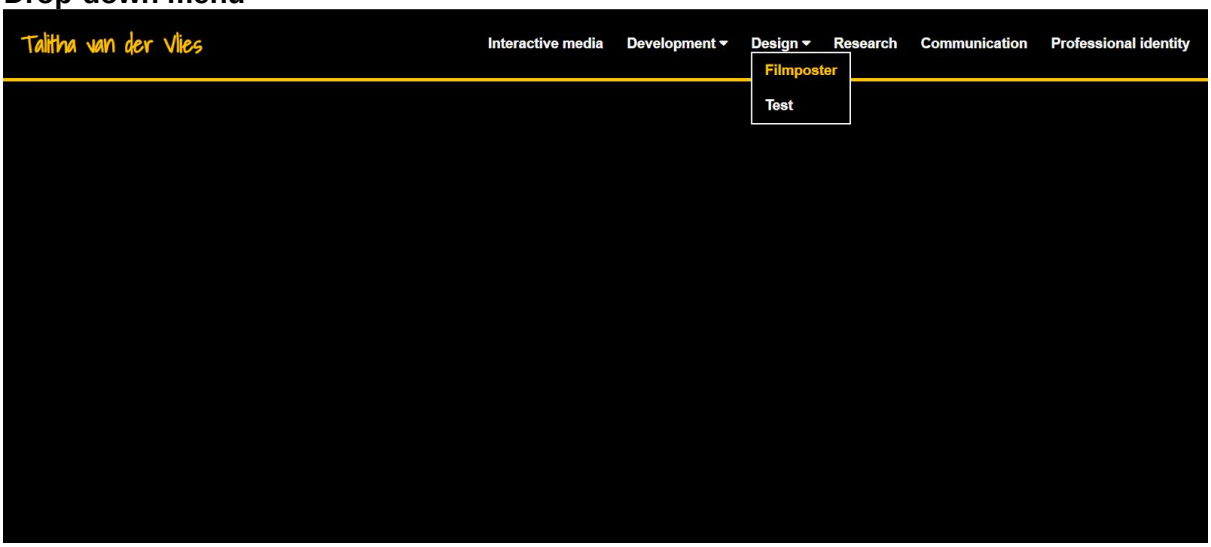
Development portfolio

Lettertype in Safari

```css
/* Tekst */
h1 {
    font-family: 'Shadows Into Light', 'Arial', 'Verdana', 'Times New Roman';
    font-size: 35px;
    color: #ffc400;
    font-weight: lighter;
    font-style: normal;
}
```

Toen ik mijn portfolio website opende op een MacBook, ontdekte ik dat het lettertype Shadows Into Light niet zichtbaar was in Safari. Om dit probleem op te lossen, begon ik te zoeken naar de reden waarom het lettertype niet werkte. Ik kwam erachter dat Safari aanvullende informatie vereist in de CSS. Daarom voegde ik de eigenschappen 'font-weight' en 'font-style' toe. Door deze aanpassingen is het lettertype nu wel zichtbaar in Safari.
Bron: https://stackoverflow.com/questions/24061808/google-font-not-working-on-safari

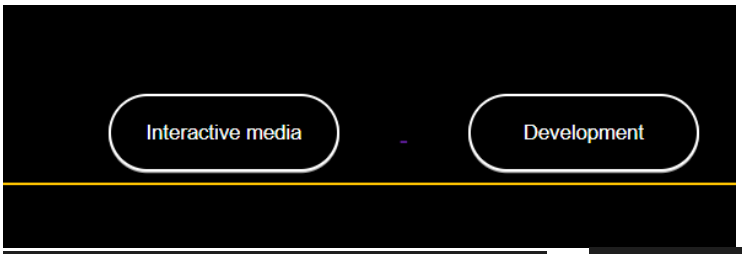
Drop-down menu




```
<li class="dropdown">
    <a href="javascript:void(0)" class="dropbtn">Design <i class="fa fa-caret-down"></i></a>
    <div class="dropdown-content">
      <a href="filmposter.html">Filmposter</a>
      <a href="#test">Test</a>
    </div>
</li>
```

Om het dropdown menu te maken heb ik gebruik gemaakt van W3schools.
Ik heb een list-item gemaakt met de class ‘dropdown’. Het eerste <a>-element is de 'Design' knop met een naar beneden wijzend pijltje. De href is ingesteld op ‘javascript:void(0)’ om ervoor te zorgen dat er niets gebeurt als iemand alleen op ‘Design’ klikt. Daarna heb ik een toegevoegd met de class 'dropdown-content', waarin twee links zijn opgenomen naar ‘Filmposter’ en ‘Test’.
Bron: https://www.w3schools.com/w3css/w3css_dropdowns.asp

Hoverline




```css
a::before {
    content: '';
    position: absolute;
    width: 100%;
    height: 2px;
    background-color: rgb(255, 196, 0);
    bottom: 0;
    left: 0;
    transform-origin: right;
    transform: scaleX(0);
    transition: transform .3s ease-in-out;
  }

a:hover::before {
    transform-origin: left;
    transform: scaleX(1);
}
```

→

```css
.navbar a::before {
    content: '';
    position: absolute;
    width: 100%;
    height: 2px;
    background-color: rgb(255, 196, 0);
    bottom: 0;
    left: 0;
    transform-origin: right;
    transform: scaleX(0);
    transition: transform .3s ease-in-out;
  }

.navbar a:hover::before {
    transform-origin: left;
    transform: scaleX(1);
}
```

Wanneer ik met de muis over de knoppen ging, verscheen er een gele streep onder. Dit gebeurde doordat ik een algemene stijl had toegepast op de tekst in de navigatiebalk, waardoor er een gele streep verscheen. Door '.navbar' ervoor te zetten, werd deze code alleen toegepast op de navigatiebalk.

Tekst in lijn


```css
h2:before,
h2:after {
    content: "";
    flex: 1 1;
    border-bottom: 3px solid #ffc400;
    margin: auto 3.5%;

}
```

Om een lijn met daarin tekst te maken zoals weergegeven in de eerste afbeelding, heb ik de code gebruikt zoals te zien is in de tweede afbeelding. Ik heb de margin aangepast naar mijn voorkeur. De eerste margin is ingesteld op 'auto' om ervoor te zorgen dat de lijn in het midden van de tekst wordt geplaatst. Deze code resulteert in extra ruimte tussen de lijn en de zijkant, waardoor de uitlijning niet meer correct is.
Bron: https://www.geeksforgeeks.org/how-to-make-horizontal-line-with-words-in-the-middle-using-css/

