Box

Handout Cursist

CSS handout

In deze handout vind je alle CSS opdrachten terug die je moet maken. Het is de bedoeling dat je bij het beantwoorden van de vragen zo uitgebreid mogelijk te werk gaat en altijd ervoor zorgt dat je goed research hebt gedaan. Om geordend te werk te gaan willen we graag dat je werkt met een goed opgebouwde mappenstructuur. Dus elk onderwerp in een eigen map en elke opdracht van elk onderwerp een eigen map. ***Voorts dien je elke opdracht uit te werken in een apart html document***. Je HTML pagina structuur ziet er als volgt uit:

Je hebt aldus een main page waar alle opdrachten in zijn gelinkt. Elke link verwijst naar een aparte HTML pagina waarin je desbetreffende opdracht hebt uitgewerkt. Wanneer je alle CSS opdrachten heb afgerond dan zorg je ervoor dat het gehele CSS onderdeel wordt geupload naar je Github repository.

Inhoud

1. Selectors & Visual rules

In CSS worden selectors gebruikt om de HTML-elementen op een webpagina te targeten die we willen stijlen. Er is een grote verscheidenheid aan CSS-selectors beschikbaar, die een grote precisie mogelijk maken bij het selecteren van elementen om te stijlen.

Een CSS-selector is het eerste deel van een CSS-regel. Het is een patroon van elementen en andere termen die de browser vertellen welke HTML-elementen moeten worden geselecteerd om de CSS-eigenschappen binnen de regel toe te passen. Het element of de elementen die door de selector worden geselecteerd, worden 'the subject of the selector' genoemd. Hieronder zie je het h1 en p element die als selectors worden gebruikt.

```
h1 {
  color: blue;
  background-color: yellow;
}

p {
  color: red;
}
```

Lees meer over selectors in de officiele documentatie.

Opdrachten

Opdracht 1
Je kunt op 3 verschillende manieren CSS in je HTML inladen. Op welke 3 manieren kun je dat doen? Leg ook voor elke wijze uit wat het precies inhoud.

Opdracht 2
CSS pas je toe met een bepaalde syntax. Hoe ziet zo'n syntax eruit? Kun je ook uitleggen wat elk element uit de syntax betekent?

Opdracht 3
Er zijn verschillende selectors, bijvoorbeeld;

- type selectors
- descendant selectors
- class selectors

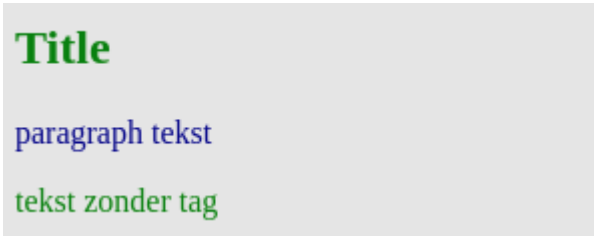
Maak van de bovengenoemde selectors voorbeelden. Geef in je antwoord ook aan wat de voordelen zijn van elke selector.

Opdracht 4

Maak een aparte html bestand en voeg de volgende code toe:

```
<div>
    <h2>Title</h2>
    <p>paragraph tekst</p>
    tekst zonder tag
</div>
```

Maak het onderstaande na door gebruik te maken van selectors. De kleuren die hiervoor gebruikt zijn is green en darkblue.

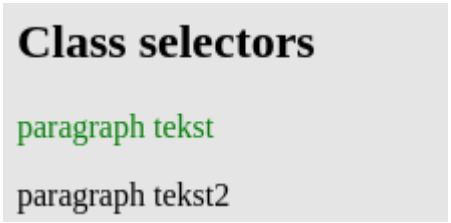


Opdracht 5

Maak een aparte html bestand en voeg de volgende code toe:

```
<h2>Class selectors</h2>
<p>paragraph tekst</p>
<p>paragraph tekst2</p>
```

Maak het onderstaande na door gebruik te maken van selectors. De kleuren die hiervoor gebruikt zijn is green en black.



Opdracht 6

Verder heb je ook nog de volgende selectors:

- child selectors
- adjacent selectors
- general selectors

Maak in een aparte html bestand voor elke selector een voorbeeld en maak een link in je antwoord naar dat html bestand. Geef in je antwoord ook aan wat de voordelen zijn van elk selector.

Opdracht 7

Maak nu een html bestand aan met voorbeelden van alle soorten selectors erin. Maak een link in je antwoord die naar dat html bestand verwijst.

Opdracht 8

Styling die je toepast wordt toegepast in een bepaalde volgorde. Het kan voorkomen dat je styling toepast en vervolgens niet het gewenste resultaat ziet. ***Cascade*** en ***Inheritance*** zijn belangrijke concepten om te begrijpen hoe styling werkt. Lees hier in de officiële documentatie meer over cascade en inheritance.
Zoek ook andere referenties om het concept te begrijpen.

Leg in eigen woorden uit wat met cascade en inheritance wordt bedoeld. Maak voorbeelden om je antwoord uit te leggen.

Meer weten?
Je kunt meer informatie vinden over deze concepten in deze video

2. Boxmodel

Alles in CSS heeft een zogenaamde ‘box’ eromheen en het begrijpen van deze ‘boxen’ is de sleutel om lay-outs met CSS te kunnen maken of om items met andere items uit te lijnen. In deze les zullen we het CSS-box-model goed bekijken zodat je complexere layouts kunt bouwen met een beter inzicht in hoe het werkt en de terminologie die daarmee verband houdt. In CSS wordt de term "box model" gebruikt wanneer we het dus hebben over ontwerp en lay-out. Elke ‘box’ in CSS bestaat uit: ‘margins’, ‘borders’, ‘padding’, en de ‘content’ oftewel de inhoud. De onderstaande afbeelding illustreert het box-model. Officiële documentatie over het box-model vind je hier op deze website.

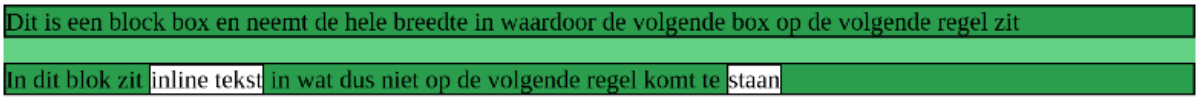
Opdrachten

Opdracht 1

Wat zijn de eigenschappen van block boxes en inline boxes?

Opdracht 2

Maak wat hieronder staat met paragraph en span tags.

