



# Revitalize.

**CVD Webentwicklung 2019**  
**Softwaredokumentation**

Simon Weck #2180135  
Benjamin Jäger #2180031  
Patryk Watola #2180138  
Clemens Bork

# Aufgabenstellung

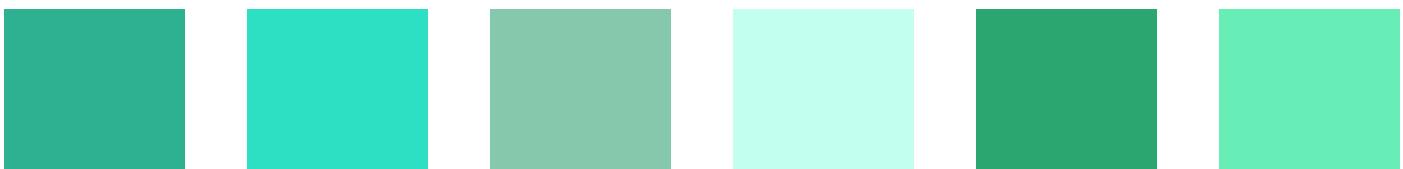
Die im Modul „Informatik III und Design III“ gestellte Aufgabe im dritten Semester des Studiengangs „Computervisualistik und Design“ war es, ein Logo sowie eine Website für eine Marke im Bereich „Sport“ zu entwickeln.

Anforderungen an diese Website waren eine Startseite, vier Kategorieseiten, vier Artikelseiten und eine Registrierungsseite. Dessenweiteren sollten Daten Client-side gespeichert werden und die Registrierungsformulare sollten den UI design patterns aus der Vorlesung entsprechen.

## Unsere Marke

Unsere selbst entwickelte Marke heißt Revitalize und vertreibt natürliche Supplements, für alle Sportler. Im Zentrum steht Natürlichkeit, Ehrlichkeit, Hochwertigkeit sowie Persönlichkeit. Wir möchten den Markt durch tatsächlich natürliche Produkte erobern, die nicht nur hochwertig produziert wurden, sondern auch fair gegenüber der Umwelt sind. Diese Thematik wollen wir auf unsere Website übertragen.

### Corporate Farben



### Corporate Logo



Revitalize.



### Corporate Typografie

Überschrift

**Open Sans**

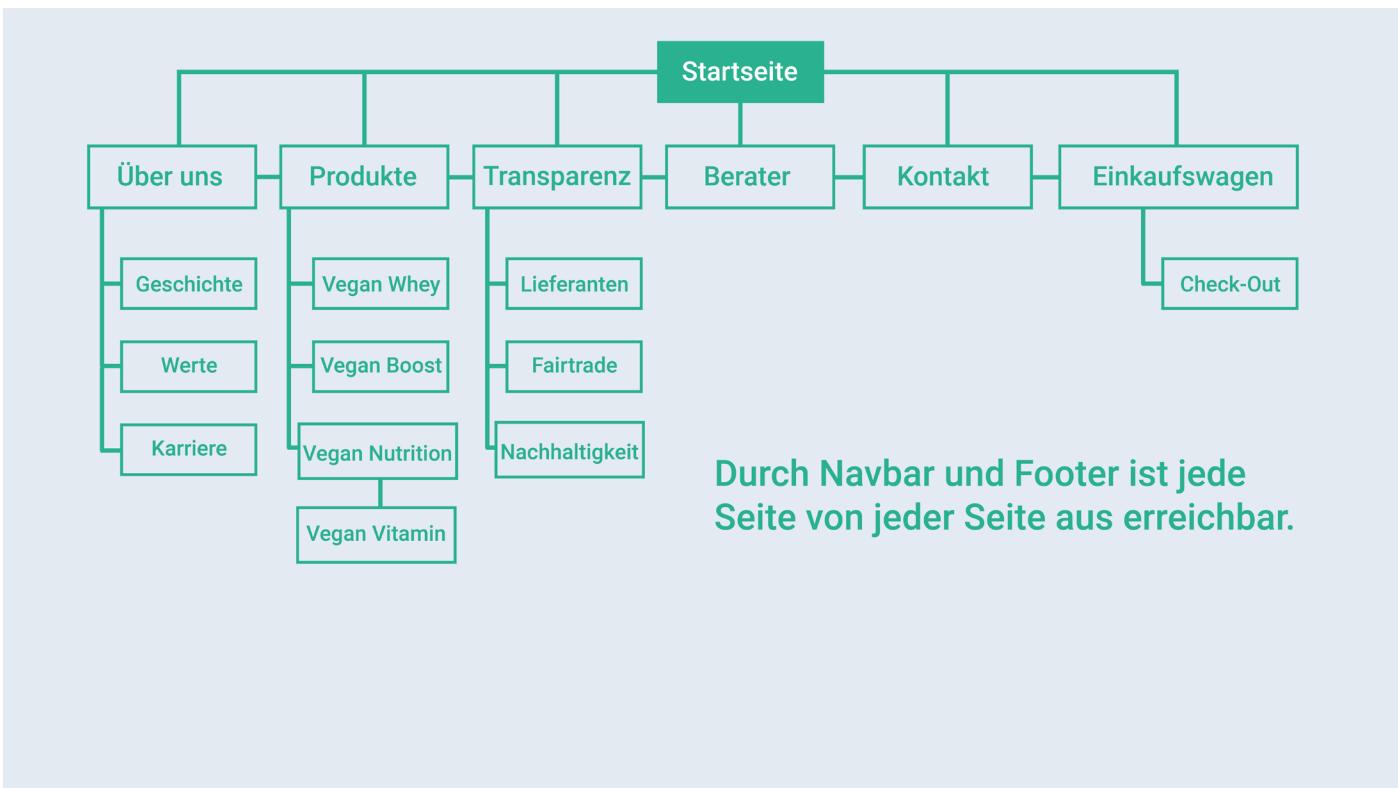
Fließtext

Open Sans

# Inhalt

- 4 Sitemap / Informationsarchitektur
- 6 Welleneffekt
- 9 Navigation Bar
- 15 Startseite
- 24 Produktseiten
- 28 Artikelseiten
- 29 Kontaktseite
- 31 Kategorieseiten
- 36 Revitalize Berater
- 39 Warenkorb
- 43 Anteile der Gruppenmitglieder
- 44 Quellen

# Sitemap / Informationsarchitektur



Generell sind alle Seiten jederzeit von der aktuellen Seite erreichbar. Im Footer ist jede Seite noch einmal einzeln verlinkt und mittels Navigation Bar können wichtige Seiten jederzeit erreicht werden. Auf unserer Seite finden Sie 3 Kategorie Seiten, 6 Artikel Seiten, 4 Produkt Seiten, 2 Registrierungs Seiten sowie eine Kontakt Seite und den Revitali ze Berater.

## Navigation Bar



## Footer

 Revitalize.	Über uns Geschichte Werte Karriere	Transparenz Lieferanten Fairtrade Nachhaltigkeit	Produkte Vegan Whey Vegan Boost Vegan Nutrition Vegan Vitamin	Einkaufswagen Check-Out	Berater	Kontakt
---	---	---	---	----------------------------	---------	---------



## **Artikelseiten**

- Geschichte
- Werte
- Karriere
- Lieferanten
- Fairtrade
- Nachhaltigkeit
- Interview Artikel

## **Kategorieseiten**

- Über uns
- Transparenz
- Produkte

## **Produktseiten**

- Vegan Whey
- Vegan Boost
- Vegan Nutrition
- Vegan Vitamin

## **Registrierungsseiten**

- Checkout
- Kontakt

## **Sonstige Seiten**

- Revitalize Berater
- Warenkorb
- Impressum
- Startseite

# Welleneffekt

Für viele Seiten haben wir einen Welleneffekt genutzt, um die Seiten natürlicher wirken zu lassen und die Seiten etwas dynamischer zu gestalten. Die Idee war es dabei Bilder mit einer Sinuswelle anzuschneiden, damit die Bilder nicht zu eckig wirken. Später sind wir dann auf die Idee gekommen, diese Wellen auch zu animieren, damit die Seite etwas mehr Bewegung bekommt. Dabei unterscheiden wir zwischen 2 Arten von Wellen. Einmal eine große Welle, die über die gesamte Bildschirmbreite verläuft und eine Welle, die über einem Bild liegt.



Für diesen Welleneffekt benutzen wir, wie bereits erwähnt, eine Sinuswelle. Diese Welle wird zudem jedes Mal pseudo zufällig generiert. Wir verändern Form und Farbe der Welle jedes Mal basierend auf zufälligen Zahlen. Bei der Farbe nutzen wir eine leichte Abweichung von unserer CD Farbe, sowie eine zufällige Transparenz, in einem Intervall zwischen 0.3 bis 0.8.

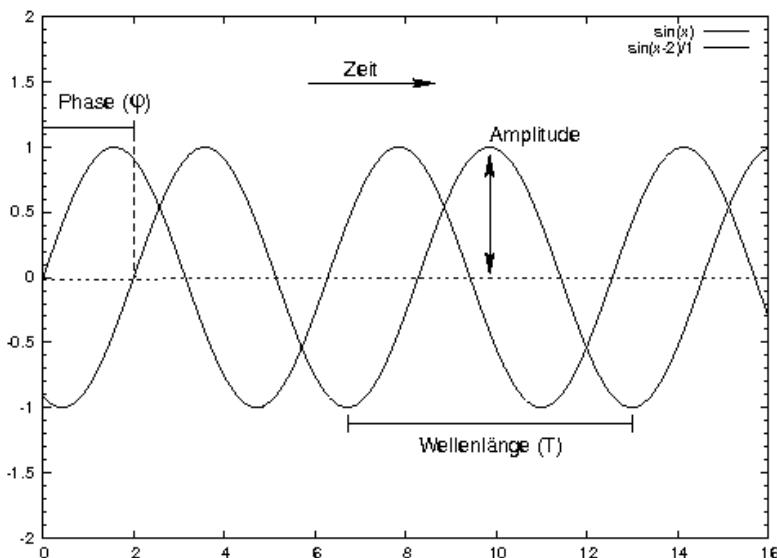
```
//set up random rgba colors based on the Coporate color (61,222,154)
var colorOffset = 0.7; //ammount of deviation from the original color (0=max deviation, 1= no deviation)
var minTransparency = 0.3; //min value for transparency
var maxTransparency = 0.8; //max value for transparency

var a = minTransparency+Math.random()*(maxTransparency-minTransparency);
var r = Math.floor(61 * (colorOffset+Math.random()*(1-colorOffset)));
var g = Math.floor(222 * (colorOffset+Math.random()*(1-colorOffset)));
var b = Math.floor(154 * (colorOffset+Math.random()*(1-colorOffset)));
```

Für die Form der Welle, passen wir Frequenz sowie Amplitude (auch basierend auf zufälligen Werten) an um, die Sinuswelle zu verändern. Dazu wählen wir eine zufällige Phase und Geschwindigkeit, damit die Wellen noch unterschiedlicher wirken.

```
//zufällige amplitude, Frequenz, phase, geschwindigkeit sowie Farbe um zufällige Wellen zu generieren
var resolution = 300; //ammount of lines
var amplitude = 4+Math.random()*(25-4); //height of function
var frequency = 1.2+Math.random()*(1.5-1.2); //width of function
var speed = 0.001+Math.random()*(0.005-0.001); //speed of the wave
var yOffset = 3/8*picture.height;
var step = Math.random()*50;
```

Die Variable Step gibt die aktuelle Phase an und wird im späteren Verlauf um die Variable speed erhöht, damit es so aussieht, als ob sich die Welle bewegt. Hier noch ein Diagramm das beschreibt, was genau wir meinen, wenn wir die Form der Sinuswelle mittels Phase, Amplitude und Frequenz verändern.



```
for(var i = 0; i<resolution ; i++){
    y = Math.sin(i*2*Math.PI/resolution*frequency+step)*amplitude+canvas.height/2+yOffset;
    x = i/resolution*canvas.width;
    ctx.lineTo(x, y);
}
```

Diese Schleife erledigt die meiste Arbeit. Sie passt die Sinus Funktion mit den entsprechenden Werten an und bestimmt x sowie y Werte basierend auf der Höhe und Breite des Canvas an. Es Werden immer Punkte berechnet, die mit Linien verbunden werden. Die Auflösung dieser Kurve, also die Anzahl an Punkten, kann mittels der Variable resolution gesteuert werden, damit die Kurve möglichst glatt aussieht. Vor dieser Schleife müssen wir lediglich die Variable step um die Geschwindigkeit erhöhen sowie das Canvas einmal aufräumen (ein weißes Viereck darüber zeichnen). Diese gesamte Funktion wird dann alle 17 ms aufgerufen, um eine flüssige Bewegung zu simulieren.