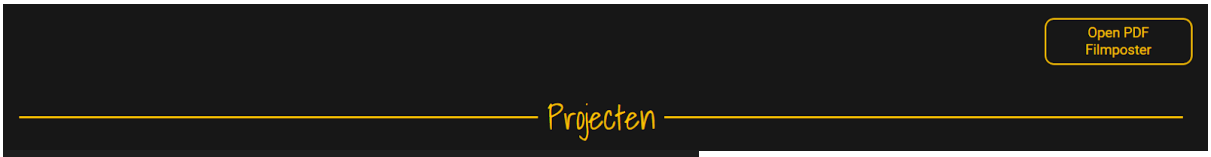
```css
h2:before,
h2:after {
    content: "";
    flex: 1 1;
    border-bottom: 3px solid #ffc400;
    margin: auto 0px;
}

h2::before{
    margin-right: 10px;
}

h2::after{
    margin-left: 10px;
}
```

Om dit op te lossen is er op een andere manier margin toegevoegd. De '::' zorgt ervoor dat de margin ontstaat tussen de lijn en de tekst. Voor de h2 is 10px margin aan de rechterkant van de lijn toegevoegd, na de h2 is 10px margin aan de linkerkant van de lijn toegevoegd. Hierdoor is er wel ruimte tussen de lijn en de tekst, maar er wordt geen extra afstand tussen de zijkant en de lijn toegevoegd.

Button in lijn


```
float: right;
```

Toen ik de knop ‘Open PDF’ toevoegde met de stijl ‘float: right’, werd deze in de lijn weergegeven.



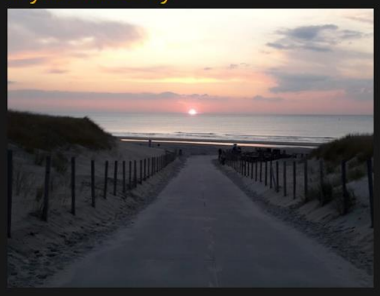

```
<div class="pdfButtonContainer">
    <a href="documentation/filmposter.pdf" target="_blank">
        <button class="pdfButton">Open PDF <br> Filmposter</button>
    </a>
</div>
```

```
.pdfButtonContainer {
    text-align: right;
}
```

Om dit probleem op te lossen, heb ik een extra toegevoegd, waardoor de PDF-knop boven de lijn komt te staan. Door de CSS-eigenschap ‘text-align: right’ toe te voegen, wordt de knop aan de rechterkant van de pagina geplaatst.

Positioneren afbeelding en tekst

Originele afbeelding



Dit is de originele afbeelding, die ik zelf gefotografeerd heb. Ik heb voor deze afbeelding gekozen, omdat ik hierin mijn idee kon uitwerken. Het was voor mij belangrijk om elementen zoals lucht, strand, zee en oppervlakte op de afbeelding te hebben. Ik ben graag op het strand en geniet van mooi weer. Om de afbeelding te transformeren tot een poster, heb ik deze afbeelding verkleind.

```
<div class="widthRowFull">
    <div class="widthFifty">
        <img src="photosPortfolio/smokeEffectBrushVersie4.png" width="70px">
    </div>
    <div class="widthFifty">
        <p>Dit is de smoke vorm die ik heb gebruikt om resultaat van versie IV te bereiken.</p>
    </div>
</div>
```

```
.widthRowFull {
    display: flex;
    flex-direction: row;
    width: 100%;
    padding: 10px 0;
}

.widthFifty {
    width: 50%;
}
```

Ik heb geprobeerd om op verschillende manieren de afbeeldingen en de tekst te positioneren. Omdat ik hier moeite mee had, heb ik hulp gevraagd aan de docent. Hij gaf me de tip om de afbeeldingen en de tekst beide op 50% te schalen nadat ik deze op totaal 100% had ingesteld. Op deze manier creëer ik een consistente positionering voor de hele website, waardoor alles dezelfde uitstraling heeft.

Overlay

```
<div class="containerSocialmedia">
    <img src="photos/gitlab.png" class="image">
        <div class="overlay">
            <img src="photos/gitlabGeel.png" class="image">
        </div>
</div>
```

Ik wil gebruik maken van een hover-effect op de logo's van LinkedIn en GitLab, zodat het duidelijker is dat erop geklikt kan worden. Om dit te bereiken, heb ik een div met de class ‘containerSocialmedia’ gemaakt. Hierin heb ik het witte logo geplaatst. Binnen deze div heb ik een extra div met de class 'overlay' gemaakt, waarin ik het gele logo heb geplaatst. Beide afbeeldingen hebben de class ‘image’ gekregen.

```
.containerSocialmedia {
    position: relative;
    width: 40px;
    height: 40px;
}
```

```
.image {
    display: block;
    width: 100%;
    height: 100%;
    object-fit: contain;
}
```

De ‘containerSocialmedia’ heeft een relatieve positionering en heeft een afmeting van 40px bij 40px. De afbeeldingen met de class 'image' zijn zichtbaar omdat ze zijn ingesteld op ‘display: block'. Ze nemen de volledige breedte en hoogte in van het ouder-element, in dit geval 40x40px, doordat ze zijn ingesteld op ‘width: 100% en height: 100%’. Dankzij het gebruik van ‘object-fit: contain’ wordt de oorspronkelijke vorm van de afbeelding behouden.

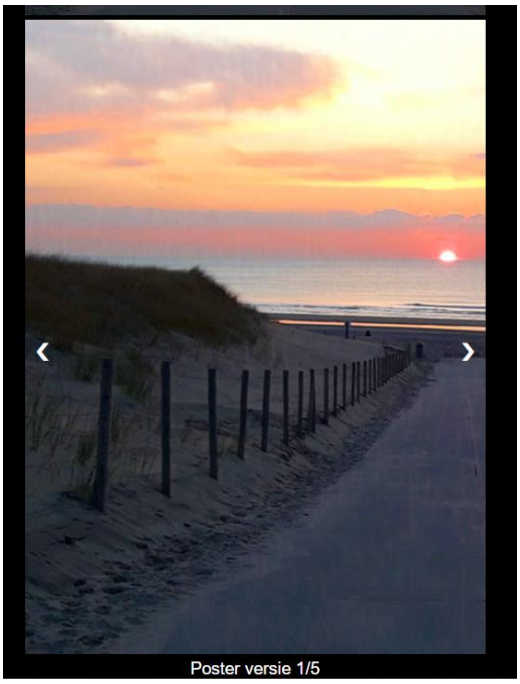
```
.overlay {
    position: absolute;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    height: 100%;
    width: 100%;
    opacity: 0;
    transition: .5s ease;
}
```

Om ervoor te zorgen dat de afbeelding met de class ‘overlay’ onzichtbaar is, wordt de ‘opacity’ ingesteld op 0. De 'position' wordt ingesteld op absolute, zodat de afbeelding over de andere elementen heen kan worden geplaatst. Door de waarden van top, bottom, left en right op 0 te zetten en de waarden van height en width op 100%, wordt deze div exact boven op het bovenliggende element geplaatst. Door de ‘transition’ in te stellen op .5s ease, wordt er een geleidelijke overgang van een halve seconde toegepast wanneer er over het logo gehoverd wordt..

```
.containerSocialmedia:hover .overlay {
    opacity: 1;
}
```

.containerSocialmedia:hover .overlay { opacity: 1; } zorgt ervoor dat wanneer de gebruiker over het element met de class ‘containerSocialmedia’ hovert, de elementen met de class ‘overlay’, een opaciteit van 1 krijgen. Hierdoor wordt de gele afbeelding zichtbaar.

