page I am on the website

(W2.2) The website provides navigation and useful links for me to get the information I need

(W2.3) I can move around the website by using the link or the Back button to the next or next page

(W2.4) The link on the website is functioning and well maintained

(W2.5) The website doesn't open too many new tabs when I browse the website

(W2.6) The placement of links or menus is standardized and I can easily figure it out

User Interface Design (W3):

(W3.1) Attractive interface design on the website

(W3.2) I feel comfortable with the colors applied to the website

(W3.3) The website does not contain features that make it difficult for me, for example scrolling or flickering text and animations that are often repeated

(W3.4) The website has a consistent feel and appearance on all pages on its website

(W3.5) The website does not contain many ads

(W3.6) The website design can be understood and studied well in using it

Performance, Effectivenss (W4):

(W4.1) I don't have to wait too long to open the page

(W4.2) I can tell the difference between visited and unvisited links

(W4.3) I can access the website anytime and anywhere

(W4.4) The website responded to the actions I took as I expected

(W4.5) I can save time, energy or money when using the website

(W4.6) The website always provides clear and useful messages or information when I don't know what to do next

7.2 Ergebnisse

7.2.1 Testablauf

An der Umfrage haben insgesamt fünf Personen teilgenommen, vier davon sind im Alter von 25 bis 31 Jahren und es gibt eine Person, die 45 Jahre alt ist. Die Teilnehmer wurden eingeladen, die Website auf demselben Laptop zu durchstöbern, der auch für die Entwicklung der Website verwendet wurde. Sie nutzten den Chrome-Browser und testeten die Website auf verschiedenen Geräten und in unterschiedlichen Bildschirmgrößen, um eine umfassende Bewertung vorzunehmen. Anschließend wurden sie gebeten, den Fragebogen auszufüllen. Die Umfrage wurde mittels Google Formulare erstellt.

Die gesammelten Daten wurden gemäß der zuvor beschriebenen Formel berechnet, um den endgültigen Merit-Wert zu ermitteln, der zur Bewertung der gesamten Usability der Website verwendet wird.

Es gibt einige Fragen, die nicht beantwortet wurden. Konkret wurde Frage *W4.3* nicht beantwortet, da es sich um eine lokale Website handelt, die für Lernzwecke erstellt wurde und noch keine offizielle Domain besitzt. Daher konnten die Teilnehmer nicht von ihren eigenen Geräten aus auf die Website zugreifen. Zudem wurden die Fragen *W2.4*, *W2.6* und *W4.2* nicht beantwortet, da die Website keine Links zu anderen Seiten enthält.

7.2.2 Ergebnisse

Im Folgenden sind die statistischen Ergebnisse sowie die Gesamtpunktzahl jeder Aussage aufgeführt.

Tabelle 7.3: Ergebniss Kategorie Content, Organisation,Readibility (W1)

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	**Total Merit Score**
W1.1	0 User	0 User	3 User	2 User	0 User	3
W1.2	0 User	0 User	4 User	1 User	0 User	2.75
W1.3	0 User	0 User	0 User	4 User	1 User	4
W1.4	0 User	1 User	1 User	2 User	1 User	3.75
W1.5	0 User	0 User	0 User	0 User	5 User	5
W1.6	0 User	0 User	0 User	0 User	5 User	5

Tabelle 7.4: Ergebniss Kategorie Kategorie Navigation, Links (W2)

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	**Total Merit Score**
W2.1	1 User	3 User	1 User	0 User	0 User	1.25
W2.2	5 User	0 User	0 User	0 User	0 User	0
W2.3	5 User	0 User	0 User	0 User	0 User	0
W2.4	x	x	x	x	x	x
W2.5	0 User	0 User	0 User	0 User	5 User	5
W2.6	x	x	x	x	x	x

Tabelle 7.5: Kategorie User Interface Design (W3)