## Die Große Welle

Um eine große Welle nun auf eine Seite zu implementieren, müssen wir lediglich ein Canvas Objekt bereitstellen und die Funktion addBigWave(canvasID) ausführen. Die Funktion nimmt sich dann die Id des Canvas Objektes und zeichnet darauf. Die Größe des Canvas müssen wir dabei nicht angeben, denn die Funktion stellt Größe und Breite automatisch ein, damit alle Canvas Objekte konsistent dieselbe Größe haben. Die große Welle funktioniert genau so wie vorhin beschrieben, nur sie wird noch einmal gespiegelt auf dem Canvas gezeichnet, damit wir eine gesamte Welle haben. Stapeln wir dann mehrere Wellen übereinander, erhalten wir die finale Welle.



## Die kleine Welle

Die kleine Welle, die über einzelnen Bildern liegt, funktioniert fast genauso wie die große Welle, nur das diese nicht noch einmal gespiegelt wird. Außerdem braucht die Methode zusätzlich die ID des Bildes, auf dem die Welle liegen soll. Basierend auf dem Bild, wird dann die Größe des Canvas Objektes angepasst, damit die Welle nicht aus dem Bild herausragt. Wir können bei der Funktion zudem angeben, ob die Welle weiß sein soll, damit die Illusion eines richtig angeschnittenen Bildes entsteht.



# Die Navigation Bar

## Grundaufbau

Die Navigation Bar ist ein zentrales Element einer jeden Website. Diese ist als zentrales Element der Informationsarchitektur zu sehen. Auf den ersten Blick macht eine Navigation Bar den Grundaufbau einer Website klar und erlaubt es dem Nutzer ohne eine große Einführung die Seite bedienen zu können - sofern die Strukturierung sinnvoll ist.

Wir entschieden uns bei unserer Website für eine Leiste mit möglichst wenig Navlinks. Thema unserer Firma sollte auch Transparenz sein und eine kleine Navbar deutet eher darauf hin, dass die Firma nichts verbergen möchte. So können sich auf den ersten Blick weniger ungewollte Informationen auf irgendwelchen Unterseiten verstecken - alles wird deutlich kundenfreundlicher orientiert. Auch wichtig war uns ein möglichst cleanes Design. Nicht zu viele Icons, alles möglichst schlicht und simpel strukturiert. Diese Entscheidung trafen wir in Verbindung mit der Entscheidung, wie der Header auf der Startseite insgesamt aussehen sollte. Die Navbar und eben dieser Header wären in der Theorie das erste, was ein jeder potenzieller Neukunde als erstes sehen würde, sofern sich für einen Blick auf unsere Seite entschieden werden würde. Letztenendes folgten wir hier der Devise „weniger ist mehr“ und bauten die Navbar dem Rest des Design entsprechend simpel strukturiert auf.

Die Navigation Bar besteht also aus den vier Navlinks „Über uns“, „Produkte“, „Berater“ und „Kontakt“. Auch die Punkte selbst wählten wir nach dem Transparenz-Motiv. „Über uns“ sollte allen voran deswegen als erstes zu sehen sein. Ohne auch nur lange Zeit auf der Startseite verbringen zu müssen, kann der Nutzer sofort nähere Informationen zu der Firma erhalten. Um den reinen Verkauf von Produkten soll es nach außen hin zwar nur sekundär gehen, jedoch ist das immer noch das Kerngeschäft der von uns erdachten Firma. Dementsprechend steht an zweiter Stelle der Link zu der Produktübersicht, sodass sich der Kunde ebenso schnell einen Eindruck über das Angebot der Firma machen kann. Dahinter folgt das Logo in zentraler Position. Die Marke soll dem Benutzer im Gedächtnis bleiben, weswegen das Logo und Herzstück der Corporate Identity in der Mitte steht. Direkt daneben befindet sich der „Revitalize Berater“, der basierend auf ein paar wenigen Eingaben dem Kunden Informationen zu Ernährungsmöglichkeiten und möglichen Produkten der Firma für seinen Lebensstil empfiehlt. Dieser erfüllt zweierlei Zwecke: einerseits besteht durch die gegebenen Tipps ein direkter Mehrwert für den Kunden, andererseits verfolgt ein solcher Service durch die Produkttipps ein monetäres Interesse. Weil dieser Service beide Funktionen erfüllt, empfanden wir die Wichtigkeit groß genug für einen prominenten Platz in der Navbar. Zuletzt folgt die Kontaktseite. Diese fanden wir vor allem deswegen so wichtig, weil in einem transparenten Unternehmen jederzeit die Möglichkeit zur Kontaktaufnahme bestehen sollte. Deswegen wollten wir diese Option ebenso präsent in der Navbar zeigen.

Technisch finden sich im HTML-Teil der Navigation Bar zwei Hauptbestandteile. Die Navlinks und das Logo befinden sich innerhalb einer Unordered List, welches sämtliches HTML-Standard-Styling innerhalb des CSS entzogen wird.

```

<ul>
    <li><a href="Über uns.html" class = "navlinks">Über uns</a>
    <li><a href="Produktübersicht.html" class = "navlinks">Produkte</a></li>
    <li><a href="index.html"></a></li>
    <li><a href="Revitalize Berater.html" class = "navlinks">Berater</a></li>
    <li><a href="contact.html" class = "navlinks">Kontakt</a></li>
</ul>

```

Die Unordered List enthält fünf Links, da jeder Bestandteil dieser Liste zu einer weiteren Seite führt. Die Navlinks führen logischerweise zu den Seiten, die sie beschreiben, und das Logo bringt den Nutzer mit einem Klick zurück zu der Startseite der Website. Nicht nur rein aus ästhetischen Gründen ist das Logo in der Navbar (und auch sonst immer, wenn es benutzt wird) eine SVG-Datei. Dies ist unter anderem auch der Performance zuträglich. Ebenso der Performance zuträglich ist das Einstellen der Breite des Bildes innerhalb des HTML-Tags. Dies sorgt wiederum für ein schnelleres Laden der Seite.

Die Navbar befindet sich am oberen Bildschirmrand und verlässt diesen auch nicht, wenn der Nutzer herunterschrollt. Dafür verwendeten wir „fixed“ für die „position“-Eigenschaft des Hauptcontainers der Navbar, der alles der Navigation zugehörige enthält.

```

nav {
    position: fixed;
    width: 100%;
    font-size: 1.3vw;
    z-index: 15;
    text-align: center;
    transition: background-color 0.4s ease-out;
}

```

Wie bereits erwähnt, verlieren alle Listenelemente ihr Standard-Styling. Dies ist auch notwendig, da diese sonst mit einem Punkt vor jedem Navlink auf der Seite erscheinen würden. Zusätzlich wird den Links auch das Styling des Anchor-Tags entzogen und die Farbe bearbeitet. Ansonsten wären diese blau, unterstrichen und würden Ihre Farbe in ein violett verändern, sofern der Nutzer bereits auf die den Link geklickt hat.

```

nav a {
    color: #43DEB9;
    text-decoration: none;
}

```

```

nav ul{
    padding: 0;
    list-style: none;
    margin: 0 auto;
}

```

Der zweite Hauptbestandteil der Navbar sind die Absolut positionierten Warenkorb-Elemente am rechten Bildschirmrand.

```
<div class="amount-of-orders">
|   <p id="WarenkorbAnzahl"></p>
</div>

<script src="Ressources/JavaScript/Navigation Bar/Warenkorb_Anzahl.js"></script>

<a href="Warenkorb.html"></a>
<a href="Warenkorb.html"></a>
```

Im HTML wird zunächst ein Div-Container für die Anzeige der aktuell eingekauften Elemente hinzugefügt. Darin befindet sich ein p-Tag, das später im JavaScript um die aktuelle Anzahl an im Warenkorb befindlicher Produkte verändert wird. Darunter wird das Script eingebunden, was sich um das Handling der Cookies zur Speicherung von Produkten im Warenkorb kümmert. Mehr dazu befindet sich im Kapitel zum Thema „Warenkorb“.

Nun folgen zwei Image-Tags die beide jeweils eine SVG-Datei eines Einkaufswagens enthalten.

Die Navbar verändert nämlich ihre Hintergrundfarbe sobald runterscrollt wird. Es geht von der Transparenz fließend in den Standard-Grün/Blauton der Website über. Dementsprechend müssen alle Navlinks und Elemente innerhalb der Navbar ihre Farbe verändern können. Der Warenkorb ist eine SVG aus zusammengesetzten Farben, weswegen sich die Umfärbung per CSS sehr schwierig gestalten würde.

Dementsprechend wird im Falle des Runterscrollens einfach der grüne Warenkorb durch einen Weißen ausgetauscht.

## Der Scroll-Effekt

Nun beschäftigen wir uns im Detail mit dem Farbwechsel der Navbar.

Um diesen zu realisieren, hat es etwas CSS in Kombination mit JavaScript gebraucht. Zunächst ist bereits in dem oben angegebenen Screenshot zum Styling des nav-Tags im CSS zu sehen, dass ein Transition-Value festgelegt ist. Dieser sorgt dafür, dass der Übergang vom transparenten Hintergrund der Navbar hin zum Grünen binnen 0.1s mittels einer Überblende passiert.

Um den Farbwechsel des Hintergrunds und der Schrift (diese muss weiß werden, da sie sonst auf dem Hintergrund zu schwer zu lesen wäre) umzusetzen, werden zwei zusätzliche Klassen im CSS verwendet.

```
nav.scroll {
    background-color: #43DEB9;
}

.scroller {
    color: white;
}
```

Diese sehr kurzen und grundsätzlich simplen Klassen werden per JavaScript immer dann zugewiesen, wenn der Nutzer um mehr als zehn Einheiten nach unten gescrolled hat. Die Klasse „scroll“ gibt dem Hintergrund den Grünton und die Klasse „scroller“ färbt die Navlinks weiß ein.

Doch was ist nun mit dem Warenkorb und der Anzahl an Inhalten in selbigem?

Ähnlich dem Hintergrund und den Navlinks liegen dem ganzen mehrere CSS-Klassen zugrunde. Eine, welche die Farbe des Kreises verändert, in welchem sich die Anzahl an Produkten im Warenkorb befindet, und eine, welche die Farbe der Nummer selbst anpasst. Neben diesen Klassen gibt es zusätzlich noch die Klasse „nonactive“. Diese ist für den Warenkorb relevant und kann nichts anderes, als den „display“-Wert des Elements auf „none“ zu setzen und ihn damit quasi von der Seite entfernt solang die Klasse zugewiesen ist.

```
.amount-color-change {  
    background-color: white;  
}  
.amount-color-change p {  
    color: #43DEB9;  
}  
  
.nonactive {  
    display: none;  
}
```

Wie schon häufiger erwähnt, passiert die eigentliche Veränderung im JavaScript-Script der Navigation Bar. Mittels des „window.onScroll“-Events wird der Moment abgefangen, ab welchem der Nutzer nach oben oder unten scrollt. Darauf folgt ein Code-Block, in welchem mittels einer zentralen if-else-Abfrage das Styling der Elemente nach und vor dem Scrollen definiert wird.

```
window.onscroll = () => {  
    const nav = document.querySelector( selectors: 'nav');  
  
    let navlinks = document.getElementsByClassName( classNames: 'navlinks');  
    let shoppingCart = document.getElementsByClassName( classNames: 'shopping-cart-icon');  
    let amount = document.getElementsByClassName( classNames: "amount-of-orders");
```

Nachdem das Scrollen erkannt worden ist, werden zunächst Variablen erstellt. Diese nutzen das DOM um Zugriff auf die relevanten Objekte der Seite zu erlangen. In drei verschiedenen Variablen werden nun einmal alle Navlinks, die beiden Icons (das grüne und das weiße) für den Warenkorb und der Zähler der Produkte im Warenkorb abgespeichert. Diese sind zur weiteren Verarbeitung essentiell. Über diese Variablen werden im folgenden die bereits thematisierten CSS-Klassen hinzugefügt oder aber entfernt.