Slideshow



Wanneer ik veel afbeeldingen heb om te laten zien over een specifiek onderwerp, maak ik gebruik van een slideshow om mijn portfolio website overzichtelijk te houden. Om het voor docenten zo eenvoudig mogelijk te maken en te voorkomen dat ze te veel moeten klikken, pas ik slideshows alleen toe wanneer dat nodig is. Dit minimaliseert de hoeveelheid interactie die nodig is om de afbeeldingen te bekijken.

```
<div class="slideshow-container">
    <div class="mySlides fade">
      <img src="afbeeldingen/posterVersie1.png" width="400px">
      <div class="slideshowText">Poster versie 1/5</div>
    </div>

    <div class="mySlides fade">
      <img src="afbeeldingen/posterVersie2.png" width="400px">
      <div class="slideshowText">Poster versie 2/5</div>
    </div>

    <div class="mySlides fade">
      <img src="afbeeldingen/posterVersie3.png" width="400px">
      <div class="slideshowText">Poster versie 3/5</div>
    </div>

    <div class="mySlides fade">
        <img src="afbeeldingen/posterVersie4.png" width="400px">
        <div class="slideshowText">Poster versie 4/5</div>
    </div>

    <div class="mySlides fade">
        <img src="afbeeldingen/posterVersie5.png" width="400px">
        <div class="slideshowText">Poster versie 5/5</div>
      </div>

    <a class="prev" onclick="plusSlides(-1)">&#10094;</a>
    <a class="next" onclick="plusSlides(1)">&#10095;</a>
</div>
```

De slideshow bevindt zich binnen de div met de class 'slideshow-container'. Elke slide, bestaande uit een foto en tekst, is ondergebracht in een aparte div met de class 'mySlides fade'. Binnen deze 'slideshow-container' zijn twee knoppen aanwezig, met de classes 'prev' en 'next', die de functie 'plusSlides' aanroepen met specifieke waarden (1 en -1), waardoor de slideshow van slide kan veranderen.
Bron: https://www.w3schools.com/howto/howto_js_slideshow.asp

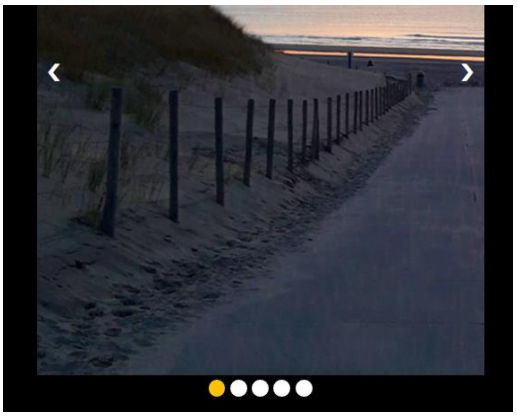
```
let slideIndex = 1;
showSlides(slideIndex);
```
De let slideIndex = 1 definieert een variabele en stelt deze in op 1, wat de huidige slideIndex van de slideshow aangeeft. De showSlides(slideIndex) roept de functie showSlides aan om de eerste afbeelding te tonen.

```
function plusSlides(slideNumber) {
showSlides(slideIndex += slideNumber);
}
```
De functie plusSlides(slideNumber) wordt aangeroepen wanneer er op de knop wordt geklikt om naar de volgende of vorige afbeelding te gaan. Deze functie veranderd de slideIndex door slideNumber erbij op of af te tellen. Vervolgens wordt de showSlides functie aangeroepen om de afbeelding te tonen die bij de nieuwe slideIndex hoort.

```
function showSlides(slideNumber) {
  let z;
  let slides = document.getElementsByClassName("mySlides");
  if (slideNumber === undefined) {slideNumber = 1}
  if (slideNumber > slides.length) {slideIndex = 1}
  if (slideNumber < 1) {slideIndex = slides.length}
  for (z = 0; z < slides.length; z++) {
    slides[z].style.display = "none";
  }
  slides[slideIndex-1].style.display = "block";
}
```
De showSlides(slideNumber) functie wordt gebruikt om de afbeelding te tonen die overeenkomt met het opgegeven slideNumber.

De functie selecteert alle afbeeldingen met de klasse 'mySlides' en verbergt ze door de style.display op none in te stellen. Vervolgens wordt de afbeelding geselecteerd met de opgegeven index en de style.display op block gezet om deze weer te geven.

De if statements in de showSlides functie controleren of de slideNumber parameter binnen het bereik van het aantal afbeeldingen valt en stelt de slideIndex variabele in op de juiste waarde als dat niet het geval is. Dit zorgt ervoor dat de slideshow altijd werkt en niet vastloopt wanneer de gebruiker op een knop klikt om naar een afbeelding te gaan die niet bestaat.



```
<div style="text-align:center">
    <span class="dot" onclick="currentSlide(1)"></span>
    <span class="dot" onclick="currentSlide(2)"></span>
    <span class="dot" onclick="currentSlide(3)"></span>
    <span class="dot" onclick="currentSlide(4)"></span>
    <span class="dot" onclick="currentSlide(5)"></span>
</div>
```

Uiteindelijk heb ik toch besloten om de tekst te veranderen in bolletjes. Hierdoor kan er op een bolletje worden geklikt (bijvoorbeeld op de vierde), waardoor je meteen naar die afbeelding gaat. Er kan dus en op de bolletjes en op de pijltjes geklikt worden.

```
function currentSlide(targetSlideIndex) {
showSlides(slideIndex = targetSlideIndex);
}
```

De functie currentSlide(targetSlideIndex) wordt aangeroepen wanneer er op een van de bolletjes wordt geklikt om direct naar een specifieke afbeelding te gaan. Deze functie stelt de slideIndex in op de waarde van de targetSlideIndex parameter en roept vervolgens de showSlides functie aan om de afbeelding te tonen met de bijgewerkte slideIndex.

```
function showSlides(slideNumber) {
  let i;
  let slides = document.getElementsByClassName("mySlides");
  let dots = document.getElementsByClassName("dot");
  if (slideNumber === undefined) {slideNumber = 1}
  if (slideNumber > slides.length) {slideIndex = 1}
  if (slideNumber < 1) {slideIndex = slides.length}
  for (i = 0; i < slides.length; i++) {
    slides[i].style.display = "none";
  }
  for (i = 0; i < dots.length; i++) {
    dots[i].className = dots[i].className.replace(" active", "");
  }
  slides[slideIndex-1].style.display = "block";
  dots[slideIndex-1].className += " active";
}
```