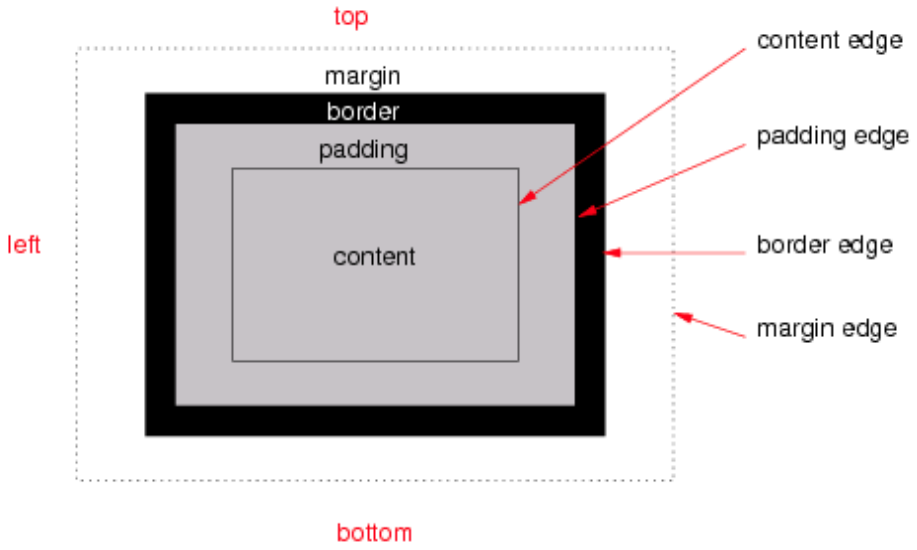


Opdracht 3

Wat gebeurt er als je een width en een height toevoegt aan de span tag ( ) van vorige opdracht?

Opdracht 4

Hieronder zie je een box-model



Leg uit wat de volgende termen betekenen: content, padding, margin en border.

Opdracht 5

Als je ruimte wilt maken tussen de border en content, welke css property gebruik je dan? Maak een voorbeeld in je antwoord.

Opdracht 6

Als je ruimte wilt maken tussen de border en buiten de box, welke css property gebruik je dan? Maak een voorbeeld in je antwoord.

Opdracht 7

Maak nu een blok met width: 100px, height: 100px, margin: 5px, padding 5px en border-width: 5px. Zie hieronder:

1. Hoe breed is de box?
2. Hoe hoog is de box?
3. Wat merk je op met width en height?

Opdracht 8

Welke CSS property kun je gebruiken om ervoor te zorgen dat de waarde van width en height de totale breedte en hoogte is van het blok van de voorgaande opdracht?

Opdracht 9

Maak een aantal voorbeelden van boxen en maak gebruik van steeds verschillende margin, padding en border properties.

3. Display & Positioning

De CSS ‘position’ property bepaalt hoe een HTML element in een document wordt gepositioneerd. Je kunt elk HTML-element op elke gewenste locatie plaatsen. Je kunt bijvoorbeeld aangeven of je het element relatief ten opzichte van de standaardpositie op de pagina wilt plaatsen of absoluut op basis van het bovenliggende element. De top, right, bottom en left properties bepalen de uiteindelijke locatie van geplaatste elementen. Officiële documentatie over display & positioning vind je hier op deze website.

Opdrachten

Opdracht 1

Welke position properties zijn er? Leg voor elke position property uit wat het doet. Geef ook aan welke default position waarde elementen hebben die je gebruikt in html.

Opdracht 2

Zie de onderstaande HTML code.

```
<h1>Dit is een titel</h1>
<p>Dit is een paragraaf.</p>
<p>Dit is nog een paragraaf.</p>
```

Hoe zorg je er voor dat het <h1> element altijd 50px van de top en 50px van de rechterkant van het scherm zichtbaar is. Maak het <h1> element rood door een RGB kleurcodering toe te passen.

Opdracht 3

Zorg er nu voor dat het <h1> element uit de voorgaande opdracht 20px links en 30px naar onder wordt geplaatst, relatief t.o.v. diens normale positie.

Opdracht 4

Zorg er nu voor dat het <h1> element uit opdracht 250px links en 100px van de top wordt geplaatst relatief t.o.v. de HTML pagina.

Opdracht 5

Positioneer het <img> element achter de tekst. Zie voorbeeld hierboven en gebruik onderstaande HTML code. Je mag zelf een eigen image gebruiken.

```
<h1>Dit is een titel</h1>
<p>Dit is een paragraaf.</p>
<p>Dit is nog een paragraaf.</p>
<img src="./bg-image-colors.jpg"/>
```

Opdracht 6

Maak het onderstaande na door gebruik te maken van relative position. De outer div block is 100px breed en 100px hoog. De inner div block is 50px breed en 50px hoog.

Opdracht 7

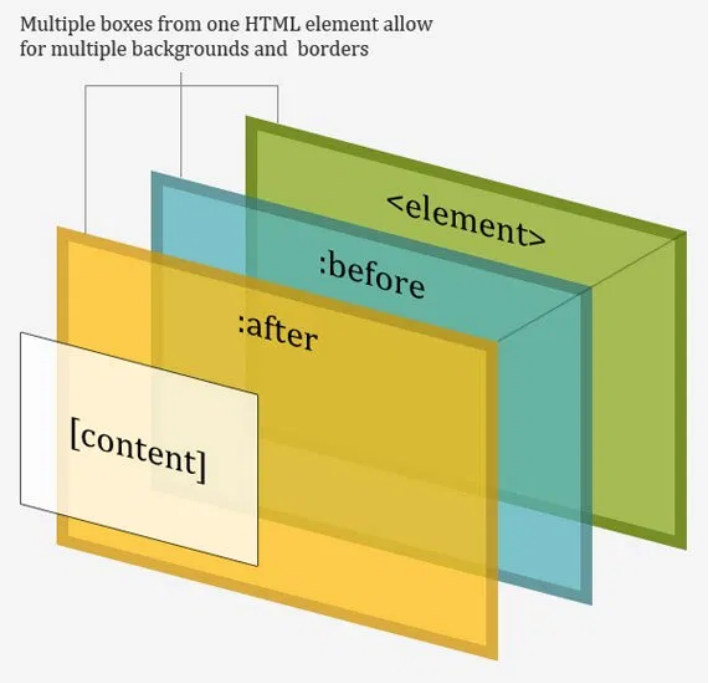
Zie de onderstaande image met tekst. Kun jij dit namaken? Gebruik zelf een foto van een zonnige bestemming. Gebruik position, maar je mag geen z-index toepassen.

Opdracht 8

Maak het onderstaande na. Je mag zelf een image zoeken en gebruiken.