Nun folgt die if-else-Abfrage, die je nach „Scrollhöhe“ die Klassen zuweist oder aber entfernt.

```

if (this.scrollY <= 10)
{
    nav.className = '';

    shoppingCart[1].classList.remove( tokens: "nonactive");
    shoppingCart[0].classList.add("nonactive");

    amount[0].classList.remove ( tokens: "amount-color-change");

    for (let i = 0; i < navlinks.length; i++)
    {
        navlinks[i].classList.remove( tokens: "scroller");
    }
}

```

Sofern der Nutzer noch nicht über zehn Einheiten nach unten gescrolled hat, verliert die Navbar die „scroll“-Klasse und wird wieder transparent. Für den Einkaufswagen passiert etwas anderes: Wie bereits erwähnt passiert der Farbwechsel beim Warenkorb durch das Austauschen des grünen Icons durch ein Weißes. Beide Icons haben im HTML dieselbe Klasse bekommen, was nicht nur dafür sorgt, dass beide Icons durch denselben Code im CSS übereinanderliegen, sie werden auch im JavaScript in der Variable in einer Liste abgespeichert. Der weiße Warenkorb hat dort den Index 0 und der Grüne den Index 1.

Nun wird die bereits erwähnte „.nonactive“-Klasse aus dem CSS-Teil relevant. In dem if-Teil der Abfrage passiert nämlich nichts anderes, als dass dem grünen Warenkorb diese Klasse entzogen und dem Weißen hinzugefügt wird. So wird das grüne Icon sichtbar und das Weiße unsichtbar.

Darunter wird der Anzahlsanzeige die Klasse entzogen, die den Hintergrund weiß und die Nummer grün einfärbt.

Zuletzt findet sich eine for-Loop. Diese geht die Liste aller Navlinks durch und entfernt die Klasse „.scroller“ von jedem. Die Klasse färbt die Navlinks weiß ein und durch das Entfernen bekommen alle Links in der Navbar wieder ihre ursprüngliche Farbe - grün.

```

else
{
    nav.className = 'scroll';

    shoppingCart[0].classList.remove( tokens: "nonactive");
    shoppingCart[1].classList.add("nonactive");

    amount[0].classList.add ("amount-color-change");

    for (let i = 0; i < navlinks.length; i++)
    {
        navlinks[i].classList.add("scroller");
    }
}

```

Im else-Teil der Abfrage werden alle Klassenänderungen durchgeführt, die der if-Teil negiert, weil dieser erst greift, wenn der Nutzer über zehn Einheiten gescrolled hat. Das nav-Element bekommt die Klasse „scroll“ und damit färbt sich der Hintergrund grün ein.

Dem weißen Warenkorb wird die .nonactive-Klasse entzogen und er wird sichtbar, während der grüne diese zugewiesen bekommt und unsichtbar wird.

Die Anzeige über die Anzahl an Produkten im Warenkorb bekommt die „amount-color-change“-Klasse zugewiesen. So wird der Kreis weiß und die Nummer grün.

Die for-Loop geht durch alle Navlinks durch, weist ihnen die „scroller“-Klasse zu und sie werden weiß.

## Die Navbar ohne die Scroll-Effekte



## Die Navbar ohne die Scroll-Effekte



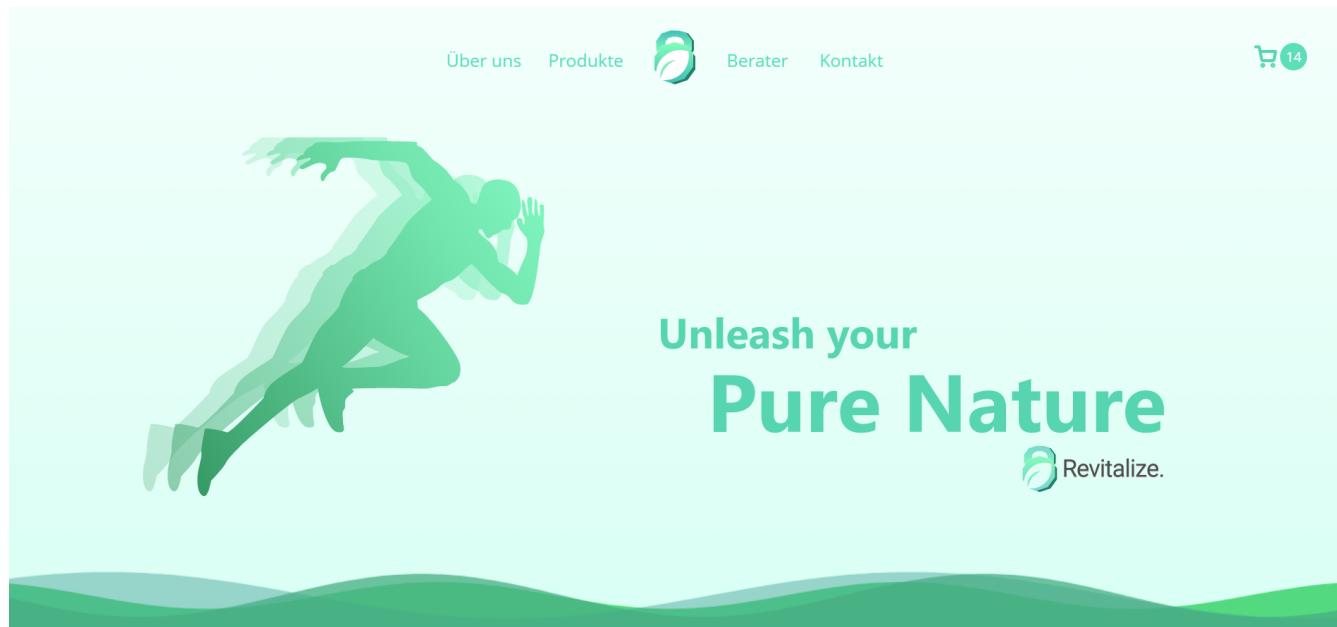
# Die Startseite

Die erste Seite, die ein jeder neuer Nutzer als erstes sehen wird, ist die Startseite (bzw. index.html). Wie im Kapitel zur Navigation Bar schon kurz angeschnitten, entstand das grundlegende Design dieser zusammen mit der Navigation. Grundlegende Voraussetzungen für die Optik waren also ein möglichst cleanes Design, eine aufgeräumte Optik und nur so viel Content wie nötig auf einmal, sodass sich kein Nutzer überfordert fühlt.

Im Folgenden wird genauer auf den technischen Aufbau der Seite eingegangen.

## Der Header

Der Header der Revitalize-Website besteht aus vier essenziellen Komponenten. Einmal der grünen Figur, die eine Laufbewegung durchführt, den Werbeslogan der Firma „Unleash Your Pure Nature“ in grüner Schrift rechts daneben und dem Logo darunter, dem Farbverlauf im Hintergrund und den animierten Wellen im Vordergrund.



Die Figur und die Schrift daneben teilen sich eine Eigenschaft: Beide sind SVG-Dateien. Zu Anfang bestand noch gewisse Unsicherheit darüber, ob die Schrift ebenso als SVG eingebunden werden sollte, da der Text ebenso einfach per HTML und CSS hätte erstellt werden können. Schlussendlich fiel die Entscheidung doch auf eine SVG. So war nie ein anderes Format als SVG für den Schriftzug vorgesehen, was auf Webseiten generell performancetechnisch kaum Probleme macht. Deshalb blieb die Schrift in der nun vorliegenden Version.

Die Schrift und die Figur links daneben sind mit wenigen Zeilen CSS nebeneinander positioniert und der Abstand zwischen ihnen erhöht worden. Hierzu wird die linke Margin des Schriftzugs erhöht.

Die beiden Elemente befinden sich innerhalb eines eigenen Div-Elements („hero-container“) und dies wiederum befindet sich in einem weiteren Div-Element („hero-gradient“).

```

<div class = "hero-gradient">
    <div class="hero-container">
        
        
    </div>

    <!--Die animierten Wellen-->
    <canvas class="hero-wellen" id="myCanvas1"></canvas>
    <script>addBigWave(myCanvas1, false);</script>
    <canvas class="hero-wellen" id="myCanvas2"></canvas>
    <script>addBigWave(myCanvas2);</script>
    <canvas class="hero-wellen" id="myCanvas3"></canvas>
    <script>addBigWave(myCanvas3);</script>
</div>

```

Diese Aufteilung macht das erstellen des Hintergrunds sehr viel einfacher. So sollte sich der Hero-Teil der Website allen voran durch einen weichen Farbverlauf auszeichnen. Dieser ist in Adobe XD skizziert und optimiert worden. So konnten schlussendlich die Hexcodes einfach im CSS der Eigenschaft „background-image“ zugewiesen werden. Diese Eigenschaft gehört zum Container „.hero-gradient“, der neben seinem charakteristischem Verlauf zusätzlich auch noch eine Größe in Prozentzahlen zugewiesen bekommt. Zu jeder Sekunde soll sich der gesamte Hero-Bereich über die Breite des Bildschirms erstrecken, weswegen die Breite bei 100% liegt und die Höhe automatisch mitskaliert. So ist zu gewährleisten, dass sich der Header auch beim Kleinziehen des Fensters nicht verzieht. Der „.hero-container“ skaliert ebenso mit.

Quasi über dem Header an und für sich befinden sich die Wellen. Auf diese wird im folgenden Kapitel genauer eingegangen. An dieser Stelle ist erstmals relevant, dass diese auf voller Breite und „absolute“ positioniert unterhalb des „.hero-containers“ liegen.

## Die „Was ist Revitalize?“-Sektion

Auch lange zur Diskussion stand die Frage, was ein neuer Kunde als erstes sehen sollte, wenn er nach unten scrollt. Einerseits wäre ein Ansatz gewesen, diesen sofort mit Produkten der Firma zu konfrontieren um einen schnellen Verkauf zu gewährleisten. Schlussendlich fiel jedoch auch diese Entscheidung nach demselben Schema wie jede andere: erst Transparenz/Kundennähe und dann der eigentliche Verkauf. Dementsprechend sollte direkt unter dem Header zunächst eine Anzeige erscheinen, die dem Nutzer die beiden wichtigsten Grundwerte der Firma (Fortschritt im Sport und Nachhaltigkeit) näher bringen sollten.

## Was ist Revitalize?



### Supplements - keine Chemiekeulen

Revitalize bietet dir gesunde Supplements, die nicht aus dem Reagenzglas kommen.  
Natürliche Hilfe für natürliche Körper.

### Umweltschutz bedingt die Sache

Als Quelle aller unserer Produkte dient die Natur - natürlich liegt uns der Schutz dieser dann sehr nah.

Jedes Produkt ist so klimaneutral wie möglich produziert worden und so wird es auch immer bleiben.



Die Grundidee war es also, dass auf einer Seite ein Bild mit einem Text-Overlay zu finden ist und auf der anderen Seite ein Text mit einer rechtsbündig ausgerichteten Überschrift.

Die Bilder sind PNG-Dateien, die in Photoshop so lange verkleinert worden sind, bis der optimale Kompromiss aus Qualität und Größe resultierte. So sind die Bilder weit kleiner als die Originale und damit performanceschonender. Leider war ein Umstieg auf JPG nicht möglich, da die Bilder durch den Wellenanschnitt Transparenz haben be halten müssen.

Die Umsetzung der Anzeige selbst steht und fällt mit der HTML-Struktur. Diese sieht auf den ersten Blick etwas unübersichtlich aus, verfolgt jedoch ein klares Konzept. Bevor genauer auf die Anzeige eingegangen wird, sei auf eine spezielle Besonderheit eingegangen. Die Seite benutzt auch semantische Tags, die einen direkten Vorteil haben: Jeder Content unterhalb des Headers (mit Ausnahme des Footers) kann mittels „main“ als Selektor im CSS angesprochen und mit einer Erhöhung der oberen Margin nach unten verschoben werden. Dies ist insbesondere deswegen so praktisch, weil die Wellen am unteren Rand des Headers absolut positioniert sind und deswegen einen Teil des eigentlichen Contents der Seite verdecken würden.