Ik heb de functie showSlides aangepast zodat de dots werken. Ik heb de variabele 'dots' toegevoegd, die alle elementen met de classname 'dot' ophaalt. Door middel van de for-loop worden alle elementen met de class ‘dot active’ vervangen door de class ‘dot, zodat de dot niet meer geel is’. Tot slot heb ik toegevoegd dat de dot met de huidige 'slidesIndex' wordt gewijzigd naar de class 'dot active', zodat de dot geel wordt.
Bron: https://www.w3schools.com/howto/howto_js_slideshow.asp

Meerdere slideshows gebruiken
Nadat ik meerdere slideshows aan de website had toegevoegd, merkte ik dat ik deze niet onafhankelijk van elkaar kon gebruiken. Het probleem lag in het feit dat de slideIndex van alle slideshows gezamenlijk werd bijgehouden. Dit betekende dat wanneer ik in de tweede slideshow op het pijltje klikte om naar de volgende foto te gaan, hij eerst alle foto's van de eerste slideshow toont, voordat de tweede slideshow verder gaat. Om dit te verhelpen, heb ik de code van de slideshow aangepast.

```
(function() {


  slideshow();
```

Door gebruik te maken van (function () {...}) kan de slideshow getoond worden, zonder een slideIndex te definieren. Hierdoor kunnen de slideshows los van elkaar gebruikt gaan worden. slideshow() toont bij het laden van de pagina de eerste afbeelding en verbergt de rest.

```
function slideshow() {

  parents = document.getElementsByClassName('slideshow-container');

  for (j = 0; j < parents.length; j++) {
    var slides = parents[j].getElementsByClassName("mySlides");
    var dots = parents[j].getElementsByClassName("dot");
    slides[0].classList.add('active-slide');
    dots[0].classList.add('active');
  }
}
```

De functie genaamd 'slideshow' haalt alle HTML-elementen op met de classnaam 'slideshow-container' en slaat ze op in de variabele 'parents'. Vervolgens wordt er een for-loop uitgevoerd voor elk element in 'parents'. In elke herhaling van de loop worden de HTML-elementen met de classnamen 'mySlides' en 'dot' opgehaald. Deze elementen worden vervolgens toegewezen aan de variabelen 'slides' en 'dots'. De eerste afbeelding wordt gemarkeerd als actief door de class 'active-slide' toe te voegen aan het eerste element in de slides array. De eerste dot wordt gemarkeerd als actief door de class 'active' toe te voegen aan het eerste element in de dots array.

```
dots = document.getElementsByClassName('dot');
for (i = 0; i < dots.length; i++) {
  dots[i].onclick = function() {
    slides = this.parentNode.parentNode.getElementsByClassName("mySlides");
    for (j = 0; j < this.parentNode.children.length; j++) {
      this.parentNode.children[j].classList.remove('active');
      slides[j].classList.remove('active-slide');
      if (this.parentNode.children[j] == this) {
        index = j;
      }
    }
    this.classList.add('active');
    slides[index].classList.add('active-slide');
  }
}
```

Om ervoor te zorgen dat gebruikers op de dots kunnen klikken, is het betreffende deel van de code aangepast. Allereerst worden alle elementen met de class 'dot' opgehaald en opgeslagen in de variabele dots. Vervolgens wordt er een loop gestart die door de dots heen loopt. Bij elk element wordt een functie toegevoegd die wordt geactiveerd wanneer erop wordt geklikt. Binnen deze functie worden eerst alle slides opgehaald die zich in dezelfde container als de dot bevinden. Vervolgens wordt er een loop gestart die door de children-elementen van de geselecteerde dot loopt. Voor elk child-element wordt de class 'active' en 'active-slide' verwijderd. Vervolgens wordt de class 'active' toegevoegd aan de geselecteerde dot en de class 'active-slide' toegevoegd aan de slide die bij de dot hoort.

```
  links = document.querySelectorAll('.slideshow-container a');

  for (i = 0; i < links.length; i++) {
    links[i].onclick = function() {
      current = this.parentNode;

      var slides = current.getElementsByClassName("mySlides");
      var dots = current.getElementsByClassName("dot");
      curr_slide = current.getElementsByClassName('active-slide')[0];
      curr_dot = current.getElementsByClassName('active')[0];
      curr_slide.classList.remove('active-slide');
      curr_dot.classList.remove('active');
      if (this.className == 'next') {

        if (curr_slide.nextElementSibling.classList.contains('mySlides')) {
          curr_slide.nextElementSibling.classList.add('active-slide');
          curr_dot.nextElementSibling.classList.add('active');
        } else {
          slides[0].classList.add('active-slide');
          dots[0].classList.add('active');
        }

      }

      if (this.className == 'prev') {

        if (curr_slide.previousElementSibling) {
          curr_slide.previousElementSibling.classList.add('active-slide');
          curr_dot.previousElementSibling.classList.add('active');
        } else {
          slides[slides.length - 1].classList.add('active-slide');
          dots[slides.length - 1].classList.add('active');
        }

      }

    }

  }
})();
```

De functie voor 'prev' en 'next' is aangepast. Eerst worden de elementen met de class ‘slideshow-container a’ geselecteerd en opgeslagen in de variabele ‘links’. Vervolgens wordt de verwijzing naar het parent-element ingesteld met behulp van de variabele ‘current’. De elementen met de class ‘mySlides’ die zich binnen het parent-element bevinden, worden opgeslagen in de variabele ‘slides’. Daarna worden alle elementen met de class ‘dot’ binnen het parent-element opgeslagen in de variabele ‘dots’. De elementen met de class ‘active-slide’ en ‘active’ worden opgeslagen in respectievelijk de variabelen ‘curr_slide’ en ‘curr_dot’. Vervolgens worden de classes ‘active-slide’ en ‘active’ verwijderd van de elementen 'curr_slide' en 'curr_dot'.

Als er wordt geklikt op een element met de class ‘next’, wordt er gecontroleerd of het volgende sibling-element de class ‘mySlides’ bevat. Als dit het geval is, wordt de class van dat element ingesteld op 'active-slide' en 'active'. Als dat niet het geval is, wordt de class ingesteld op de eerste elementen van de slideshow.

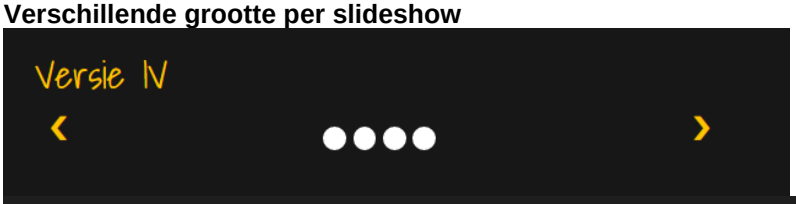
Als er wordt geklikt op een element met de class ‘prev’, wordt er gecontroleerd of het vorige sibling-element de class ‘mySlides’ bevat. Als dit het geval is, wordt de class van dat element ingesteld op 'active-slide' en 'active'. Als dat niet het geval is, wordt de class ingesteld op de laatste elementen van de slideshow.