4. Pseudo elements

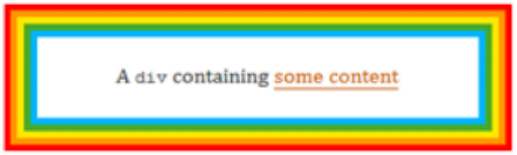
In CSS worden stijlen normaal gedefinieerd ten behoeve van een element. Soms is het echter wenselijk effecten te bereiken, die niet mogelijk zijn als je alleen beschikt over element- of attribuut-selectors. Bijvoorbeeld het in een bepaalde opmaak weergeven van de eerste letter of de eerste regel van de inhoud van een element. Om dat soort effecten mogelijk te maken, zijn pseudo-elementen geïntroduceerd. Een pseudo-element kun je zien als een denkbeeldig element, dat weliswaar niet in het document voorkomt, maar waarvoor je wel een stijl kunt definiëren.
Het is supergaaf wat je kunt doen met pseudo-elementen. Ze ontgrendelen een heleboel interessante ontwerpmogelijkheden zonder de semantiek van de opmaak negatief te beïnvloeden. De meest gebruikte pseudo-elementen zijn '**:: before**' en '**:: after**'. In het plaatje hieronder kun je zien hoe een pseudo-element werkt.

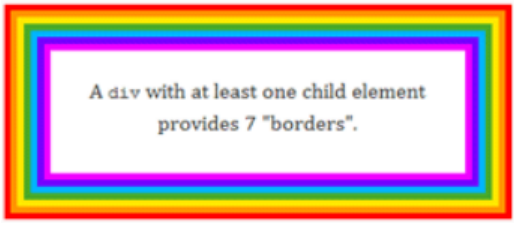


Omdat je pseudo elementen absoluut kunt positioneren, relatief t.o.v. de 'parent' element, kun je ze zien als twee extra lagen waar je mee kunt spelen voor elk element. Er zijn tal van mogelijkheden met pseudo-elementen, bijvoorbeeld meervoudige borders maken of meervoudige achtergronden. Op de website van Nicolas Gallagher kun je zien wat er nog meer mogelijk is met pseudo-elementen. Hieronder nog enkele voorbeelden van de mogelijkheden.

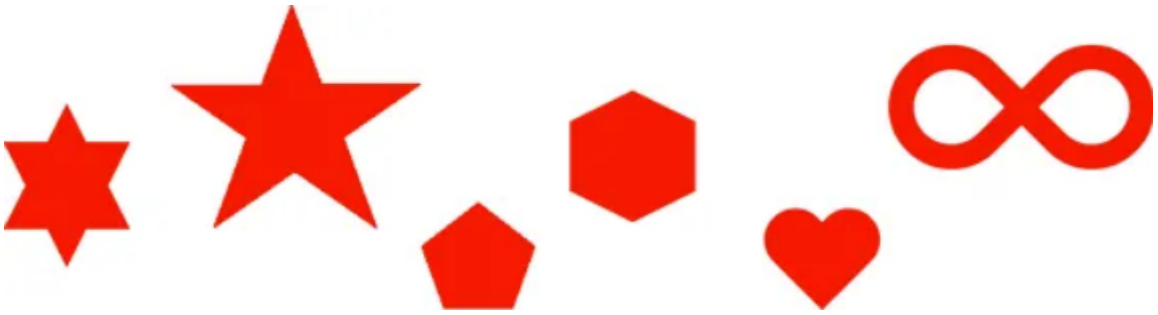
Meervoudige borders

Meervoudige achtergronden





Allerlei vormen



In de officiële documentatie kun je meer informatie terugvinden over pseudo elementen. Zie de volgende links: Pseudo elements en Pseudo classes.

Opdrachten

Opdracht 1

Kun je enkele voorbeelden geven van pseudo-elementen (**minimaal 4**)? Beschrijf met je eigen woorden wat deze pseudo-elementen inhouden.

Opdracht 2

Er zijn 14 verschillende pseudo elementen. Je hebt in de voorgaande opdracht reeds 4 kunnen benoemen. Kun je de andere 10 ook benoemen en aangeven wat ze precies doen?

Opdracht 3

Er zijn aldus 14 verschillende pseudo elementen. Je hebt ze in de voorgaande opdracht reeds opgezocht en uitgezocht wat ze doen. Kun je van al deze pseudo-elementen een voorbeeld maken?

Opdracht 4

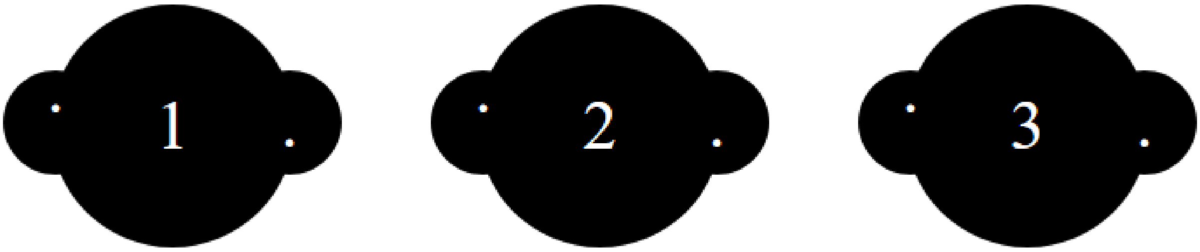
Hieronder vind je een voorbeeld van het gebruik van een pseudo-element. Kun jij dit namaken?

Opdracht 5

Hieronder vind je nog eens een voorbeeld van het gebruik van een pseudo-element ::before. Kun jij dit namaken? Zoek een afbeelding op internet.

Opdracht 6

Hieronder vind je wederom een voorbeeld van het gebruik van een pseudo-element. Kun jij dit namaken?



Opdracht 7

De laatste opdracht vergt wat creativiteit. Zie het onderstaande voorbeeld en maak het na.

5. Flexbox

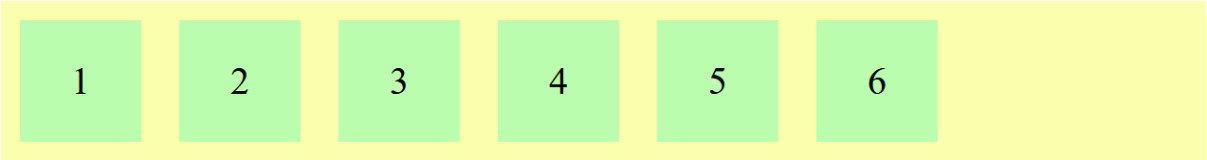
Flexbox is een CSS model voor het positioneren van HTML-elementen. Het is een ideale techniek voor de ontwikkeling van een responsive layout. Het gebruik van 'float' en 'absolute' positioning is met deze techniek verleden tijd. Het grote voordeel van Flexbox is dat het een volwaardig CSS model is die je volledige controle geeft over de zaken die belangrijk zijn bij een responsive layout van je webpagina. Denk daarbij bijvoorbeeld aan de uitlijning en sortering van HTML-elementen. Daarnaast heb je geen onnodige divs of clears meer nodig.

