Design interaktiver Oberflächen

Abgabe von

Jeremy Grunert

Matrikelnummer: 29265

Hochschule Harz
Medieninformatik | SoSe 2021
Dozent: Prof. Daniel Ackermann
Datum: 31.08.21

Inhaltsverzeichnis

1. Konzept

Das Konzept ist eine responsive Webseite, welche meine Arbeit als Fotograf veranschaulichen soll. Eine interessante Landingpage, eine Übersicht meiner Bilder und Dienstleistungen und ein Kontaktformular sollen erkennbar sein. Zwei Portfolio-Seiten zeigen meine Aufnahmen im Masonry-Layout, sofern die Fotos unterschiedliche Dimensionen haben. Eine konsistente Seitengestaltung mit Fokus auf Weißraum und Farbe. Die Seite besteht im Großteil aus Graustufen und hat lediglich eine Akzentfarbe. Damit soll der Schwerpunkt auf die Aufnahmen gerichtet werden.

2. Herangehensweise

2.1 Grundlegendes

Bevor das Grundgerüst stand, habe ich erst einmal verschiedene Herangehensweisen ausprobiert.

Die erste Idee war eine einfache Webseite mit HTML, CSS, Javascript und responsivem Design. Die Navigation war das erste Element, das Einzug gefunden hat. Den Wrapper habe ich mit dem Container ersetzt. Der Container kommt dann zum Einsatz, sobald eine maximale Breite für den Inhalt nicht überschritten werden soll.

2.2 Framework

Um das responsive Design leichter zu gestalten, habe ich mich entschieden, ein Frontend-CSS-Framework zu benutzen. Zuerst habe ich Bootstrap und Materialize ausprobiert. Von beiden Frameworks war ich nicht überzeugt und habe mich letztlich für Tailwind CSS entschieden. Die Umstellung mit Frameworks bzw. mit Tailwind CSS zu arbeiten war zuerst ungewohnt. Nach der Umgewöhnungsphase ermöglichen diese Frameworks jedoch eine einfachere Umsetzung eines konsistenten responsiven Webauftritts. Dem Framework habe ich noch zusätzliche Utility-Classes hinzugefügt, um auf diese leicht im HTML-Code zugreifen zu können. Ich habe mein eigenes Farbthema und Klassen, um ein responsives Masonry-Layout zu ermöglichen, hinzugefügt. Die Masonry-Klassen teilen den Bereich lediglich in Spalten. Parallel dazu habe ich noch eine Custom-CSS. In dieser sind eine Custom-Scrollbar, die Hero-Landingpage, das Fullscreen-Navigation-Overlay und die Wellen definiert. Schließlich stellt die Hamburger-CSS das Hamburger-Icon und die Klick-Animation zur Verfügung.

2.3 Navigation & Footer

Nach dem erfolgreichen Installieren von Tailwind CSS habe ich die Navigation noch mal neu aufgebaut. Abhängig vom Inhalt des Menüs habe ich nach den passenden Breakpoints gesucht, wann die maximale Breite angepasst werden muss. Bevor der Medium-Breakpoint (@media (min-width: 768px)) erreicht wird, ist ein Hamburger-Menü klickbar. Dieses Hamburger-Menü führt beim Anklicken das Ham-

burger-Menü-Javaskript aus, welches wiederum die Höhe des Fullscreen-Overlays zwischen 0 % und 100 % Viewport-Höhe hin- und herschaltet. Ich habe verschiedene Hamburger ausprobiert und mich letztlich für einen der sehr stylishen Burger von march08 entschieden. Ich habe lediglich die Farbe und Größe vom Hamburger bearbeiten müssen. Das Fullscreen-Overlay beinhaltet die primäre Navigation, welche auch sichtbar ist, sobald das Fenster eine Breite von mindestens 768 Pixel aufweist. Die Navigationselemente haben einen leichten Hover-Effekt erhalten. Im ähnlichen Design habe ich den Footer erstellt und darauf geachtet, dass die Breakpoints auch mit der Navigationsleiste übereinstimmen. Um eine Wort-Bild-Marke bzw. die kleinen Icons im Footer darzustellen, habe ich ein SVG-Kamera-Bild und Font-Awesome-Icons eingesetzt.