Daarna wordt er een loop gemaakt die door alle links loopt. Voor elke link wordt een 'onclick' functie toegevoegd, zodat deze geactiveerd wordt wanneer erop geklikt wordt. In deze functie wordt eerst het parent-element opgeslagen als 'current'. Vervolgens worden alle slides en dots die zich bevinden in de parent container opgeslagen als variabelen slides en dots. Daarna worden de slide en de dot die momenteel actief zijn opgeslagen in de variabelen ‘curr_slide’ en 'curr_dot'. Vervolgens worden 'active-slide' en 'active' verwijderd van de curr_slide en curr_dot.

Daarna wordt gecontroleerd of er op de 'next'-knop is gedrukt. Als dat het geval is, dan wordt de volgende slide en dot actief gemaakt. Als er geen volgende slide is, dan wordt de eerste slide en dot actief gemaakt.

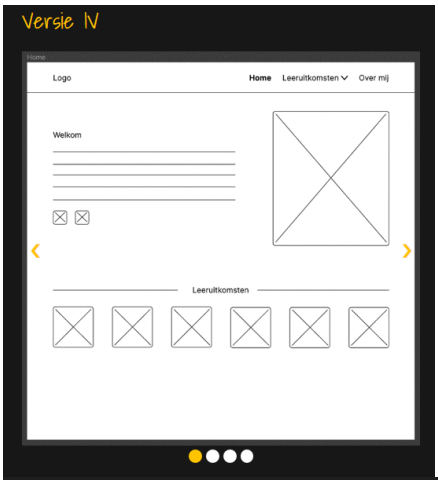
Daarna wordt gekeken of er op de 'prev'-knop is gedrukt. Als dat het geval is, dan wordt de vorige slide en dot actief gemaakt. Als er geen vorige slide is, dan wordt de laatste slide en dot actief gemaakt.

Verschillende grootte per slideshow




```
parents = document.getElementsByClassName('slideshow-container', '.slideshow-containerPortfolio');
```

De reden waarom de code niet werkt voor de classnaam 'slideshow-containerPortfolio' is omdat de getElementsByClassName-methode slechts één argument kan accepteren, één enkele string met de classnaam.



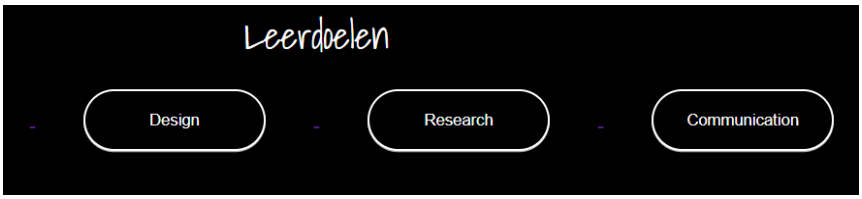

```
parents = document.querySelectorAll('.slideshow-container, .slideshow-containerPortfolio');
```

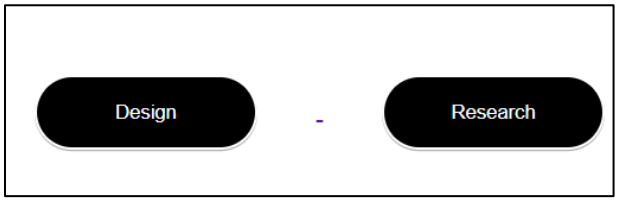
Om meerdere elementen te selecteren met verschillende classnamen, moet ik de querySelectorAll-methode gebruiken en deze een selector geven die alle classnamen bevat die ik wil selecteren.

```
links = document.querySelectorAll('.slideshow-container a', '.slideshow-containerPortfolio a');
links = document.querySelectorAll('.slideshow-container a, .slideshow-containerPortfolio a');
```

De reden waarom de next en prev functionaliteit niet werken voor .slideshow-containerPortfolio is omdat in de links querySelector alleen de eerste selector wordt geselecteerd. Om zowel .slideshow-container als .slideshow-containerPortfolio te selecteren, moet ik de selector veranderen van '.slideshow-container a', '.slideshow-containerPortfolio' a naar '.slideshow-container a, .slideshow-containerPortfolio a'.

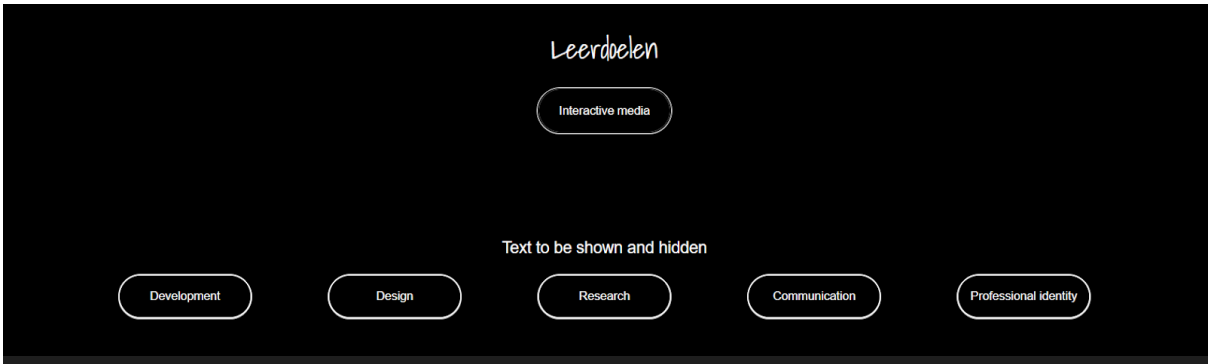
Filterfunctie buttons






```
<a href="filmposter.html">
    <button class="leerdoelen">Design</button>
</a>
```

Op de homepagina heb ik voor elk leerdoel een aparte button aangemaakt, zoals te zien is op de eerste afbeelding. Om deze buttons te verbinden met andere pagina's, heb ik gebruikgemaakt van de code die te zien is op de laatste afbeelding. Door het gebruik van kwamen er streepjes tussen de buttons. Op de afbeelding met de zwarte achtergrond is dit moeilijk te zien. Hierdoor heb ik de achtergrond even wit gemaakt, zodat het duidelijker te zien is.