Ein weiterer Vorteil besteht darin, dass alle Seiten einen konsistenten Rahmen haben können. Jede Seite, auch die Startseite, bindet die sogenannte „Standard CSS“-Datei ein, welche die Breite des main-Bereichs auf 90% festgelegt und eben diesen zentriert. Der Abstand eines jeden Contents zum Rand ist also immer jeweils fünf Prozent der Gesamtbreite des Viewports. Auch beim Skalieren der Website.



Die Anzeige selbst besteht insgesamt aus zwei Reihen. Der Reihe mit einem Bild auf der linken und Text auf der rechten Seite und der Reihe, die mit einem Text auf der linken und einem Bild auf der rechten Seite aufwartet. Dazu kommt noch die Überschrift über der ersten Reihe. Innerhalb des „section“-Tags, das erst einmal alles zu dieser Anzeige hält, findet sich als erstes die Überschrift, die zentriert wird. Darunter finden sich die beiden div-Container „first-row“ und „second-row“, die jeglichen Content zu den beiden Reihen enthalten. Das wären für beide Reihen je ein Bild, ein Container mit dem Text, der über den Bildern steht, und ein Container mit dem Text, welcher sich neben den Bildern befindet. Einzig und allein die Anordnung der Komponenten unterscheidet sich. Bild und Text werden innerhalb ihres Div-Containers (also „first-row“ oder „second-row“) mittels inline-block nebeneinander positioniert. Da der Text selbst in eine „h2“ und einen Text in „p“-Tags aufgeteilt wird, ist das Styling der Textblöcke einfacher. Die Stylings der Textabschnitte passieren im CSS-Code für die Bestandteile beider Reihen gleichzeitig. So werden einfach alle „h2“- und „p“-Tags innerhalb der „section“ angesprochen und allgemeingültig verändert.

Ausgenommen aus dem Styling für die beiden Tags sind die Overlay-Texte der beiden Bilder. Diese werden noch einmal einzeln gestyled, weil hier eine spezielle Optik vorgesehen ist. Beide Overlays enthalten ein „Wir“, welches sehr viel größer über einem für jedes Bild individuellen kleinen Text steht. Das „Wir“ sieht bei beiden Bildern gleich aus und deswegen werden auch beide Bilder gleichzeitig mittels einer eigenen Klasse im CSS angesprochen. Dies geht beim jeweils darunterliegenden Text nicht, weil sich hier ein paar Eigenschaften unterscheiden müssen. So ist beispielsweise der Text der unteren Reihe linksbündig, während der im oberen Bereich rechtsbündig erscheint. Eine Eigenschaft teilen sich jedoch die Schriften: sie werden in vw angegeben. Dies dient erneut der Responsivität und gewährleistet dieselbe Größe beim zusammenziehen des Browsers.

Beide Overlay-Texte haben ihren eigenen Div-Container, welche „absolute“ über den Bildern platziert werden. Die genaue Positionierung mittels der vier Positionseigenschaften in CSS werden für beide Container einzeln vorgenommen, weil diese natürlich individuell ausfallen muss. Auch bei der Positionierung wird auf vw gesetzt, da so die Position bei einer Verkleinerung des Browsers beibehalten werden kann und die allgemeine Optik nicht gestört wird.

Zu den Overlays gehört noch eine Art Hover-Effekt, der sich erkenntlich macht, wenn man mit der Maus über die beiden Schriften geht. Dabei verwenden wir die Hover-Pseudoklasse, die eine CSS-Animation auslöst.

```
.pic-overlay-wir:hover, .pic-overlay-first-row p:hover, .pic-overlay-second-row p:hover {  
    animation-name: change-color;  
    animation-duration: 1s;  
    animation-fill-mode: forwards;  
}  
  
@keyframes change-color {  
    0% {  
        color: white;  
    }  
  
    100% {  
        color: #43DEB9;  
    }  
}
```

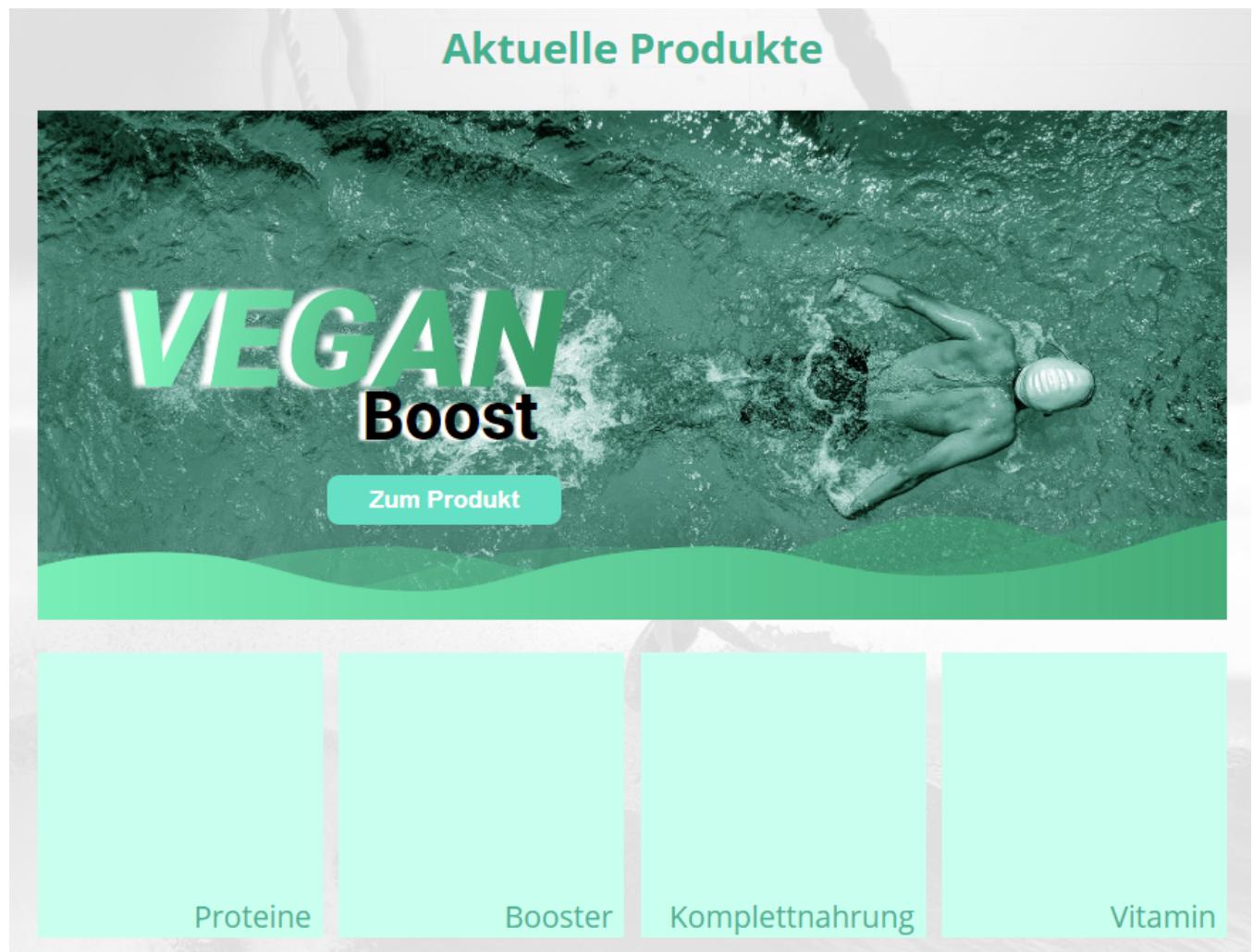


Die Animation wird deswegen benutzt, weil sie eine größere Kontrolle über den Verlauf der Farben zu gewährleistet. Das reine Verändern der Schriftfarbe, wenn die Maus über der Schrift liegt, sah zu schnell aus, weswegen die Animation diesen Vorgang deutlich verlangsamt und den Übergang genauer zeigt. Die Einstellungen lassen den Wechsel der Farbe über eine Sekunde hinweg geschehen.

Hat der Nutzer sich mit den vier Blöcken beschäftigt, geht es weiter zu den aktuellen Produkten.

## Die „Aktuelle Produkte“-Sektion

Sämtliche Designentscheidungen der Website basieren auf dem Prinzip „Erst Transparenz/Kundennähe, dann Verkauf“. Nachdem der Nutzer sich also mittels der zwei Bilder und den kurzen Texten einen Eindruck davon machen können, welche Prinzipien hinter der Firma stehen, geht es nun an die Werbung und den Verkauf der eigenen Produkte. Dafür beginnt innerhalb des main-Bereichs eine neue „section“, die sich rein der Anzeige zu aktuellen Produkten verschrieben hat.



Die gesamte Sektion der Startseite zu den Produkten besteht aus zwei Komponenten. Einerseits einer Anzeige mit dem aktuellsten Produkt und darunter vier große Buttons, die zu den Produktseiten führen.

```

<section class="products">
    <h1>Aktuelle Produkte</h1>
    <div class="products-pic-plus-overlay">
        

        <div class="products-overlay">
            
        </div>

        <div class="products-button-container">
            <a href="Vegan_Booster.html"><button class = "go-to-product">Zum Produkt</button></a>
        </div>
    </div>
    <div class="category-blocks">
        <div class="proteine">
            <a href="Vegan_Whey.html"><p>Proteine</p></a>
        </div>

        <div class="booster">
            <a href="Vegan_Booster.html"><p>Booster</p></a>
        </div>

        <div class="komplettahrung">
            <a href="Vegan_Nutrition.html"><p>Komplettahrung</p></a>
        </div>

        <div class="vitamine">
            <a href="Vegan_Vitamin.html"><p>Vitamin</p></a>
        </div>
    </div>
</section>

```

Das HTML dieser Sektion ist weit simpler aufgebaut als das der vorherigen. Nach der Überschrift des Abschnitts folgt ein Div-Container, der sowohl das Bild für die Anzeige als auch das Overlay (das Logo und den Button um zur Produktseite zu gelangen) enthält. Das Bild und das dazugehörige Overlay erhalten allen voran deswegen einen eigenen Container, weil dies die Positionierung von absoluten Elementen vereinfacht. Das Logo und der Button müssen absolut über dem Bild platziert werden und es wäre unpraktisch, wenn dies von der Sektion selbst aus geschehen müssten. Ohne ihren eigenen Container würden sich die absoluten Elemente nur relativ zu der Sektion verhalten, weil das das Elternelement wäre. Dann müssten beide Elemente von der Überschrift aus positioniert werden. Durch den eigenen Container verhalten sie sich quasi relativ zur oberen Kante des Bildes, da das nun das Elternelement ist.

Ansonsten weist dieser Teil des HTML-Codes keine Besonderheiten mehr auf. Einzig das Format und die Größe des Hintergrundbildes wären noch zu erwähnen. Hier wird auf ein JPG gesetzt, da keine Transparenz benötigt wird und dieses Format der Performance am zuträglichsten ist. Darüberhinaus ist auch dieses Bild solange verkleinert worden bis der optimale Kompromiss aus Qualität und Performance herauskam.

Unter der Anzeige befinden sich die vier Buttons. Diese sind nichts anderes, als vier Div-Container innerhalb eines Containers, die alle ein mit einem Link versehenes „p“-Tag enthalten. In diesem Tag steht die Produktkategorie. Während das HTML zu dieser Art von Menü sehr unscheinbar ist, setzen wir gerade im CSS verschiedenste Tricks und Techniken ein. So ist der Div-Container, der die vier Quadrate erhält, eine Flexbox. Diese Technik machte gerade das Positionieren und Skalieren der Buttons sehr einfach. So konnte in der Theorie jedem Button schlicht die richtige Hintergrund- und Schriftfarbe gegeben und der Text in die untere Ecke positioniert werden und die Buttons wären fertig, weil die Flexbox durch ihre „flex-direction“ und das „space-between“ für die „justify-content“-Eigenschaft die Knöpfe automatisch mit dem richtigen Abstand positionieren würden. Soweit hat das auch geklappt, nur sollten die Buttons quadratisch sein. Die Buttons waren schlicht zu schmal.

```
.proteine, .vitamine, .kompletnahrung, .booster {  
    background-color: #C3FFEF;  
    width: 24%;  
    height: 0;  
    padding-bottom: 24%;  
    text-align: right;  
}
```

Nun kam ein kleiner Trick zum Einsatz, der die Umsetzung der eigentlichen Idee doch weit einfacher machte als ursprünglich gedacht. Den Div-Containern der vier Buttons ist ein unteres Padding in der Größe der Breite des Containers zugewiesen worden. Es ergaben sich perfekte Quadrate, die auch einwandfrei mit der Seite skalierten. Die Positionierung des Textes warf jedoch ein weiteres Problem auf. Diese konnten nur innerhalb der Content-Region des Divs, aber nicht im Padding positioniert werden. Leider war genau das vorgesehen. Am Ende wurde einfach das Padding der Schrift angepasst und diese dadurch an die richtige Stelle geschoben.  
Der letzte Schritt für die vier Quadrate war das Zuweisen eines Hover-Effektes mittels der Pseudoklasse im CSS.