Het belangrijkste idee achter de flexibele lay-out is om de container de mogelijkheid te geven de breedte / hoogte (en volgorde) van de items te wijzigen om de beschikbare ruimte zo goed mogelijk te vullen (meestal voor alle soorten displays en schermformaten). Een flexibele container breidt items uit om beschikbare vrije ruimte te vullen of verkleint ze om overflow te voorkomen.
Het belangrijkste is dat de lay-out van de flexbox niet richtingsafhankelijk is, in tegenstelling tot de reguliere lay-outs (block dat verticaal is gebaseerd en inline block dat horizontaal is gebaseerd). Hoewel die goed werken voor pagina's, missen ze flexibiliteit om grote of complexe toepassingen te ondersteunen (vooral als het gaat om het veranderen van oriëntatie, formaat, uitrekken, verkleinen, enz.). Om meer te lezen over flexbox kun je de volgende websites bezoeken van CSS-tricks en Mozilla developer.

Opdrachten

Opdracht 1

Zie het onderstaande voorbeeld en maak het na met flex. Gebruik **geen** width, height of line-height.

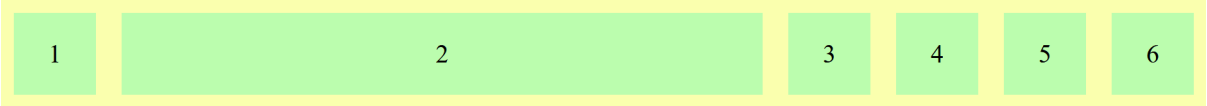


Opdracht 2

Zie het onderstaande voorbeeld, de items staan nu anders uitgelijnd. Maak het na met flex. Gebruik **geen** width, height of line-height.

Opdracht 3

Zie het onderstaande voorbeeld, een van de items is breder dan de rest. Maak het na met flex. Gebruik **geen** width, height of line-height.



Opdracht 4

De property ‘justify-content’ is een property die door flex gebruikt wordt. Maak verschillende voorbeelden door gebruik te maken van de justify-content property.

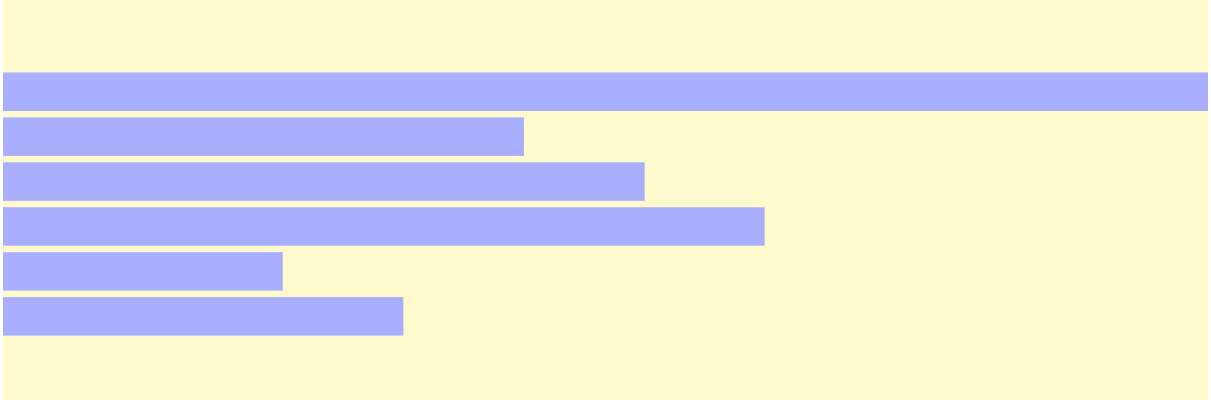
Opdracht 5

De property ‘align-items’ is een property die door flex gebruikt wordt. Maak verschillende voorbeelden door gebruik te maken van de align-items property.

Opdracht 6

Maak het onderstaande na met flexbox. Properties die je o.a dient te gebruiken zijn:

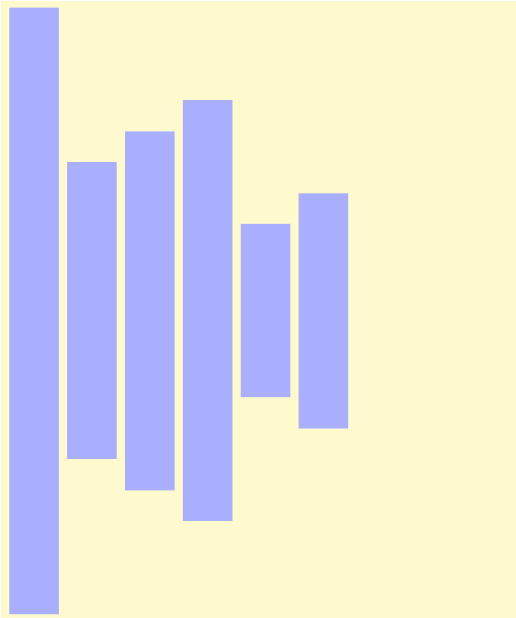
- display
- justify-content
- flex-direction



Opdracht 7

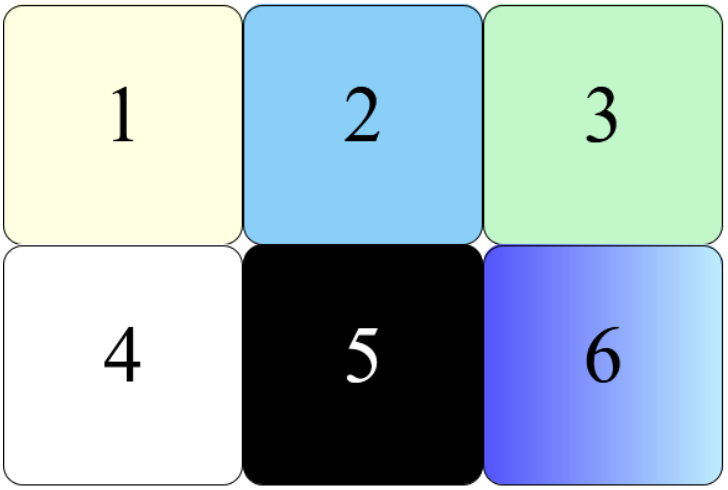
Maak het onderstaande na met flexbox. Properties die je o.a dient te gebruiken zijn:

- display
- justify-content
- flex-direction
- align-items



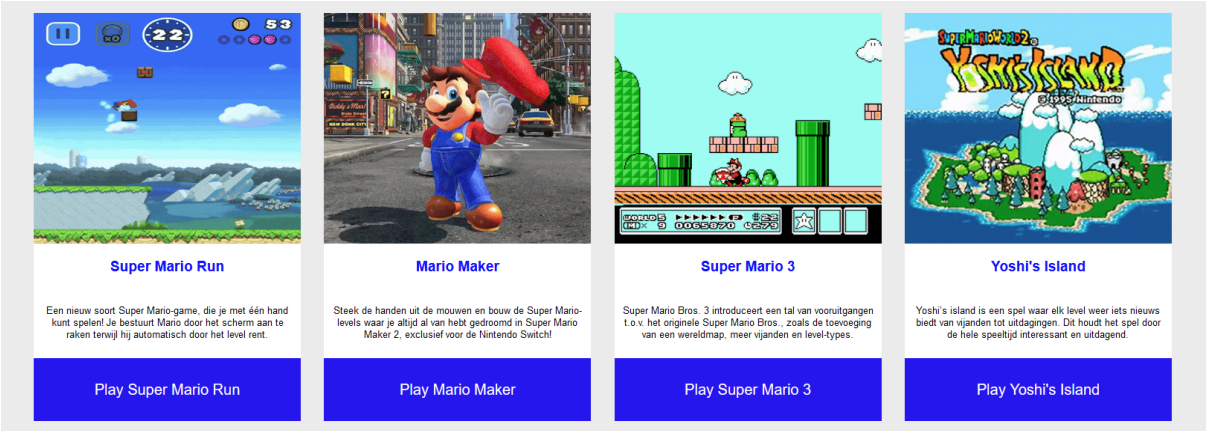
Opdracht 8

Zie het onderstaande voorbeeld en maak het na met flexbox.



Opdracht 9

Zie het onderstaande voorbeeld en maak het na met flexbox. Zorg voor een leuk effect als je met de muis over de ‘play’ knop heen gaat. Optioneel: gebruik .gif bestanden als images voor een levendig effect.



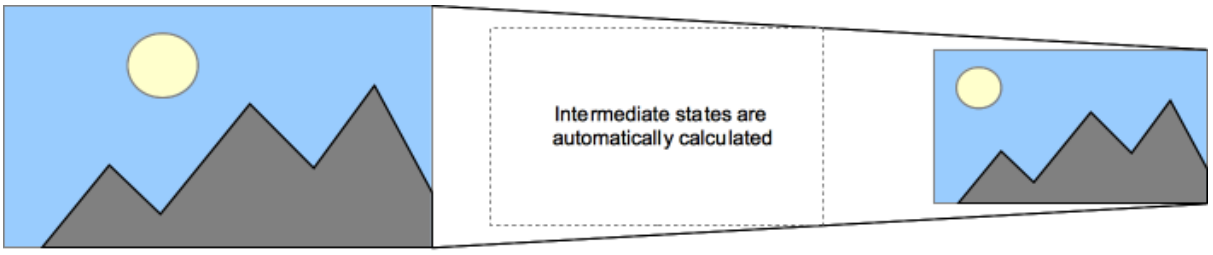
Opdracht 10

Verzin zelf een opdracht met flexbox. Irriteer je medecursist ermee door je medecursist te vragen om het na te maken.

6. Transitions

CSS-transitions bieden een manier om de animatiesnelheid te regelen bij het wijzigen van CSS-properties. In plaats van het onmiddellijk wijzigen van de properties, kun je ervoor zorgen dat de wijzigingen in een property gedurende een bepaalde periode plaatsvinden. Als je bijvoorbeeld de kleur van een element verandert van wit naar zwart, dan is de verandering meestal onmiddellijk. Als CSS-transitions zijn toegepast, vinden er wijzigingen plaats met tijdsintervallen die een versnellingscurve volgen, die allemaal kunnen worden aangepast naar wens.

Animaties die een transitie tussen twee toestanden met zich meebrengen, worden vaak 'implicit transitions' genoemd, omdat de staat tussen de start- en eindstaat impliciet wordt gedefinieerd door de browser.



Met CSS-transitions kun je zelf beslissen welke properties jij wilt animeren (door ze expliciet te vermelden), wanneer de animatie zal starten (door een 'delay' in te stellen), hoe lang de transitie zal duren (door een 'duration' in te stellen) en hoe de transitie zal verlopen (door de 'timing function' te definiëren, bijv. 'linear' of 'ease-in-out').

Het is goed om te weten dat niet alle CSS-properties te animeren zijn. In de officiële documentatie is een lijst beschikbaar met een opsomming van de properties die je kunt animeren, oftewel, waar je transitions op kunt toepassen. Zie hier de volledige lijst van properties die je kunt animeren.
Wil je meer lezen over transitions? Bekijk de officiële documentatie en verdiep je verder in transitions en hoe het werkt.

Opdrachten

Opdracht 1

Zoek uit wat transitions zijn en beschrijf met eigen woorden wat het betekent.