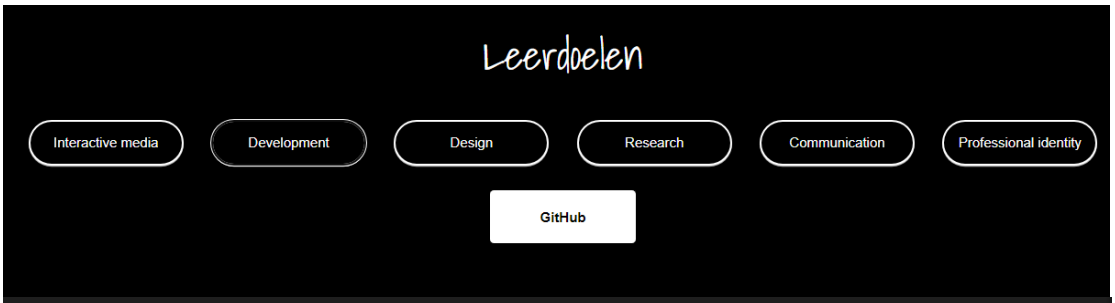



```
<button ONCLICK="ShowAndHide()" class="leerdoelen">Interactive media</button>
<DIV ID="SectionName" STYLE="display:none"><p>Text to be shown and hidden</p></DIV>
```

```
function ShowAndHide() {
  var x = document.getElementById('SectionName');
  if (x.style.display == 'none') {
      x.style.display = 'block';
  } else {
      x.style.display = 'none';
  }
}
```

Door het gebruik van een ONCLICK-functie verdwenen de strepen tussen de buttons. Wanneer er op een button werd geklikt, verplaatsten de andere buttons zich onder de tekst 'text to be shown and hidden'.

De functie genaamd ShowAndHide() zorgt ervoor dat het element met de id 'SectionName' zichtbaar wordt als het verborgen is en onzichtbaar wordt als het al zichtbaar is. Hierdoor kan door middel van een klik op een knop de weergave van het element wijzigen.
Bron: https://html-shark.com/HTML/ShowHideSections.htm




```
<div class="centerText" id="myBtnContainer">
    <button class="buttonLeerdoelen" onclick="filterSelection('interactiveMedia')"> Interactive media</button>
    <button class="buttonLeerdoelen" onclick="filterSelection('development')"> Development</button>
    <button class="buttonLeerdoelen" onclick="filterSelection('design')"> Design</button>
    <button class="buttonLeerdoelen" onclick="filterSelection('research')"> Research</button>
    <button class="buttonLeerdoelen" onclick="filterSelection('communication')"> Communication</button>
    <button class="buttonLeerdoelen" onclick="filterSelection('identity')"> Professional identity</button>
</div>

<div class="centerText">
    <div class="filterDiv interactiveMedia"><button class="buttonUitwerking" onclick="window.location.href='#'">Test 1</button></div>

    <div class="filterDiv development"><button class="buttonUitwerking" onclick="window.location.href='#'">GitHub</button></div>

    <div class="filterDiv design"><button class="buttonUitwerking" onclick="window.location.href='filmposter.html'">Filmposter</button></
    <div class="filterDiv design"><button class="buttonUitwerking" onclick="window.location.href='#'">Test 3</button></div>

    <div class="filterDiv research"><button class="buttonUitwerking" onclick="window.location.href='#'">Test 4</button></div>

    <div class="filterDiv communication"><button class="buttonUitwerking" onclick="window.location.href='#'">Test 5</button></div>

    <div class="filterDiv identity"><button class="buttonUitwerking" onclick="window.location.href='#'">Test 6</button></div>
</div>
```

Met behulp van de functie 'filterSelection' zorg ik ervoor dat verschillende div-elementen kunnen worden getoond. Afhankelijk van de geselecteerde button wordt een specifieke div met bijbehorende inhoud zichtbaar of verborgen gemaakt.
Bron: https://www.w3schools.com/howto/howto_js_filter_elements.asp

```
function filterSelection(leerdoelen) {
  var alleLeerdoelen, i;
  alleLeerdoelen = document.getElementsByClassName("filterDiv");
  for (i = 0; i < alleLeerdoelen.length; i++) {
    RemoveClass(alleLeerdoelen[i], "show");
    if (alleLeerdoelen[i].className.indexOf(leerdoelen) > -1) AddClass(alleLeerdoelen[i], "show");
  }
}
```

De functie ‘filterSelection’ zorgt ervoor dat specifieke div-elementen getoond of verborgen worden op basis van de geselecteerde leerdoelen. Het haalt alle elementen op met de classnaam ‘filterDiv’ en voert vervolgens een loop uit om de class 'show' toe te voegen of te verwijderen, afhankelijk van de overeenkomst met de geselecteerde leerdoelen.

```
function AddClass(element, name) {
  var i, elementClasses, newClasses;
  elementClasses = element.className.split(" ");
  newClasses = name.split(" ");
  for (i = 0; i < newClasses.length; i++) {
    if (elementClasses.indexOf(newClasses[i]) == -1) {
      element.className += " " + newClasses[i];
    }
  }
}
```

In de functie genaamd 'AddClass' worden twee parameters gebruikt, namelijk 'element' en 'name'. Om te beginnen, splitst de classnaam van het element op in een array van klassen met behulp van de methode 'element.className.split(' ')' en slaat deze op in de variabele 'elementClasses'. Op dezelfde manier splitst de parameter 'name' op in een array van klassen en slaat deze op in de variabele 'newClasses'. Daarna start een for-loop waarin elke klasse in 'newClasses' wordt doorlopen. Binnen de loop wordt gecontroleerd of de huidige klasse nog niet aanwezig is in de array 'elementClasses' door gebruik te maken van de voorwaarde 'elementClasses.indexOf(newClasses[i]) == -1'. Indien de

klasse nog niet aanwezig is, wordt deze toegevoegd aan de classnaam van het element door 'element.className += ' ' + newClasses[i]' uit te voeren.

```
function RemoveClass(element, name) {
  var i, elementClasses, newClasses;
  elementClasses = element.className.split(" ");
  newClasses = name.split(" ");
  for (i = 0; i < newClasses.length; i++) {
    while (elementClasses.indexOf(newClasses[i]) > -1) {
      elementClasses.splice(elementClasses.indexOf(newClasses[i]), 1);
    }
  }
  element.className = elementClasses.join(" ");
}
```

Deze code bevat een functie genaamd 'RemoveClass' die twee parameters verwacht: 'element' en 'name'. Het doel van deze functie is om een bepaalde klasse te verwijderen van een HTML-element.

Om dit te bereiken, wordt de bestaande lijst van klassen van het element opgesplitst in een array genaamd 'elementClasses'. Daarna wordt de klasse die verwijderd moet worden ook opgeslagen in een array genaamd 'newClasses'.

Vervolgens wordt er door elke klasse in de 'newClasses' array gelopen, en voor elke klasse wordt gecontroleerd of deze aanwezig is in de 'elementClasses' array. Dit wordt gedaan met behulp van de 'indexOf' methode. Als de klasse gevonden wordt in de 'elementClasses' array, wordt deze verwijderd met behulp van de 'splice' methode.

