# Nutzen der Webseite

Bei vielen Webseiten, bei denen es um Wanderungen geht, ist schwierig abzuschätzen wie es dort aussieht und man kann sich schlecht ein Bild von der Wanderung mache. Ein Grund dafür ist, man kann die verfügbaren Bilder nicht chronologisch einordnen. Es ist schwierig sich die Wanderung bildlich vorzustellen. Dieses Problem wir mit der Webseite gelöst, man kann genau sehen wie, es wo aussieht, aufgebaut ist es ziemlich ähnlich wie „Google Maps Street View“ nur halt für die Berge.

Durch die Webseite soll man eine bessere Übersicht über die Wanderroute bekommen, so dass man besser abschätzten kann, ob sie für einen passt.

# Zielgruppe

Die Zielgruppe befindet sich im alter von 18 bis 40 und sind tendenziell gerne draußen. Sicherlich gibt es auch älter Personen, für die die Informationen auf der Webseite nützlich sind, jedoch kann es sein das für älter Gruppen das User Interface kompliziert erscheint. Man kann sich durchaus vorstellen das die Knöpfe / Markierungen auf der Wanderrouten nicht verstanden werden. Wen die Knöpfe / Markierungen auf der Wanderrouten nicht angeklickt werden bekommt man deutlich weniger Informationen über die Wanderung als sonst.

# Programmierung

## HTML

### Struktur der HTML Seite

Die Webseite ist in 5 Bereiche eingeteilt.

   <!-- Startseite der Webseite -->

   <section id="part\_1">

   </section>

   <!-- Überblick der Wanderung -->

   <section id="part\_2">

   </section>

   <!-- Erster Tag -->

   <section id="part\_3">

   </section>

   <!-- Zweiter Tag -->

   <section id="part\_4">

   </section>

   <!-- Dritter Tag -->

   <section id="part\_5">

   </section>

*HTML Code 1.1*

1. Startseite der Webseite
2. Beschreibung der gesamten Wanderung
3. Beschreibung des ersten Tags
4. Beschreibung des zweiten Tags
5. Beschreibung des dritten Tags

In der HTML Struktur wird das in sogenannte „part’s“ eingeteilt (sieh HTML Code 1.1)

Diese „part’s“ sind auch dafür da, dass die Links in der Navigationbar wissen, wo hin sie springen müssen.

### Bilder

Da die Webseite die Wanderung möglichst grafisch darstellen soll werden viele Bilder verwendet. Insgesamt sind es 34 Bilder, die auf der Webseite dargestellt werden. Alle Bilder, außer die Routen Bilder, wurden selbst fotografiert und bearbeitet. Die Routen Bilder stammen von Google Maps.

Die Bilder haten anfangs eine Auflösung von 6000 x 4000 Pixel das ergab im Durchschnitt eine Gösse von 10 MB. Alle Bilder wurden verkleinert.

#### Bilder Auflösungen

Nach Bearbeitung haten alle Bilder diese Auflösungen:

* Titelbild:

Auflösung: 1280 x 720 Pixel

Grösse: 2.64 MB

* Routenbilder:

Auflösung: 1280 x 604 Pixel

Grösse: ca. 2.21 MB

* Bilder auf der Route:

Auflösung: 960 x 640 Pixel

Grösse: 1.76 MB

### Animieret Elemente

Auf der Webseite wurden Animationen teils mit CSS oder mit JavaScript erstellt. Die Anzeige der Bilder auf der Roadmap wurde mit Hilfe von JavaScript erstelle. Genau gesagt mit einer Bibliothek von JavaScript namens „jQuery“. Auf die genaue Funktion wird im Kapitel JavaScript eingegangen. Die Bibliothek kann auf zwei Wege installiert werden mithilfe von „npm“ („Node Paktet Manager“) oder so wie es in der Webseite gelöst ist, über ein API-Schnittstelle. (siehe HTML Code 1.2)

<!-- Einbinden von JQuery für die Bilder Animationen und „load data on scroll“-->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">

</script>

HTML Code 1.2

Über das CSS wurde die Navigationsbar animiert, die Animation besteht darin das sich die Navigationsbar in 0.3 Sekunden nach oben verschwinden wen nach oben gescrollt wird. Für den fließenden Übergang ist ein CSS-Befehl zuständig. (siehe HTML Code 1.3)

/\* Lässt Navigationsbar fließend verschwinden \*/

transition: top 0.3s;

HTML Code 1.3

Jedoch reicht diese Animation nicht das die Navigationsbar verschwinde zusätzlich nötig ist JavaScript, mehr in Kapitel JavaScript.