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	**Total Merit Score**
W3.1	0 User	0 User	1 User	2 User	2 User	3
W3.2	0 User	0 User	0 User	3 User	2 User	4.25
W3.3	0 User	0 User	3 User	2 User	0 User	3
W3.4	0 User	0 User	1 User	3 User	1 User	3.75
W3.5	0 User	0 User	0 User	0 User	5 User	5
W3.6	0 User	0 User	1 User	4 User	0 User	3.5

7.2.3 Datenanalyse

Die Usability-Wert für jede einzelne Kategorie wird mittels die Formel 7.1 berechnet. Hierbei wird der durchschnittliche Merit-Wert berechnet, indem die Summe der Merit-Werte aller Fragen innerhalb einer Kategorie durch die Anzahl der Fragen in dieser Kategorie geteilt wird. Dadurch erhält man den spezifischen Usability-Wert für die jeweilige Kategorie.

Tabelle 7.6: Ergebniss Kategorie erformance, Effectiveness (W4)

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	**Total Merit Score**
W4.1	0 User	0 User	0 User	0 User	5 User	5
W4.2	x	x	x	x	x	x
W4.3	x	x	x	x	x	x
W4.4	0 User	0 User	2 User	3 User	0 User	3.25
W4.5	0 User	0 User	3 User	2 User	0 User	3
W4.6	0 User	4 User	1 User	0 User	0 User	1.5

Tabelle 7.7: Merit-Wert Kategorie Content, Organisation,Readibility (W1)

	W1.1	W1.2	W1.3	W1.4	W1.5	W1.6
Total Merit Score	3	2.75	4	3.75	5	5
Average Merit	0.6	0.55	0.8	0.75	1	1
Average Number of Merit Scores			4.7			
Usability Value for each Category			0.78			
Usability Level			Good			

Der durchschnittliche Merit-Wert jeder Frage (Average Merit) wird berechnet, indem die Gesamtpunktzahl (Total Merit Score) der jeweiligen Frage durch die Gesamtzahl der Teilnehmer (also 5) geteilt wird. Basierend auf der Tabelle 7.7 wird die Usability der Kategorie (W1) aus dem Durchschnitt der erreichten Punktzahlen ermittelt. Dieser beträgt 4,7, geteilt durch die Anzahl der Fragen, also 6. Daraus ergibt sich ein Usability-Wert von 0,78 für alle Aussagen (W1.1-W1.6) mit einem Usability-Level von „Gut“.

Tabelle 7.8: Merit-Wert Kategorie Navigation, Links (W2)

	W2.1	W2.2	W2.3	W2.4	W2.5	W2.6
Total Merit Score	1.25	0	0	x	5	x
Average Merit	0.25	0	0	x	1	x
Average Number of Merit Scores			1.25			
Usability Value for each Category			0.31			
Usability Level			Poor			

Basierend auf der Tabelle 7.8 wird die Usability der Kategorie (W2) aus dem Durchschnitt

der erreichten Punktzahlen ermittelt. Dieser beträgt 1.25, geteilt durch die Anzahl der Fragen, also 4. Daraus ergibt sich ein Usability-Wert von 0,31 für alle Aussagen (W2.1-W2.6) mit einem Usability-Level von „Schlecht“.

Tabelle 7.9: Merit-Wert Kategorie User Interface Design (W3)

	W3.1	W3.2	W3.3	W3.4	W3.5	W3.6
Total Merit Score	3	4.25	3	3.75	5	3.5
Average Merit	0.6	0.85	0.6	0.75	1	0.7
Average Number of Merit Scores			4.5			
Usability Value for each Category			0.9			
Usability Level			Excellent			

Basierend auf der Tabelle 7.9 wird die Usability der Kategorie (W3) aus dem Durchschnitt der erreichten Punktzahlen ermittelt. Dieser beträgt 4.5, geteilt durch die Anzahl der Fragen, also 6. Daraus ergibt sich ein Usability-Wert von 0,9 für alle Aussagen (W3.1-W3.6) mit einem Usability-Level von „Sehr gut“.