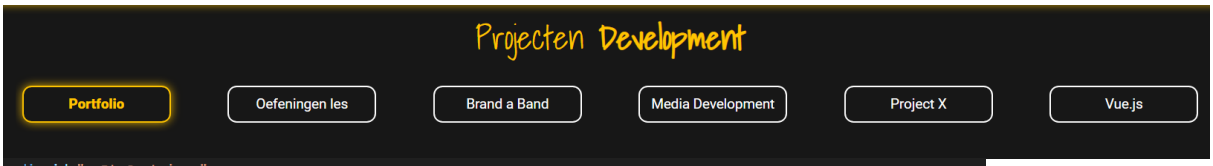
Uiteindelijk wordt de 'className' eigenschap van het HTML-element bijgewerkt met de overgebleven klassen die zijn opgeslagen in de 'elementClasses' array. Dit wordt gedaan door de 'elementClasses' array samen te voegen tot een enkele string met behulp van de 'join' methode, en deze string wordt toegewezen aan de 'className' eigenschap van het element.

```
var btnContainer = document.getElementById("myBtnContainer");
var btns = btnContainer.getElementsByClassName("buttonLeerdoelen");
for (var i = 0; i < btns.length; i++) {
    btns[i].addEventListener("click", function() {
    var current = document.getElementsByClassName("active");
    if (current.length > 0) {
      current[0].className = current[0].className.replace(" active", "");
    }
    this.className += " active";
    });
}
```

Deze code selecteert alle knoppen met de class 'buttonLeerdoelen' die zich binnen een HTML-element met de ID 'myBtnContainer' bevinden. Vervolgens voegt de code een click-functie toe aan elk van de geselecteerde knoppen.

Wanneer een knop wordt geklikt, wordt er eerst gezocht naar het HTML-element met de klasse 'active' met behulp van de 'getElementsByClassName' methode en de lengte van het resulterende object wordt gecontroleerd. Als er een element met de klasse 'active' wordt gevonden, wordt deze klasse van het element verwijderd met behulp van de 'replace' methode.

Na de klik wordt de 'active' toegevoegd aan de betreffende knop door gebruik te maken van de 'className' eigenschap. Dit wordt gedaan door het toevoegen van de string 'active' aan het einde van de huidige klassen die al zijn toegewezen aan de knop met behulp van de ' += ' operator.




```
<div id="myBtnContainer">
    <h1>Projecten <b>Development</b></h1>
        <div class="RowMyBtnContainer">
            <button class="buttonLeerdoelen active" onclick="filterSelection('showThisButton')"> Portfolio</button>
            <button class="buttonLeerdoelen" onclick="filterSelection('oefeningen')"> Oefeningen les</button>
            <button class="buttonLeerdoelen" onclick="filterSelection('brandaBand')"> Brand a Band</button>
            <button class="buttonLeerdoelen" onclick="filterSelection('mediaDevelopment')"> Media Development</button>
            <button class="buttonLeerdoelen" onclick="filterSelection('projectX')"> Project X</button>
            <button class="buttonLeerdoelen" onclick="filterSelection('vueJS')"> Vue.js</button>
        </div>
</div>

<div class="filterDiv showThisButton">

</div>
```

De buttons zijn aangepast, zodat er nu tekst verschijnt in plaats van buttons.

```
function filterSelection(leerdoelen) {
    var alleLeerdoelen, i;
    alleLeerdoelen = document.getElementsByClassName("filterDiv");
    for (i = 0; i < alleLeerdoelen.length; i++) {
    RemoveClass(alleLeerdoelen[i], "show");
    if (alleLeerdoelen[i].className.indexOf(leerdoelen) > -1) AddClass(alleLeerdoelen[i], "show");
}

scrollToTop();
}
```

Er is een scrollToTop()-functie toegevoegd, waardoor gebruikers niet handmatig naar boven hoeven te scrollen nadat ze de pagina hebben bekeken. Nadat de loop is voltooid, wordt de functie 'scrollToTop' aangeroepen, waardoor de pagina automatisch naar de bovenkant van het scherm wordt gescrold indien er op een button wordt geklikt.

```
function scrollToTop() {
window.scrollTo({ top: 0, behavior: 'smooth' });
}
```

De functie scrollToTop wordt gebruikt om naar de bovenkant van de pagina te scrollen. In JavaScript maakt deze functie gebruik van de window.scrollTo-methode, die het mogelijk maakt om naar een specifieke positie op de pagina te scrollen. Door de parameter van scrollTo op top 0 in te stellen, wordt er naar de bovenkant van de pagina gescrold. De behavior smooth zorgt ervoor dat er vloeiend door de pagina wordt gescrold.

```
filterSelection("showThisButton")
function filterSelection(leerdoelen) {
  var alleLeerdoelen, i;
  alleLeerdoelen = document.getElementsByClassName("filterDiv");
  for (i = 0; i < alleLeerdoelen.length; i++) {
    RemoveClass(alleLeerdoelen[i], "show");
    if (alleLeerdoelen[i].className.indexOf(leerdoelen) > -1) AddClass(alleLeerdoelen[i], "show");
  }
}
```

Om ervoor te zorgen dat er altijd een actieve button is en bij de bijbehorende button tekst te zien is, heb ik filterSelection('showThisButton') toegevoegd.

Animatie effecten - Buttons leeruitkomsten

Ik heb de inspiratie opgedaan om dit effect toe te passen op mijn portfolio website via de website van Vegan Heroes. In de footer van hun website gaat dit icoon bewegen als een ‘heartbeat’ wanneer je eroverheen beweegt met je muis.
Bron: https://www.veganheroes.nl/

Vervolgens heb ik een YouTube video bekeken en de stappen gevolgd om het effect te kunnen toepassen op mijn portfolio website. Ik heb ervoor gekozen om dit effect op de ‘Buttons Leerdoelen’ toe te passen. De bron van de YouTube video die ik heb gebruikt is: https://www.youtube.com/watch?v=D2NDORRN4sU.

```
<div class="leeruitkomst">
    <img src="interactiveMediaOverlay.png" width="170px">
</div>
```

```
.leeruitkomst{
    position: absolute;
    top:50%;
    left:50%;
    animation: leeruitkomst 1.5s infinite;
}

@keyframes leeruitkomst {
    0%{
        transform: scale(1);
    }
    50%{
        transform: scale(1.4);
    }
}
```

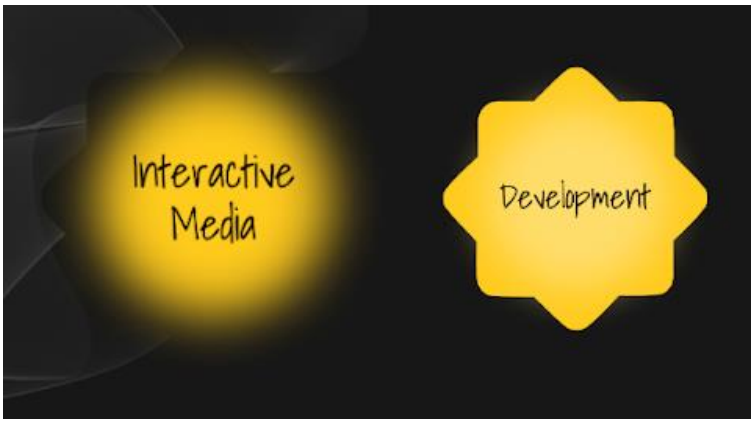
Ik heb de code van de YouTube video aangepast naar mijn eigen wensen, zodat de button op een mooie manier zou bewegen.