Zu den Animationen könnte man noch die „Hover“ Elemente dazuzählen, das sind:

* Navigationsbar
  + Wechselt die Farbe von Hellblau zu Dunkelbau. (siehe HTML Code 1.4)
* Sozial Media Icons
  + Wechselt die Farbe von Weiß zu Hellgrau und wird zusätzlich ein paar Pixel grösser. (siehe HTML Code 1.5)
* Pins / Markierungen auf der Route
  + Vergrößert sich und wechselt die Farbe von Dunkelblau zu Hellblau. (siehe HTML Code 1.6)
* Smooth Scrolling wen der Link in der Navigationsbar angeklickt wird
  + Wen der Link in der Navigationsbar angeklickt wird „slidet“ die Webseite zur richtigen Stelle (siehe HTML Code 1.7)

html {

  scroll-behavior: smooth;

}

HTML Code 1.7

.pin:hover {

  border-color: rgb(70, 164, 218); /\* Setzt die Farbe des Rahmens \*/

  height: 1.5vh; /\* Setzt die Höhe des Punktes \*/

  width: 1.5vh; /\* Setzt die Breite des Punktes \*/

  border-width: 0.35vh; /\* Setzt die Dicke des Rahmens \*/

  z-index: 1000; /\* Setzt die Position über einen andern Kreis \*/

}

HTML Code 1.6

/\* Vergrössert und wechsel die Farbe wen darübergefahren wird \*/

.icon:hover{

  font-size: 28px;

  color:#9b9b9b;

}

/\* Vergrössert und wechsel die Farbe wen darübergefahren wird \*/

#mail:hover{

  transform: scale(1.25);

  color:#9b9b9b;

}

HTML Code 1.5

/\* Verändert die Backgroundfarbe, wen die Maus darüber kommt \*/

nav ul a:hover, #btnDropdown:hover, #dropdownContent a:hover

{

  background-color: rgb(0, 60, 255);

}

HTML Code 1.4

## CSS (Konzeption, Umsetzung des Designs in CSS)

Aufbau aller Elemente (z.B. Boxen) graphisch zeigen (Schriften, Farben, float, Abstände in px oder em,...) und kurz erklären. Graphiken sollen z.B. aus Visio Dokument stammen.