Tabelle 7.10: Merit-Wert Kategorie Performance, Effectiveness (W4)

	W4.1	W4.2	W4.3	W4.4	W4.5	W4.6
Total Merit Score	5	x	x	3.25	3	1.5
Average Merit	1	x	x	0.65	0.6	0.3
Average Number of Merit Scores			2.55			
Usability Value for each Category			0.63			
Usability Level			Good			

Basierend auf der Tabelle 7.10 wird die Usability der Kategorie (W4) aus dem Durchschnitt der erreichten Punktzahlen ermittelt. Dieser beträgt 2.55, geteilt durch die Anzahl der Fragen, also 4. Daraus ergibt sich ein Usability-Wert von 0,63 für alle Aussagen (W4.1-W4.6) mit einem Usability-Level von „Gut“.

Der gesamte Usability-Wert der Website wird mittels die Formel 7.2 berechnet. Hierbei wird die Summe der Usability-Werte aller Kategorien genommen und durch die Anzahl der Kategorien (also 4) geteilt. Diese Berechnung ergibt den durchschnittlichen Usability-Wert der gesamten Website und gibt somit einen allgemeinen Überblick über die Usability der Website.

Tabelle 7.11: Website Usability-Level

	Score	Usability Level
Total Usability Scores for all Categories	2.62	
Total Website Usability Scores	0.65	Good

Der Gesamtwert der Usability aller Kategorien beträgt 2.55. Dadurch ergibt sich ein Gesamtwert der Usability für die Website von 0.63, was dem Usability-Level „Gut“ entspricht.

7.3 Evaluation

Nach der Durchführung der Usability-Test wurden einige Schwachstellen identifiziert, die im Folgenden detailliert analysiert werden.

Fehlende Navigation

Der größte Nachteil der Website besteht darin, dass keine Hauptnavigation existiert. Daher bekommen die Aussagen *W2.4* und *W2.6* die niedrigste Merit-Wert von 0. Da es sich bei diesem Projekt um eine Single-Page-Website handelt, wurde auf eine klassische Hauptnavigation verzichtet. Die Teilnehmer wurden nach ihrer Meinung gefragt, ob ein Navigation-Menu für diese Website notwendig ist. Zwei Teilnehmer sagten, dass dies unnötig ist. Eine Navbar wäre sinnvoller, wenn die Marke zwei oder mehrere unterschiedliche Produkte präsentieren möchte. In diesem Fall jedoch, wo nur ein Produkt (Schokolade) vorgestellt wird, empfinden die zwei Teilnehmer eine Navbar als überflüssig und gaben an, dass sie diese auch nicht nutzen würden, selbst wenn eine Navbar vorhanden wäre.

Zwei andere Teilnehmer hingegen hatten die gegenteilige Meinung und fanden es sehr nachteilig, dass die Website keine Navigation bietet. Sie empfanden es als zeitraubend, durch Scrollen zu einem anderen Abschnitt, wie zum Beispiel vom Abschnitt „The Unique“, zurück zum Seitenanfang zu gelangen. Obwohl die Marke nur ein Produkt, nämlich Schokolade, anbietet, sollte die Website dennoch eine Navbar haben, die zu Abschnitten wie “Intro”, “The Classic” und “The Unique” führt.

Der verbleibende Teilnehmer äußerte, dass ihn das Scrollen zwar nicht störe, aber eine Website für ein Unternehmen ohne Navigation als sehr unprofessionell wirkt. Er fügte hinzu, dass die Seite mehr Inhalte wie z.B ein Kontaktformular enthalten sollte. Eine Navbar wäre dafür sehr sinnvoll.

Durch das Feedback der Teilnehmer kann geschlossen werden, dass es nach wie vor notwendig ist, Navigation zu der Website hinzuzufügen. Dies hilft den Nutzern, sich besser auf der Website zurechtzufinden, schnell und gezielt zu den gewünschten Inhalten zu gelangen, ohne langwierig scrollen zu müssen. Eine mögliche Lösung wäre die Implementierung einer Sticky-Navigation, das es dem Benutzer ermöglicht, direkt zu den verschiedenen Abschnitten der Seite zu springen.