Als letzter der Startseite spezifischer Inhalt folgte die „Aktionen und Specials“-Sektion.

## Die „Aktionen und Specials“-Sektion

Der letzte Inhaltsbereich der Startseite sollte nun die „Aktionen und Specials“-Sektion werden. Nachdem es einen doch recht großen dedizierten Bereich für den Verkauf für Produkte auf der Startseite gibt, war die Intention hinter dem letzten Bereich wiederum eine Mehrwertidee. Gerade wiederkehrende Kunden hätten von der Website nichts mehr, weil das reine Kaufen eine Routineaktion werden würde. Um diesen Kunden etwas mehr zu bieten, führten wir die „Aktionen und Specials“-Sektion ein. Wenn die Website ein reales Projekt für eine Firma gewesen wäre, hätte dies ein Bereich mit regelmäßig wechselnden Artikeln und Videos werden können, die unterschiedlichste Themen behandeln, wie Sportler interviewen oder ähnliches. Deswegen entschieden wir uns für einen Artikel, der dort unten beworben werden sollte.

## Aktionen und Specials

**One on One  
mit dev1ce**

Voller Fokus, pure Konzentration, Adrenalin bis in die Fingerspitzen - das ist E-Sport. Im Interview erzählt uns Nicolai "dev1ce" Reedtz von seinem Weg an die Spitze, das Gefühl auf der Bühne und wie man in deartiger Konsistenz fokussiert bleibt.

Zum Interview

Der HTML-Aufbau dieser Sektion ist im Prinzip genauso zu sehen wie der der Werbeanzeige darüber.

```
<section class="specials">
    <h1>Aktionen und Specials</h1>

    <div class="specials-pic-plus-overlay">
        

        <div class = "specials-overlay">
            <h1>One on One <br><span>mit dev1ce</span></h1>
            <p>Voller Fokus, pure Konzentration, Adrenalin bis in die Fingerspitzen - das ist E-Sport. Im Interview erzählt uns Nicolai "dev1ce" Reedtz von seinem Weg an die Spitze, das Gefühl auf der Bühne und wie man in deartiger Konsistenz fokussiert bleibt. </p>
        </div>

        <div class="specials-button-container">
            <a href="deviceinterview.html"><button class = "go-to-product">Zum Interview</button></a>
        </div>
    </div>
</section>
```

Wieder enthält die „section“ zu dem Bereich eine Überschrift und einen Container, der sowohl das Hintergrundbild sowie das Overlay mit einem Text und einem Button enthält. Das Bild ist diesmal wieder eine PNG, da der Schnitt an der Seite Transparenz erforderte.

Ansonsten ist der Aufbau grundsätzlich derselbe wie der der Anzeige zum aktuellen Produkt. Der Text wird „absolute“ auf dem Bild positioniert. Einzig spannend ist hier die verwendete Überschrift erster Ordnung. So steht die gesamte Überschrift („One on One mit dev1ce“) in einem „h1“-Tag und das spezielle Styling des „mit dev1ce“ wird mittels eines „span“-Tags realisiert. Der Bereich lässt sich so im CSS einzeln ansprechen und entsprechend bearbeiten. Darunter befindet sich ein Fließtext in einem „p“-Tag, was wiederum einzeln bearbeitet wird.

Als allerletztes findet sich auf der Startseite der Footer, der auf allen Seiten gleich aussieht.

# Produktseite

Die Produktseite soll dem Benutzer das Produkt näherbringen und ihm zum Kauf anregen. Im Vordergrund unserer Produktseite soll daher das Produkt stehen. Das Produkt soll möglichst detailliert beschrieben werden, während die Oberfläche einfach und übersichtlich bleibt. Öffnen wir die Seite, sehen wir zuerst ein Titelbild, mit einem passendem Slogan zum Produkt (hier „Natur Kick“). Dieser Slogan ragt etwas aus dem Bild heraus, um mit dem Rest der Seite zu verschmelzen. Des Weiteren hat der Nutzer die Möglichkeit, sich durch ein Bilderkarussell durchzuklicken. In diesem Karussell werden Bilder des Produktes angezeigt, aber aus Zeitgründen haben wir hier hauptsächlich Platzhalter Bilder eingesetzt als tatsächlich Bilder von Produkten mit unserem Logo zu erstellen.

Darunter befindet sich eine große Welle, die bereits zuvor angesprochen wurde. Der Rest der Seite bietet wenig erwähnenswerte Elemente, bis auf die Nährwerttabelle die mit selbst angegebenen Werten errechnet wird und der Navigation Bar, die sich beim runter bzw. rauf scrollen zusammen bzw. aufklappt. Ein weiterer Hingucker sind leichte Einblendeffekte der einzelnen Abschnitte, wenn man herunterscrollt.



## Produktbeschreibung

Der Vegan Boost wurde entwickelt, um Müdigkeit zu bekämpfen und das tägliche Leistungsniveau zu unterstützen. Es kombiniert Vitamine mit Kräutern, die in der traditionellen chinesischen Medizin verwendet werden, und ist ideal für allgemeine Müdigkeit, unnachgiebige Arbeitszeiten und hektischen Lebensstil.

## Inhaltsstoffe

### Ingwer



Ingwer ist reich an Vitamin C und enthält darüber hinaus Magnesium, Kalium, Kalium, Natrium und Phosphor. Das Rhizom wirkt antibakteriell und kann somit zu einer gesunden Darmflora beitragen. Ingwer wirkt auch entzündungshemmend und gegen die Vermehrung von Viren. Besonders die enthaltenen Scharstoffe regen außerdem die Durchblutung und den Kreislauf an.

### Rote Beete



Das Rote Beete Rhizom stammt aus den Knollen des Rübenpflanzen. Diese werden geerntet und dann zu Pulver gemahlen. Da die Knollen reich an Mineralstoffen und Vitaminen sind, stärken sie unser Immunsystem. Rote Beete ist außerdem mit wichtigen Nährstoffen. Zudem ist die Rote Beete reich an Eiweiß und Ballaststoffen und enthält wenig Zucker und Kalorien.

### Guaraná



Der bedeutendste Inhaltsstoff in Guaraná ist das Koffein: Zwischen 0,9 und 7,6 Prozent liegt das Koffeingehalt von Guaraná-Pulpa und es ist deutlich höher als im Koffeingehalt von Kaffee, der es nur auf 1,2 bis 1,3 Prozent schafft. Noch dazu ist Koffein in Guaraná für den menschlichen Körper um einiges besser verträglich als das von Kaffee.

## Nährwertangaben

### Durchschnittliche Nährwerte pro 100 g

Brennwert	975kJ/229kcal
Fett	12,5g
davon gesättigte Fettsäuren	5,4g
Kohlenhydrate	0,6g
davon Zucker	0,6g
Salz	5,0g

## Lieferanten

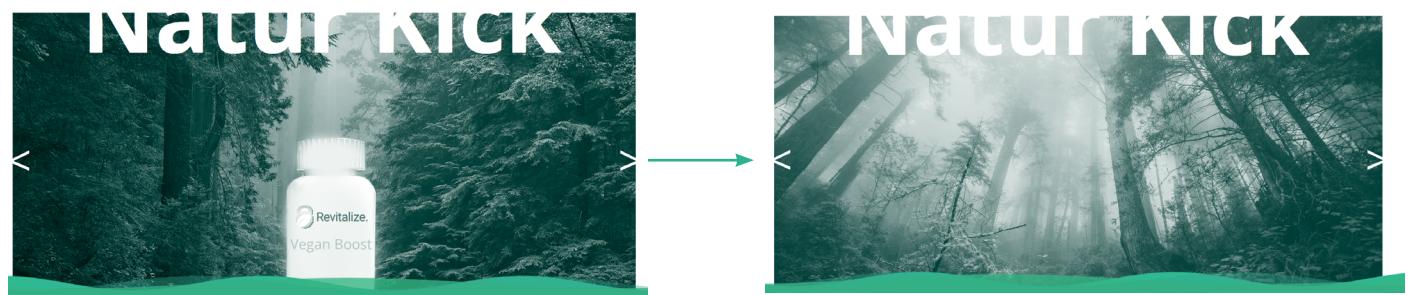


## Landlinie

Wir vertreiben seit 1989 Produkte aus biologischen Anbau für den Handel. Wir sind Ihr Biohofhändler, der Wert auf eine faire und vertraulose Zusammenarbeit mit engagierten und vor allem zertifizierten Produzenten legt. Wir kennen die einzelnen Anbaugebiete, Produktionsstätten und die Menschen dahinter persönlich. Aus den langjährigen Beziehungen sind sehr verlässliche Kontakte gewachsen, die es uns ermöglichen, Ihnen erstklassige Bio-Produkte anbieten zu können.

Die LANDLINE Lebensmittel-Ventrieb GmbH mit Sitz in North bei Köln ist ein Lebensmittel-Großhändler für biologisch erzeugte Produkte und Spezialist im Bio-Oliven- & Gemüse-Bereich. Wir sind seit über 25 Jahren erfolgreich deutschlandweit tätig, seit 1991 Deineter Vertrieghändler.

## Das Bilderkarussell



Das Bilderkarussell ist bedienbar durch die beiden Pfeile links und rechts. Klickt man auf einen dieser Pfeile, wird das aktuelle Bild langsam transparent und das nächste Bild langsam sichtbar. Im Hintergrund werden die einzelnen Bilder dann mittels javascript auf transparent bzw. sichtbar gesetzt.

```
function slide(n=1){  
    //altes Bild transparent setzen  
    x[currentImage].style.opacity = 0;  
  
    currentImage+=n;  
  
    //Ausnahmefälle abfangen  
    if(currentImage==x.length)  
        currentImage=0;  
  
    if(currentImage<0)  
        currentImage=x.length-1;  
  
    //Das Neue Bild sichtbar setzen  
    x[currentImage].style.opacity = 1;  
}
```

Die Variable, die wir der Funktion slide übergeben, soll -1 oder 1 betragen wobei -1 nach links und 1 nach rechts im Karussell ist. Dabei können beliebig viele Bilder in das Karussell eingefügt werden und die Funktion erfüllt weiter ihren Zweck, da sie nicht mit einer festen Anzahl an Bildern arbeitet, sondern weil die Funktion immer mit der Anzahl an Bildern arbeitet, denen wir eine bestimmte CSS Klasse gegeben haben. X ist hier die Liste aller Bilder, die diese Klasse besitzen. Die Funktion wird auch alle 10 Sekunden selbst ausgelöst, damit sich das Karussell automatisch bewegt. Um den Übergang der einzelnen Bilder sanfter zu gestalten, fügen wir noch dies zu der CSS Klasse hinzu.

```
transition: opacity 2s ease-out;
```

Eine weiter kleine Verbesserung bei diesem Karussell ist im Bereich User Experience, denn der Benutzer muss nicht direkt über die einzelnen Pfeile Hovern um sich im Karussell fortzubewegen. Über den beiden Pfeilen liegen nämlich 2 div Elemente die onmouseover bzw. onmouseout die beiden Pfeiltasten einfärben und bei einem Klick die oben erwähnte Funktion auslösen.