De ‘animation’ eigenschap geeft aan dat de 'leeruitkomst' animatie moet worden uitgevoerd, 1.5 seconde duurt en ‘infinite’ geeft aan dat de animatie eindeloos moet worden herhaald.

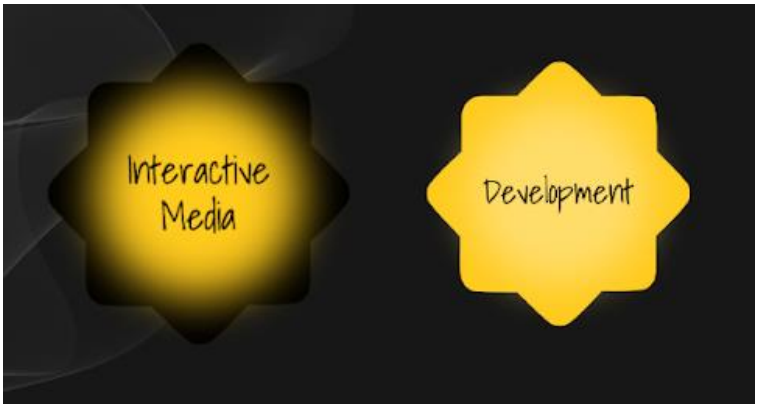
In de ‘@keyframes leeruitkomst’ worden de stappen van de animatie beschreven. De animatie begint met een schaal van 1 (de standaardgrootte van het element). Na 50% van de animatie, wordt het element vergroot tot een schaal van 1.4. Bij 100% van de animatie is het element terug bij zijn oorspronkelijke grootte.

```css
.leeruitkomst{
    position: absolute;
    top:50%;
    left:50%;
}

.leeruitkomst:hover{
    animation: leeruitkomst 1.5s infinite;
}

@keyframes leeruitkomst {
    0%{
        transform: scale(1);
    }
    50%{
        transform: scale(1.4);
    }
}
```

Door het toevoegen van .leeruitkomst:hover, wordt de button alleen geanimeerd wanneer de muis er overheen beweegt.

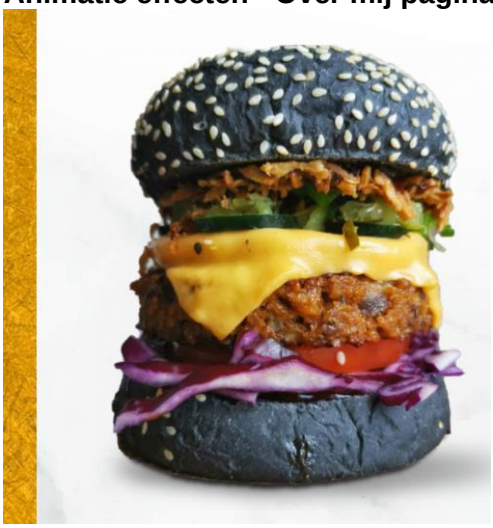


De zwarte kleur is door het animatie effect niet meer goed zichtbaar. Dit was namelijk dezelfde zwarte kleur als de achtergrond.



Hierdoor heb ik de kleur aangepast naar volledig zwart, waardoor het animatie effect nu wel opvalt.

Animatie effecten - Over mij pagina



Ik heb mijn inspiratie voor het toepassen van dit effect op mijn portfolio website gevonden op de website van Vegan Heroes. Wanneer je op hun pagina naar beneden scrollt, verschijnt er een hamburger die vanaf de linkerkant de pagina binnenkomt. Dit blijft zichtbaar totdat de pagina opnieuw wordt geladen.
Bron: https://www.veganheroes.nl/

Dit effect wilde ik toepassen op mijn skate foto op de 'Over mij'-pagina om de nadruk te leggen op de achtergrond. Om de achtergrond verder op te laten vallen, wilde ik de tekst laten verschijnen.

```css
.skatefoto {
    margin-top: 80px;
    position: relative;
    animation: slide-in 5s ease 0.5s 1 alternate forwards;
    transform: translateX(-350px);
}

@keyframes slide-in {
    to {
        opacity: 1;
        transform: translateX(0);
    }
}
```

In de ‘animation’ eigenschap worden verschillende waarden gebruikt om de animatie vorm te geven. De animatie duurt 5 seconden, begint na een vertraging van 0.5 seconden en heeft een timing-functie van ‘ease’ voor een vloeiende overgang. De animatie wordt slechts één keer afgespeeld, totdat de pagina opnieuw wordt geladen. De forwards-optie onthoud de eindpositie van de animatie en zorgt dat de afbeelding hier blijft staan. Daarnaast wordt de ‘transform‘ eigenschap gebruikt om de afbeelding bij de start van de animatie 350 pixels naar links te verschuiven met ‘transform: translateX(-350px)’.

Dit wordt gevolgd door de ‘@keyframes slide-in’ regel die beschrijft welke veranderingen er moeten plaatsvinden tijdens de animatie. In dit geval begint de animatie bij de startpositie en eindigt bij de uiteindelijke positie met ‘transform: translateX(0);’ waar de afbeelding niet meer verschoven is. De ‘opacity’ eigenschap blijft ongewijzigd en behoudt de standaardwaarde van 1 (volledig zichtbaar).
Bron: https://www.w3schools.com/css/css3_animations.asp


```
.aboutText {
    margin-top: 150px;
    animation: fadeIn 10s;
}

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

Er wordt een animatie genaamd 'fadeIn' toegevoegd aan de .aboutText class die 10 seconden duurt. De ‘fadeIn’ animatie begint op 0% van de duur, waarbij de opacity van het element op 0 wordt gezet, wat betekent dat het element niet zichtbaar is. Op 100% van de animatieduur, wordt de opacity op 1 gezet, waardoor het element volledig zichtbaar wordt. Dit zorgt voor een geleidelijke overgang van een onzichtbaar element naar een volledig zichtbaar element in 10 seconden.
Bron: https://blog.hubspot.com/website/css-fade-in