Scroll-Geschwindigkeit

Ein weiteres identifiziertes Problem betrifft die Scroll-Geschwindigkeit beim horizontalen Scrollen, die sich je nach Bildschirmgröße und Auflösung des Geräts verändert. Beim Testen wurde festgestellt, dass die Scroll-Geschwindigkeit bei der Anpassung der Bildschirmgröße ungleichmäßig wird: In einigen Fällen scrollt die Seite schneller als normal, in anderen Fällen langsamer. Die Scroll-Geschwindigkeit wird wieder normal, wenn die Seite neu geladen wird. Die Teilnehmer gaben an, dass sie sich nicht allzu sehr durch dieses Problem gestört fühlten, da die Änderungsgeschwindigkeit nicht zu hoch war und die anderen Animationen weiterhin einwandfrei funktionierten.

Das Problem tritt auf, weil die Scroll-Animation von GSAP basierend auf der aktuellen Breite des Fensters berechnet wird. Wenn sich diese Breite ändert, stimmt die vorherige Berechnung nicht mehr. Der zentrale Punkt dieses Problems liegt darin, dass GSAP-Animationen und Scroll-Trigger nicht automatisch auf die neuen Abmessungen reagieren, wodurch Diskrepanzen in der Scroll-Geschwindigkeit auftreten. Um eine nachhaltige Lösung zu finden, muss der Grundsatz beachtet werden, dass Animationen dynamisch auf Größenänderungen reagieren können. Dies erfordert eine regelmäßige Aktualisierung der Berechnungen bei jeder Änderung der Fenstergröße.

Die Analyse zeigt, dass die Website tatsächlich noch viele Mängel aufweist und erheblich überarbeitet werden muss. Trotz ihrer Schwächen wurde die Website als 'gut' bewertet. Dies könnte darauf zurückzuführen sein, dass der Usability-Test nur mit fünf Teilnehmern durchgeführt wurde, was möglicherweise zu wenig ist. Zudem haben sich die meisten Teilnehmer hauptsächlich auf das Design der Website konzentriert und andere wichtige Faktoren nicht ausreichend berücksichtigt.

8 Fazit

Durch dieses Projekt wurden grundlegende Kenntnisse in der Erstellung von Animationen und der Entwicklung von responsiven Websites erlangt. Erreicht wurde die Erstellung von CSS-Animationen; die Verwendung von Intersection Observer API, um Animationen beim Scrollen zu aktivieren; der Einsatz von Media Queries für Responsives Webdesign; sowie die Anwendung von GSAP, um horizontales Scrollen und komplexe Animationen zu erstellen.

Die getestete Website zwar einige Stärken aufweist, jedoch auch mehrere Schwächen, die die Benutzererfahrung beeinträchtigen. Als Stärke wird die Website für ihr schönes Design und attraktive Animationseffekt gelobt. Allerdings liegt die Schwäche in der Usability: Die fehlende Navigation, die ungleichmäßige Scroll-Geschwindigkeit sind allesamt Probleme, die behoben werden müssen, um die Benutzerfreundlichkeit und Zugänglichkeit der Website zu verbessern. Ohne eine klare Navigation kann es für die Benutzer schwierig sein, schnell zu dem gewünschten Inhalt zu gelangen. Zudem können Benutzer, die an traditionelle mehrseitige Websites gewöhnt sind, verwirrt oder frustriert sein, wenn sie keine Navigation vorfinden.

Dieses Projekt hat nicht die erhofften Ergebnisse erzielt, was auf eine unzureichende Auseinandersetzung mit dem Thema Usability zurückzuführen ist. Es wurde klar, wie wichtig gründliche Usability-Tests sind, um potenzielle Probleme zu identifizieren, gezielte Verbesserungen vorzunehmen und zukünftige Projekte besser zu planen.