# Die Nährwerttabelle

Durchschnittliche Nährwerte pro 100 g		Durchschnittliche Nährwerte pro 250 g	
Brennwert	975kJ/229kcal	Brennwert	2437.5kJ/572.5kcal
Fett	12.5g	Fett	31.25g
davon gesättigte Fettsäuren	5.4g	davon gesättigte Fettsäuren	13.5g
Kohlenhydrate	0.6g	Kohlenhydrate	1.5g
davon Zucker	0.6g	davon Zucker	1.5g
Salz	5g	Salz	12.5g

Die Nährwerttabelle besteht aus einer gewöhnlichen HTML Tabelle, wobei die einzelnen Elemente mittels Javascript basierend auf dem obigen Input Feld verändert werden. Die Funktion zur Errechnung wird oninput ausgelöst, was der User Experience zugutekommt. Des Weiteren wurde die Tabelle mit CSS so gestaltet, dass der Text der linken Spalte auch linksbündig ist bzw. die rechte Spalte rechtsbündig sowie die Unterbereiche einer Zeile (z.B. davon gesättigte Fettsäuren) einen größeren Abstand zur linken Kante haben, damit sie leichter als Unterbereich wahrgenommen werden. Das Input Feld ist auch nur auf Nummern eingestellt, sodass kein Unfug hineingeschrieben werden kann.

## Die Navigation Bar



Die Navigation Bar besteht bei der Produktseite aus einer weiteren Leiste, die unterhalb der originalen Navigation Bar liegt. Durch diese Leiste kann der Nutzer sich durch die Produktseite navigieren, sowie dem Nutzer immer die Möglichkeit geben, das Produkt in seinen Warenkorb abzulegen, um es später zu kaufen. Viele Seiten wie z.B. Amazon bieten diesen „In den Warenkorb“ Button nur statisch auf der Seite an, und sollte man sich für den Kauf entscheiden, muss man erst zurück zu diesem Button scrollen, um das Produkt in den Warenkorb zu schieben. Mit dieser zusätzlichen Navigationsleiste wollen wir ein verbessertes Nutzererlebnis erzielen, da er nicht immer nach diesem Button auf der Seite suchen muss.

Ein Problem ist, das die Navigation Bar zu groß ist (mit der zusätzlichen Leiste), weshalb wir uns entschieden haben die originale Navigation Bar nach oben zu schieben, sollte der Nutzer nach unten scrollen. Nun kann der Nutzer nur den Bereich der Leiste sehen, der relevant ist, um sich durch die Produktseite zu navigieren. Scrollt der Benutzer wieder nach oben, erscheint wieder die gesamte Navigation Bar. Möchte der Benutzer die Seite verlassen, scrollt er intuitiv nach oben, weshalb die originale Navigation Bar wieder von Vorteil ist, weil sie von der Produktseite weg navigiert. Mit Javascript errechnen wir aus der letzten scroll position und der aktuellen scroll position ob der Nutzer nun herauf bzw. herunter gescrollt hat und verschieben darauf basierend die gesamte Navigation Bar herauf bzw. herunter.

```
function moveNavBar(){
    var z = document. getElementsByTagName("nav");

    if  (this.scrollY <= 100) //schieb navigationbar nach unten wenn man am anfang der Seite ist
        z[0].style.top = -this.scrollY.toString()+"px";
    else //schieb navigations bar nach oben wenn man runter scrollt
        z[0].style.top = "-5.5vw";

    //wenn nach oben gescrollt wird schieb die original Navigation bar nach unten
    if(lastScrollY-this.scrollY > 0)
        z[0].style.top = "0";

    lastScrollY=this.scrollY;
}
```

Dessweiteren werden die einzelnen Elemente der zusätzlichen Leiste der Navigation Bar auch verfärbt sollte man sich nicht am Anfang der Seite befinden, wie bei der originalen Navigation Bar. Interessant ist auch der innere Abschnitt der zusätzlichen Navigationsleiste, denn diese sind Links, die zu dem bestimmten Abschnitt auf der Seite verlinkt.



Befindet man sich wie beim Screenshot bei dem Bereich Lieferanten auf der Seite, wird der untere Rand des Textelementes in weiß eingefärbt.

# Artikelseiten

## Fortschrittsbalken



Der Fortschrittsbalken auf Artikelseiten (der schmale dunkelgrüne Balken unter der Navigationsleiste) soll zeigen, wie weit ein Artikel bereits gelesen wurde bzw. wie weit es bis zum Ende des Artikels ist.

```
1  addEventListener("scroll", function () {  
2      var viewport = window.innerHeight;  
3      var offset = window.pageYOffset;  
4      var site_length = document.body.offsetHeight;  
5      var footer = document.getElementsByTagName("footer")[0].offsetHeight;  
6  
7      var progress = (offset*100)/(site_length-footer-viewport);  
8      document.getElementsByClassName("inner_bar")[0].style.width=progress+"%";  
9  })
```

In einer ersten Version wurden lediglich das Y-Offset und die Seitenlänge miteinander dividiert um einen groben Wert zu erhalten. Dieser hat dann aber nicht angezeigt wie weit der Artikel gelesen wurde, sondern wie weit heruntergescrollt wurde.  
Um nun nur den Artikel in die Berechnung einzubeziehen wurden von der Seitenlänge noch zusätzlich die Höhe des Footers und der Viewport subtrahiert.

Der Footer wurde subtrahiert, da dieser das Element direkt nach dem Artikel ist und zwischen beiden dieser folglich endet. Der Viewport wiederum wurde subtrahiert damit der Balken bereits bei 100% ist, sobald die letzte Zeile eines Artikels sich an der unteren Kante befindet.

# Kontaktseite

## Konaktformular



Über uns    Produkte     Berater    Kontakt

**Haben Sie noch Fragen?  
Kontaktieren Sie uns.**

Etiam sodales volutpat dolor quis consectetur. Sed interdum sapien vel ex luctus, quis convallis justo pretium. Cras finibus, nisi eget suscipit congue, ligula ligula tincidunt neque, in facilisis tellus magna ut augue.

Ihre Email\*:

Ihre Nachricht\*:

(\*Pflichtfelder)

Nachricht absenden

Das Kontaktformular bietet dem Nutzer die Möglichkeit bei spezifischen Fragen mit dem Unternehmen in Kontakt zu treten. Hierfür wird die Nachricht selbst sowie eine Email Adresse für eine Antwort benötigt.

Das Abspeichern wird durch ein PHP Skript ermöglicht, welches hier aber nie wirklich ausgeführt wird, da es sehr umständlich wäre eine Datenbank inklusive Server dafür aufzusetzen. Das Skript wurde so verfasst, dass es die Aufgabe erfüllen würde.

# Kontaktformular

## Code

Bei dem Code haben wir darauf geachtet, dass SQL-Injection Angriffe und XSS nicht möglich sind, indem besagte Fälle vor dem Erstellen der SQL Anweisung abgefangen und bearbeitet werden.

```
$quotedmail = $con->quote($mail);
$quotedmessage = $con->quote($message);
$processedmail = htmlentities($quotedmail, ENT_QUOTES, "UTF-8");
$processedmessage = htmlentities($quotedmessage, ENT_QUOTES, "UTF-8");
```

Zudem befinden sich die Daten für den Zugriff auf die Datenbank in einer separaten Datei, damit diese sicherer sind. Das zusätzliche Skript wird mit „require“ eingebunden, da es wichtige Konstanten enthält, welche nicht fehlen dürfen. „Include“ wäre da dann nicht ausreichend.

```
require("sqldata.php");
```

```
const SQLSERVER = "modelpfad";
const USER = "conatct";
const PASSWORD = "12345";
```

In der Theorie würde das Skript den Kommentar/ die Frage eines Kunden zusammen mit einer angegebenen Email in einer Datenbank speichern, wo diese dann weiterverarbeitet werden kann.

Die Modelldatenbank hat die drei Spalten „ID“ (als Primary Key), „mail“ und „message“.

```
$stmt = $con->prepare("INSERT INTO messages ('id', 'mail', 'message') VALUES (NULL, '".$processedmail."', '".$.
$processedmessage."');");
$stmt->execute();
```



# Die Kategorieseiten

Die Revitalize-Website hat verschiedenste Kategorieseiten. Dazu zählen die Produktseiten, die Produktübersicht, die Über-uns-Seite und die Transparenz-Seite. Die Produktseiten haben ihr eigenes Kapitel, weswegen in diesem Teil nur auf die sonstigen Produktseiten eingegangen wird. An dieser Stelle sei auch erwähnt, dass ausschließlich Besonderheiten und Strukturen thematisiert werden, die nicht bereits an anderer Stelle benutzt und dementsprechend erklärt wurden oder aber keine Besonderheit aufweisen.

## Die Produktübersicht

Die Produktübersicht zeigt dem Nutzer einerseits, warum er sich überhaupt für die Produkte der Firma entscheiden sollte und zeigt zusätzlich eben alle diese Produkte an. Für diese beiden Ziele gibt es je eine Sektion auf der Seite, welche nun der Reihe nach genauer besprochen werden.

Zunächst sei jedoch auf das Titelbild eingegangen. Die Titelbilder sehen auf jeder Kategorie- und Artikelseite so ziemlich gleich aus. Einzig und allein die Positionen der Schriften unterscheiden sich um etwas Spannung beizubehalten.



Realisiert werden diese Titelbilder über ein Bild und einer in weiß „absolute“ darüber positionierten Schrift. Das Bild selbst wird durch eine Breitebegrenzung von 90% im selben Rahmen gehalten wie der Hauptcontent einer jeden Seite, während die Welle darunter mit ein Breite von 100% den ganzen Horizont des Viewports einnimmt. Dies war gewollt, weil so eine spannende Ästhetik entsteht. Positioniert wird die Welle hier genauso wie auf jeder Startseite. Zu der Animation der Wellen gibt es mehr Informationen in dem Kapitel dazu.

Nach dem Standardkonstrukt des Headers einer jeden Unterseite folgt die Sektion „Warum Revitalize?“, die dem Nutzer die Vorteile und speziellen Eigenschaften der Produkte näher bringen soll. Um diese Darstellung möglichst spannend zu gestalten, wird hier auf eine interaktive Anzeige mit drei Blöcken gesetzt.

## Warum Revitalize?



Beim Hovern über einen der jeweiligen Blöcke fährt sich der Informationstext aus, der zu den entsprechenden Überschriften gehört.

Damit dieser Effekt auch so funktioniert, ist eine spezielle Struktur im HTML-Code notwendig.

Die drei Blöcke befinden sich zunächst in einem großen Div-Container. Die einzelnen Blöcke sind in zwei Div-Containern untergebracht. Der erste trägt die Klasse „whole-block“, die jeder Block hat. So kann jedem ein und dasselbe Styling zugewiesen werden. Die Klasse des zweiten Div-Containers ist nach der jeweiligen Stelle benannt, an welcher der Block zum Einsatz kommt. So meint „wave-one“ den ersten Block, „wave-two“ den Zweiten und „wave-three“ den Dritten. Eigentlich sind alle Blöcke gleich gestyled. Die Klassen der Div-Container werden jedoch später für den Hover-Effekt relevant. Innerhalb des zweiten Containers befinden sich wiederum zwei Divs - „wave-head“ und „wave-text“. Diese sind auch für alle Blöcke gleich.

Der „wave-head“ meint die Welle, das Icon darauf und die darunterliegende Überschrift des Blocks, während „wave-text“ den Teil meint, der nach unten ausfährt, wenn über den Block gehovert wird.

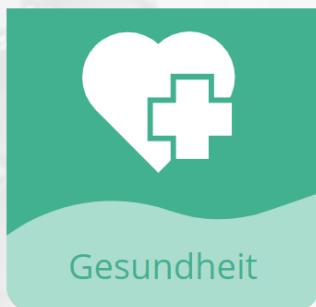
Spannend wird es jedoch erst im CSS. Zunächst werden die Blöcke durch „inline-block“ nebeneinander platziert, die Wellen richtig ausgerichtet, die Icons „absolute“ auf den Wellen platziert, die Hintergrundfarbe von „wave-head“ auf das richtige Grün gebracht und die Überschrift richtig gestyled und positioniert - so weit bekannt.

Interessant ist jedoch das Styling der Elemente mit der „wave-text“-Klasse. Diese erhalten nämlich ein „translateY(-94%)“ für die „transform“-Eigenschaft und einen geringeren z-index als die mit „wave-head“ versehenen Elemente. Dadurch verschwinden sie hinter den Teil der Blöcke mit der Welle, den Icons und der Überschrift.