2.4 Landingpage

Als Index-Seite habe ich mich für eine Hero-Landingpage entschieden, welche ein Bild in Höhe des Viewports und im unteren Bereich noch überlappende Wellen darstellt. Die einfachen, übereinander liegenden Wellen sind von Goodkatz. Damit die Höhe auch ohne Scrollen funktioniert, habe ich die Navigationsleistenhöhe von der Höhe des Viewports abgezogen. Den Footer mit den Wellen hatte ich erst später implementiert und dann die CSS-Position-Property auf fixed gesetzt. Insofern habe ich beide Lösungsmöglichkeiten eingebaut, um die Inhaltsgröße der Viewport-Höhe anzupassen.

2.5 Portfolio & Portraits

Die Portfolio- und Portraits-Seite zeigen meine Bilder im Masonry-Look, sofern das für die Bilddimensionen notwendig ist. Für die Portraits-Seite hätte ich auch ein einfaches Flexbox-Layout aufbauen können. Das habe ich nicht realisiert, weil ich mich erst nach Fertigstellen der Seiten für Bilder mit identischen Dimensionen entschieden habe. Die Bilder haben zusätzlich noch ein Lazy-Loading-Skript. Bei Auswahl eines Bildes wird dieses in einem neuen Tab geöffnet. Ich habe leider zu spät mit dem Projekt angefangen und konnte eine Modal-Lightbox nicht rechtzei-

tig einbauen. Auf beiden Seiten ist am Ende ein responsives Call-To-Action Element.

2.6 Pricing

Auf der Pricing-Seite benutze ich angepasste responsive Blöcke von Tailblocks. Zwei Rubriken werden auf dieser Seite präsentiert. Eine bewirbt die fairen Preise meiner Dienstleistung als renommierter Fotograf. Die andere zeigt eines meiner erwerbbaren Prints.

2.7 Contact

Die Contact-Page zeigt im oberen Bereich ein angepasstes Kontaktformular von Tailblocks. Im Unteren ist ein Inlineframe mit eingebetteter Google-Karte.

2.8 Production

Meine Abgabe beinhaltet zwei Ordner, „Production“ und „Development“. Im Development-Ordner sind alle unveränderten Dateien, um einen klaren Einblick in den Code zu erhalten. Des Weiteren sind auch alle „node-modules“ und die vollständige „tailwind.css“ dort vorhanden. Um die Ordnergröße so klein wie möglich zu halten, habe ich mit „Vite“ einen Production-Build erstellt. Vite ist ein Entwicklungsserver, der effektives und schnelles Hot-Module-Replacement zur Verfügung stellt. Des Weiteren bietet Vite einen Build-Command, der unter anderem alle CSS-Dateien „minified“ bzw. den ganzen Code mit „Rollup“ zusammenfügt.

2.9 Webserver-Hinweise

Den Development-Ordner kann man mit Visual Studio Code und installiertem „Node.js“ starten. Dazu den Development-Ordner in Visual Studio Code öffnen und ein neues Terminal starten. Im Terminal dann „npm run dev“ eingeben.

Einfacher ist es den Production-Ordner in Visual Studio Code mit vorinstallierter „Live Server“-Extension von Ritwick Dey zu öffnen und den Live Server zu starten.

3. Quellen

3.1 Genutzte Assets

https://github.com/tuupola/lazyload
— Lazy Load Remastered von „tuupola”

https://www.w3schools.com/howto/howto_js_fullscreen_overlay.asp
— Fullscreen-Overlay-Navigation

https://march08.github.io/animated-burgers/
— Animated CSS Hamburger Icon von march08

https://tailblocks.cc/
— Tailblocks von mertJF

https://blog.marclucraft.co.uk/masonry-layout-with-tailwindcss
— Masonry Layout von Marc Lucraft

https://codepen.io/goodkatz/pen/LYPGxQz
— Simple CSS Waves von Goodkatz

3.2 Zusätzliches

https://youtube.com/playlist?list=PL5f_mz_zU5eXWYDXHUDOLBE0scnuJofO0
— Tailwind CSS Screencasts

https://youtu.be/puaX_nhTMRU
— Tailwind CSS Navbar Tutorial

https://tailwindcss.com/
— Tailwind CSS Documentation