Literatur

Chiew, T. K. (2003). WEBUSE: WEBSITE USABILITY EVALUATION TOOL. *Malaysian Journal of Computer Science*, *16*(1), 47–57. https://doi.org/10.1109/ICTIIA54654.2022.9936027

Dirksen, J. (2023). *Learn Three.js: Program 3D animations and visualizations for the web with JavaScript und WebGL* (4. Aufl.). Packt Publishing.

Domin, A. (2018). *Wie der Mosaic Browser das Internet revolutionierte*. Verfügbar 23. Mai 2024 unter https://www.welt.de/wirtschaft/webwelt/article176090768/Wie-der-Mosaic-Browser-das-Internet-revolutionierte.html

Hadwich, K. (2022). *Smart Services: Band 3: Kundenperspektive – Mitarbeiterperspektive – Rechtsperspektive* (1. Aufl.). Springer.

Herbst, M. (1999). *Informationsmanagement in der Medizin: Beispiele und Perspektiven* (1. Aufl.). Springer.

Höllings, C. (2022). *Was ist Mobile First?* https://www.heise-regioconcept.de/glossar/mobile-first#:~:text=Bei%20all%20den%20Vorteilen%20gibt,und%20Desktop%20gleicherma%C3%9Fen%20zu%20finden.

Horster, E. (2022). *Digitales Tourismusmarketing: Grundlagen, Suchmaschinenmarketing, User-Experience-Design, Social-Media- Marketing und Mobile Marketing* (1. Aufl.). Springer.

Jannah, F. A. (2022). Evaluation of Bunga Bali Florist Website Usability Using The Website Usability Evaluation (Webuse) Method. *2022 1st International Conference on Technology Innovation and Its Applications (ICTIIA)*. https://doi.org/10.1109/ICTIIA54654.2022.9936027

Janne, U. (2024). *Web Design and CSS Animation* (1. Aufl.). Blue Rose.

Kadlec, T. (2013). *Praxiswissen Responsive Webdesign* (1. Aufl.). Pearson Education.

Kielholz, A. (2008). *Online-Kommunikation: Die Psychologie der neuen Medien für die Berufspraxis: E-Mail, Website, Newsletter, Marketing, Kundenkommunikation* (1. Aufl.). Springer.

Michaelis, A. (2021). Responsive Webdesign. *WiSt - Wirtschaftswissenschaftliches Studium*, *44*(7), 50–53. https://doi.org/10.15358/0340-1650-2015-7

Raaf, U. (2021). *Der SEO Planer: Suchmaschinenoptimierung in Unternehmen richtig organisieren und umsetzen (mit Checklisten)* (1. Aufl.). Springer.

Riempp, R. (2019). *Digital Storytelling im Web: am Beispiel von scroll-activated animations* (1. Aufl.). Springer.

Sinner, D. (2018). *Webtechnologien: JavaScript . PHP . Datenbank* (1. Aufl.). Springer.

Staab, P. (2019). *Don't worry, be digital: Tipps für einen angstfreien Umgang mit Digitalisierung* (1. Aufl.). Springer.

Steinberg, M. (2020). *Entwicklung von Multimediasystemen 2: Animationen mit GSAP*. https://f3-studip.fh-h.de/sendfile.php?type=0&file_id=dd66dde4af01d46ba6be41092a5a2707&file_name=05b_MM_SYS2_2022.pdf

Streich, M. (2000). Das World Wide Web (WWW) als neues Medium für Verbraucherinformationen. *Diskussionsbeiträge*, *1*(306). https://www.econstor.eu/bitstream/10419/68862/1/685626482.pdf

Süss, Y. (2016). *E-Commerce für klein- und mittelständische Unternehmen: Konkrete Schritte zum digitalen Erfolg* (1. Aufl.). Springer.

Wilde, E. (1999). *World Wide Web: Technische Grundlagen* (1. Aufl.). Springer.

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel

Konzeption und Entwicklung einer Single-Page-Website mit scrollbasiertem Animationseffekt

selbstständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

Hamburg, 3. September 2024