Opdracht 2

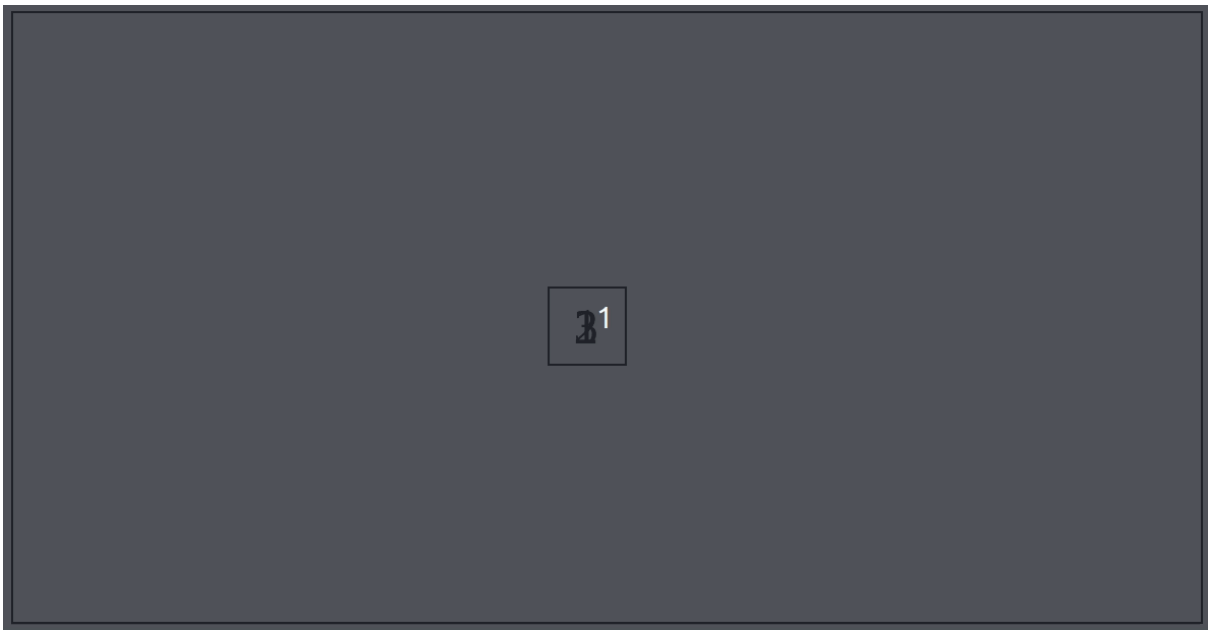
Maak een simpele transition wat de kleur van een tekst aanpast.

Opdracht 3

Transitions kent verschillende transition-timing. Zie hieronder een voorbeeld. Maak het na.

Opdracht 4

Je kunt op verschillende properties transitions aangeven. Zie hieronder een voorbeeld. Zoek uit op welke properties de transitions zitten en maak het ongeveer hetzelfde na.



Opdracht 5

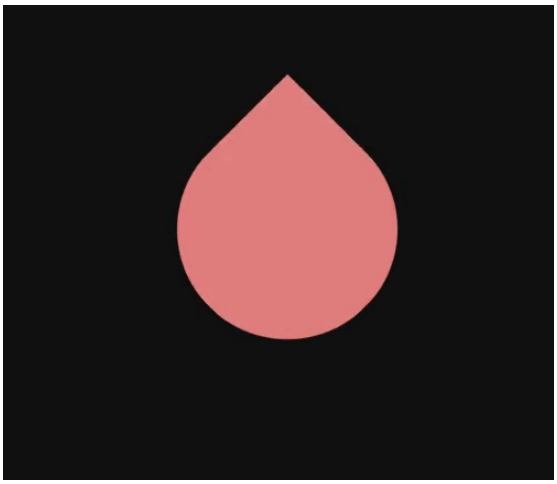
Hieronder zie je wederom een voorbeeld van een transition. Je kunt op verschillende properties transitions aangeven. Zoek uit op welke properties de transitions zitten en maak het na. Je eigen creativiteit gebruiken is uiteraard toegestaan.

Opdracht 6

Zie de onderstaande card deck. Kun jij deze zelf namaken? Je mag uiteraard weer je eigen creativiteit erop los laten

Opdracht 7 - extra opdracht (optioneel)

Zie de onderstaande animatie en probeer deze na te maken. Hier is gebruik gemaakt van keyframe animatie.

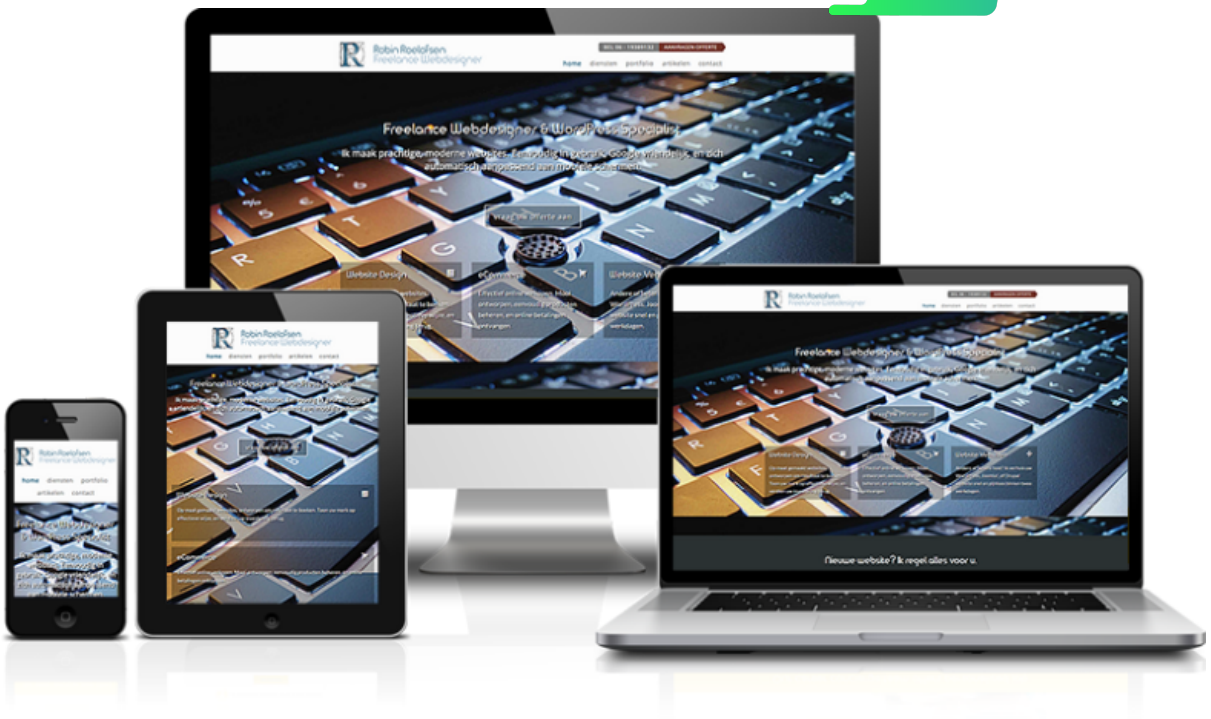


7. Responsive web design

Liefst wil je dat je website zich automatisch aanpast naar het apparaat dat je gebruikt, of eigenlijk naar de grootte van het browserscherm op dat apparaat.