#### Part 1

##### Idee

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

PC Handy

##### Umsetzung

#### Part 2

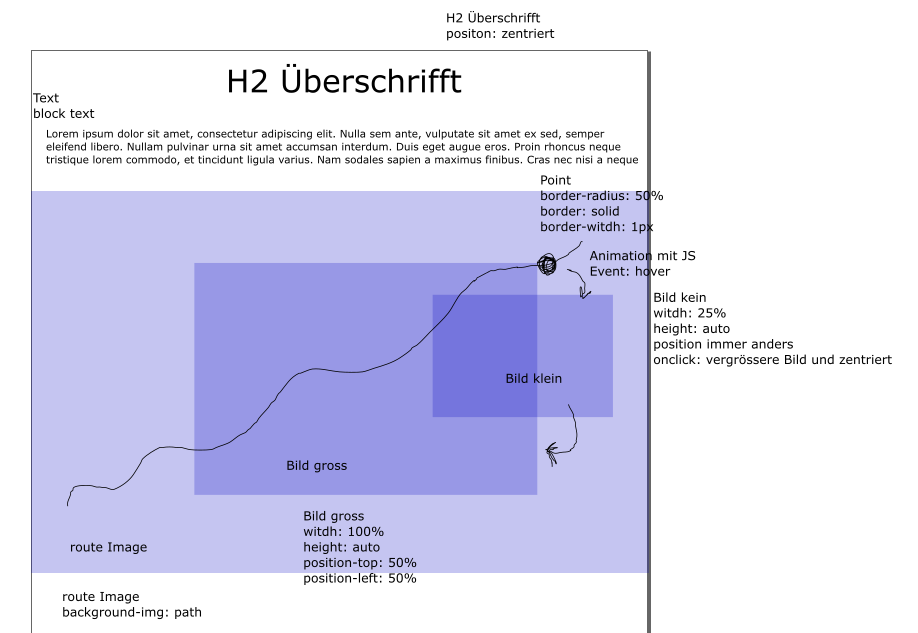
##### Idee

PC Handy

##### Umsetzung

#### Part 3

##### Idee



PC Handy

##### Umsetzung

#### Part 4

##### Idee

PC Handy

##### Umsetzung

#### Part 5

##### Idee

PC Handy

##### Umsetzung

fad

## JavaScript (Serlbs erarbeitet Themen und übernommene Elemente)

Neue, nicht im Kurs behandelte Elemente, die verwendet wurden, sollen erklärt werden (falls vorhanden)

Erklärung der Funktion von kopierten Elementen inkl. Link zur Source

### jQuery

jQuery ist eine JavaScript-Bibliothek, die Webentwicklern die Arbeit mit JavaScript erleichtern soll. Sie überbrückt technische Unterschiede zwischen verschiedenen Browsern und ermöglicht komfortablen und kompakten Code, der im direkten Vergleich zu JavaScript häufig sehr viel kürzer ist.

jQuery wird für das Verschwinden und Auftauchen der Bilder verwendet.

#### Bilder Animationen

Um die HTML Elemente besser anzusprechen und durch zu iterieren wurde ein Array erstellt mit allen HTML Elementen. (HTML Code 3.1)

// Holt das HTML Element

var ImagesDay1 = [

    document.getElementById("posOfImg1D1"),

    document.getElementById("posOfImg2D1"),

    document.getElementById("posOfImg3D1"),

    document.getElementById("posOfImg4D1"),

    document.getElementById("posOfImg5D1"),

    document.getElementById("posOfImg6D1"),

    document.getElementById("posOfImg7D1"),

    document.getElementById("posOfImg8D1"),

    document.getElementById("posOfImg9D1"),

    document.getElementById("posOfImg10D1")

]

HTML Code 3.1

Da die Funktion „zoomImageD1“ (HTML Code 3.3) die Position des Bildes verändert müssen die Standartwerdet der Positionen der Bilder abgespeichert werden. Jedes Mal, wenn ein Knopf / Markierung auf der Route darübergefahren wird setzt es bei allen Bildern die Standartwerte. (HTML Code 3.2)

// Default Werte der Positionen der Bilder

var defaultValuesOfImgDay1 = [

    ["20%", "70%"], //Position des "posOfImg1"

    ["45%", "70%"], //Position des "posOfImg2"

    ["52%", "65%"], //Position des "posOfImg3"

    ["50%", "62%"], //Position des "posOfImg4"

    ["46%", "55%"], //Position des "posOfImg5"

    ["25%", "42%"], //Position des "posOfImg6"

    ["22%", "38%"], //Position des "posOfImg7"

    ["22%", "36%"], //Position des "posOfImg8"

    ["20%", "33%"], //Position des "posOfImg9"

    ["75%", "20%"]  //Position des "posOfImg10"

]

HTML Code 3.2

Die Funktion „zoomImageD1“ ( HTML Code 3.3) ist zuständig dafür das das Bild wen es angeklickt wir in der Maximalen grösse angezeigt wird. Auch verändert es die Position des Bildes, das wiederum in der Funktion „showImageD1“ (HTML Code 3.4) rückgängig gemacht werden muss. Die Standardposition bekommt es vom Array „defaultValuesOfImgDay1“ (HTML Code 3.2)

// Wird aufgerufen wen auf das Bild gecklickt wird

function zoomImageD1(numberOfElement){

    ImagesDay1[numberOfElement].style.width = "100%";

    ImagesDay1[numberOfElement].style.top = "50%";

    ImagesDay1[numberOfElement].style.left = "50%";

    ImagesDay1[numberOfElement].style.zIndex = 1001;

}

HTML Code 3.3

Die mit Abstand größte Funktion ist die „showImageD1“ (HTML Code 3.4) sie ist dafür zuständig, dass die Bilder auf der Route langsam auftauchen und wieder langsam verschwinden. Das Funktioniert so:

Die Funktion wir aufgerufen wen über Knopf / Markierung gefahren wird. Das HTML übergibt der Funktion ein Parameter in Form einer Zahl, in der Variable „numberOfElement“. Diese Zahl bestimmt auf welchem Bild eine Animation stattfindet. Die Funktion iteriert jedes Bild, so kann viel Code gespart werden. Bei jedem Schleifen Durchgang wird überprüft, ob die For-Schiefe am richtigen Bild ist. Wenn es am richtigen Bild angekommen ist, kommt jQuery ins Spiel, jQuery frägt als erste ab, ob das HTML Dokument schon geladen ist, was meistens der Fall ist. Wen es geladen ist führt es die Funktion „.show“ aus, dies Funktion ist von jQuery geschrieben. Diese Funktion ist dafür zuständig das das Bild innerhalb von 0.6 Sekunden angezeigt wird. So wird das Bild angezeigt.

Wen aber die For-Schleife an eine Bild ist auf das nicht gefahren wurde, wird es mit Hilfe von jQuery verschwinden, aber nur, wie auch vorher, nur wen das HTML Dokument geladen wurde. Wen das Bild verschwinden ist bekommt es die Standartwerte der Position, wie schon erklärt.

// Funkton benutzt jQuery für die Bilder Animation

function showImageD1(numberOfElement){

    // Iteriert durch alle Bilder

    for(let i = 0; i < ImagesDay1.length; i++){

        if(i == numberOfElement){

            // JQerry animiert das Aufrufen der Bilder dauert 600ms

            $(document).ready(function(){

                $(ImagesDay1[i]).show(600);

            });

        }

        else{ // Alle Elemente, die nicht berührt werden

            // Alle Elemente die noch Angezeigt werden

            if(ImagesDay1[i].style.display == "inline" ||

            ImagesDay1[i].style.display == "inline-block"){

                // JQerry animiert, das Verschwinden der Bilder dauert 600 ms

                $(document).ready(function(){

                    $(ImagesDay1[i]).hide(600, function(){

                        // Stellt den Default-Wert ein nach dem Verschwinden der

                        // Bilder

                        ImagesDay1[i].style.width = "25%";

                        ImagesDay1[i].style.zIndex = 0;

                        ImagesDay1[i].style.top = defaultValuesOfImgDay1[i][0]

                        ImagesDay1[i].style.left = defaultValuesOfImgDay1[i][1]

                    });});}}}}

HTML Code 3.4

#### Autoload nach Scrollen

Nicht nur die Bilder Animationen brauchen jQuery auch das automatisch nachladen der Bilder wird mit jQuery realisiert. Das Automatisch nachladen der Bilder wurde hinzugefügt um den „Lade-Traffic“ beim Aufrufen der Webseite zu minimieren.

jQuery frägt als erste ab ob auf der Webseite gescrollt wird wen das der Fall ist, führ jQuery alle HTML-Tag’s mit „img“ eine Funktion aus. Wen das HTML-Tag auch noch das Attribut „data-src“ hat und in der nähe des unter Bildschirm Rands ist wird es geladen. Das Laden der Bilder funktioniert so, jQuery holst sich die Adresse die im „data-src“ vorhanden ist Speicher sie ab und fügt sie in das Attribut „src“ hinein. Sobald das HTML in dem Attribut „src“ eine Adresse zu einem Bild hat, lädt es das Bild in die Webseite hinein. Schlussendlich wird das „data-src“ Attribut gelöscht.

// Wen gescrolt wird

$(window).scroll(function() {

    // Holt sich alle HTML-Tag's mit 'img'

    $.each($('img'), function() {

        // Alle img werden beim Ersten scrollen geladen

        if ( $(this).attr('data-src') &&

        $(this).offset().top < ($(window).scrollTop() + $(window).height()) ) {

            // Holt sich die gespeicherte Source von 'data-src',

            // in 'data-src' liegt der Link zum Bild

            var source = $(this).data('src');

            // Überschreibt did leeren HTML-Tag's 'src' mit den 'data-src' Tag's

            $(this).attr('src', source);

            // Löscht die 'data-src' Tag's

            $(this).removeAttr('data-src');

        }

    })

})

HTML Code 3.3

### Navigationsbar

# Ladezeit Analyse

# Kopieret und Übernommene Elemente

// link: https://stackoverflow.com/questions/5117421/how-to-load-images-dynamically-or-lazily-when-users-scrolls-them-into-view