Doch warum „-94%“? Dieser Wert platziert den Text so hinter den Kopf des Blocks, dass nur noch die abgerundeten Ecken rausgucken und zusammen mit dem unteren Teil des Kopfes eine angenehme Optik erzeugen.

Um den Effekt auszulösen, wird die Hover-Pseudoklasse verwendet. Sobald über die Blöcke gehovert wird, wird der „translateY“-Wert von -94% auf -30% gesetzt und die Textblöcke somit sichtbar nach unten verschoben. Um daraus eine flüssige Animation zu machen, wird innerhalb der „wave-text“-Klasse der „transition“-Wert auf eine halbe Sekunde gesetzt, was den Text in optimaler Geschwindigkeit ein- und ausfahren lässt.

## Warum Revitalize?



Gesundheit



Umweltschutz



Natürlichkeit

Lorem ipsum dolor sit amet,  
consectetur adipiscing elit, sed  
do eiusmod tempor incididunt ut  
labore tempor incididunt ut  
labore

Unter der „Warum Revitalize“-Sektion befindet sich die Auswahl der Produkte. Links zu sehen sind die Kategorien und von rechts fährt das Produkt in das Bild, während der angeklickte Button sich ebenso nach rechts bewegt und synchron mit dem Produkt nach links animiert wird.

### Kategorien

Protein	 <b>VEGAN</b> <b>Whey</b>
Booster	
Kompletnahrung	
Vitamin	<a data-bbox="1176 1423 1351 1455" href="#">Zum Produkt</a>

Diese Form einer Tabliste basiert auf dem Tutorial von W3Schools zu Tablisten in HTML, CSS und JavaScript, wurde jedoch für unsere Zwecke abgeändert.

Im HTML befindet sich so ein Div-Container, der fünf weitere Div-Container enthält. Im ersten sind nacheinander die Buttons der Tabliste zu finden. Diese öffnen beim Klicken die Funktionen „showProduct“ und „slideIn“ aus dem JavaScript-Code. Darauf egehen wir im Folgenden genauer ein. Der letzte Button hat eine zusätzliche ID „last-product-button“, die im CSS genutzt wird um die untere Margin des Buttons zu entfernen. Die Buttons haben alle eine Margin erhalten um einen Abstand zu den darunterliegenden Buttons zu gewinnen, diese ist jedoch beim unteren Button nicht gewünscht und wird deswegen für diesen wieder entfernt.

Die vier weiteren Div-Container enthalten die Produktinformationen - also jeweils eine Produktillustration, ein Produktlogo und einen Button zur Produktseite des jeweiligen Produkts. Die HTML-Klassen dieser vier Div-Container sind dieselben, jedoch hat jeder Container seine eigene ID erhalten um einzeln ansprechbar zu sein.



Im CSS wird zunächst der Container der fünf Divs zu einer Flexbox, was das nebeneinander darstellen der Inhalte stark vereinfacht.

Die Buttons bekommen alle dasselbe Styling (mit Ausnahme des Letzten dem seine Margin entzogen wird) und auch die Content-Struktur der Produktinformationen ist immer exakt dieselbe. Auf die Produktillustration wird das Logo gesetzt und in die untere rechte Ecke der Button um zum Produkt zu gelangen. So weit so bekannt.

Spannend wird es bei der Animation. Im CSS wird die „.active“-Klasse definiert, die eine einsekündige Animation startet, die mittels „transform: translateX()“ ein Objekt von rechts nach links ins Bild laufen lässt. Jetzt muss diese Klasse nur passend zugewiesen werden um die Animation auch durchzuführen - dazu dient der JavaScript-Code.

```
function showProduct(event, productName){  
  
    let productInfo = document.getElementsByClassName( classNames: "product-tab-content");  
    let productLinks = document.getElementsByClassName( classNames: "product-tab");  
  
    for (let i = 0; i < productInfo.length; i++) {  
        productInfo[i].style.display = "none";  
    }  
  
    for (let v = 0; v < productLinks.length; v++) {  
        productLinks[v].className = productLinks[v].className.replace( searchValue: " active", replaceValue: "");  
    }  
  
    document.getElementById(productName).style.display = "block";  
    event.currentTarget.className += " active";  
}
```

Zunächst gehen wir auf die Funktionsweise der „showProduct“-Funktion ein. Diese wird ausgeführt, wenn der Nutzer auf einen Button klickt. Der Klick auf den Button über gibt der Funktion auch gleich die ID des Produkts, dass der Nutzer sich ansehen möchte. Auch über gibt die Funktion ein sogenanntes Event, was dem JavaScript-Code das Element markiert, was zuletzt den Event Listener ausgelöst hat.

Diese Informationen nutzt die Funktion nun. Zuerst werden Variablen erstellt. Die Variable „productInfo“ speichert alle Divs ab, die die eigentliche Produktvorschau enthalten, während „productLinks“ die Buttons der Tablist enthält.

In der ersten for-Loop wird durch das Ändern des Wertes der „display“-Eigenschaft auf „none“ erstmals alle Produktvorschauen ausgeblendet.

In der zweiten for-Loop wird allen Buttons der Tabliste die „.active“-Klasse entzogen, sodass diese nicht mehr als ausgewählt angezeigt werden.

Nun folgt der eigentliche Effekt: In der zweit letzten Zeile nimmt sich der JavaScript-Code das Element mit der ID, die in der Parameterliste übergeben wurde, und setzt da den „display“-Wert auf „block“, was nur diesen wieder sichtbar macht.

Zuletzt wird mittels des übergebenen „Events“ der Button auf „.active“ gesetzt, der die Produktvorschau ausgelöst hat.

Für die Animation sorgt die „slideIn“-Funktion.

```
function slideIn(productName)
{
    let productContent = document.getElementById(productName);
    productContent.className += " active";
}
```

Diese wird ebenso ausgeführt, wenn ein Button in der Tabliste angeklickt wird. Ebenso wird hier die ID der Produktvorschau übergeben. Mittels dieser ID wird auf genau die eine Produktvorschau zugegriffen und diese in der Variable „productContent“ abgespeichert. Dieser Produktvorschau wird die im CSS definierte Klasse „active“ angehängt, die wiederum die Animation startet.

Doch warum bewegt sich der Button auch in das Bild, wenn man draufklickt?

Ganz simpel: die Klasse, die die Animation triggert heißt genauso wie die Klasse, die einen Button als ausgewählt markiert. Der Button erhält die Klasse in der „showProduct“-Methode, wenn jemand dort draufklickt. Das startet die Animation dann auch für den Button.

## Die „Über uns“- und die „Transparenz“-Seite

Die letzten beiden Kategoriseiten sind soweit sehr simpel aufgebaut und funktionieren nach demselben Schema wie bereits besprochene Seiten. Deswegen wird an dieser Stelle nur kurz auf die Seiten eingegangen.



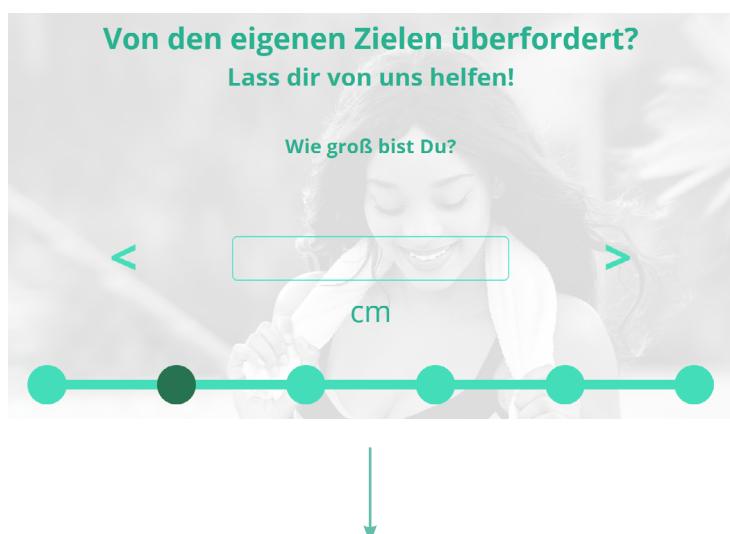
Grundsätzlich sind beide Seiten ein Aufeinanderfolgen von Bildern, zugehörigem Text und einem Button, der zur entsprechenden Unterseite führt. Hier sei nur kurz erwähnt, dass für die Kombination aus einem Bild und Text daneben eine Flexbox verwendet wird, während für die Teile mit einem Bild im Hintergrund und Text und Buttons auf dem Bild dieselbe Struktur wie auf der Startseite verwendet wird.

# Revitalize Berater

Der Revitalize Berater soll dem Benutzer Tipps und Produkte individuell basierend auf einigen Eingaben des Nutzers geben. Im oberen Bereich haben wir wieder das gewohnte Titelbild und eine große animierte Welle. Darunter befindet sich dann der eigentliche Berater. Hier befindet sich immer eine Frage und eine Eingabemöglichkeit für den Benutzer. Darunter befindet sich eine Anzeige, die dem Nutzer anzeigt, wie weit er sich im Berater befindet bzw. wie viele Fragen noch kommen, bis er sein Ergebnis erhält. Mittels Javascript wird dann die Frage sowie die Eingabemöglichkeit verändert und außerdem der Fortschritt innerhalb des Beraters korrekt angezeigt. Mittels Pfeiltasten kann der Benutzer dann vorwärts oder ggf. rückwärts gehen.



The screenshot shows a large green header bar with the word "Berater" in white. Below it is a black and white photograph of a basketball player jumping to dunk a ball. A green progress bar at the bottom indicates the user's progress through the advisor. To the right, a woman is shown holding a smartphone, with a text overlay asking if she is overwhelmed by her goals and offering help. The footer contains navigation links for Revitalize, transparency, products, shopping cart, advisor, and contact, along with links for terms of service, privacy policy, and imprint.



The screenshot shows a slider for "Wie groß bist Du?" (How big are you?) with a range from < to >. The slider is currently set to 0 cm. A large green arrow points downwards towards the second screenshot.



The screenshot shows a slider for "Was ist dein persönliches Ziel?" (What is your personal goal?). The slider has options like "Fettreduktion" (Fat reduction) and "Gesundheit" (Health). The slider is currently set to "Fettreduktion".

Texteingabe

Dropdown-Liste

## Fortschrittsbalken



Der Fortschrittsbalken funktioniert ähnlich wie das Bilderkarussell und ändert durch eine Variable den momentan gefärbten Kreis.

```
function changeFortschritt(){
    //liste mit allen Kreisen
    var x = document.getElementById("circles").getElementsByTagName("li");

    //Farbe des vorherigen Kreises zurück auf die normale Fabe ändern (wenn man einen Schritt nach vorne geht)
    if(currentInput!=0)
        x[currentInput-1].style.backgroundColor = "#43DEB9";

    //Farbe des aktuellen Kreises auf einen dunklen Ton färben
    x[currentInput].style.backgroundColor = "#27734f";

    //Farbe des vorherigen Kreises zurück auf die normale Fabe ändern (wenn man einen Schritt zurück geht)
    if(currentInput<x.length-1)
        x[currentInput+1].style.backgroundColor = "#43DEB9";
}
```

Die variable `currentInput` gibt dabei den momentanen Fortschritt innerhalb des Beraters an und liegt immer zwischen 0 und 5 also 6 Schritte im Berater. Diese Variable wird hoch bzw. runter gezählt sollte der Nutzer auf den rechten bzw. den linken Pfeil klicken.