Je maakt je site responsive met media queries. Een media query is een stukje code in de CSS stylesheet van je website, dat informatie zoals de grootte van het browserscherm van de bezoeker opvraagt tijdens het laden van jouw website. Weet je site wat de grootte van het browserscherm is, dan laat je site de opgevraagde informatie zien op de manier waarop je dat hebt opgezet. Dit stel je in met specifieke CSS stijlregels voor de verschillende schermgroottes.

Een website die zich met behulp van media queries aanpast aan de grootte van het browserscherm waarop het getoond wordt, heet een **responsive website**.
De kleinere browserschermen worden voornamelijk op mobiele apparaten gebruikt, maar je kunt het responsive effect ook zien wanneer je het browserscherm op je desktop verkleint. De media queries kijken namelijk alleen maar naar de grootte het browserscherm, en niet naar de grootte van het scherm zelf. Bekijk hier de officiële documentatie om meer te leren over media queries.



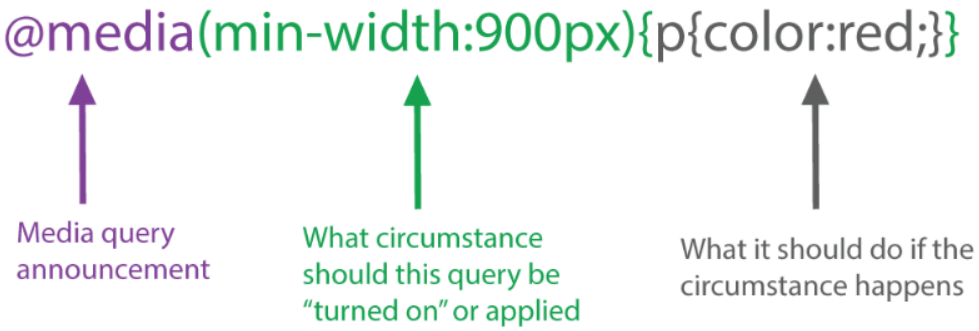
Media queries werken eigenlijk heel simpel. De media query die je gebruikt zoekt uit wat de grootte is van het browserscherm en past vervolgens de door jou gewenste CSS toe als desbetreffende schermgrootte wordt gevonden.

Hieronder zie je een media query staan die we stap voor stap gaan ontleden.

```
@media(min-width:900px){p{color:red;}}
```

Wat hier eigenlijk staat is "Als het scherm wijder is dan 900 pixels dan moet de kleur van de tekst rood worden.

Het onderstaande plaatje laat nog eens goed zien hoe de media query is opgedeeld.



Hoe je media queries moet gebruiken om CSS-regels toe te passen op basis van schermgrootte en resolutie heb je hierboven kunnen zien, maar hoe bepaal je welke queries je moet instellen?

De punten waarop media queries worden ingesteld, worden 'breakpoints' genoemd. Breakpoints zijn de schermformaten waarop jouw webpagina niet correct meer wordt weergegeven. Als we bijvoorbeeld tablets willen targeten die in liggende stand staan, kunnen we de volgende breakpoint maken:

```
@media only screen and (min-width: 768px) and (max-width: 1024px) and (orientation:
landscape) {
   /* hier plaats je dan je gewenste CSS regels */
}
```

In het bovenstaande voorbeeld wordt een schermformaat gecreëerd ter grootte van een tablet in liggende modus en wordt ook de oriëntatie aangegeven.
Het instellen van breekpunten voor elk denkbaar apparaat zou echter ongelooflijk moeilijk zijn, omdat er veel apparaten zijn met verschillende vormen en maten. Daarnaast komen er elk jaar nieuwe toestellen uit met nieuwe schermformaten.

In plaats van breekpunten in te stellen op basis van specifieke apparaten, is het het beste om het formaat van de browser aan te passen om te zien waar de website van nature breekt op basis van de inhoud. De afmetingen waarop de lay-out 'breekt' oftewel, er vreemd uitziet, worden de breekpunten van jouw media query. Binnen die breekpunten kunnen we de CSS aanpassen om het formaat van de webpagina te wijzigen en te reorganiseren.

Door de afmetingen te observeren waarop een website van nature 'breekt', kun je mediaquery-breekpunten instellen die de best mogelijke gebruikerservaring geven bij desbetreffende specifieke project waar je mee bezig bent. Dit is veel beter dan dat je elk project dwingt om op een bepaalde schermgrootte te passen. Verschillende projecten hebben verschillende behoeften, en het maken van een responsief ontwerp zou niet anders moeten zijn.

Bekijk deze lijst met breekpunten op basis van apparaat breedte. Gebruik het als referentie voor schermbreedtes om je website te testen om er zeker van te zijn dat deze er op verschillende apparaten goed uitziet.

Opdrachten

Om de opdrachten uit te kunnen voeren heb je natuurlijk als eerste een webpage nodig die **niet** responsive is. Die hoef je voor het gemak niet helemaal zelf te maken. Zie hieronder de code die jij kunt kopiëren en plakken in Visual Studio Code. Maak voor de HTML code een apart HTML bestand en voor de CSS code een apart CSS bestand en link de CSS aan de HTML. Op de plaatsen in de CSS code waar images zijn gebruikt kun je die images vervangen door eigen images te gebruiken of je download de in het project gebruikte images hieronder. Vergeet niet je imagepath ook nog even juist aan te passen als je de onderstaande code kopieert.
space.jpg (achtergrond image)
spaceship.png (logo)
spaceship2.png (logo voor grote resolutie)

HTML Code

```
<html>

<head>
 <title>Amazing Space</title>
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <link rel="stylesheet" type="text/css" href="responsive.css">
 <link href="https://fonts.googleapis.com/css?family=Space+Mono:400,700" rel="stylesheet">
</head>

<body>
 <header>
  <div class="page-title">
   <div class="logo"></div>
   <h1>Amazing Space</h1>
   <h5>At Amazing Space, there's always space for you.&trade;</h5>
  </div>
 </header>
 <div class="main">
  <p class="page-description clearfix">
    Otherworldly views, uncompromising service, unparallelled luxury. This is Amazing Space, the world's first, and finest, space adventure company. With private and group voyages embarking weekly, you can book your tickets in advance or on the fly.
   <a class="learn-more" href="#">Learn More >></a>
  </p>
  <div class="gallery">
   <figure class="gallery-item">
                                                    <img class="thumbnail" src="https://s3.amazonaws.com/codecademy-content/projects/make-a-website/lesson-3/space01.jpg">
   </figure>
   <figure class="gallery-item">
                                                    <img class="thumbnail" src="https://s3.amazonaws.com/codecademy-content/projects/make-a-website/lesson-3/space02.jpg">
   </figure>
   <figure class="gallery-item">
                                                    <img class="thumbnail" src="https://s3.amazonaws.com/codecademy-content/projects/make-a-website/lesson-3/space03.jpg">
   </figure>
   <figure class="gallery-item">
```

```
                                                    <img class="thumbnail"
src="https://s3.amazonaws.com/codecademy-content/projects/make-a-website/lesson-3/space
04.jpg">
    </figure>
    <figure class="gallery-item">
                                                    <img class="thumbnail"
src="https://s3.amazonaws.com/codecademy-content/projects/make-a-website/lesson-3/space
05.jpg">
    </figure>
    <figure class="gallery-item">
                                                    <img class="thumbnail"
src="https://s3.amazonaws.com/codecademy-content/projects/make-a-website/lesson-3/space
06.jpg">
    </figure>
    <figure class="gallery-item">
                                                    <img class="thumbnail"
src="https://s3.amazonaws.com/codecademy-content/projects/make-a-website/lesson-3/space
07.jpg">
    </figure>
    <figure class="gallery-item">
                                                    <img class="thumbnail"
src="https://s3.amazonaws.com/codecademy-content/projects/make-a-website/lesson-3/space
08.jpg">
    </figure>
  </div>
 </div>
 <nav class="footer">
  <ul>
   <li>ABOUT</li>
   <li>MISSIONS</li>
   <li>ARCHIVE</li>
  </ul>
  <div class="contact-btn">
   <a>CONTACT US</a>
  </div>
 </nav>
</body>

</html>
```

CSS Code

```
html,
body {
```

```
  max-width: 1200px;
  margin: auto;
  font-family: "Space Mono", sans-serif;
  font-size: 18px;
  line-height: 1.4em;
  color: GhostWhite;
}
body {
  background-image: url("/CSS-test/space.jpg");
  height: 100%;
  background-attachment: fixed;
  background-size: cover;
}
.clearfix {
  clear: both;
}
/* Title and Description */
.page-title {
  text-align: center;
  margin: auto;
  line-height: 2em;
}
.logo {
  height: 80px;
  width: 80px;
  background-image: url("/CSS-test/spaceship.png");
  background-size: contain;
  background-repeat: no-repeat;
  display: inline-block;
  position: relative;
  top: 1em;
}
.page-title h1 {
  display: inline-block;
}
.main {
  text-align: center;
}
.page-description {
  font-size: 0.8rem;
  padding: 30px;
}
.learn-more {
```

```
  color: DarkOrange;
  text-decoration: none;
  display: block;
  margin: 25px 0 0 0;
  font-weight: bold;
}
/* Gallery */
.gallery {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  margin-top: 20px;
}
.gallery-item {
  margin: 20px;
}
.gallery-item .thumbnail {
  width: 240px;
  border: 2px solid GhostWhite;
  border-radius: 4px;
}
/* Footer Navigation */
nav {
  margin-top: 30px;
}
nav ul {
  display: block;
}
nav li {
  display: inline;
  margin: 0 20px 0 0;
  color: GhostWhite;
}
nav a {
  line-height: 60px;
  border: 1px solid GhostWhite;
  padding: 7px;
  border-radius: 4px;
  color: DarkOrange;
  text-decoration: none;
}
.contact-btn {
  cursor: pointer;
```

```
  margin: 20px 30px;
  text-align: center;
}
.contact-btn a:active {
  position: relative;
  bottom: 2px;
}
```

Opdracht 1

Schrijf onderaan je CSS bestand een media query voor een maximale breedte van 480px. Hierdoor kunnen we de breedte van het page-title element op kleinere schermen verkleinen. Als het scherm minder dan 480px breed is, geeft je de .page-title class een breedte van 270px. Hierdoor wordt het page-title element duidelijker weergegeven op kleine schermen. Test je code door het formaat van de browser aan te passen in element inspection.

Opdracht 2

Laten we de gallery afbeeldingen groter laten lijken als de schermgrootte klein tot middelgroot is. Schrijf een media query voor schermformaten met een bereik tussen 320px en 480px. Gebruik min-breedte en max-breedte om het bereik te definiëren.

Selecteer in de media query de thumbnails van de gallery met .gallery-item .thumbnail en geef ze een breedte van 95%. Als het goed is zul je zien dat de gallery afbeeldingen breder lijken wanneer de schermgrootte tussen 320 en 480 pixels breed is.

Opdracht 3

Schrijf een media query om een logo van een hogere kwaliteit te tonen als de bezoeker naar de Amazing Space website kijkt op een scherm met hoge resolutie. Een beeldscherm met hoge resolutie heeft mogelijk een min-resolutie van 150 dpi.

Als je je pagina opent op een scherm met een resolutie groter dan 150 dpi, zul je de verandering van het logo zien. Als dit niet het geval is (omdat je een kleinere resolutie hebt), kun je in de media query de min-resolution op een lagere waarde zetten om de verandering te zien.

Opdracht 4

De tekst van de website moet groter zijn voor gebruikers met kleine schermen met een lage resolutie. Schrijf een media query die van toepassing is wanneer de maximale resolutie 150 dpi is en het scherm een maximale breedte heeft van 480px.
Maak binnen de media query de font-size van het page-description element 20px.

Opdracht 5

Ga nog eens terug naar de eerste media query waarin je je richtte op schermen met een minimale breedte van 320 px en een maximale breedte van 480 px. Laten we ook het logo en de tekst verticaal verschijnen als het scherm in ‘portrait’ oriëntatie staat.

Voeg nog een mediafunctie toe aan de regel door een komma te gebruiken om regels te scheiden. De tweede mediafunctie moet controleren of de oriëntatie van het scherm daadwerkelijk 'portrait' is.

Opdracht 6

De laatste breakpoint waar we rekening mee willen houden is een tablet in liggende stand (landscape). De Amazing Space website zou het formaat moeten veranderen om de gallery afbeeldingen aan de rechterkant weer te geven, terwijl het logo en de beschrijving aan de linkerkant staan. Schrijf een media query die aan de volgende vereisten voldoet:

- Het scherm heeft een minimale breedte van 768px;
- Het scherm heeft een maximale breedte van 1024px;
- Het scherm heeft als oriëntatie 'landscape'.

Deze query zorgt ervoor dat de paginatitel en beschrijving links van de gallery afbeeldingen komen te zweven. Pas het formaat van de browser aan om deze veranderingen op verschillende schermbreedtes te observeren via element inspection.

Test nu je project op responsiveness door het op verschillende schermen te tonen om te kijken of jouw breakpoints werken. Je kunt dit doen door 'element inspection' te gebruiken, dat ingebouwd is in je browser.