## Input Feld

Wie bereits erwähnt gibt die Variable `currentInput` an bei welchem Schritt wir uns im Berater befinden. Darauf basierend ändern wir nun den Input des Benutzers. Hier ein Beispiel für den vierten Schritt im Berater, die Frage nach der Ernährung durch 3 Radio Buttons.

```
if(currentInput==3){ //Schritt 3 = Frage nach der Ernährung durch 3 radio buttons
    x.innerHTML = "Wie ernährst du dich?";
    addButtons(); //hinzufügen der Buttons und entfernen der Einheit
}
```

Zuerst wird direkt die Frage geändert, was eine simple Veränderung des inneren HTML ist. Danach werden die Buttons hinzugefügt.

```
function addButtons(){
    document.getElementById("InputFeld").innerHTML =
    `<label class="ErnährungsTypText" for="Omnivor">Omnivor</label>
    <input class="ErnährungsTypButton" type="radio" id="Omnivor" name="Ernährung" value="Omnivor">

    <label class="ErnährungsTypText" for="Vegetarisch">Vegetarisch</label>
    <input class="ErnährungsTypButton" type="radio" id="Vegetarisch" name="Ernährung" value="Vegetarisch">

    <label class="ErnährungsTypText" for="Vegan">Vegan</label>
    <input class="ErnährungsTypButton" type="radio" id="Vegan" name="Ernährung" value="Vegan">`;

    var y = document.getElementById("Einheit").style.display = "none";
}
```

Das Element mit der ID Inputfeld ist ein Div Element, worin sich der Input des Benutzers befindet.

```
<div id="InputFeld"><input type="text"></div>
```

Zu Beginn ist es ein simples text Input Feld aber wollen wir daraus nun 3 Radio Buttons machen, müssen wir den inneren HTML-Bereich des Inputfeld div Elementes ändern. Dazu müssen wir einen langen String angeben mit allen Elementen, die innerhalb dieses div Elementes eingefügt werden sollen. Bei diesem Beispiel müssen wir also 3 Labels hinzufügen sowie 3 Input Elemente als Radio Typ. Dabei müssen wir beachten, das Label auch korrekt auf die Buttons verweisen, damit die Buttons auch über das Label anklickbar sind und das alle Radio Buttons denselben Namen haben, damit sie in einer Radio Button Gruppe sind.



# Warenkorb

The screenshot shows the shopping cart page of the Revitalize website. At the top, there are navigation links: Über uns, Produkte, Berater, Kontakt, and a shopping cart icon with a '4' indicating four items. Below this, the shopping cart area displays four products:

Produkt	Aktionen	Anzahl	Preis
Vegan Whey - Veganes Protein	Löschen	1	25.99€
Vegan Boost - Veganer Booster	Löschen	1	19.99€
Vegan Nutrition - Vegane Kompletnahrung	Löschen	1	18.99€
Vegan Vitamin - Veganer Vitaminpräparat	Löschen	1	8.99€

Total price: Sie zahlen: 73.96€ **Zur Kasse**

At the bottom of the cart area, it says Gesamtpreis: 73.96€.

The footer of the page includes the Revitalize logo, navigation links (Über uns, Transparenz, Produkte, Einkaufswagen, Berater, Kontakt), social media icons (Facebook, Instagram, YouTube, Twitter), and a footer note: "Revitalize. Geschichten Werte Karriere Lieferanten Fairtrade Nachhaltigkeit Check-Out".

Der Warenkorb speichert alle Produkte, die der Nutzer ausgewählt hat, um sie später zu kaufen. Betätigt man den zur Kasse Button wird man zum Check-out weitergeleitet, bei dem man Kontaktdaten eingeben muss um den Kauf zu bestätigen. Das Besondere an dem Warenkorb ist, dass sämtliche Daten über die Produkte über Cookies gespeichert werden. Legt der Kunde über eine Produktseite ein Produkt in den Einkaufswagen, wird direkt ein Cookie mit der entsprechenden Anzahl des Produktes gespeichert. Die kleine Nummer neben dem Einkaufswagen Icon in der Navigation Bar zeigt an wie viele Produkte im Einkaufswagen liegen. Dies wird durch Auslesen der Cookies errechnet, damit die kleine Anzeige auf allen Seiten funktioniert. Durch dieses Icon oder durch den Footer wird man direkt zur Warenkorb Seite weitergeleitet. Hier werden durch Auslesen des Cookies sämtliche Produkte angezeigt, die der Nutzer ausgewählt hat. Hier kann der Nutzer Produkte wieder aus dem Warenkorb löschen oder mittels Input Feld eine beliebige Anzahl an Produkten bestellen. Löscht oder verändert er die Anzahl der Produkte, wird direkt der Cookie aktualisiert. Außerdem wird der Gesamtpreis des Warenkorbs errechnet sowie der Gesamtpreis eines Produktes.

## Produktanzahl Anzeige



Diese kleine Nummer neben dem Warenkorb Icon zeigt an, wie viele Objekte insgesamt im Warenkorb sind. Hier könnten das zum Beispiel 2x vegan Vitamin, 1x vegan Protein und 1x vegan Nutrition sein. Die Zahl wird durch Auslesen eines Cookies errechnet.

```
function calculateWarenkorbAnzahl(){
    var x = document.cookie.split(";");
    var product1 = x[3].split("=")[1];
    var product2 = x[2].split("=")[1];
    var product3 = x[1].split("=")[1];
    var product4 = x[0].split("=")[1];
    var anzahl = parseInt(product1) + parseInt(product2) + parseInt(product3) + parseInt(product4);
    document.getElementById("WarenkorbAnzahl").innerHTML = anzahl;
}
```

Wir speichern 4 Cookies ab, für jedes Produkt 1 Cookie. Ein Cookie wird immer mit Produktnamen und Anzahl abgespeichert. Der gesamte Cookie wird hier in die 4 einzelnen Produkt Cookies aufgespaltet und dann wird die Anzahl ausgelesen und aufaddiert. Diese Funktion wird auf jeder Seite nach Laden des Dokumentes ausgeführt. Des Weiteren wird zuvor überprüft, ob bereits ein Cookie angelegt wurde. Ist dies nicht der Fall, wird ein neuer Cookie angelegt, wobei die Anzahl aller Produkte 0 ist. Alle Cookies laufen in 20 Tagen nach Erstellung ab.

# Checkout

## Lieferadresse

Vorname\* Nachname\*

Email\*

Straße\* Hausnummer\*

Postleitzahl\* Stadt\*

Land\*

## Bezahlung

Kreditkarte PayPal Rechnung

Kreditkarteninhaber\*

Kreditkartennummer\* (xxxx-xxxx-xxxx-xxxx)

Monat\* Jahr\*

(\*Pflichtfelder)

**Bezahlen**

Der Checkout besteht aus insgesamt 4 div-Elementen, eines für die Lieferadresse und drei weitere für die Bezahlmethoden. Von den drei Bezahlmethoden sind zu jeder Zeit zwei deaktiviert (display: none) und nur das aktuelle Element, dessen Button gedrückt wurde, wird angezeigt.  
Sobald ein Button gedrückt wird (zB. „PayPal“) werden einmal alle div-Elemente deaktiviert und das „PayPal“-Element danach separat aktiviert.

```
function paypal () {
    deactivateAll();
    var buttons = document.getElementsByClassName( classNames: "paypal_button");
    for(var i = 0; i < buttons.length; i++) {
        buttons[i].classList.add("active");
    }
    hideAll ();
    document.getElementById( elementId: "paypal").style.display = "block";
}
```

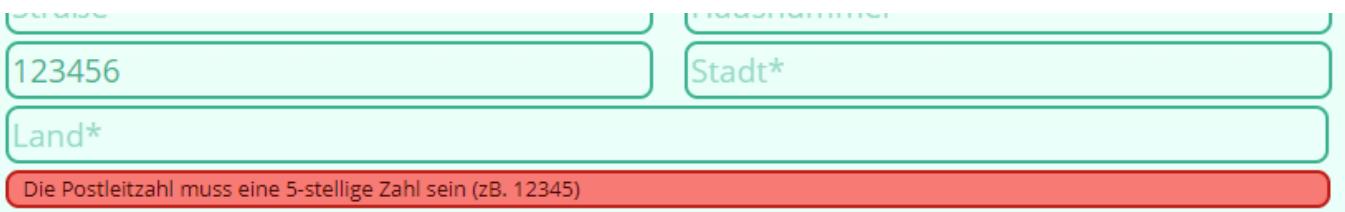
Dabei entsteht ein Problem bei der Validierung: Sobald das Element für die Kreditkarte deaktiviert wird ließ sich das gesamte Formular nicht mehr Absenden. Dies lag daran, dass die Felder in besagtem div-Element weiterhin als „required“ galten und durch die Formvalidation geprüft wurden.  
Dies ließ sich dadurch beheben, dass jedes mal wenn eine andere Bezahlweise als „Kreditkarte“ gewählt wurde bei allen Eingabefeldern das Attribut „required“ entfernt wird und wieder hinzugefügt wird. Sobald wieder die „Kreditkarte“ gewählt wird werden alle Attribute neu gesetzt.

## Checkout

```
function deactivateValidation () {  
    var inputs = document.getElementsByClassName( classNames: "novalid");  
    for(var i = 0; i < inputs.length; i++) {  
        inputs[i].required = false;  
    }  
}  
  
function activateValidation () {  
    var inputs = document.getElementsByClassName( classNames: "novalid");  
    for(var i = 0; i < inputs.length; i++) {  
        inputs[i].required=true;  
    }  
}
```

Der Checkout besteht aus insgesamt 4 div-Elementen, eines für die Lieferadresse und drei weitere für die Bezahlmethoden. Von den drei Bezahlmethoden sind zu jeder Zeit zwei deaktiviert (display: none) und nur das aktuelle Element, dessen Button gedrückt wurde, wird angezeigt.

Sobald ein Button gedrückt wird (zB. „PayPal“) werden einmal alle div-Elemente deaktiviert und das „PayPal“-Element danach separat aktiviert.



The screenshot shows a form with three input fields and one error message. The first field contains '123456' and has a green border. The second field contains 'Stadt\*' and has a green border. The third field contains 'Land\*' and has a green border. Below these fields is a red horizontal bar containing the text 'Die Postleitzahl muss eine 5-stellige Zahl sein (zB. 12345)'.

Zudem wird (aus Zeitgründen nur bei der Eingabe der Postleitzahl) geprüft ob die Eingabe das richtige Format hat. Dabei wird der Inhalt nach der Eingabe ausgelesen und überprüft. Die Visibility eines Span-Elements unterhalb der Form wird dann, je nach Ergebnis der Überprüfung, die Visibility verändert um dem Nutzer auf seinen Fehler hinzuweisen.

```
function checkPLZ () {  
    var plz_content = document.getElementById( elementId: "plz").value;  
  
    if(plz_content.length > 5 || plz_content.length < 5) {  
        document.getElementById( elementId: "plz_warning").style.visibility = "visible";  
    }  
    else if (plz_content.includes( searchElement: "e") || plz_content.includes( searchElement: "E") ||  
    plz_content.includes( searchElement: ",") || plz_content.includes( searchElement: "/") ||  
    plz_content.includes( searchElement: ".") || plz_content.includes( searchElement: "+")) {  
        document.getElementById( elementId: "plz_warning").style.visibility = "visible";  
    }  
    else if (plz_content.match("[0-9]{5}")){  
        document.getElementById( elementId: "plz_warning").style.visibility = "hidden";  
    }  
}
```

# Anteile der Gruppenmitglieder

Simon Weck: 33,33%

Benjamin Jäger: 33,33%

Patryk Watola: 33,33%

Clemens Bork: 0%

Clemens Bork hat sich nicht bei der Programmierung der Website beteiligt.

## Programmierung der Seiten

Artikelseiten - **Patryk Watola**

-Geschichte

-Werte

-Karriere

-Lieferanten

-Fairtrade

-Nachhaltigkeit

-Interview Artikel

Kategorie Seiten - **Benjamin Jäger**

-Über uns

-Transparenz

-Produkte

Produkt Seiten - **Simon Weck**

-Vegan Whey

-Vegan Boost

-Vegan Nutrition

-Vegan Vitamin

Registrierungs Seiten - **Patryk Watola**

-Check-Out

-Kontakt

Sonstige Seiten

-Revitalize Berater -**Simon Weck**

-Warenkorb - **Benjamin Jäger/Simon Weck**

-Impressum - **Patryk Watola**

-Startseite -**Benjamin Jäger**

Navigation Bar -**Benjamin Jäger**

Footer -**Benjamin Jäger**

# Quellen

Alle Quellen der Bilder oder sonstige externe Quellen, die auf der Website genutzt wurden sind, mit entsprechenden Verweisen, in der Design Dokumentation zu finden.



# Revitalize.

**CVD Webtentwicklung 2019**  
**Softwaredokumentation**

Simon Weck #2180135  
Benjamin Jäger #2180031  
Patryk Watola #2180138  
Clemens Bork